



**Database Rx Performance Portal for
Monthly Review Client Services Summary**

April 23, 2009

The **Database Rx Performance Portal** is the proprietary tool that the DBA Pro team consultants use for mining data from our repository of database metrics gathered by our resident agent. Many portal screens are also available to customer staff members. We maintain a substantial history of metric data on each database, allowing for the identification of larger trends and for capacity planning. We do not seek to impress with flashy graphics and colored dials. We are focused on substance, simplicity, and keeping databases running and running well.

As part of the DBA Pro service we install our Database Rx® agent which checks the database every few minutes for connectivity and to assure its basic functions. Every hour the agent does a more thorough check, gathering a significant amount of metadata about the database and sending it to our server for analysis. The analysis engine on our server checks over 100 different areas, including free space in tablespaces, segment statistics, log archiving status, system resources approaching parameter limits, resource-intensive queries, broken or failed jobs, users with inappropriate defaults, and many more. Custom metrics can be defined that are specific to your environment. If any of the tests performed result in values outside desired thresholds (for instance, less than 10% free space in a tablespace) you can be notified via e-mail or pager and contact us during business hours, depending on the severity of the alert. If you want the alert to come to us, you would need to sign up for our daily review service, which also has the benefit of proactive support.

Since this is not a daily review program it is up to the customer to look at the portal and contact us during business hours. Customer staff may be paged depending on the severity level, however. The same goes for lesser severity alerts that are typically emailed. Any number of your staff members may receive alerts from our monitoring system which may be customized to your specific needs.

Performance of an Oracle database can be greatly impacted through the careful handling of an experienced Oracle specialist. Our team has many years of experience dealing with a variety of challenges including performance tuning, disaster recovery, capacity planning, architecture, training, and troubleshooting. That depth of experience is not easy to come by when relying solely on internal resources. That is why establishing and maintaining a relationship with a trusted partner is critical. Our long-term customers also share the benefit of historical data collected on their databases which helps identify trend changes and increase troubleshooting efficiency.

Using information from the Performance Portal we prepare a **Monthly Review** report that you will receive every month on the state of your databases. Our goal is to summarize events of the past 30 days and highlight areas that need specific attention. A sample monthly review is attached after some sample portal screen shots to give you an idea of what we look at and how we report it to you.

Instance Status - Windows Internet Explorer

https://dbrx.dbspecialists.com/pls/dbrx/view_rep

Instance Status

DatabaseRx

Performance Portal

Powered By: DatabaseSpecialists

Select instance: DBRx production Launch report: Instance Status

Select **Launch**

[Personal settings](#) [About Database Rx](#)
[Instance settings](#) [About Database Specialists](#)
[Sign up new instance](#) [Give feedback about Database Rx](#)
[Download Database Rx Agent](#) [Contact Database Specialists](#)
[Log out](#) [Contact users](#)
 [DBA Pro Data Entry](#)

Welcome Gary Sadler

Instance Status

Customize Refresh

Active Alerts and Events for DBRx production as of 04-23-2009 13:00

Functional Area	Severity 1	Severity 2	Severity 3	Severity 4	Severity 5	Total
Total						0

Past Alerts and Events for DBRx production - 04-20-2009 13:00 through 04-23-2009 13:00

Date	Severity	Brief Description	Acknowledgement
04-21-2009 01:00	3	Tablespaces approaching space limits	
04-21-2009 22:00	4	Monitored tables with stale statistics	

[Back to top](#)

Copyright © 2009 Database Specialists, Inc.
 Database Rx is a registered trademark of Database Specialists, Inc., worldwide rights reserved.

