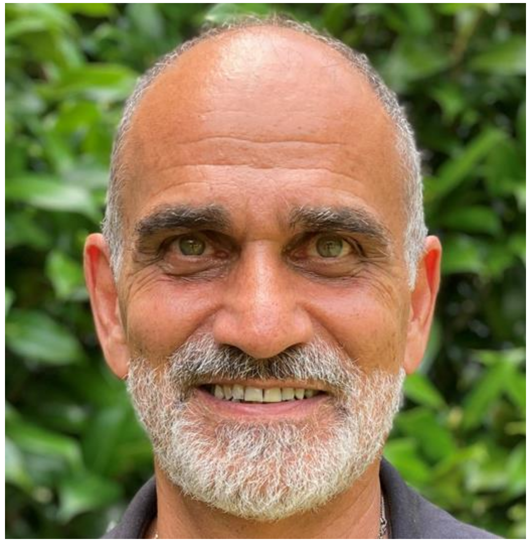
AWR Deep Dive: Truths, Troubles, and Tuning Techniques from the Field

EAST Coast Oracle Conference
November 4, 2025

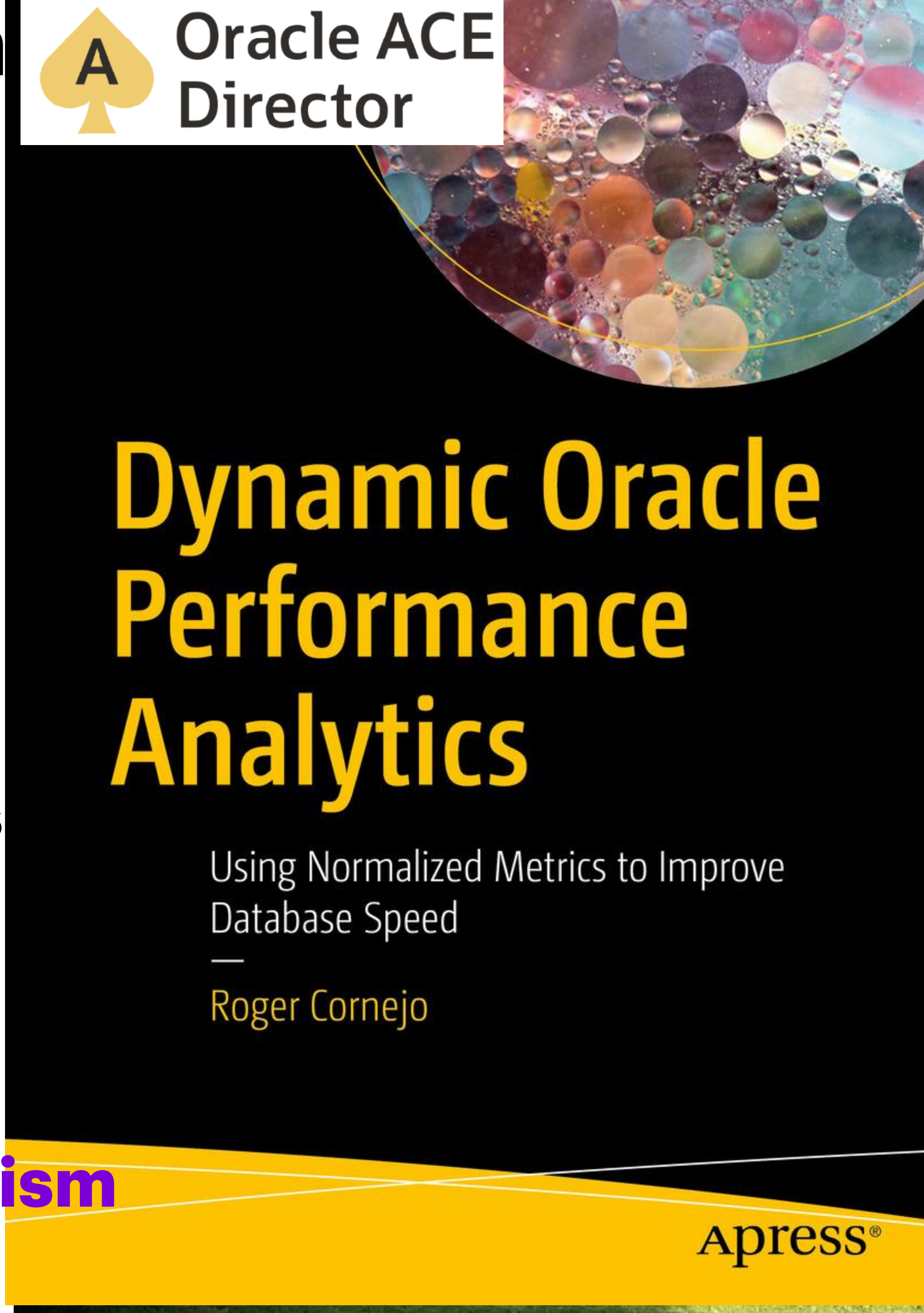
Roger Cornejo
Technology Innovation Principal Director, GenAI
Accenture Enkitec Group
Roger.Cornejo@Accenture.com
X @OracleDBTuning
RogerCornejo.com



Oracle ACE Director

roger.cornejo@accenture.com
Technology Innovation Principal Director, GenAI
Accenture Enkitec Group

- 40+ yrs Oracle experience (since V4)
  - Developer / Tech Lead / Project Manager / Application DBA / Enterprise Architect
- ~15 years deep dive in Performance
  - 300 Production Instances
  - 2500 Applications
  - ECO / OCW 19&24 / QUEST / RMOUG / Hotsos
- OCI and Database Architecture
- **Focus: Oracle/OCI GenAI capabilities**
- Dynamic Oracle Performance Analytics
  Statistical Anomaly Detection Mechanism



>

AWR Deep Dive: Truths, Troubles, and Tuning Techniques from the Field

Agenda

- ❖ASH: SQL Arrival Times; CPU by User
- ❖SQLSTAT: Anomaly Detection intro.
- ❖Interesting ways to use SQL Plan History
- ❖Ways to Evaluate Session Leaking
- ❖Stats: History and Gather Operations
- ❖Parameter Changes
- ❖Where to Look in AWR for problems related to CPU/Memory/IO/Network/REDO/UNDO/TEMP
- ❖Evaluating Concurrency Issues
- ❖Statistical Anomaly Detection
- ❖Metric Correlation Analysis

Goal: Explore the uncharted waters of Oracle performance metrics and ignite a fire for learning and experimentation.

154 AWR Views

Quick TIP: LAG fn

lag - Analytics Function

See the value for a previous row in the table

Use Case: Convert Cumulative values to Deltas for metric analysis

- ❖ Some Oracle metrics are persisted as cumulative values
- ❖ Example using LAG function to get deltas for a period:

```
select snap_id , value cumulative_value
, value - lag(value)
  over (order by instance_number, snap_id)
  as IO_wait_time_delta
from dba_hist_sysstat
     natural join dba_hist_snapshot
where begin_interval_time >= trunc(sysdate)
  and instance_number = 1
  and stat_name = 'user I/O wait time'
order by snap_id ;
```

value in an interval

SNAP ID	CUMULATIVE VALUE	IO WAIT TIME DELTA
13950	18,695,752	3,295
13951	18,704,917	9,165
13952	18,727,003	22,086
13953	18,732,526	5,523
13954	18,795,706	63,180
13955	18,797,583	1,877

Dynamic Oracle Performance Analytics

ASH

DBA_HIST_ACTIVE_SESS_HISTORY

ASH: Use Case - SQL Arrival Times

Feature Engineering with Date Intervals

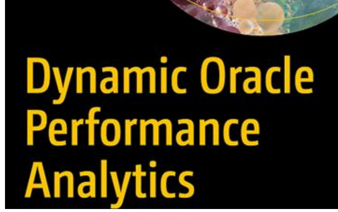
- ❖ **Analytics Concept: Feature Engineering**
  - ❖ **"Art/Science" of creating new metrics from existing data**

- ❖ **Example: Interval between date/times**
  a.k.a. Duration between events or Time since last/next event
  often more useful than the date/time itself
    - ❖ **Example: time until next purchase [Amazon ... use case]**

- ❖ **SQL Performance Use Case - SQL Arrival Times**
  - ❖ **# seconds until next execution of a query**
  - ❖ **i.e. the actual SQL queuing**

DBA_HIST_ACTIVE_SESS_HISTORY

ASH: Use Case - SQL Arrival Times



Feature Engineering with Date Intervals

lead - Analytics Function

See the value for the next row

```
with ash as select sql_id, sql_plan_hash_value
, ((lead(sql_exec_start) over (order by sql_exec_id)
  - sql_exec_start )  * (24*60*60)) seconds_to_next_exec
, sum(10) secs, sql_exec_start as sql_exec_start_time
from dba_hist_active_sess_history   a
where sql_exec_start >= trunc(sysdate-nvl(:days_back, 0))
  and SQL_ID = 'g7pa3kp7524ha'
  and sql_exec_id is not null
group by sql_id, sql_plan_hash_value, sql_exec_id, sql_exec_start
)
select ash.*
, case when seconds_to_next_exec < secs then '*******' end flag
from ash order by sql_id, sql_exec_start_time ;
```

ASH: Use Case - SQL Arrival Times

Feature Engineering with Date Intervals

Dynamic Oracle Performance Analytics

lead - Analytics Function

See the value for the next row

```
with ash as select sql_id, sql_plan_hash_value
, ((lead(sql_exec_start) over (order by sql_exec_id)
  - sql_exec_start )  * (24*60*60)) seconds_to_next_exec
, sum(10) secs, sql_exec_start as sql_exec_start_time
from dba_hist_active_sess_history   a
where sql_exec_start >= trunc(sysdate-nvl(:days_back, 0))
  and SQL_ID = 'g7pa5kp7524ha'
  and sql_exec_id is not null
group by sql_id, sql_plan_hash_value, sql_exec_id, sql_exec_start
)
select ash.*
, case when seconds_to_next_exec < secs then '*******' end flag
from ash order by sql_id, sql_exec_start_time ;
```

When the Execution of a SQL Statement Started

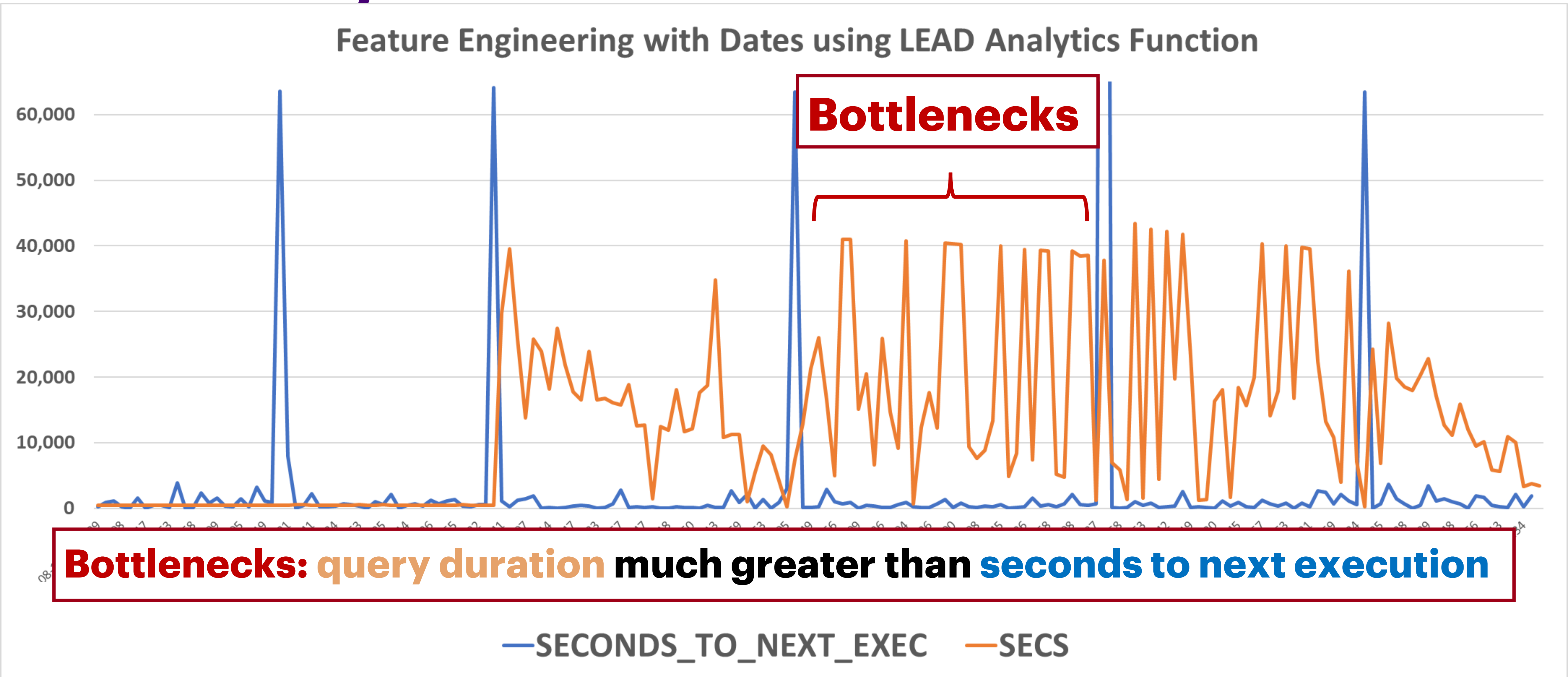
An Individual Execution of a SQL Statement

ASH: Use Case - SQL Arrival Times

Feature Engineering with Date Intervals

lead - Analytics Function

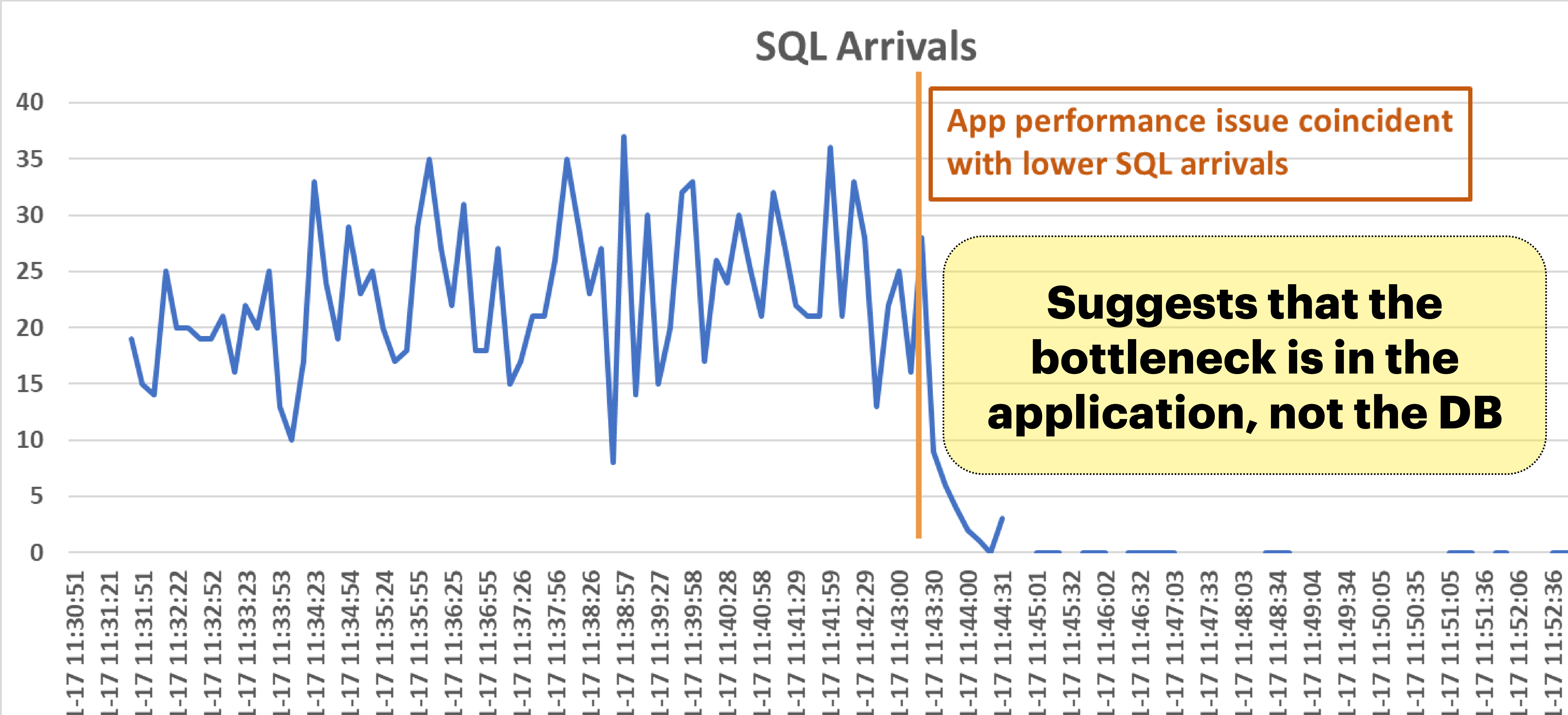
See the value for the next row



Bottlenecks: query duration much greater than seconds to next execution

ASH: Use Case - SQL Arrival Times

Feature Engineering with Date Intervals



ASH: CPU/WAITS by User or Service

AAS = All session states

Where is my CPU going? Which users contribute most to AAS?

```
-- ASH User and Service CPU only
-- =============================
select username
, name service_name
, sum(10) "CPU (sec)"
from dba_hist_active_sess_history ash
, dba_users du
, dba_services s
where trunc(sample_time) >= trunc(sysdate)
  and session_state = 'ON CPU'
  and du.user_id      = ash.user_id
  and s.name_hash     = ash.service_hash
group by du.username, s.name
;
```

ASH - Core Diagnostic Queries

Identify Sessions

- **Blocked Session** at various aggregation levels
  - Group by Session
  - Group by SQL_ID

- **Top Sessions** at various aggregation levels
  - Group by Event, SQL, User
  - Group by SQL, User
  - Group by User

Anomaly Detection against ASH Metrics <later>

DBA_HIST_ACTIVE_SESS_HISTORY

SQLSTAT

SQLSTAT: Which SQL executions are Anomalous?

