如何执行对数据库的健康状况检查 (Doc ID 1548891.1)

适用于:

Oracle Database Cloud Schema Service - 版本 N/A 和更高版本
Oracle Database Exadata Cloud Machine - 版本 N/A 和更高版本
Oracle Database Exadata Express Cloud Service - 版本 N/A 和更高版本
Oracle Cloud Infrastructure - Database Service - 版本 N/A 和更高版本
Oracle Database Backup Service - 版本 N/A 和更高版本
本文档所含信息适用于所有平台

用途

本文章解释如何执行对数据库的基本健康状况检查，以检查一些配置问题。本文章给出了一些一般准则，来指导用户应该对哪些领域进行调查，才能更好地了解数据库运

如果您想进行更深入的健康状况检查来检查数据库结构和数据字典完整性，请查看第 11 章的相关链接。

此处调查的领域主要是基于一些脚本，我们不保证这些脚本的完整性和正确性。对于以后的数据库版本和功能，这些脚本可能需要改写。
本文章的内容可能需要进一步扩充，以适应特定应用程序检查的需要。

尽管本文章中讨论了一些性能方面的问题，但是本文的目的不在于提供优化数据库性能的完整详细解释

适用范围

详细信息

1. 参数文件

参数文件有 2 种形式。首先，我们有基于文本的版本，通常称为 init.ora 或 pfile，和一个基于二进制的文件，通常称为 spfile。可以使用标准操作系统编辑器调整 pfile，而 spfile 需要通过实例本身进行管理。

一定要注意，spfile 优先于 pfile，即除非另有说明，只要有 spfile 可用，都将被自动应用。

注意：建议在数据库配置发生变化后，生成一份 RDA 报告。保留历史 RDA 报告，可以确保您随着数据库演化对数据库配置有个大概了解。

参考：
Note 249664.1 Pfile vs SPfile

2. 控制文件

强烈建议至少拥有两份控制文件。可以通过镜像控制文件完成该操作，并且强烈建议放在不同的物理磁盘上。如果由于磁盘崩溃等原因，导致控制文件丢失，则您可以使用镜像文件启动数据库。通过这种方法，丢失的控制文件可以
轻松简单地被恢复。

```
connect as sysdba
SQL> select status, name from v$controlfile;

STATUS NAME
------- ---------------------------------
/u01/oradata/<SID>/control01.ctl
/u02/oradata/<SID>/control02.ctl
```

控制文件的位置和数量可以通过初始化参数 "control_files" 进行控制。

3. 重做日志文件

Oracle 服务器维护联机重做日志文件，以尽量减少数据库中的数据丢失。重做日志文件用于示例失败等情况，以恢复还未写进数据文件的已提交数据。强烈建议在不同物理磁盘上对重做日志文件进行镜像，从而在因磁盘崩溃、用户删除等原因导致其中一个重做日志文件丢失时，恢复起来更加容易。

```
connect as sysdba
SQL> select * from v$logfile;

GROUP# STATUS TYPE MEMBER
--------- ------- ------ ----------------------------------
1 ONLINE /u01/oradata/<SID>/redo01_A.log
1 ONLINE /u02/oradata/<SID>/redo01_B.log

2 ONLINE /u01/oradata/<SID>/redo02_A.log
2 ONLINE /u02/oradata/<SID>/redo02_B.log

3 ONLINE /u01/oradata/<SID>/redo03_A.log
3 ONLINE /u02/oradata/<SID>/redo03_B.log
```

虽然至少需要两个重做日志组，但是在启用归档时最好具有至少三个重做日志组。（请参阅下一章）。在存在大量日志切换的环境中，通常会看到 ARCHiver 后台进程归档的速度落后于 LGWR 后台进程生成日志的速度。在这种情况下，LGWR 进程需要等待 ARCH 进程完成归档重做日志文件。