Done Internet 100%

DatabaseRx

Performance Portal

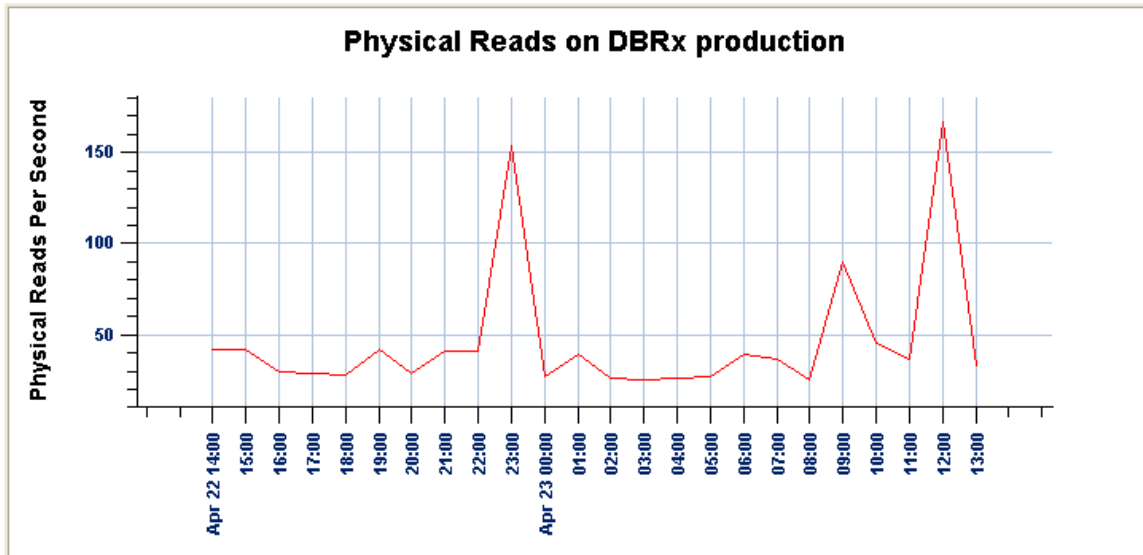
Powered By:
DatabaseSpecialists

Select instance: DBRx production **Select** Launch report: Instance Activity **Launch**

[Personal settings](#) [About Database Rx](#)
[Instance settings](#) [About Database Specialists](#)
[Sign up new instance](#) [Give feedback about Database Rx](#)
[Download Database Rx Agent](#) [Contact Database Specialists](#)
[Log out](#) [Contact users](#)
[DBA Pro Data Entry](#)

Welcome Gary Sadler

Instance Activity



© B.V. Meza-Sandoa Corp. 1.6

Time Interval			Physical Reads		
Start	End	Duration	Per Second	Per Transaction	Total
04-23-2009 12:00	04-23-2009 13:00	60 minutes (3600 seconds)	33	76	117,930
04-23-2009 11:00	04-23-2009 12:00	60 minutes (3600 seconds)	167	371	601,875
04-23-2009 10:00	04-23-2009 11:00	60 minutes (3600 seconds)	37	79	132,391
04-23-2009 09:00	04-23-2009 10:00	60 minutes (3600 seconds)	45	102	160,936
04-23-2009 08:00	04-23-2009 09:00	60 minutes (3600 seconds)	90	203	324,550
04-23-2009 07:00	04-23-2009 08:00	60 minutes (3600 seconds)	25	60	91,516

Top 5 Wait Events (entire instance)

[View detail](#)

Wait Event	Waits	Timeouts	Average Seconds Per Wait	Total Seconds Waited	Percent Of All Non-idle Event Wait Time
db file sequential read	2,894,599	0	0.003	7,936.67	93.31
db file scattered read	41,615	0	0.011	466.14	5.48
log file parallel write	73,373	0	0.001	56.04	0.66
log file sync	12,508	0	0.001	14.59	0.17
control file parallel write	29,664	0	0.000	13.85	0.16

Top 5 Wait Events (background sessions only)

[View detail](#)

Wait Event	Waits	Timeouts	Average Seconds Per Wait	Total Seconds Waited	Percent Of All Non-idle Event Wait Time
log file parallel write	73,373	0	0.001	56.04	52.50
db file sequential read	21,102	0	0.001	22.09	20.70
control file parallel write	29,664	0	0.000	13.86	12.98
Log archive I/O	2,332	0	0.002	4.69	4.39
log file sequential read	2,288	0	0.002	4.11	3.85