- ❖ **Use Case: SQL is suspected as needing tuning**
- ❖ **What are the occurrences of the SQL when it behaved unusually?**

```
with ranges as
(
select sql_id
, round( avg(elapsed_time_delta/executions_delta) ) as avg_elap_per_exec
, round(  avg(elapsed_time_delta/executions_delta)
+ (2 * stddev(elapsed_time_delta/executions_delta) )) as upper_elap_per_exec
, round( percentile_cont(0.95) within group (order by
elapsed_time_delta/executions_delta )) as pctile_95
from dba_hist_sqlstat natural join dba_hist_snapshot
where sql_id = :sql_id
  and begin_interval_time >= trunc(sysdate) - 15
  and executions_delta > 0
group by sql_id
)
-- ...
```

DBA_HIST_SQLSTAT

SQLSTAT: Which SQL executions are Anomalous?

- ❖ **Use Case: SQL is suspected as needing tuning**
- ❖ **What are the occurrences of the SQL when it behaved unusually?**

```
with ranges as
(
select sql_id
, round( avg(elapsed_time_delta/executions_delta) ) as avg_elap_per_exec
, round(  avg(elapsed_time_delta/executions_delta)
+ (2 * stddev(elapsed_time_delta/executions_delta) )) as upper_elap_per_exec
, round( percentile_cont(0.95) within group (order by
elapsed_time_delta/executions_delta )) as pctile_95
from dba_hist_sqlstat natural join dba_hist_snapshot
where sql_id = :sql_id
  and begin_interval_time >= trunc(sysdate) - 15
  and executions_delta > 0
group by sql_id
)
-- ...
```

Normal Range Upper Bound

DBA_HIST_SQLSTAT

SQLSTAT: Which SQL executions are Anomalous?

❖ **Use Case: SQL is suspected as needing tuning**

❖ **What are the occurrences of the SQL when it behaved unusually?**

```
with ranges as
(
select sql_id
, round( avg(elapsed_time_delta/executions_delta) ) as avg_elap_per_exec
, round(  avg(elapsed_time_delta/executions_delta)
+ (2 * stddev(elapsed_time_delta/executions_delta) )) as upper_elap_per_exec
, round( percentile_cont(0.95) within group (order by
elapsed_time_delta/executions_delta )) as pctile_95
from dba_hist_sqlstat natural join dba_hist_snapshot
where sql_id = :sql_id
  and begin_interval_time >= trunc(sysdate) - 15
  and executions_delta > 0
group by sql_id
)
-- ...
```

95th Percentile

Deep Dive on Normal Ranges and Percentiles for Anomaly Detection is in the Appendix.

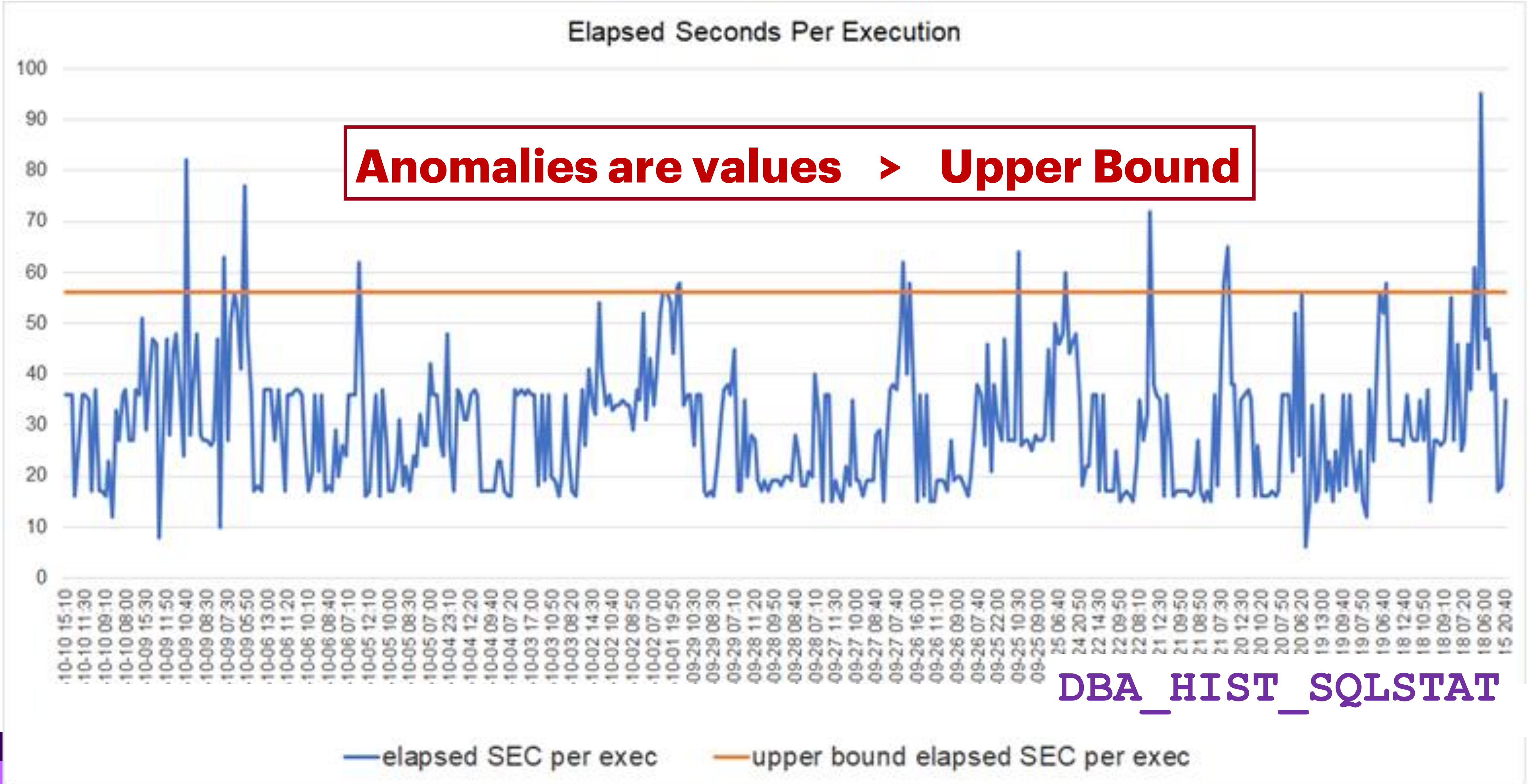
SQLSTAT: Which SQL executions are Anomalous?

❖ **Use Case: SQL is suspected as needing tuning**

❖ **What are the occurrences of the SQL when it behaved unusually?**

```
select to_char(trunc(begin_interval_time, 'MI'), 'YYYY-MM-DD HH24:MI') as
begin_interval_time
, ss.sql_id, plan_hash_value
, case when elapsed_time_delta/executions_delta > upper_elap_per_exec
          then '****' else null end as anomalies
, round(elapsed_time_delta / executions_Delta / 1000000, 3) "elap sec per
exec"
, round(avg_elap_per_exec / 1000000, 3) "AVG elap sec per exec"
, round(upper_elap_per_exec / 1000000, 3) "UPPER BOUND elap sec per exec"
, round(pctile_95 / 1000000, 3) "95th PCTILE elap sec per exec"
from dba_hist_sqlstat ss natural join dba_hist_snapshot
, ranges
Where ss.sql_id = ranges.sql_id  and executions_delta > 0
  and begin_interval_time >= trunc(sysdate) - 15
order by snap_id desc ;
```

SQLSTAT: Which SQL executions are Anomalous?



SQLSTAT - Core Diagnostic Queries

Expensive SQL

- **by Day**
  - **Identify Plan flips**
  - **Visually Compare metrics**

- **by snap_id**
  - **Most granular you can get with SQLSTAT**

Anomaly Detection against SQLSTAT Metrics <later>

SQL Plan History

<SKIP>

SQL Plan History

- **Use Cases:**
- **What SQL are involved with an object?**
- **What objects are involved with a specific SQL ID?**
- **Get the stats for the objects for a specific SQL ID.**

DBA_HIST_SQL_PLAN

SQL_ID's for an Object

Evaluate all queries that touch an object

❖**Use Case:** **Focus in on SQL from an important object (or set).**

```
-- SQL_ID's for an object:
with objects as
(
select 'TABLE' object_type, owner, table_name object_name
from dba_tables
where owner = :owner and table_name = :object_name
union
select 'INDEX' object_type, owner, index_name object_name
from dba_indexes
where table_owner = :owner and table_name = :object_name
)
-- ...
```

Get the table in a row

Get the indexes

DBA_HIST_SQL_PLAN

SQL_ID's for an Object

Evaluate all queries that touch an object

❖**Use Case:** **Focus in on SQL from an important object (or set).**

```
-- ...
select distinct sql_id
from dba_hist_sql_plan sp
, dba_objects do
, objects o
where object#  = object_id
  and do.owner = o.owner
  and do.object_name = o.object_name
  and do.object_type = o.object_type
  and object# is not null
order by sql_id
;
```

Get the SQL_ID's from the `objects` common table expression

DBA_HIST_SQL_PLAN

Objects for a SQL_ID

Evaluate all objects touched by a query

❖**Use Case:** Evaluate all table and index stats for a query.

```
-- Objects for a SQL_ID:
select distinct sp.object_owner
, sp.object_name, do.object_type
from dba_hist_sql_plan sp
, dba_objects do
where object#  = object_id
  and sql_id = :sql_id
  and object# is not null
order by do.object_type desc, object_owner, object_name
;
```

DBA_HIST_SQL_PLAN

Objects for a SQL_ID

Evaluate all objects touched by a query

❖**Use Case:** **Evaluate Stats for all objects involved with a query**

```
-- Compare Table Stats for a SQL_ID:
with all_tabs_for_plan as
(select distinct sql_id, sp.object_owner table_owner
, sp.object_name table_name, do.object_type
from dba_hist_sql_plan sp, dba_objects do
where do.object_type = 'TABLE' and object#  = object_id
  and sql_id = :sql_id
  and object# is not null)
select sql_id, table_owner, table_name, report
from all_tabs_for_plan
, table(dbms_stats.diff_table_stats_in_history(table_owner,
table_name, trunc(sysdate-1), trunc(sysdate),0) )
;
```

DBA_HIST_SQL_PLAN

Objects for a SQL_ID

Evaluate all objects touched by a query

❖**Use Case:** Evaluate Stats for all objects involved with a query

```
-- Compare Table Stats for a SQL_ID:
with all_tabs_for_plan as
(select distinct sql_id, sp.object_owner table_owner
, sp.object_name table_name, do.object_type
from dba_hist_sql_plan sp, dba_objects do
where do.object_type = 'TABLE' and object#  = object_id
  and sql_id = :sql_id
  and object# is not null)
select sql_id, table_owner, table_name, report
from all_tabs_for_plan
, table(dbms_stats.diff_table_stats_in_history(table_owner,
table_name, trunc(sysdate-1), trunc(sysdate),0) )
;
```

How much a tables stats have changed over time

DBA_HIST_SQL_PLAN

<SKIP>

Session Leaking

Session Leaking

- **Use Case:**
- **App is having session leaking as identified by**
  - **connection pool errors**
  - **failed logons**
- **What metrics are involved?**
  - **DBA_HIST_SERVICE_STAT**
  - **Get service name for user from ASH**
  - **DBA_AUDIT_TRAIL**
  - **DBA_HIST_SQLSTAT** **(SYS SQL writing to Audit Trail)**

Logons for a Service

Evaluate logons to help detect session leaking

```
-- Logons for a Service
select to_char(begin_interval_time, 'YYYY-MM-DD HH24:MI') as
begin_interval_time
, service_name, snap_id, instance_number, stat_name
, value, value - lag(value)
over (partition by instance_number, stat_name order by snap_id)
delta_value
from dba_hist_service_stat natural join dba_hist_snapshot
where stat_name = 'logons cumulative' and service_name =
:service_name
  and begin_interval_time >= trunc(sysdate) - 2
  and instance_number = 1
order by 1;
```

DBA_HIST_SERVICE_STAT

Service Stat

Side Bar Note: Research 28 metrics of resource usage by service

[Blog: Discover the Hidden Secrets of CPU Utilization in Oracle Databases]

[Blog: Discovering Hidden Secrets of CPU Utilization in Oracle Databases, pt2]

[Oracle Doc.: SYSSTAT Statistics Descriptions]

[Oracle Doc.: Stat Names documented in V$SESS_TIME_MODEL]

DBA_HIST_SERVICE_STAT

Logons for a Service

Side Bar: if you only know username, can get service name from ASH

```
-- get usernames for a service from ASH
select username, name service_name, count (*) *10 ash_seconds
from dba_hist_active_sess_history
, dba_users
, dba_services
where trunc(sample_time) > trunc(sysdate)
  and dba_users.user_id = dba_hist_active_sess_history.user_id
  and name_hash = service_hash
  and username like nvl(:username, username) and name like
nvl(:service_name, name)
group by username, name
order by name
;
```

Logons and Logoffs per second for a User

Evaluate audited logons to help detect session leaking

```
-- logons and logoffs per second for a user
select to_char(timestamp, 'YYYY-MM-DD HH24:MI:SS') date_time
, username
, sum(decode(action_name, 'LOGON',  1, 0)) logon_per_sec
, sum(decode(action_name, 'LOGOFF', 1, 0)) logoff_per_sec
from dba_audit_trail
where username = nvl(:username, username)
  and timestamp >= trunc(sysdate) - nvl(:days_back, 14)
  and action_name in ('LOGON', 'LOGOFF')
group by to_char(timestamp, 'YYYY-MM-DD HH24:MI:SS'), username
order by 1
;
```

Not part of AWR DBA_AUDIT_TRAIL

<SKIP>

Stats

STATS issues:

- **Metrics:**
  - **<none in AWR>**
  - **DBA_TAB_STATS_HISTORY – when the stats changed**
- **Health Checks:**
  - Check stale stats and non-gathered stats
  - DBA_OPTSTAT_OPERATIONS –
    what stats gathering operations were done and when

- **Rules of Thumb:**
  - Need to know the life-cycle of data/objects
  - **No historic stats in AWR**
    → Stats used by tuning advisor are current stats
  - Reference doc's and Oracle Real World Performance

Objects for a SQL_ID

Evaluate all objects touched by a query

❖ **Use Case:** Evaluate Stats for all objects involved with a query

```
-- Compare Table Stats for a SQL_ID:
with all_tabs_for_plan as
(select distinct sql_id, sp.object_owner table_owner
, sp.object_name table_name, do.object_type
from dba_hist_sql_plan sp, dba_objects do
where do.object_type = 'TABLE' and object#  = object_id
  and sql_id = :sql_id
  and object# is not null)
select sql_id, table_owner, table_name, report
from all_tabs_for_plan
, table(dbms_stats.diff_table_stats_in_history(table_owner,
table_name, trunc(sysdate-1), trunc(sysdate),0) )
;
```

dbms_stats.diff_table_stats_in_history

Objects for a SQL_ID

Evaluate all objects touched by a query

❖ **Use Case:** Evaluate Stats for all objects involved with a query

```
-- Compare Table Stats for a SQL_ID:
with all_tabs_for_plan as
(select distinct sql_id, sp.object_owner table_owner
, sp.object_name table_name, do.object_type
from dba_hist_sql_plan sp, dba_objects do
where do.object_type = 'TABLE' and object#  = object_id
  and sql_id = :sql_id
  and object# is not null)
select sql_id, table_owner, table_name, report
from all_tabs_for_plan
, table(dbms_stats.diff_table_stats_in_history(table_owner,
table_name, trunc(sysdate-1), trunc(sysdate),0) )
;
```

How much a tables stats have changed over time

dbms_stats.diff_table_stats_in_history

<SKIP>

Parameter Changes

Parameter Changes

Evaluate parameter changes to understand configuration

Ex: DB "broken" after refresh from PRD: → delta to *db_file_multi_block_read_count* and *db_cache_size*

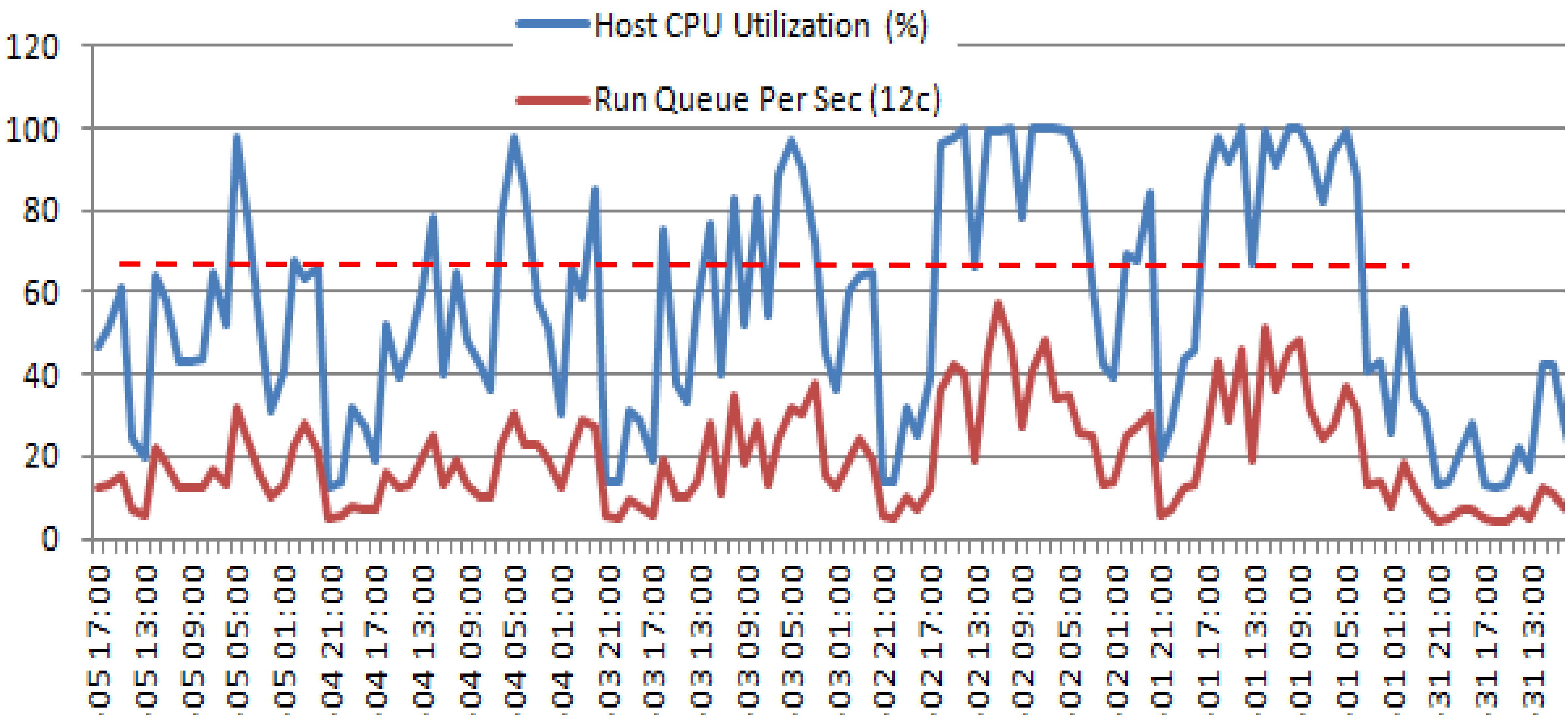
```
-- parameter changes
with all_parameters as
(select snap_id, parameter_name, value
, lag(value) over (partition by dbid, instance_number,
parameter_hash order by snap_id) prior_value
from dba_hist_parameter)
select to_char(begin_interval_time, 'YYYY-MM-DD HH24:MI')
change_date, parameter_name, value, prior_value
from all_parameters p, dba_hist_snapshot s
where value != prior_value
  and s.snap_id = p.snap_id
  and begin_interval_time >= trunc(sysdate) - nvl(:days_back, 14)
  and parameter_name like nvl(:parameter_name, parameter_name)
order by parameter_name, begin_interval_time desc ;
```

DBA_HIST_PARAMETER

<SKIP>

CPU

CPU issues:



DBA_HIST_SYSMETRIC_SUMMARY

CPU issues:

Metrics:

- OS Load — dba_hist_osstat
- **CPU Queue** / CPU usage% — dba_hist_sysmetric_summary
- Logon Count — dba_hist_sysmetric_summary

Blog: Discover the Hidden Secrets of CPU Utilization in Oracle Databases

Blog: Discovering Hidden Secrets of CPU Utilization in Oracle Databases, pt2

Health Checks:

- High CPU Consuming SQL from ASH or SQLSTAT
- **Check for Connection Storms – can max out CPU**

Rules of Thumb:

- # connections: < 100 / CPU Core
- Best response time: < 65% CPU usage

Memory

<SKIP>

Memory issues:

- **Metrics:**
  - SGA/PGA Usage from DB dba_hist_sga / dba_hist_pgastat
    - **and other DB's on same machine**
  - OS paging dba_hist_osstat
- **Check:**
  - ADDM Report findings
    Action as per % impact to DB Time
  - ASH.pga_allocated: Identify Sessions + SQL Monitor Report
- **Rules of Thumb – also consider:**
  - Memory demand from other DB's on machine
  - OS memory needs

Memory issues: ERROR: ORA-04036:

<SKIP>

ORA-04036: PGA memory used by the instance exceeds PGA_AGGREGATE_LIMIT

Script: AWR - PGA Allocation and Usage.sql

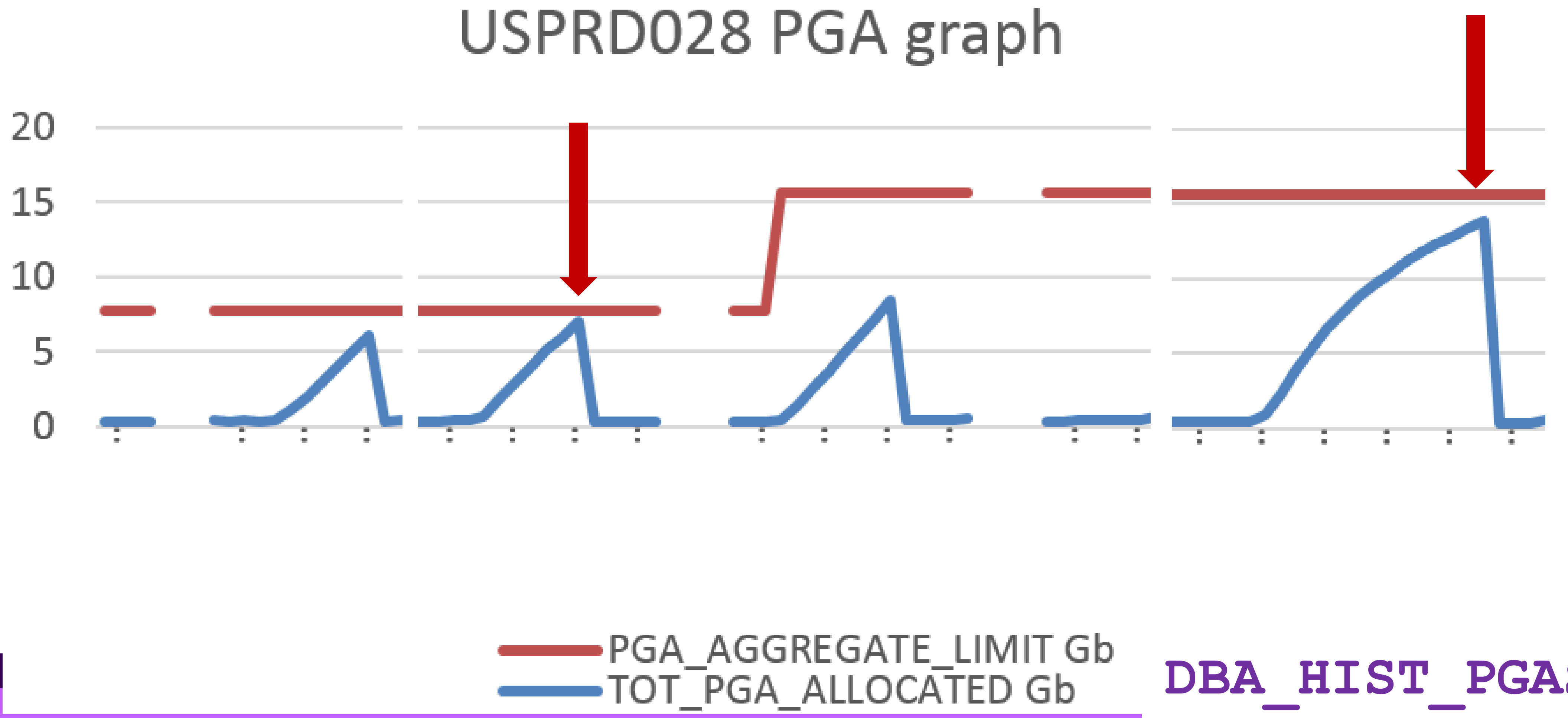
- **Rules of Thumb**
- Prior to 12c: PGA usage could exceed the target
  => high rate of swapping => performance issues
- In 12c and above:
  - **avoid setting this limit to a very high # just to avoid the error**
  - rather take the opportunity to
    highlight the SQL that could be tuned to use less PGA

DBA_HIST_PGASTAT

Memory issues: ERROR: ORA-04036:

Script: AWR - PGA Allocation and Usage.sql

Counter-Example: "chasing ORA-04036 by raising the limit":

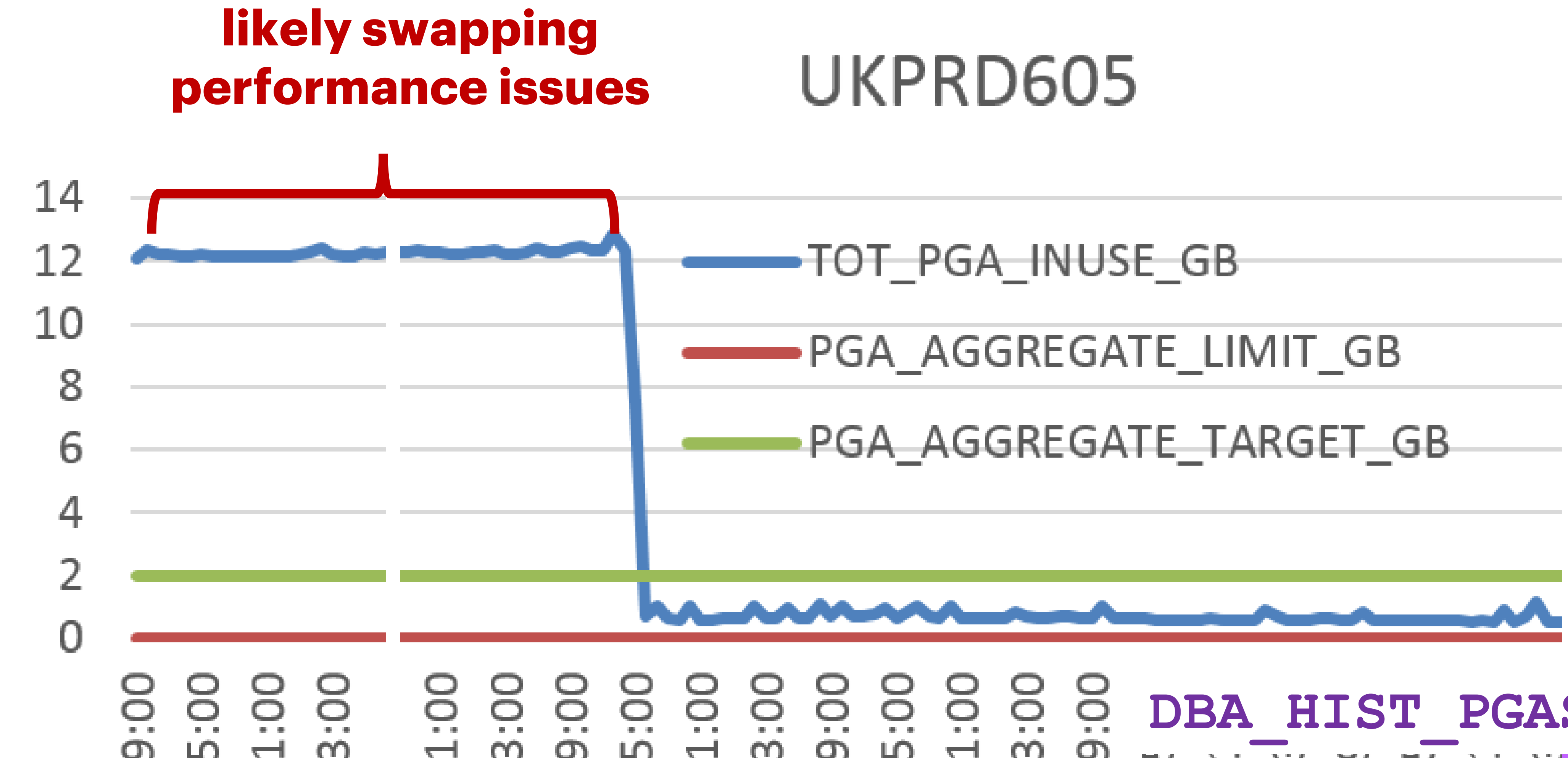


DBA_HIST_PGASTAT

Memory issues: ERROR: ORA-04036:

Script: AWR - PGA Allocation and Usage.sql

Example: "no limit and high PGA in-use"

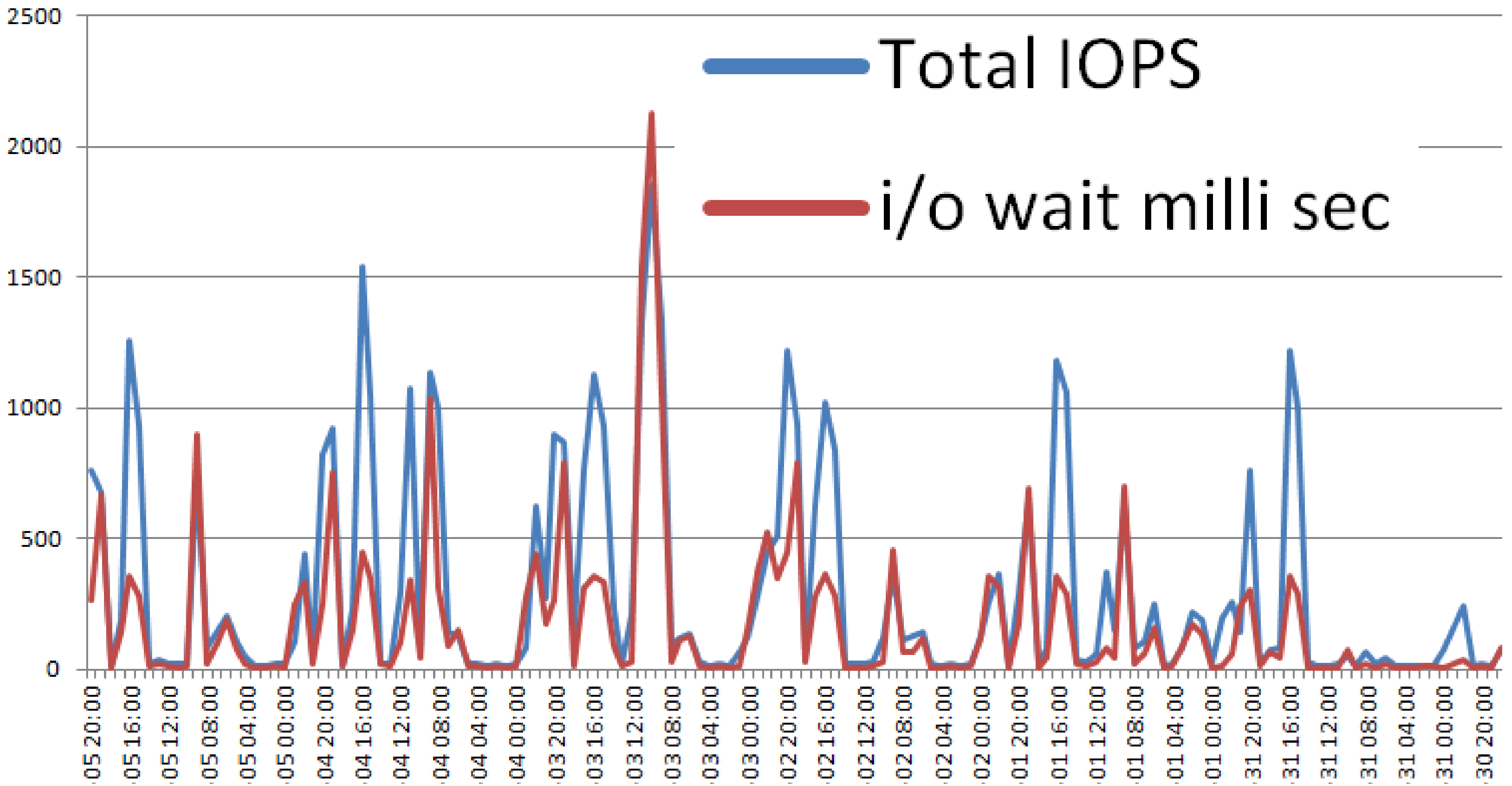


DBA_HIST_PGASTAT

IO Subsystem

<SKIP>

I/O Subsystem issues:



DBA_HIST_SYSSTAT

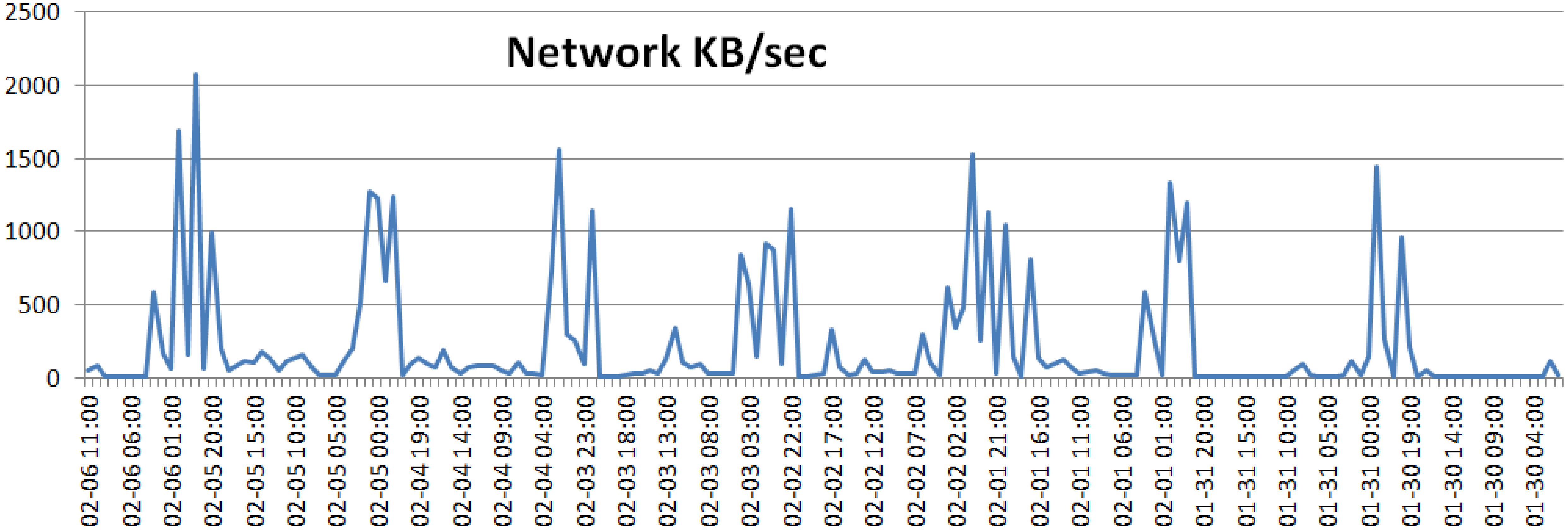
I/O Subsystem issues:

- **Metrics:**
  - User I/O wait time / I/O Rates [IOPS / BPS]
    dba_hist_sysstat dba_hist_sysmetric_summary
- **Health Checks**:
  - a. health check - AWR - Sysmetric_Summary.sql
- **Rules of Thumb:**
  - I/O Latency: average <= ~ 1-10 milliseconds (**10^{-3}**)
    ~ .5 miliseconds for Solid State Storage
    🚫 maximum > 50 milliseconds