Top 5 Resource Intensive SQL (ranked by total buffer gets)

[View detail](#)

SQL ID	Address and Child Number	Buffer Gets	Executions	Gets Per Execution	Percent Of All Gets On Instance	Parsing User ID	CPU Seconds
SQL Statement							
cm0qzbn4c8rj	00000000807324F8 / 0	1,099,577	525,253	2.09	0.64	56	15.13
SELECT LOWER (OMIT_VALUE) OMIT_VALUE FROM ANALYSIS_INSTANCE_OMIT_LISTS WHERE INSTANCE_ID = :B2 AND OMIT_LIST_ID = :B1							
36j9zuz8ntzx	00000000806EB328 / 0	740,698	46,973	15.77	0.43	56	17.82

Execution Plan

ID	Parent	Operation
0		SELECT STATEMENT Optimizer=ALL_ROWS (Cost=9)
1	0	SORT (ORDER BY) (Cost=9 Card=2 Bytes=762)
2	1	NESTED LOOPS (Cost=8 Card=2 Bytes=762)
3	2	NESTED LOOPS (Cost=6 Card=2 Bytes=38)
4	3	INDEX (RANGE SCAN) OF "ANALYSIS_RESULTS_PK" (Cost=3 Card=2 Bytes=22)
5	3	TABLE ACCESS (BY INDEX ROWID) OF "ANALYSIS_COMMON_RESULTS" (Cost=2 Card=1 Bytes=)
6	5	INDEX (UNIQUE SCAN) OF "ANALYSIS_COMMON_RESULTS_PK" (Cost=1 Card=1 Bytes=)
7	2	TABLE ACCESS (BY INDEX ROWID) OF "ANALYSIS_TESTS" (Cost=1 Card=1 Bytes=362)
8	7	INDEX (UNIQUE SCAN) OF "ANALYSIS_TESTS_PK" (Cost=0 Card=1 Bytes=)

Available Versions

Address	Child Number	Parsing User ID	Optimizer Mode	Data Points
00000000806EA1A8	0	56	ALL_ROWS	11

Statement Statistics

	04-20-2009 23:00:01	04-23-2009 13:00:01	Difference	Per Execution During Time Interval
SQL ID	3j60jbayq9z3u	Same		
Hash value	3177512058	Same		
Address	00000000806EA1A8	Same		
Child Number	0	Same		
Executions	48,776	54,002	5,226	1
Buffer gets	566,942	656,848	89,906	17
Disk reads	312	341	29	0
Rows processed	35,069	38,251	3,182	1
CPU time (seconds)	10.259	12.477	2.217695	0.000424
Elapsed time (seconds)	10.895	13.168	2.272827	0.000435
Sorts	48,776	54,002	5,226	1
Loads	1	Same	0	0
Invalidations	0	Same	0	0
Parse calls	2	Same	0	0
Sharable memory	31175	6863		
Persistent memory	9704	Same		



www.dbspecialists.com

Acme Widgets
Monthly Review Report for February 2008
Prepared by Gary Sadler, Staff Consultant (gsadler@dbspecialists.com)

PROD instance

Summary:

We began monitoring this database as of February 6, 2008, so this first review represents an incomplete snapshot of monthly activity. Our first impression is that the database is performing reasonably well, but features a lot of underperforming SQL statements that can be improved through the use properly placed indexes. Other than that, there are just a few items that need attention, none of which is critical in nature.