Network

<SKIP>

Network issues:



DBA_HIST_SYSMETRIC_SUMMARY

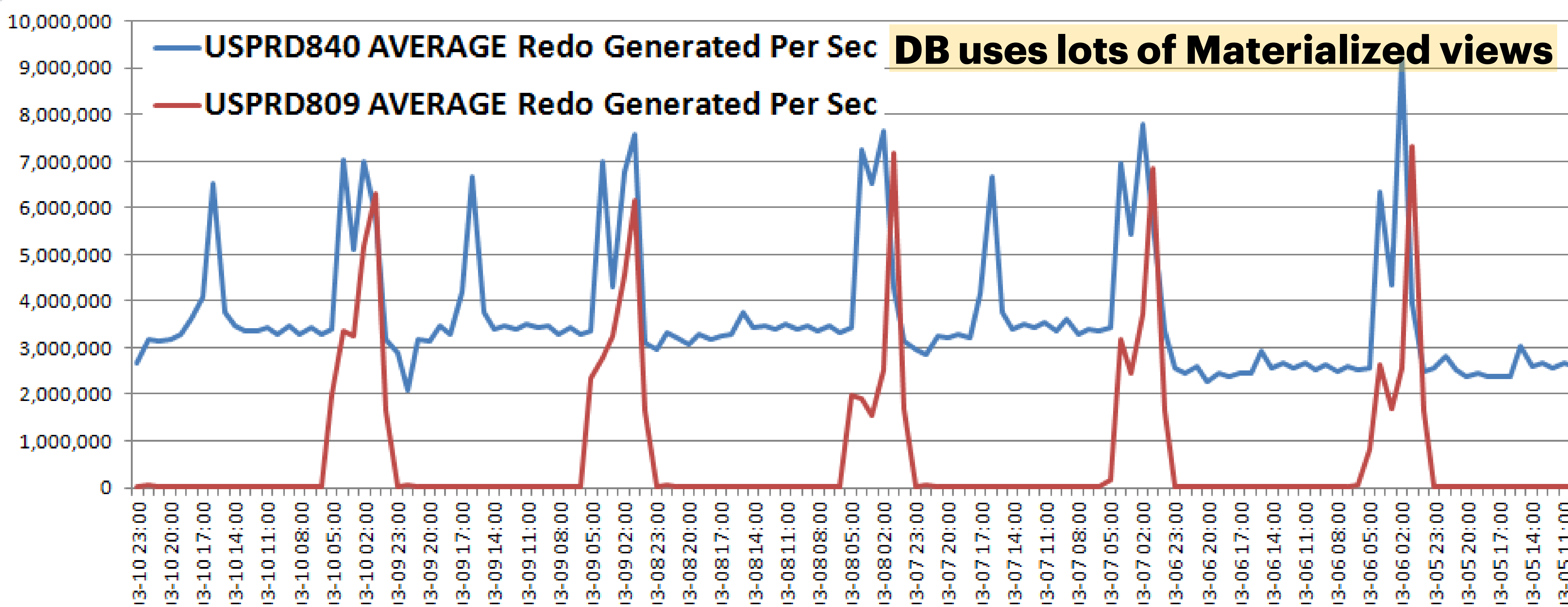
Network issues:

- **Metrics:** dba_hist_sysmetric_summary
  - Network KB/Sec
    ‘Network Traffic Volume Per Sec’ (unit: Bytes Per Second)
- **Also Check:**
  - rows per fetch from SQLSTAT => **<array fetch size>**
  - or rows per execution for inserts => **<array insert>**
  - ASH SQL duration vs SQLSTAT elapsed time
- **Rules of Thumb:**
  - App server on same LAN with DB
  - Tune for throughput

<SKIP>

REDO

REDO issues:



DBA_HIST_SYSMETRIC_SUMMARY

REDO issues:

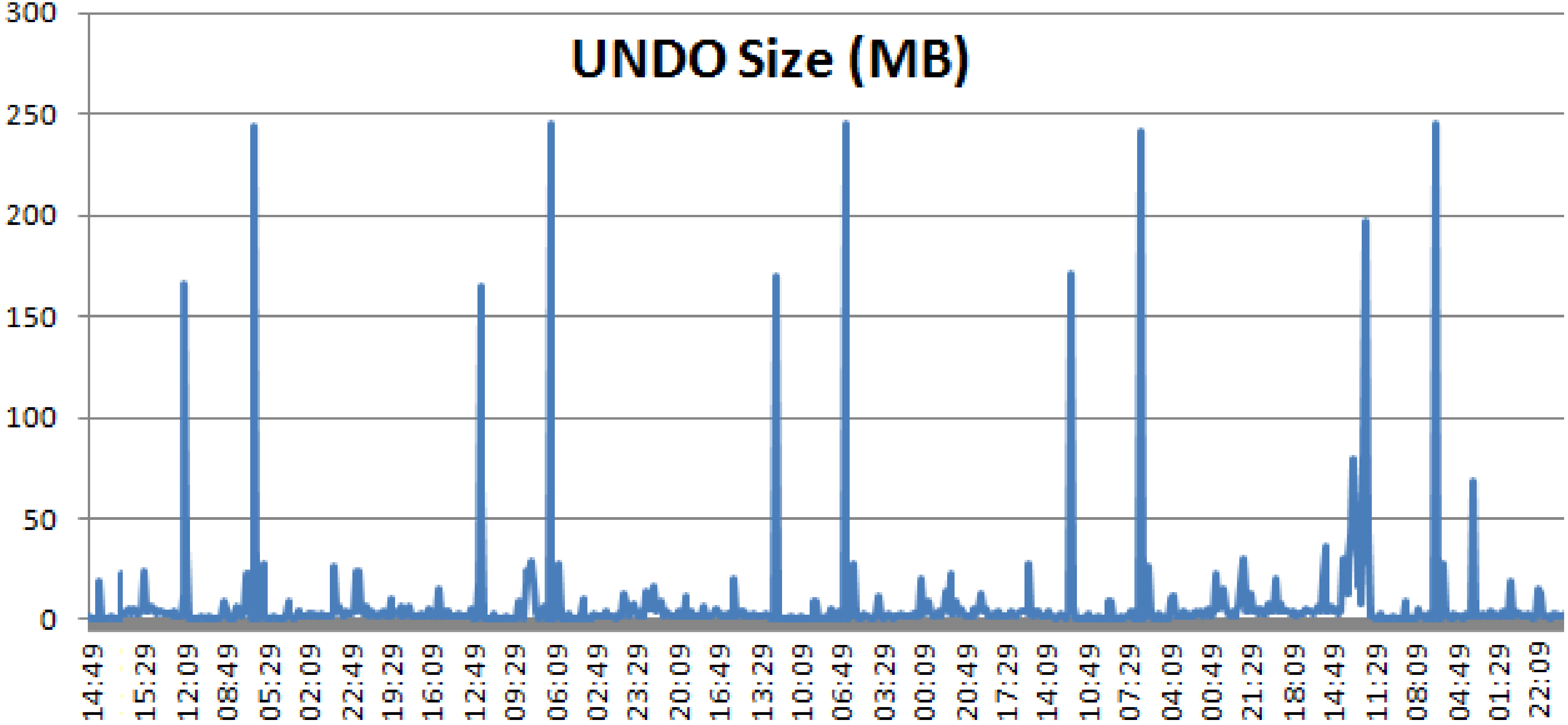
- **Metric:**
  - 'Redo Generated Per Sec' dba_hist_sysmetric_summary
- ***Surrogate* Metric:**
  - DBA_HIST_SEG_STAT and %_OBJ **Not all DB's have this populated**
- **Health Checks:**
  - *a. health check - AWR – REDO.sql*

  what objects and SQL are responsible for the *most block changes*
- **Rules of Thumb:**
  - Mviews are high REDO consumers
  - High Commit and Transaction Rates

<SKIP>

UNDO

UNDO issues:



DBA_HIST_UNDOSTAT

UNDO issues:

- **Metrics:**
  - dba_hist_undostat [*AWR - undostat.sql*]
    includes SQL_ID for top UNDO consuming SQL
- **Health Checks:**
  - *a. health check - undo information.sql*
  - ***dbms_undo_adv.undo_health* *– UNDO Advisor***
- **Rules of Thumb:**
  - Tablespace size and undo retention needs to be big enough to avoid: ORA-01555: snapshot too old
  - Can tune SQL to reduce UNDO requirement

<SKIP>

TEMP

TEMP issues:

- **Metrics:**
  - dba_hist_tbspc_space_usage
    - **periods of high temp consumption**
  - dba_hist_active_sess_history.temp_space_allocated (11g+)
    - **SQL with high TEMP consumption**
- **Health Checks:**
  - a. health check - AWR – TEMP.sql
  - Sort SQL from ASH by TEMP consumption

- **Rules of Thumb:**

Applications can be victim of rather than cause of

ORA-01652: *unable to extend temp segment by <> in tablespace <>*

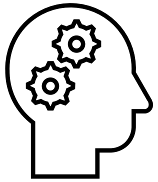
- Can tune SQL to reduce TEMP requirement

<SKIP>

Contention / Concurrency

Contention / Concurrency   Many sources of information

Concurrency problems	DBA_HIST_SQLSTAT.CCWAIT_DELTA → high values
Blocking Sessions	DBA_HIST_ACTIVE_SESS_HISTORY.BLOCKING_%
SQL w/ latch waits	DBA_HIST_ACTIVE_SESS_HISTORY.EVENT enq: TX - index contention, library cache lock
Service Concurrency	DBA_HIST_SERVICE_STAT.stat_name concurrency wait time
Latch Related Waits	DBA_HIST_SYSTEM_EVENT.TIME_WAITED_MICRO cache buffer chains enq: TX - row lock contention latch: redo writing, latch: redo allocation
Latches	DBA_HIST_LATCH.LATCH_NAME cache buffer chains, session allocation redo writing, redo allocation



Anomaly Detection

Anomaly Detection: Flag key influencing metrics

- ❖ **Use Cases:**
  - ❖ understanding why performance degraded or improved
  - ❖ comparing workloads / application behavior analysis
  - ❖ resource contention - root cause analysis

- ❖ **Approach:**
  1. **Unpivot** metrics stored in multi columns to key-value pairs
  2. **Feature Engineering** (i.e. collecting metrics/creating new ones)
  3. **Flag** metrics based on their values exceeding a cutoff
     - ❖ Normal Ranges or Percentile **Feature Selection** - data science term
  4. **Prioritize** (subset/sort) metrics
     - ❖ amount above the cutoff / # intervals where metric > cutoff

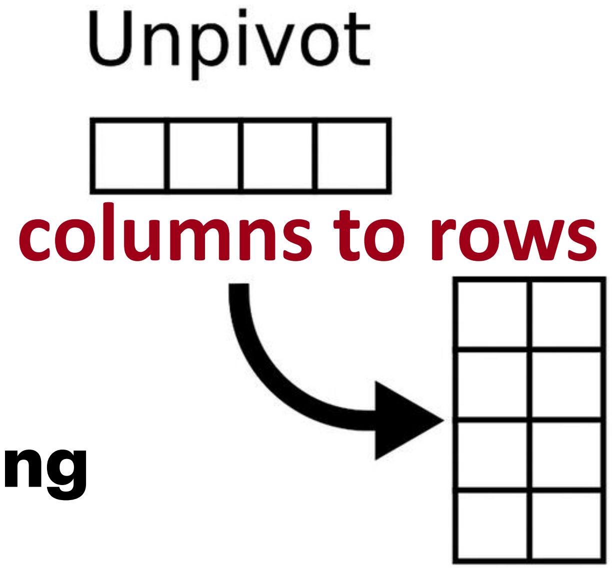
Unpivoting Metrics

Unpivoting Metrics - Feature Engineering:

Dynamic Oracle Performance Analytics

- **Creating metrics (a.k.a features; a.k.a. variables) by:**
  "unpivoting" traditional short/wide structured data (MCT)
  ➔ tall/skinny structure: key-value pair (KVP)
  Columns / variables become KVP rows
  - **Trend: analytics moving away from traditional ER structures to KVP structures (e.g. NoSQL (e.g. MongoDB, Redis); JSON ; XML ; . . .)**

- **"Unpivoted" KVP metrics from several sources are UNIONed together for analysis**
  ➔ massively expand the set of metrics



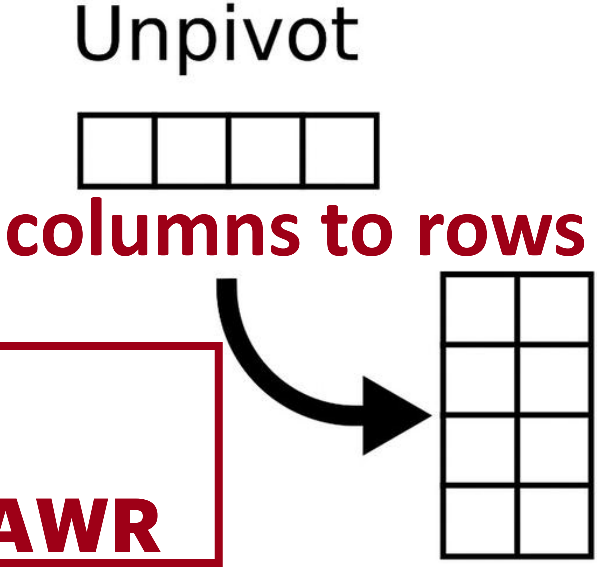
- **Without unpivoting/key-value-pair structures: each measurement would require specific programming**

Unpivoting Metrics - Feature Engineering:

Dynamic Oracle Performance Analytics

- Creating metrics (a.k.a features; a.k.a. variables) by:
  "unpivoting" traditional short/wide structured data (MCT)
  ➔ tall/skinny structure: key-value pair (KVP)
  Columns / variables become KVP rows
  - Trend: analytics moving away from traditional ER structures to KVP structures (e.g. NoSQL (e.g. MongoDB, Redis); JSON ; XML ; . . .)

- "Unpivoted" KVP metrics from several sources are UNIONed together for analysis
  ➔ massively expand the set of metrics



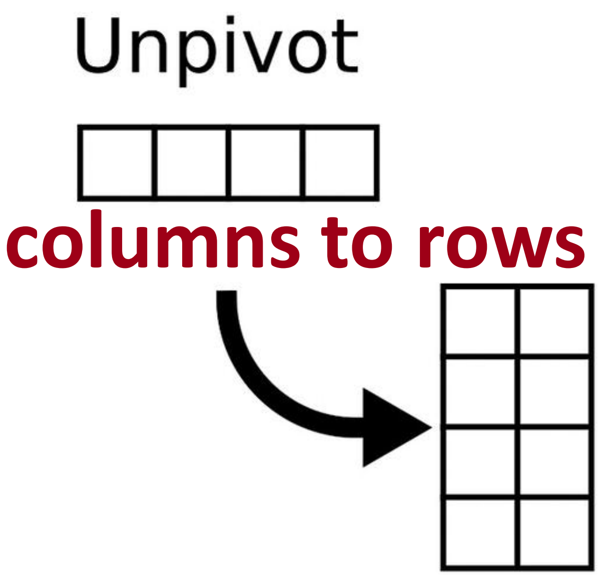
Few Metrics, hand curated [small-model approach]

➔ Not Scalable to thousands of metrics available in AWR

Unpivot – Example (create "key-value pairs")

dba_hist_sqlstat - multi-column table

```
select snap_id, sql_id
, FETCHES_DELTA, SORTS_DELTA, EXECUTIONS_DELTA
, PARSE_CALLS_DELTA, DISK_READS_DELTA
, BUFFER_GETS_DELTA, ROWS_PROCESSED_DELTA
, DIRECT_WRITES_DELTA,
PHYSICAL_READ_REQUESTS_DELTA
, PHYSICAL_WRITE_REQUESTS_DELTA
from dba_hist_sqlstat
where sql_id = nvl(:sql_id_X, sql_id)
order by snap_id
;
```



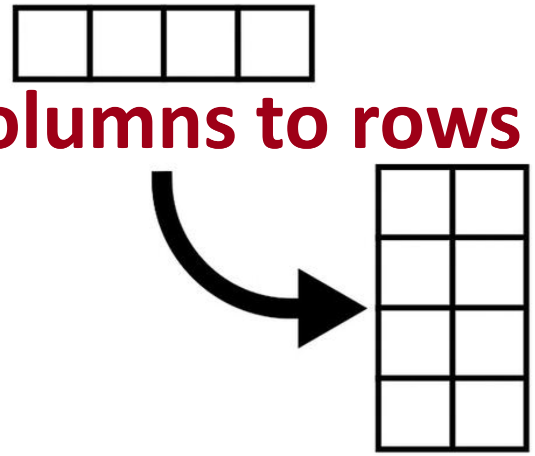
Unpivot – Example (create "key-value pairs")

dba_hist_sqlstat - multi-column table

SNAP_ID	SQL_ID	FETCHES_DELTA	SORTS_DELTA	EXECUTIONS_DELTA	PARSE_CALLS_DELTA	DISK_READS_DELTA	BUFFER_GETS_DELTA	R
28240	60nmtddad7mhm	9141	9133	9141	9141	27362	123010451	
28258	60nmtddad7mhm	4394	4395	4395	4395	16109	1033828229	
28259	60nmtddad7mhm	0	0	0	0	5619	2945502590	
28260	60nmtddad7mhm	1	1	1	1	1002	0	
28261	60nmtddad7mhm	1	1	1	1	1836	0	
28262	60nmtddad7mhm	0	0	0	0	5258	2908006919	
28263	60nmtddad7mhm	1	1	1	1	3614	0	
28264	60nmtddad7mhm	1	1	1	1	3742	0	
28265	60nmtd							
28266	60nmtd							
28267	60nmtd							
28283	60nmtd							
28356	60nmtd							

SNAP_ID	SQL_ID	METRIC_NAME	DELTA_VALUE
28240	60nmtddad7mhm	BUFFER_GETS_DELTA	123,010,451
28240	60nmtddad7mhm	DIRECT_WRITES_DELTA	0
28240	60nmtddad7mhm	DISK_READS_DELTA	27,362
28240	60nmtddad7mhm	EXECUTIONS_DELTA	9,141
28240	60nmtddad7mhm	FETCHES_DELTA	9,141
28240	60nmtddad7mhm	PARSE_CALLS_DELTA	9,141
28240	60nmtddad7mhm	PHYSICAL_READ_REQUESTS_DELTA	27,362
28240	60nmtddad7mhm	PHYSICAL_WRITE_REQUESTS_DELTA	0
28240	60nmtddad7mhm	ROWS_PROCESSED_DELTA	9,141
28240	60nmtddad7mhm	SORTS_DELTA	9,133

Unpivot

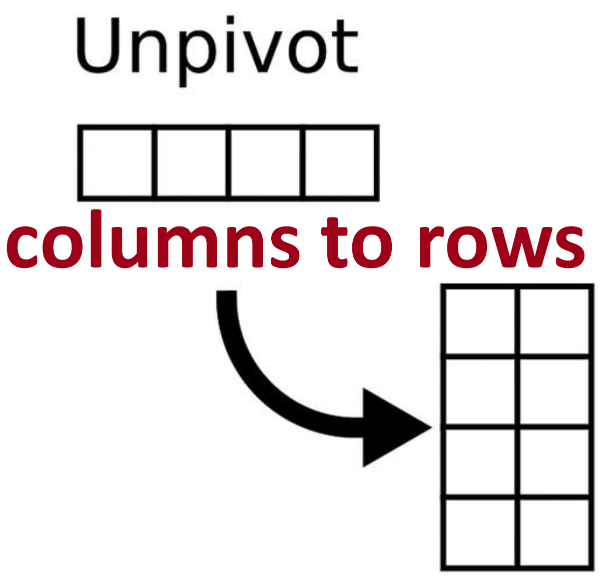


Unpivot – Example (create "key-value pairs")

the Unpivot SQL:

dba_hist_sqlstat - unpivot to KVP

```
select snap_id, sql_id, metric_name, delta_value
from dba_hist_sqlstat sqlstat
unpivot include nulls
( delta_value for metric_name in
  (FETCHES_DELTA,SORTS_DELTA,EXECUTIONS_DELTA
  ,PARSE_CALLS_DELTA,DISK_READS_DELTA
  ,BUFFER_GETS_DELTA,ROWS_PROCESSED_DELTA
  ,DIRECT_WRITES_DELTA
  ,PHYSICAL_READ_REQUESTS_DELTA
  ,PHYSICAL_WRITE_REQUESTS_DELTA)
)
where sql_id = nvl(:sql_id, sql_id)
order by snap_id, metric_name;
```



Unpivot – Example (create "key-value pairs")

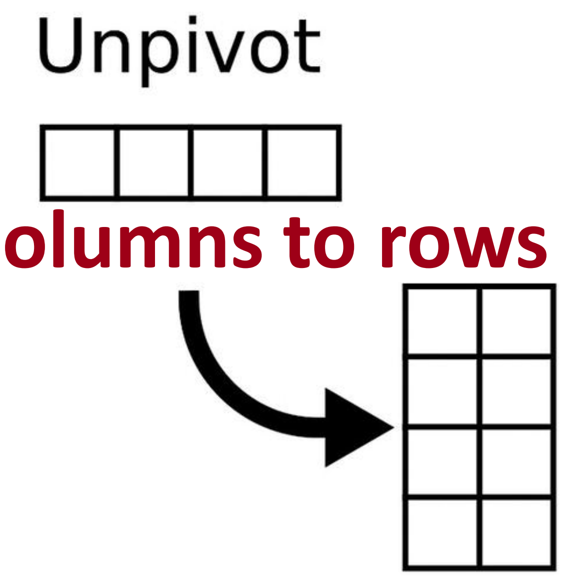
the data:

dba_hist_sqlstat - unpivot to KVP

SNAP_ID	SQL_ID	METRIC_NAME	DELTA_VALUE
28240	60nmtddad7mhm	BUFFER_GETS_DELTA	123,010,451
28240	60nmtddad7mhm	DIRECT_WRITES_DELTA	0
28240	60nmtddad7mhm	DISK_READS_DELTA	27,362
28240	60nmtddad7mhm	EXECUTIONS_DELTA	9,141
28240	60nmtddad7mhm	FETCHES_DELTA	9,141
28240	60nmtddad7mhm	PARSE_CALLS_DELTA	9,141
28240	60nmtddad7mhm	PHYSICAL_READ_REQUESTS_DELTA	27,362
28240	60nmtddad7mhm	PHYSICAL_WRITE_REQUESTS_DELTA	0
28240	60nmtddad7mhm	ROWS_PROCESSED_DELTA	9,141
28240	60nmtddad7mhm	SORTS_DELTA	9,133
28258	60nmtddad7mhm	BUFFER_GETS_DELTA	1,033,828,229
28258	60nmtddad7mhm	DIRECT_WRITES_DELTA	0
28258	60nmtddad7mhm	DISK_READS_DELTA	16,109
28258	60nmtddad7mhm	EXECUTIONS_DELTA	4,395
28258	60nmtddad7mhm	FETCHES_DELTA	4,394

Unpivot into KVP format:

→ Easy to perform same Analysis to all "columns"



columns-by-column evaluation / calculation requires a lot more effort / programming

SQL-Level Metric Anomaly Detection

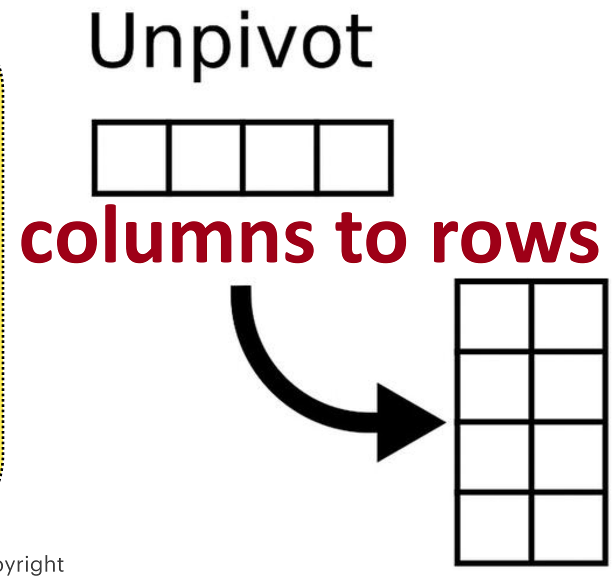
KVP View of SQLSTAT Metrics

~ 120 SQLSTAT metrics every snapshot interval

STAT SOURCE	METRIC_NAME
dba_hist_sqlstat	SUM: apwait (seconds)
dba_hist_sqlstat	SUM: apwait_per_execution (seconds)
dba_hist_sqlstat	SUM: buffer_gets (count)
dba_hist_sqlstat	SUM: buffer_gets_per_execution (count)
dba_hist_sqlstat	SUM: ccwait (seconds)
dba_hist_sqlstat	SUM: ccwait_per_execution (seconds)
dba_hist_sqlstat	SUM: cell_uncompressed (meg)
dba_hist_sqlstat	SUM: clwait (seconds)
dba_hist_sqlstat	SUM: clwait_per_execution (seconds)
dba_hist_sqlstat	SUM: cpu_time (seconds)
dba_hist_sqlstat	SUM: cpu_time_per_execution (seconds)
dba_hist_sqlstat	SUM: direct_writes (count)

Feature Engineering includes calculations

- **milliseconds to seconds**
- **bytes to megabytes**
- **per execution calculations**
- **sums and averages**



Unpivoting ASH Metrics

SQL-Level Metric Anomaly Detection

MCT View of Active Session History Metrics

```
with ash as /* get all the ASH data of interest ; unpivot later */
(
select
SNAP_ID
,SAMPLE_ID
,SAMPLE_TIME
,SQL_ID
,IS_SQLID_CURRENT
,decode(QC_INSTANCE_ID || QC_SESSION_ID || QC_SESSION_SERIAL#, null, 'N', 'Y')
  as IS_SQL_EXECUTING_IN_PARALLEL
, EVENT || decode(BLOCKING_SESSION||BLOCKING_SESSION_SERIAL#,null,null, ' [blocked event]')
  as EVENT
,decode(SESSION_STATE, 'WAITING', 'Y', 'N') as IS_SESSION_WAITING
,decode(SESSION_STATE, 'ON CPU', 'Y', 'N') as IS_SESSION_ON_CPU
,IN_CONNECTION_MGMT,IN_PARSE,IN_HARD_PARSE,IN_SQL_EXECUTION,IN_PLSQL_EXECUTION
,IN_PLSQL_RPC,IN_PLSQL_COMPILATION,IN_JAVA_EXECUTION,IN_BIND
,IN_CURSOR_CLOSE,IN_SEQUENCE_LOAD
,CAPTURE_OVERHEAD,REPLAY_OVERHEAD
,IS_CAPTURED,IS_REPLAYED
,TM_DELTA_TIME,TM_DELTA_CPU_TIME
,TM_DELTA_DB_TIME
,TM_DELTA_DB_TIME - TM_DELTA_CPU_TIME as TM_DELTA_IDLE_TIME
,DELTA_TIME,DELTA_READ_IO_REQUESTS,DELTA_WRITE_IO_REQUESTS,DELTA_READ_IO_BYTES
,DELTA_WRITE_IO_BYTES,DELTA_INTERCONNECT_IO_BYTES
,PGA_ALLOCATED
,TEMP_SPACE_ALLOCATED
from dba_hist_active_sess_history
where 1=1 /* in practice, subset on intervals, sql_id's, and sessions */
  and sql_id = '988a1hx9zc5rr'
)
```

```
select *
from ash
order by sample_id fetch
first 10 rows only
;
```

Leverage the fact that one row in ASH = ~ 10 seconds

e.g. # of rows * 10 where IN_SQL_EXECUTION = 'Y' ~= the number of seconds so non-numeric metrics can be engineered as numeric

SQL-Level Metric Anomaly Detection

KVP View of Active Session History Metrics

1	STAT SOURCE	METRIC_NAME
2	dba_hist_active_sess_history	SUM: capture_overhead (seconds)
3	dba_hist_active_sess_history	SUM: delta_interconnect_io_bytes (seconds)
4	dba_hist_active_sess_history	SUM: delta_read_io_bytes (seconds)
5	dba_hist_active_sess_history	SUM: delta_read_io_requests (seconds)
6	dba_hist_active_sess_history	SUM: delta_time (seconds)
7	dba_hist_active_sess_history	SUM: in_bind (seconds)
8	dba_hist_active_sess_history	SUM: in_connection_mgmt (seconds)
9	dba_hist_active_sess_history	SUM: in_cursor_close (seconds)
10	dba_hist_active_sess_history	SUM: in_hard_parse (seconds)
11	dba_hist_active_sess_history	SUM: in_java_execution (seconds)
12	dba_hist_active_sess_history	SUM: in_parse (seconds)
13	dba_hist_active_sess_history	SUM: in_plsql_compilation (seconds)
14	dba_hist_active_sess_history	SUM: in_plsql_execution (seconds)

SQL-Level Metric Anomaly Detection

KVP View of Active Session History Metrics

~ 300 ASH metrics every snapshot interval

	STAT SOURCE	METRIC_NAME
1		
2	dba_hist_active_sess_history	SUM: capture_overhead (seconds)
3	dba_hist_active_sess_history	SUM: delta_interconnect_io_bytes (seconds)
4	dba_hist_active_sess_history	SUM: delta_read_io_bytes (seconds)
5	dba_hist_active_sess_history	SUM: delta_read_io_requests (seconds)
6	dba_hist_active_sess_history	SUM: delta_time (seconds)
7	dba_hist_active_sess_history	SUM: in_bind (seconds)
8	dba_hist_active_sess_history	SUM: in_connection_mgmt (seconds)
9	dba_hist_active_sess_history	SUM: in_cursor_close (seconds)
10	dba_hist_active_sess_history	SUM: in_hard_parse (seconds)
11	dba_hist_active_sess_history	SUM: in_java_execution (seconds)
12	dba_hist_active_sess_history	SUM: in_parse (seconds)
13	dba_hist_active_sess_history	SUM: in_plsql_compilation (seconds)
14	dba_hist_active_sess_history	SUM: in_plsql_execution (seconds)

Feature Engineering includes time spent on:

- **Wait events [flagged blockers]**
- **Plan steps**
- **Objects being processed**
- **sums and averages**

Unpivoting SQL-Level Metrics Summary

Session/SQL-Level Metrics

Key-Value Pair Normalization – Feature Engineering

~ 300 DBA_HIST_ACTIVE_SESS_HISTORY

All the columns in ASH plus Events, objects, and plan operations are named metrics

~ 120 DBA_HIST_SYS_SQLSTAT

All the columns in SQLSTAT plus calculated metrics (e.g. per execution calculations)

Unpivoting Overall Metrics Summary

System-Wide metric analysis included with SQL-Level metrics

Same Key-Value Pair Normalization

161	DBA_HIST_SYSMETRIC_SUMMARY		~sysstat w/ Per Sec/Txn
31	DBA_HIST_SYS_TIME_MODEL		CPU / Elapsed Time
2036	DBA_HIST_SYSSTAT		Oracle Resources
~261	DBA_HIST_SYSTEM_EVENT	new	Wait events also in ASH
~300	DBA_HIST_ACTIVE_SESS_HISTORY		
~120	DBA_HIST_SYS_SQLSTAT		

Nearly 3000 metrics every snapshot

Example Anomaly Detection Run

Case Study: Regressed SQL

ash-sqlstat - flag unpivoted metrics.sql

✓**Select good and bad intervals to compare**

INPUT_PARAMETER_NAME	INPUT PARAMETER VALUE	
:sql_id_string_comma_sep_lst	4w1mftxfptyyk	**id1,id2,id3,...**
:sessions_for_interval		
:ssessions_for_normal		
:bad_run_st_MM_DD_YYYY_HH24_MI	08_18_2022_07_00	**Experimental**
:bad_run_end_MM_DD_YYYY_HH24_MI	08_18_2022_07_20	
:good_run_st_MM_DD_YYYY_HH24_MI	08_12_2022_07_00	**Baseline**
:good_run_end_MM_DD_YYYY_HH24_MI	08_12_2022_07_20	
:metric_name_ash		
:flag_percentile	0	
:stat_source		

Case Study: Regressed SQL

ash-sqlstat - flag unpivoted metrics.sql

✓ **Determine which metrics likely to most influence performance**

STAT SOURCE	SQL_ID	METRIC_NAME	NORMAL VALUE	BAD RUN VALUE	DELTA TO GOOD RATIO	STDDEV VALUE	MIN VALUE	AVG VALUE	MAX VALUE
active_sess_history	4w1mftxfptyyk	**SUM: wait class: Concurrency (seconds)**	10	1,870	18,600	1,821	10	896	6,250
active_sess_history	4w1mftxfptyyk	**SUM: wait event: buffer busy waits [blocked event] ; wait class: Concurrency (seconds)**	10	**1,560**	15,500	630	10	380	1,550
active_sess_history	4w1mftxfptyyk	**SUM: wait class: Other (seconds)**	10	1,530	15,200	563	10	256	1,530
active_sess_history	4w1mftxfptyyk	**SUM: wait event: enq: US - contention [blocked event] ; wait class: Other (seconds)**		**1,280**		0	1,280	1,280	1,280
active_sess_history	4w1mftxfptyyk	**SUM: wait event: enq: US - contention ; wait class: Other (seconds)**		240		0	240	240	240
active_sess_history	4w1mftxfptyyk	SUM: is_blocked_session (seconds)	20	**2,840**	14,100	579	0	79	6,210
active_sess_history	4w1mftxfptyyk	SUM: is_session_blocked (seconds)	20	2,840	14,100	579	0	79	6,210
dba_hist_sqlstat	4w1mftxfptyyk	**SUM: elapsed_time_per_execution (seconds)**	0.1880	0.4920	162	0	0	0	0
active_sess_history	4w1mftxfptyyk	SUM: sql_plan_line_id: hash: 3983693627 | line: | operation: DELETE STATEMENT | options: (seconds)	100	3,510	3,410	488	10	115	3,410
active_sess_history	4w1mftxfptyyk	SUM: is_session_waiting (seconds)	430	3,870	800	635	0	223	6,250
dba_hist_sqlstat	4w1mftxfptyyk	SUM: elapsed_time (seconds)	1,228	4,239	245	2,130	1,228	2,733	4,239
active_sess_history	4w1mftxfptyyk	SUM: sql_plan_operation: DML Delete	1,690	5,400	220	495	10	374	4,110

Active Session History

Hot Block Contention

UNDO Space Contention

SQL Stat

Case Study: Regressed SQL

ash-sqlstat - flag unpivoted metrics.sql

✓ **Determine which metrics likely to most influence performance**

STAT SOURCE	SQL_ID	METRIC_NAME	NORMAL VALUE	BAD RUN VALUE	DELTA TO GOOD RATIO	STDDEV VALUE	MIN VALUE	AVG VALUE	MAX VALUE
active_sess_history	4w1mftxfptyyk	SUM: wait class: Concurrency (seconds)	10	1,870	18,600	1,821	10	896	6,250
active_sess_history	4w1mftxfptyyk	SUM: wait event: buffer busy waits [blocked event] ; wait class: Concurrency (seconds)	10	**1,560**	15,500	630	10	380	1,550
active_sess_history	4w1mftxfptyyk	**SUM: wait class: Other (seconds)**	10	1,530	15,200	563	10	256	1,530
active_sess_history	4w1mftxfptyyk	**SUM: wait event: enq: US - contention [blocked event] ; wait class: Other (seconds)**		**1,280**		0	1,280	1,280	1,280
active_sess_history	4w1mftxfptyyk	**SUM: wait event: enq: US - contention ; wait class: Other (seconds)**		240		0	240	240	240
active_sess_history	4w1mftxfptyyk	SUM: is_blocked_session (seconds)	20	**2,840**	14,100	579	0	79	6,210
active_sess_history	4w1mftxfptyyk	SUM: is_session_blocked (seconds)	20	2,840	14,100	579	0	79	6,210
dba_hist_sqlstat	4w1mftxfptyyk	**SUM: elapsed_time_per_execution (seconds)**	0.1880	0.4920	162	0	0	0	0
active_sess_history	4w1mftxfptyyk	SUM: sql_plan_line_id: hash: 3983693627 | line: | operation: DELETE STATEMENT | options: (seconds)	100	3,510	3,410	488	10	115	3,410
active_sess_history	4w1mftxfpt[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	635	0	223	6,250
dba_hist_sqlstat	4w1mftxfpt[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible],130	1,228	2,733	4,239
active_sess_history	4w1mftxfptyyk	SUM: sql_plan_operation: DML Delete	1,690	5,400	220	495	10	374	4,110

Active Session History

Hot Block Contention

UNDO Space Contention

SQL Stat

Ordered by how big a problem this metric is.