Current Alerts

As of February 29, there are five active alerts as follows:

- Many SQL statements were flagged as resource-intensive for their number of logical and physical reads. Please see the section below titled "Resource Intensive SQL" for details.
- The instance is configured with three online redo log groups. We generally recommend a minimum of four groups on a busy system, but yours is heavy on reads and light on writes. There is no evidence of waits on "log file sync" or similar. Therefore, three groups are sufficient and we will not mention this alert in future reviews.
- There are a number of invalid objects. The list follows. Some or all of them may become validated through use or by compiling them manually.

```
PUBLIC SYNONYM X$KSPPI
PUBLIC SYNONYM X$KSPPSV
FUNCTION SYSMAN.DECRYPT
FUNCTION SYSMAN.DECRYPTBYTES
FUNCTION SYSMAN.ENCRYPT
FUNCTION SYSMAN.ENCRYPTBYTES
PACKAGE SYSMAN.MGMT_LOCK_UTIL
PACKAGE BODY SYSMAN.EMD_BCNTXN
PACKAGE BODY SYSMAN.EMD_LOADER
PACKAGE BODY SYSMAN.EM_SEC
PACKAGE BODY SYSMAN.MGMT_CREDENTIAL
PACKAGE BODY SYSMAN.MGMT_JOB_ENGINE
PACKAGE BODY SYSMAN.MGMT_LOCK_UTIL
PACKAGE BODY SYSMAN.MGMT_LOGIN_ASSISTANT
PACKAGE BODY SYSMAN.MGMT_TARGET
```

```

PACKAGE BODY SYSMAN.MGMT_TARGET_UPDATE
PACKAGE BODY SYSMAN.MGMT_USER
PACKAGE BODY SYSMAN.MGMT_VIEW_PRIV
TRIGGER SYSMAN.MGMT_CREDS_INS_UPD

```

- The ADMIN, DIP, ICRM_DBA, and TSMSYS user accounts have the SYSTEM tablespace as their default for object creation. There is the potential for these users to interfere with overall database function if large objects are created without specifying a tablespace where the objects will reside. TSMSYS and DIP are Oracle-created accounts and should create objects in the SYSAUX or TOOLS tablespaces. The USERS tablespace might be a good choice of default tablespace for the others.
- The DATA tablespace has 9% free space and is not set to autoextend so allocating additional free space is recommended.

Historical Alerts

There were no other relevant alerts beyond those that were mentioned in the previous section.

Database Size and Growth

Allocated space within the database grew by 2% during the 23-day monitoring period. All tablespaces with the exception of DATA currently appear to have sufficient space for growth or are configured to extend automatically.

Resource Intensive SQL

There are quite a few resource intensive SQL statements being executed, so we will not address them all right now. But the performance of the worst offenders can be improved greatly with some quick changes.

The worst offender is a PL/SQL block with hash value 2334543584. It is executed once a minute during the day, averaging about 57,000 logical reads per execution. It accounts for 60% of the logical reads performed by the database during daytime hours. The text of the block is:

```

DECLARE      -- variables for declaration
v_cur_date  c_ticker_restriction.expires%TYPE;
v_cur_user  c_ticker_restriction.user_id%TYPE;
v_ticker    c_ticker_restriction.ticker%TYPE;

CURSOR c_direct_holdings IS
  SELECT DISTINCT ticker_sym
  FROM positions
  WHERE (abs(shares) > 0.99) or (abs(pend_shares) > 0.99);
BEGIN
  SELECT SYSDATE INTO v_cur_date FROM DUAL;
  SELECT USER INTO v_cur_user FROM DUAL;
  OPEN c_direct_holdings;
  LOOP
    FETCH c_direct_holdings INTO v_ticker;
    UPDATE ticker SET restriction=3,
      port_type='HELD', user_id=v_cur_user, expires=trunc(v_cur_date)+15
    WHERE symbol=v_symbol;
    EXIT WHEN c_direct_holdings %NOTFOUND;
  END LOOP;
  CLOSE c_direct_holdings;
  COMMIT;
END;

```

This block goes through every qualifying row in the POSITIONS table (typically 100-200) and, for each row, must do a full table scan of the TICKER table. An index on TICKER(SYMBOL) would help a lot, and would help several other queries. But an even better solution would be to do all the updates as one SQL statement, rather than separate statements for each qualifying POSITIONS row.