Case Study: Regressed SQL

ash-sqlstat - flag unpivoted metrics.sql

✓**System-Wide metrics inform Root Cause Analysis**

STAT SOURCE	SQL_ID	METRIC_NAME	NORMAL VALUE	BAD RUN VALUE	DELTA TO GOOD RATIO	STDDEV VALUE	MIN VALUE	AVG VALUE	M
_sysmetric_summary	overall	AVG: Total Sorts Per User Call	0	40,239	13,502,881	11,447	0	530	
_sysmetric_summary	overall	AVG: Executions Per User Call	0	45,815	13,052,562	41,340	0	2,085	
Sysmetric Summary		AVG: Total Table Scans Per User Call	0	1,519	12,657,575	4,145	0	114	
_sysmetric_summary	overall	AVG: DB Block Gets Per User Call	11	1,417,099	12,357,963	382,247	0	19,585	
_sysmetric_summary	overall	AVG: DB Block Changes Per User Call	7	862,830	12,140,473	240,859	0	12,559	
_sysmetric_summary	overall	AVG: Logical Reads Per User Call	57	6,037,852	10,556,693	2,116,268	6	120,724	
_sysmetric_summary	overall	AVG: CR Undo Records Applied Per Txn	69	12,236	17,632	78,656	0	3,860	
_sysmetric_summary	overall	AVG: Physical Writes Per Txn	102	15,506	15,174	32,687	0	6,903	
_sysmetric_summary	overall	AVG: Consistent Read Changes Per Txn	103	14,822	14,236	91,228	0	21,561	
_sysmetric_summary	overall	AVG: Logons Per Txn							
_sysmetric_summary	overall	AVG: Total Table Scans Per Txn							
_sysmetric_summary	overall	AVG: Enqueue Timeouts Per Txn							
_sysmetric_summary	overall	AVG: Long Table Scans Per Txn							
_sysmetric_summary	overall	AVG: CR Blocks Created Per Txn							
_sysmetric_summary	overall	AVG: Enqueue Requests Per Txn	19.3940	783.1830	3,938	16,599	8	10,695	
_sysmetric_summary	overall	AVG: Redo Writes Per Txn	1.1410	31.8180	2,689	1,751	0	576	
hist_sys_time_model	overall	AVG: hard parse (bind mismatch) elapsed	16,036	434,591	2,610	8,861,275	0	2,048,105	1
Sys Time Model	verall	AVG: Total Parse Count Per Txn	12	215	1,726	5,604	2	3,189	
_sysmetric_summary	overall	AVG: Physical Reads Per Txn	36	540	1,417	275,778	0	36,818	

Note: high system-wide metrics related to IO, concurrency, and UNDO

Case Study: Regressed SQL

ash-sqlstat - flag unpivoted metrics.sql

Root Cause:

- ❑ buffer busy waits [blocked event] ; wait class: Concurrency
  - ❖ Hot blocks contention → read consistency and data-related bottleneck
- ❑ enq: US - contention [blocked event] ; wait class: Other
  - ❖ UNDO Segment contention → application rollback-related bottleneck
- ❑ high system-level metrics related to concurrency
  - ❖ Confirm that the bottleneck is mostly data read and rollback-related
- ❑ high system-level metrics related to IO and workload
  - ❖ Contributes to IO delays

Anomaly Detection Wrapup

Anomaly Detection: Flag key influencing metrics

- Use Cases:
  - understanding why performance degraded or improved
  - comparing workloads / application behavior analysis
  - resource contention - root cause analysis

- Approach:

1. Unpivot metrics stored in multi columns to key-value pairs
2. Feature Engineering (i.e. collecting metrics/creating new ones)
3. Flag metrics based on their values exceeding a cutoff
   - Normal Ranges or Percentile Feature Selection - data science term
4. Prioritize (subset/sort) metrics
   - amount above the cutoff / # intervals where metric > cutoff

Anomaly Detection: Flag key influencing metrics

- ❖Observations of system behavior (different per workload) :
  - ❖high ccwaits_delta in SQLSTAT → high contention waits in ASH
    - ❖Check total contention waits in dba_hist_system_event
    - ❖High SQL/ASH contention → check for blocking locks
  - ❖Sorts in SQLSTAT → high TEMP in ASH and sys metrics
  - ❖App waits on log buffer space → log file sync; redo % sys metrics
  - ❖high Average Active Sessions → CPU vs Wait
  - ❖High redo metrics → high network metrics
  - ❖Session spikes (logon storms) → high CPU
  - ❖High Asynchronous Single Block Read Latency associated with:
    - ❖High IO demand
    - ❖Potential IO Sub-System issues if not high IO demand

specific to SQL?

specific to App?

Sysmetric Ratio's

What's the root cause?

What disks?

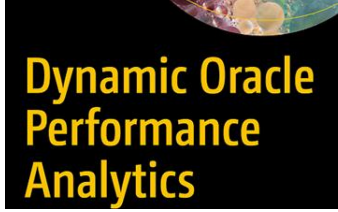
Metric Correlation

<SKIP>

`corr` - Correlation

Understanding the data value relationship between variables

- `corr`
  aggregate function produces values in the range of -1 to 1
  - 1 = perfect correlation (i.e. values go up and down together)
  - 0 = no correlation
  - -1 = inversely correlated
- https://oracle-base.com/articles/misc/corr-analytic-function



corr - Correlation

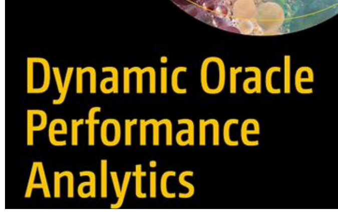
Understanding the data value relationship between variables

```
with metric_set_1 as (select begin_time, metric_name, value
from v$sysmetric_history
where upper(metric_name) like upper(nvl(:metric_name1, metric_name)))
, metric_set_2 as (select begin_time, metric_name, value
from v$sysmetric_history
where upper(metric_name) like upper(nvl(:metric_name2, metric_name)))
/* main select CORR function */
select s1.metric_name "Metric Name 1", s2.metric_name "Metric Name 2"
, round(CORR(s1.value, s2.value), 7) AS "Pearson's Correlation"
from metric_set_1 s1, metric_set_2 s2
where s1.begin_time = s2.begin_time and s1.METRIC_NAME <> s2.metric_name
group by s1.METRIC_NAME, s2.metric_name
having CORR(s1.value, s2.value) is not null
   and (CORR(s1.value, s2.value) >= .75
    or  CORR(s1.value, s2.value) <= -.75)
order by 3 desc ;
```

Query: Cross Product of all metrics against all metrics

`corr` - Correlation

Understanding the data value relationship between variables



Metric Name 1	Metric Name 2	Pearson's Correlation
Average Active Sessions	Database Time Per Sec	1.00
Open Cursors Per Txn	Executions Per Txn	0.99
Total Parse Count Per Sec	Executions Per Sec	0.99
CPU Usage Per Txn	Logical Reads Per Txn	0.97
CPU Usage Per Txn	Response Time Per Txn	0.97
Logical Reads Per Txn	Redo Writes Per Txn	0.95
Redo Writes Per Txn	Logical Reads Per Txn	0.95
Enqueue Requests Per Txn	Response Time Per Txn	0.95
Response Time Per Txn	Enqueue Requests Per Txn	0.95
Logical Reads Per Txn	DB Block Gets Per Txn	0.95
DB Block Gets Per Txn	Logical Reads Per Txn	0.95
DB Block Changes Per Txn	Logical Reads Per Txn	0.95
Soft Parse Ratio	Library Cache Miss Ratio	-0.92
Library Cache Miss Ratio	Soft Parse Ratio	-0.92
Database CPU Time Ratio	Database Wait Time Ratio	-0.99
Disk Sort Per Sec	Memory Sorts Ratio	-1.00

AAS is calculated from DB Time

increase CPU → worse response

Compliment metrics (add to 100)

Expect to see these diverge

Metric Correlation: Discover Surrogate Metrics

- **Use Case: Discover relationships across different metrics.**

- **What database behavior is related to Data Guard synchronous replication (Primary to Secondary)?**
  - **DB Block Changes Per Sec highly correlated with:**
    - **Redo Generated Per Sec**
    - **Background CPU Usage Per Sec**
    - **Background Checkpoints Per Sec**
    - **Leaf Node Splits Per Sec**
    - **Branch Node Splits Per Sec**

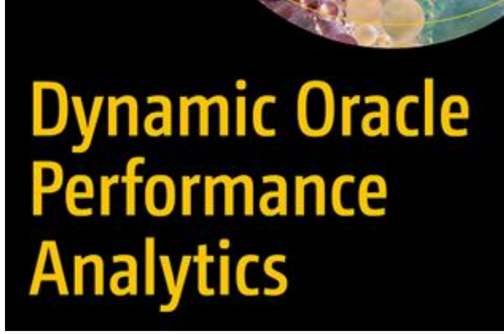
Good candidates for instrumentation

Index maintenance significantly contributing to block changes and REDO generation

References

<SKIP>

References



- ❖ **Analytics Using Feature Selection for Anomaly Detection**
  - ❖ **Medium Blog post: Summary of the DOPA process**

    https://medium.com/gsktech/analytics-using-feature-selection-for-anomaly-detection-4c1474501157
- ❖ **Book: Dynamic Oracle Performance Analytics**
  Using Normalized Metrics to Improve Database Speed

  http://www.apress.com/9781484241363

  https://www.amazon.com/Dynamic-Oracle-Performance-Analytics-Normalized/dp/1484241363/ref=cm_cr_arp_d_product_top?ie=UTF8
- ❖ **Anomaly Detection SQL code (from Book) is available on GitHub**

  https://github.com/Apress/dynamic-oracle-perf-analytics
- ❖ **RMOUG Article: Recognizing and Overcoming Limitations of Standard Performance Tuning Tools**

  https://rmoug.org/resources/Documents/2016-summer16web160803.pdf

Helpful Links

- TAMING THE AWR TSUNAMI – AN INNOVATIVE APPROACH
- https://www.rogercornejo.com/oracle-blog/2022/5/2/taming-the-awr-tsunami-an-innovative-approach

- THE BACK STORY – DYNAMIC ORACLE PERFORMANCE ANALYTICS
- https://www.rogercornejo.com/oracle-blog/2022/5/10/the-back-story-dynamic-oracle-performance-analytics

- **TUNING SQL USING FEATURE ENGINEERING AND FEATURE SELECTION - PART I**
- https://www.rogercornejo.com/oracle-blog/2022/4/18/tuning-sql-using-feature-engineering-and-feature-selection-part-i

- Medium Article on Analytics Using Feature Selection for Anomaly Detection
- https://medium.com/gsktech/analytics-using-feature-selection-for-anomaly-detection-4c1474501157

- Book: *Dynamic Oracle Performance Analytics, Using Normalized Metrics to Improve Database Speed*
- http://www.apress.com/9781484241363

- Oracle Blog – RogerCornejo.com
- https://www.rogercornejo.com/oracle-blog

- War Stories from the DBA Trenches [Blog] – RogerCornejo.com

https://www.rogercornejo.com/war-stories

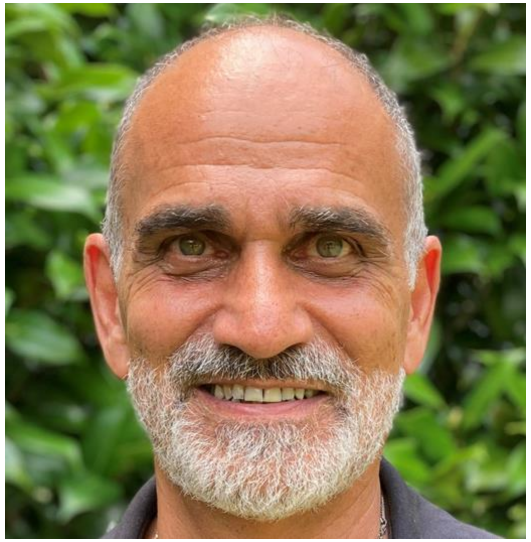
AWR Deep Dive: Truths, Troubles, and Tuning Techniques from the Field

EAST Coast Oracle Conference

November, 2025

Questions?

Roger Cornejo
Technology Innovation Principal Director, GenAI
Accenture Enkitec Group
Roger.Cornejo@Accenture.com
@OracleDBTuning
RogerCornejo.com



Oracle ACE Director

ENKITEC DATABASE ENGINEERING

Architect, Engineer, Operate, Govern Hybrid, Multi-Cloud, Edge Distributed Physical & Virtual infrastructure "Everywhere"

ELITE

- World renown deep engineering skills in Oracle and cloud platforms
- 1,000+ engineered systems configured / 200+ patched annually
- 500+ clients served annually on Oracle technologies and solutions
- Delivery Unit dedicated to Oracle (Accenture Oracle Practice)

EXPERTISE

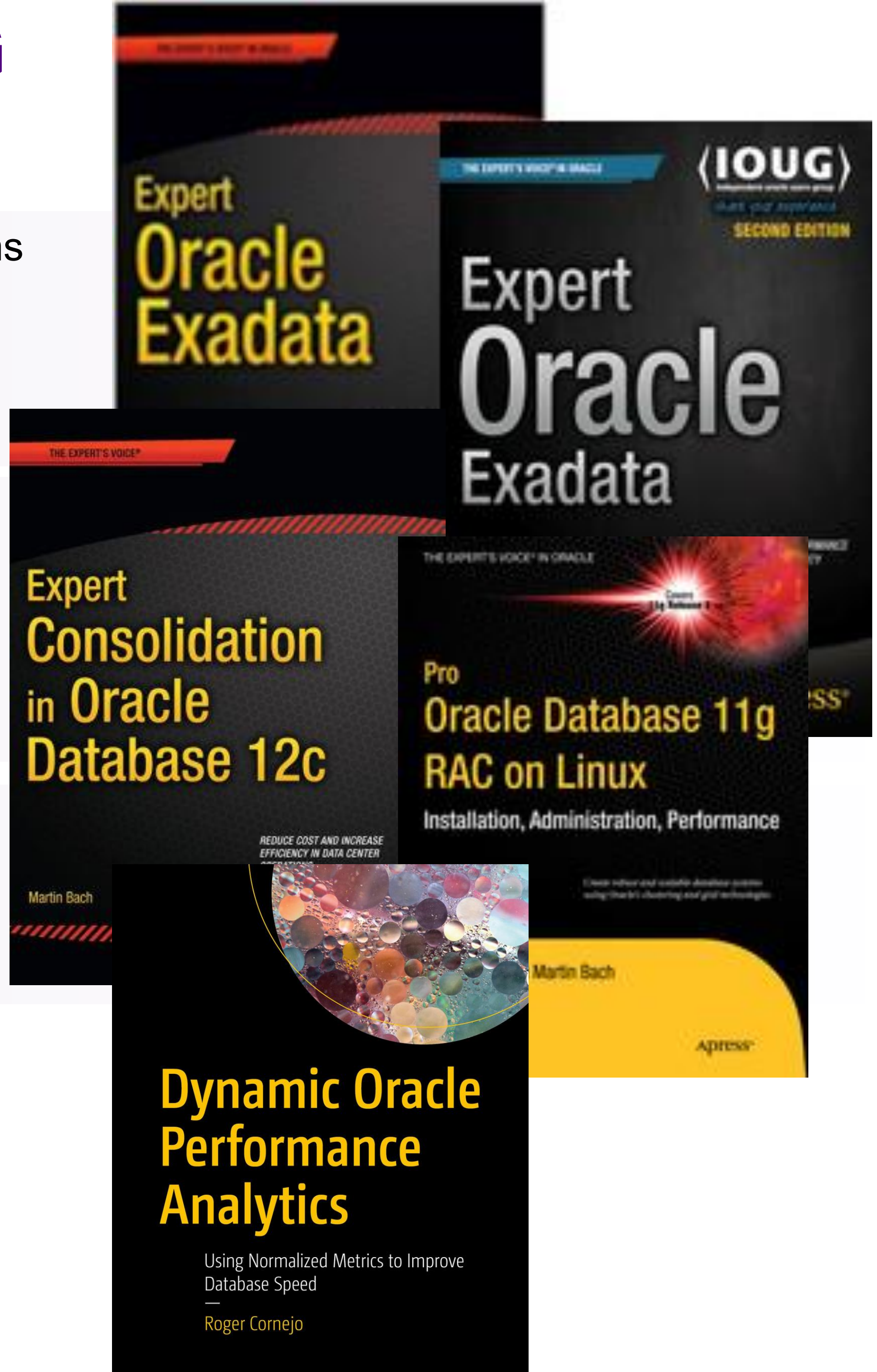
- 120 people specializing in Oracle and other cloud providers
- Average of 15+ years experience
- 17 Oracle ACEs in Accenture Enkitec Group
- 54,000 people in the Accenture Oracle Practice
- Bare metal solutions for Azure and Google

SUCCESS

- Many awards and recognitions
- Thousands of engineered systems configured and patched
- Hundreds of clients
- Massive library of performance and cloud assessment tools

THOUGHT LEADERSHIP

- Well-published in multiple subject areas
- Many online resources curated
- Sought after for conference presentations
- Many white papers produce on advanced topics such as Oracle cloud performance and TCO on the Oracle cloud

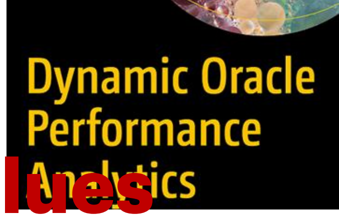


Thank You

Appendix

Normal Ranges

Statistical Analysis: Normal Range



Identify Abnormal Values

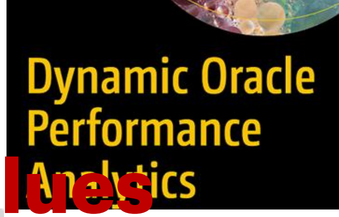
- **High and low values based on the "rule of thumb" + / - 2 <u>standard deviations</u> from the mean**

```
, avg_average - (2 * STDDEV_average) as lower_bound
, avg_average as average_value
, avg_average + (2 * STDDEV_average) as upper_bound
```