This statement would be:

```
update ticker SET restriction=3,
                port_type='HELD', user_id=:v_cur_user, expires=trunc(sysdate)+15
where symbol in (SELECT ticker_sym
                FROM positions
                WHERE (abs(shares) > 0.99) or (abs(pend_shares) > 0.99));
```

This would require one full scan of the POSITIONS table and one full scan of the TICKER table, reducing the logical reads performed by the block by about 99%. As a result the database would be working about half as hard during the day.

We've also noticed a general lack of indexes on some tables in the database. Adding just a few indexes will help performance substantially. For instance, several queries on the ANA_PERF table would benefit. Some of the queries are below:

Hash value 870012409

```
SELECT * FROM ANA_PERF WHERE MKT_END_DATE > (CURRENT_DATE - 5) AND
TICKER='TOTAL' AND PERIOD_CODE='Q' AND STYLE_CODE=:1 ORDER BY
MKT_END_DATE DESC
```

Hash value 735458541

```
SELECT * FROM ANA_PERF WHERE MKT_END_DATE > (CURRENT_DATE - 5) AND
TICKER='TOTAL' AND PERIOD_CODE='M' AND STYLE_CODE=:1 ORDER BY
MKT_END_DATE DESC
```

Hash value 1464391819

```
SELECT TICKER, STYLE_CODE, PERIOD_CODE, MKT_START_DATE, MKT_END_DATE,
PCT_GL_MKT, IRR_POS FROM ANA_PERF WHERE TICKER='TOTAL' AND
MKT_END_DATE=:1 AND STYLE_CODE=:2 AND PERIOD_CODE='Q' ORDER BY
STYLE_CODE
```

Hash value 2635912821

```
SELECT TICKER, STYLE_CODE, PERIOD_CODE, MKT_START_DATE, MKT_END_DATE,
PCT_GL_MKT, IRR_POS FROM ANA_PERF WHERE TICKER='TOTAL' AND
MKT_END_DATE=:1 AND STYLE_CODE=:2 AND PERIOD_CODE='M' ORDER BY
STYLE_CODE
```

Hash value 1919076454

```
SELECT TICKER, STYLE_CODE, PERIOD_CODE, MKT_START_DATE, MKT_END_DATE,
PCT_GL_MKT, IRR_POS FROM ANA_PERF WHERE TICKER='TOTAL' AND
MKT_END_DATE=:1 AND STYLE_CODE=:2 AND PERIOD_CODE='Y' ORDER BY
STYLE_CODE
```

An index on ANA_PERF(TICKER, PERIOD_CODE, STYLE_CODE, MKT_END_DATE) would reduce the logical and physical reads performed by each of these queries by more than 99%.

If you make these changes, we think you'll notice a dramatic improvement in performance, and an overall reduction of workload for the database. We'll examine additional queries next month.

Logical and Physical Read Volume

Logical reads for the 23 day period (partial month) totaled 1,977 million, or 982 per second. Physical reads totaled 118 million in the period, or 59 per second. In future monthly reviews, we will report the percentage change from the previous month.

Wait Events

For the 23 day period (partial month), the most prominent wait event was " RMAN backup & recovery I/O", which accounted for 23 thousand seconds of wait time, 56% of all non-idle wait time. This is likely due to the arrangement where the RMAN catalog is at another location and data is traveling across the WAN.

Performance Ratios

For the 23-day period, common performance ratios were as follows:

Buffer cache hit ratio:	96.14%
Library cache hit ratio:	99.95%
Latch hit ratio:	100.0%
Sorts performed in memory:	99.87%
Parses that are soft parses:	99.75%

All ratios are in the acceptable range.

CPU Usage

CPU seconds used by Oracle server processes for the 23 day period (partial month) totaled 35,286 seconds (0.02 CPU second per elapsed second). This is generally a very low level of CPU usage. In future months we will compare that rate to the previous month.