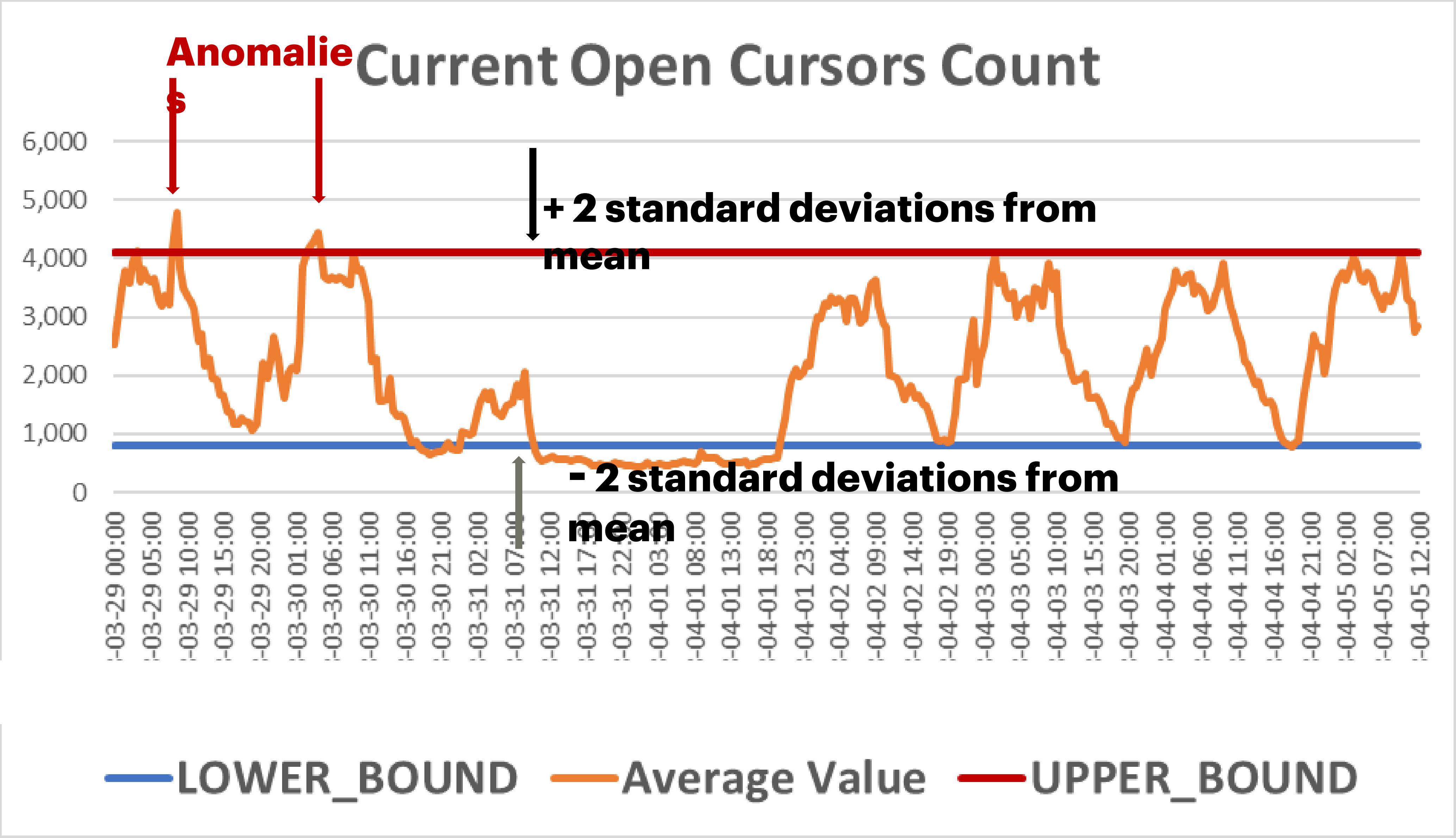
- **Calculation uses simple aggregate functions**

```
,   AVG(average)          as   avg_average
,   STDDEV(average)       as   stddev_average
```

Statistical Analysis: Normal Range



Identify Abnormal Values



Statistical Analysis:
Normal Range Housekeeping - Removing Outliers

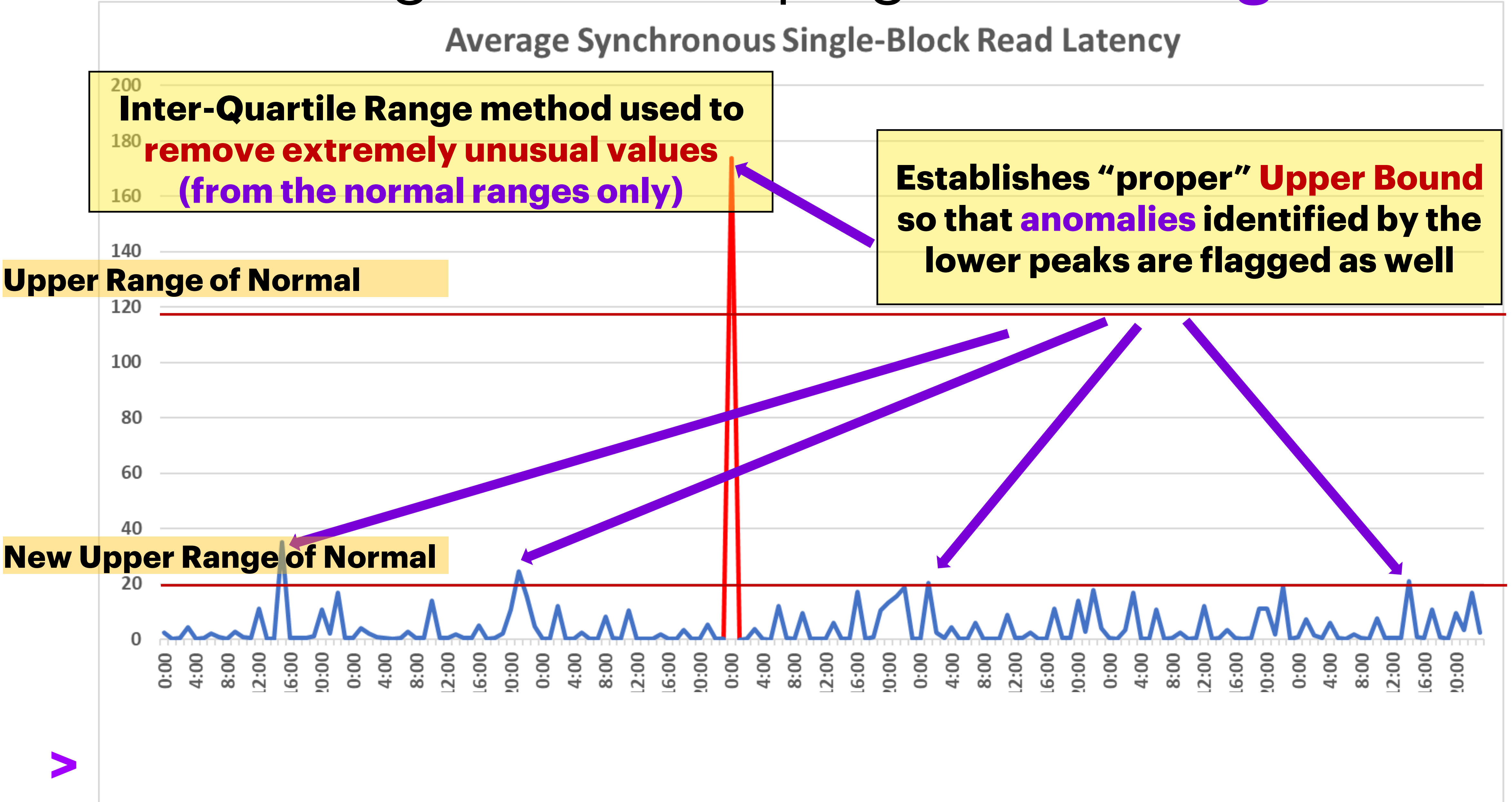
- **AWR occasionally has some data spikes (outliers) that distort statistical analysis**
- **For example:**
  - **Average Synchronous Single-Block Read Latency = ~3000**
  - **Falsely elevates the NR upper bound to ~150 milliseconds**
  - ***real* Normal Range upper bound ~2 milliseconds**
- **Remove Outliers: Use only those metric values between:**
  - **Standard Inter-Quartile Range method:**

    < Q1 – (1.5 * IQR) and > Q3 + (1.5 * IQR)

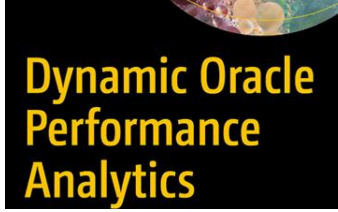
```
,  Q3 - Q1 as IQR
,  percentile_cont(0.25) within group (order by average) as Q1
,  percentile_cont(0.75) within group (order by average) as Q3
```

https://blogs.oracle.com/utilities/post/iqr-batch-threads

Statistical Analysis:
Normal Range Housekeeping - **Removing Outliers**



percentile_cont – Percentiles Example

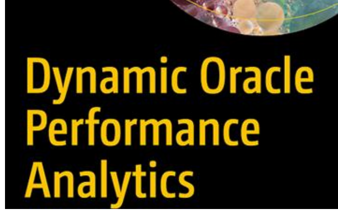


Quantify importance of a variable in s

Often used for Anomaly Dete

```
select metric_name
      , round(avg(average)) "Avg Value"
      , round(max(average)) "Max Value"
      , round(percentile_cont(0.98) within group (order by
average)) "98th Percentile"
      , round(percentile_cont(0.99) within group (order by
average)) "99th Percentile"
from dba_hist_sysmetric_summary
where metric_name like nvl(:metric_name_x, metric_name)
group by metric_name
order by 1
;
```

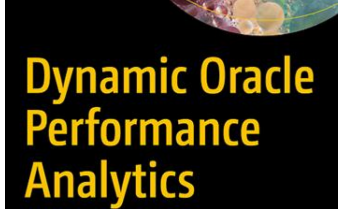
percentile_cont – Percentiles Example



Quantify importance of a variable in s
Often used for Anomaly Dete

METRIC_NAME	Avg Value	Max Value	98th Percentile	99th Percentile
Active Parallel Sessions	1	14	7	9
Active Serial Sessions	3	18	9	10
Average Active Sessions	3	20	12	14
Average Synchronous Single-Block Read Latency	3	40	14	16
Background CPU Usage Per Sec	14	663	68	77
Background Checkpoints Per Sec	0	0	0	0
Background Time Per Sec	0	7	2	2
Branch Node Splits Per Sec	0	1	0	0
Branch Node Splits Per Txn	0	2	0	1
Buffer Cache Hit Ratio	97	100	100	100
CPU Usage Per Sec	150	746	533	567
CPU Usage Per Txn	333	4711	1711	1981

percentile_cont – Percentiles Example



Quantify importance of a variable in s

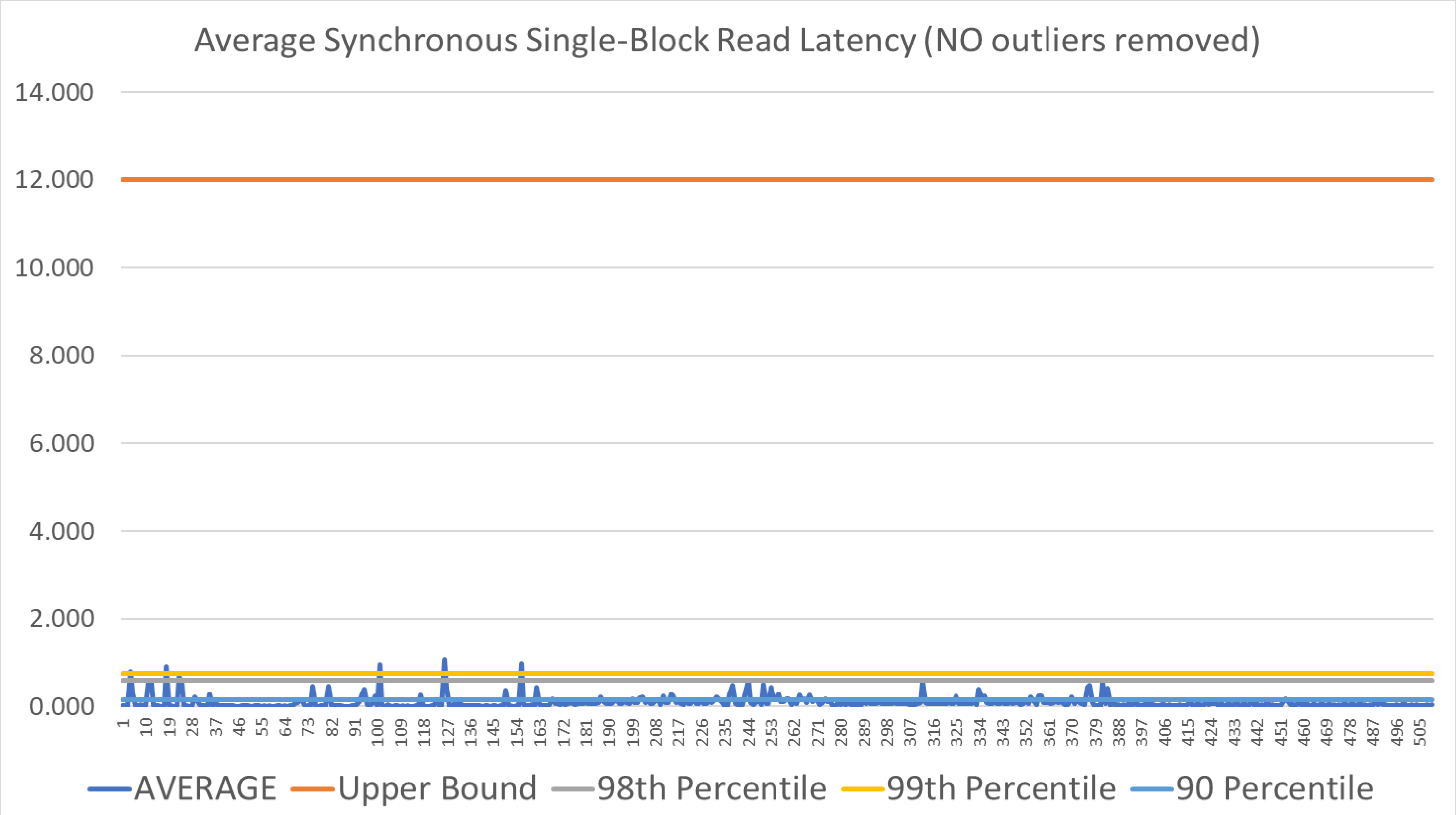
Often used for Anomaly Dete

```
-- intervals with most metric anomolies
with percentile as
(select /*+ MATERIALIZE */ metric_name
, round(percentile_cont(0.99) within group (order by average))
"99th Percentile"
from dba_hist_sysmetric_summary
where metric_name like nvl(:metric_name_x, metric_name)
group by metric_name)
, flagged_metrics as
(select /*+ MATERIALIZE */
hist.snap_id, hist.metric_name, pct."99th Percentile"
from dba_hist_sysmetric_summary hist, percentile pct
where hist.metric_name = pct.metric_name
  and hist.AVERAGE > pct."99th Percentile"
  and pct."99th Percentile" <> 0)
/* main query */
select snap_id, count(*) count_flagged_metrics
from flagged_metrics
group by snap_id order by 2 desc fetch first 10 rows only;
```

SNAP_ID	COUNT_FLAGGED_METRICS
28548	34
28784	26
28880	25
28377	22
28717	22
29220	21
28716	21
28238	21
29216	20
29052	19

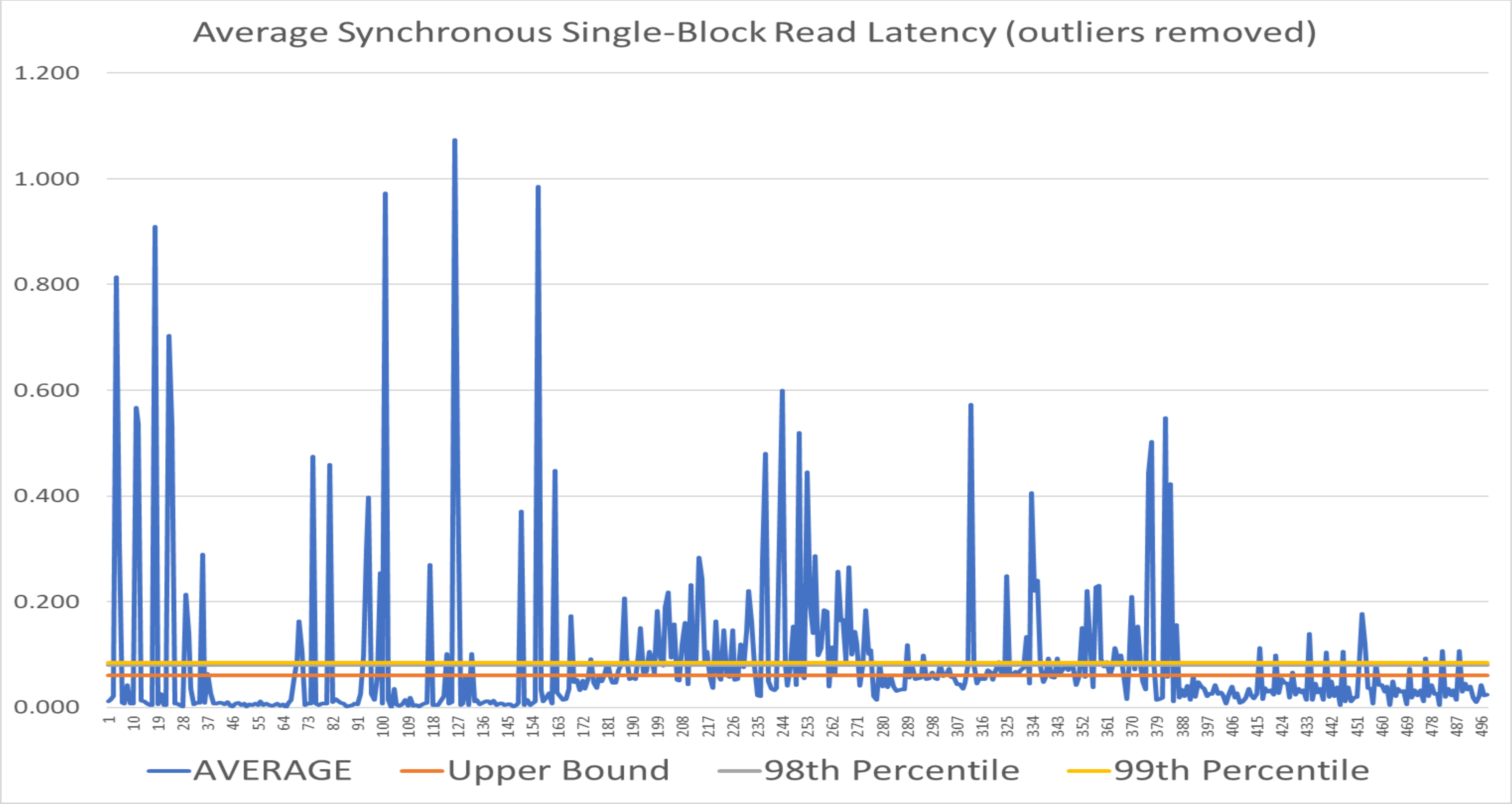
Removing Outliers – NO Outliers Removed

Dynamic Oracle Performance Analytics



Removing Outliers – Outliers Removed

Dynamic Oracle Performance Analytics



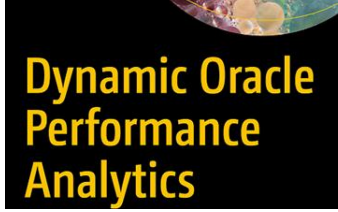
Notes on Anomaly Detection

- No perfect Anomaly detection mechanism
  - False Positives and False Negatives
    - False Positives → False anomaly flagged
      - Too many false positives → ignore distracting warnings
    - False Negatives → True anomalies missed
      - Too many false negatives → miss essential anomalous observations
- Use an anomaly detection mechanism that is
  - sensitive enough (catches true anomalies)
  - but is not too sensitive as to falsely flag anomalies
  - has other metrics to allow you to decide relevant anomalies

Percentiles vs Normal Ranges

<SKIP>

percentile_cont – Percentiles Example



Quantify importance of a variable in set

Often used for Anomaly Detection

```
select metric_name
     , round(avg(average)) "Avg Value"
     , round(max(average)) "Max Value"
     , round(percentile_cont(0.98) within group (order by
average)) "98th Percentile"
     , round(percentile_cont(0.99) within group (order by
average)) "99th Percentile"
from dba_hist_sysmetric_summary
where metric_name like nvl(:metric_name_x, metric_name)
group by metric_name
order by 1
;
```

`percentile_cont` – Percentiles Example

Dynamic Oracle Performance Analytics

Quantify importance of a variable in set

Often used for Anomaly Detection

METRIC_NAME	Avg Value	Max Value	98th Percentile	99th Percentile
Active Parallel Sessions	1	14	7	9
Active Serial Sessions	3	18	9	10
Average Active Sessions	3	20	12	14
Average Synchronous Single-Block Read Latency	3	40	14	16
Background CPU Usage Per Sec	14	663	68	77
Background Checkpoints Per Sec	0	0	0	0
Background Time Per Sec	0	7	2	2
Branch Node Splits Per Sec	0	1	0	0
Branch Node Splits Per Txn	0	2	0	1
Buffer Cache Hit Ratio	97	100	100	100
CPU Usage Per Sec	150	746	533	567
CPU Usage Per Txn	333	4711	1711	1981

Threshold Mechanisms for Anomaly Detection

- ❖ No perfect Anomaly detection mechanism
  - ❖ False Positives and False Negatives
    - ❖ False Positives → False anomaly flagged
      - ❖ Too many false positives → ignore distracting warnings
    - ❖ False Negatives → True anomalies missed
      - ❖ Too many false negatives → miss essential anomalies

- ❖ Use an anomaly detection mechanism that is
  - ❖ sensitive enough (catches true anomalies)
  - ❖ but is not too sensitive as to falsely flag anomalies
  - ❖ use other metrics to allow you to decide relevant anomalies

Threshold Mechanisms for Anomaly Detection

- **are more sensitive with outliers removed**
- **Normal ranges works well with outliers removed**
- **Percentile (say 99th) works well *with or without* outliers removed**
- **Metric data distributions influences how well a threshold mechanism flags anomalies**
- **SQL-Level anomaly detection code uses Percentiles**

Misc KVP Slides

SQL-Level Metric Anomaly Detection

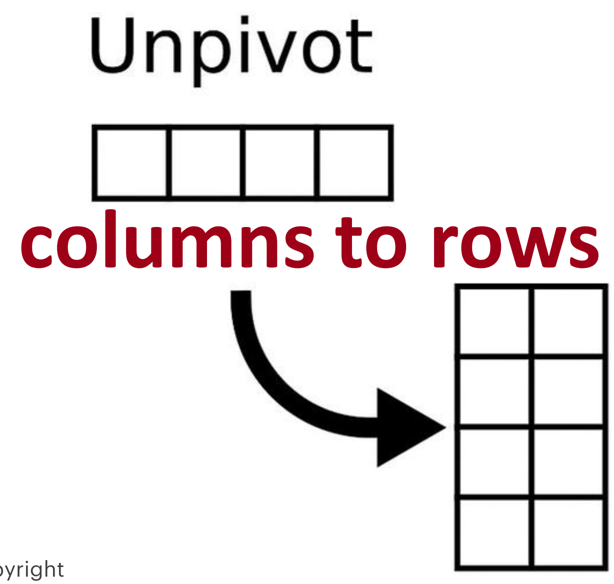
MCT View of SQLSTAT Metrics

```sql
/* Example: simple select on the
multi-column table version dba_hist_sqlstat for a sql_id */
select SNAP_ID
,SQL_ID,PLAN_HASH_VALUE
,FETCHES_DELTA
,END_OF_FETCH_COUNT_DELTA
,EXECUTIONS_DELTA
,LOADS_DELTA
,INVALIDATIONS_DELTA
,PARSE_CALLS_DELTA
,DISK_READS_DELTA
,BUFFER_GETS_DELTA
,ROWS_PROCESSED_DELTA
,CPU_TIME_DELTA
,ELAPSED_TIME_DELTA
,IOWAIT_DELTA
,CLWAIT_DELTA
,APWAIT_DELTA
,CCWAIT_DELTA
,DIRECT_WRITES_DELTA
,PLSEXEC_TIME_DELTA
,JAVEXEC_TIME_DELTA
,IO_OFFLOAD_ELIG_BYTES_DELTA
,IO_INTERCONNECT_BYTES_DELTA
,PHYSICAL_READ_REQUESTS_DELTA
,PHYSICAL_READ_BYTES_DELTA
,PHYSICAL_WRITE_REQUESTS_DELTA
,PHYSICAL_WRITE_BYTES_DELTA
,OPTIMIZED_PHYSICAL_READS_DELTA
,CELL_UNCOMPRESSED_BYTES_DELTA
,IO_OFFLOAD_RETURN_BYTES_DELTA
from dba_hist_sqlstat
where sql_id = '988a1hx9zc5rr'
order by sql_id, snap_id
;
```

SQL-Level Metric Anomaly Detection

KVP View of SQLSTAT Metrics

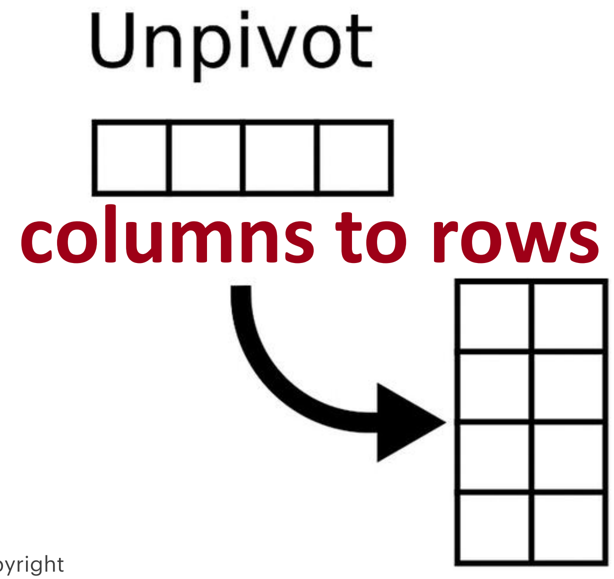
STAT SOURCE	METRIC_NAME
dba_hist_sqlstat	SUM: apwait (seconds)
dba_hist_sqlstat	SUM: apwait_per_execution (seconds)
dba_hist_sqlstat	SUM: buffer_gets (count)
dba_hist_sqlstat	SUM: buffer_gets_per_execution (count)
dba_hist_sqlstat	SUM: ccwait (seconds)
dba_hist_sqlstat	SUM: ccwait_per_execution (seconds)
dba_hist_sqlstat	SUM: cell_uncompressed (meg)
dba_hist_sqlstat	SUM: clwait (seconds)
dba_hist_sqlstat	SUM: clwait_per_execution (seconds)
dba_hist_sqlstat	SUM: cpu_time (seconds)
dba_hist_sqlstat	SUM: cpu_time_per_execution (seconds)
dba_hist_sqlstat	SUM: direct_writes (count)



SQL-Level Metric Anomaly Detection

KVP View of SQLSTAT Metrics

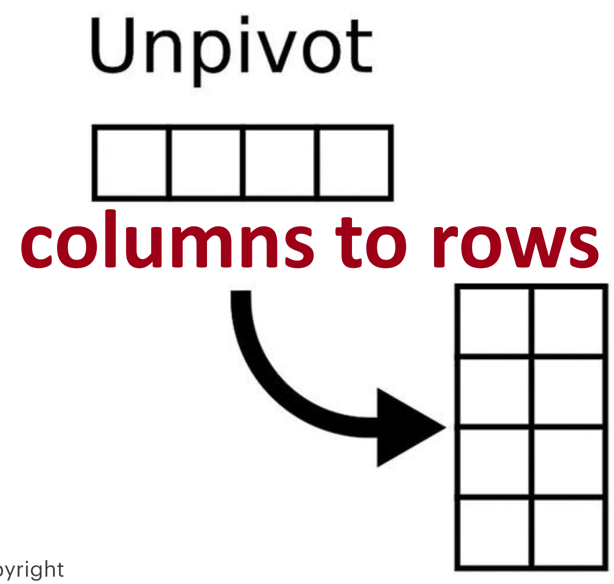
STAT SOURCE	METRIC_NAME
dba_hist_sqlstat	SUM: disk_reads (count)
dba_hist_sqlstat	SUM: elapsed_time (seconds)
dba_hist_sqlstat	SUM: elapsed_time_per_execution (seconds)
dba_hist_sqlstat	SUM: end_of_fetch_count (count)
dba_hist_sqlstat	SUM: executions (count)
dba_hist_sqlstat	SUM: fetches (count)
dba_hist_sqlstat	SUM: invalidations (count)
dba_hist_sqlstat	SUM: io_interconnect (meg)
dba_hist_sqlstat	SUM: io_offload_elig (meg)
dba_hist_sqlstat	SUM: io_offload_return (meg)
dba_hist_sqlstat	SUM: iowait (seconds)
dba_hist_sqlstat	SUM: iowait_per_execution (seconds)



SQL-Level Metric Anomaly Detection

KVP View of SQLSTAT Metrics

STAT SOURCE	METRIC_NAME
dba_hist_sqlstat	SUM: javexec_time (seconds)
dba_hist_sqlstat	SUM: loads (count)
dba_hist_sqlstat	SUM: optimized_physical_reads (count)
dba_hist_sqlstat	SUM: parse_calls (count)
dba_hist_sqlstat	SUM: physical_read (meg)
dba_hist_sqlstat	SUM: physical_read_requests (count)
dba_hist_sqlstat	SUM: physical_write (meg)
dba_hist_sqlstat	SUM: physical_write_requests (count)
dba_hist_sqlstat	SUM: plsexec_time (seconds)
dba_hist_sqlstat	SUM: px_servers_execs (count)
dba_hist_sqlstat	SUM: rows_processed (count)
dba_hist_sqlstat	SUM: rows_processed_per_execution (count)
dba_hist_sqlstat	SUM: sorts (count)



SQL-Level Metric Anomaly Detection

KVP View of Active Session History Metrics

	STAT SOURCE	METRIC_NAME
1	STAT SOURCE	METRIC_NAME
15	dba_hist_active_sess_history	SUM: in_plsql_rpc (seconds)
16	dba_hist_active_sess_history	SUM: in_sequence_load (seconds)
17	dba_hist_active_sess_history	SUM: in_sql_execution (seconds)
18	dba_hist_active_sess_history	SUM: is_background_session (seconds)
19	dba_hist_active_sess_history	SUM: is_blocked_session (seconds)
20	dba_hist_active_sess_history	SUM: is_captured (seconds)
21	dba_hist_active_sess_history	SUM: is_foreground_session (seconds)
22	dba_hist_active_sess_history	SUM: is_replayed (seconds)
23	dba_hist_active_sess_history	SUM: is_session_blocked (seconds)
24	dba_hist_active_sess_history	SUM: is_session_on_cpu (seconds)
25	dba_hist_active_sess_history	SUM: is_session_waiting (seconds)
26	dba_hist_active_sess_history	SUM: is_sql_executing_in_parallel (seconds)
27	dba_hist_active_sess_history	SUM: is_sqlid_current (seconds)

SQL-Level Metric Anomaly Detection

KVP View of Active Session History Metrics

1	STAT SOURCE	METRIC_NAME
28	dba_hist_active_sess_history	SUM: max pga_allocated (meg)
29	dba_hist_active_sess_history	SUM: max temp_space_allocated (meg)
30	dba_hist_active_sess_history	SUM: object processing: TABLE SUBPARTITION: S. S.S. T (SYS_SUBP 031) (seconds)
31	dba_hist_active_sess_history	SUM: object processing: TABLE SUBPARTITION: S S.S T (SYS_SUBP 032) (seconds)
32	dba_hist_active_sess_history	SUM: object processing: TABLE SUBPARTITION: S S.S. T (SYS_SUBP 033) (seconds)
33	dba_hist_active_sess_history	SUM: object processing: TABLE SUBPARTITION: S S.S. T (SYS_SUBP 034) (seconds)
34	dba_hist_active_sess_history	SUM: object processing: TABLE SUBPARTITION: S S.S. T (SYS_SUBP 035) (seconds)

SQL-Level Metric Anomaly Detection

KVP View of Active Session History Metrics
… many partitions

1	STAT SOURCE	METRIC_NAME
44	dba_hist_active_sess_history	SUM: ON CPU (seconds)
45	dba_hist_active_sess_history	SUM: replay_overhead (seconds)
46	dba_hist_active_sess_history	SUM: session_and_serial# (count distinct)
47	dba_hist_active_sess_history	SUM: sql_child_number (count distinct)
48	dba_hist_active_sess_history	SUM: sql_execution_id (count distinct)
49	dba_hist_active_sess_history	SUM: sql_plan_hash_value (count distinct)
50	dba_hist_active_sess_history	SUM: sql_plan_hash_value: 0
51	dba_hist_active_sess_history	SUM: sql_plan_hash_value: 1510797065
52	dba_hist_active_sess_history	SUM: sql_plan_line_id: hash: 0 | line: | operation: INSERT STATEMENT | options: (seconds)
53	dba_hist_active_sess_history	SUM: sql_plan_line_id: hash: 1510797065 | line: | operation: | options: (seconds)

SQL-Level Metric Anomaly Detection

KVP View of Active Session History Metrics

	STAT SOURCE	METRIC_NAME
1	STAT SOURCE	METRIC_NAME
54	dba_hist_active_sess_history	SUM: sql_plan_line_id: hash: 1510797065 | line: | operation: INSERT STATEMENT | options: (seconds)
55	dba_hist_active_sess_history	SUM: sql_plan_line_id: hash: 1510797065 | line: 1 | operation: LOAD TABLE CONVENTIONAL | options: (seconds)
56	dba_hist_active_sess_history	options: (seconds)
57	dba_hist_active_sess_history	SUM: sql_plan_line_id: hash: 1510797065 | line: 3 | operation: PX COORDINATOR | options: (seconds)
58	dba_hist_active_sess_history	SUM: sql_plan_line_id: hash: 1510797065 | line: 6 | operation: TABLE ACCESS | options: FULL (seconds)
59	dba_hist_active_sess_history	SUM: sql_plan_operation: DML Insert (seconds)
60	dba_hist_active_sess_history	SUM: sql_plan_operation: Load Table (CONVENTIONAL) (seconds)
61	dba_hist_active_sess_history	SUM: sql_plan_operation: Parallel (seconds)
62	dba_hist_active_sess_history	SUM: sql_plan_operation: Sequence Access (seconds)

SQL-Level Metric Anomaly Detection

KVP View of Active Session History Metrics

1	STAT SOURCE	METRIC_NAME
63	dba_hist_active_sess_history	SUM: sql_plan_operation: Table Access (FULL) (seconds)
64	dba_hist_active_sess_history	SUM: sql_plan_operation: Unspecified (seconds)
65	dba_hist_active_sess_history	SUM: tm_delta_cpu_time (seconds)
66	dba_hist_active_sess_history	SUM: tm_delta_db_time (seconds)
67	dba_hist_active_sess_history	SUM: tm_delta_idle_time (seconds)
68	dba_hist_active_sess_history	SUM: tm_delta_time (seconds)
69	dba_hist_active_sess_history	SUM: transaction_id (count distinct)
70	dba_hist_active_sess_history	SUM: user executing: SAI_UMA (seconds)
71	dba_hist_active_sess_history	SUM: wait class: Other (seconds)
72	dba_hist_active_sess_history	SUM: wait class: User I/O (seconds)
73	dba_hist_active_sess_history	SUM: wait event: db file scattered read ; wait class: User I/O (seconds)
74	dba_hist_active_sess_history	SUM: wait event: reliable message [blocked event] ; wait class: Other (seconds)
75	dba_hist_active_sess_history	SUM: wait_event_sequence_number (count distinct)