参考：
Note 102995.1 Maintenance of Online Redo Log Groups and Members

4. 归档

归档提供了备份数据库变化所需的机制。归档文件非常重要，它提供了恢复数据库所必须的信息。建议在归档日志模式下运行数据库，虽然可能有理由不这样做，比如处于 TEST（测试）环境中时，丢失在当前时间与最后一次备份之间所做的更改对您来说是可以接受的。
当数据库不处于归档日志模式时，您可以略过本章。

检查归档配置的方法有数种，以下是其中一种：

```
connect as sysdba
SQL> archive log list

Database log mode No Archive Mode --OR-- Archive Mode
Automatic archival Disabled --OR-- Enabled
Archive destination <arch. dest.> --OR-- USE_DB_RECOVERY_FILE_DEST
Oldest online log sequence seq. no
Current log sequence seq. no
```

对于 10g 之前的版本，如果数据库在归档日志模式下运行，但是禁用了自动归档程序进程，则您需要手动归档重做日志文件。
如果未及时执行该操作，则数据库会被冻结，任何活动都会被阻止。
因此，当数据库处于归档日志模式时，您应该启动自动归档。通过在参数文件中将 "log_archive_start" 参数设置为 true，可以完成此操作。
从 10g 开始，该参数已经弃用，不再要求明确进行设置。归档文件的指定磁盘上必须有足够的可用空间，否则 ARCHiver 进程无法写入，必定会导致数据库崩溃。

参考：
Note 69739.1 How to Turn Archiving ON and OFF
Note 122555.1 Determine how many disk space is needed for the archive files

5. 数据文件

5.1 Autoextend

autoextend 命令选项可以启用或禁用数据文件的自动扩展。如果自动扩展的数据文件无法分配所需的空间，它会自动增加数据文件的大小以获取更多空间来给对象增长使用。

标准的 Oracle 数据文件最多可以包含 4194303 个 Oracle 数据块。
所以这也表示单个数据文件大小的上限取决于所用的 Oracle 块大小。

```
DB_BLOCK_SIZE Max Mb value to use in any command
~~~~~~~~~~~~~ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
2048 8191 M
4096 16383 M
8192 32767 M
16384 65535 M
```

从 Oracle 10g 开始，我们增加了一个称为 BIGFILE 的新功能，该功能允许创建更大的文件。请注意，每个操作系统都有其一定的限制，因此您需要确保数据文件的最大大小不超过操作系统允许的限制。

要确定数据文件进而表空间是否具有 AUTOEXTEND 功能：

```
SQL> select file_id, tablespace_name, bytes, maxbytes, maxblocks, increment_by, file_name
from dba_data_files
where autoextensible = 'YES';
```

参考：
Note 112011.1 ALERT: RESIZE or AUTOEXTEND can "Over-size" Datafiles and Corrupt the Dictionary
Note 262472.1 10g BIGFILE Type Tablespaces Versus SMALLFILE Type

5.2 位置

验证数据文件的位置。随着时间推移，数据库可能会增长，并会向数据库中添加数据文件。要避免基于"哪儿有空间就放哪儿"来随意放置数据文件，因为这会使备份策略和维护变得复杂。

以下为不良使用的一个示例：

```
SQL> select * from v$dbfile;

FILE# NAME
--------- --------------------------------------------------
1 D:\DATABASE\SYS1D806.DBF
2 D:\DATABASE\D806\RBS1D806.DBF
3 D:\DATABASE\D806\TMP1D806.DBF
5 D:\DATABASE\D806\USR1D806.DBF
6 D:\USR2D806.DBF
7 F:\ORACLE\USR3D806.DBF
```

参考：
Note 115424.1 How to Rename or Move Datafiles and Logfiles

6. 表空间

6.1 SYSTEM 表空间

用户对象不应在系统表空间中创建。如果创建，将导致不必要的碎片，并阻止系统表的增长。以下查询会返回一个列表，列出在系统表空间中已经创建的，但不属于 SYS 或 SYSTEM的对象。

```
SQL> select owner, segment_name, segment_type
from dba_segments
where tablespace_name = 'SYSTEM'
and owner not in ('SYS','SYSTEM');
```

6.2 SYSAUX 表空间（10g 及更高版本）

创建或升级数据库时，SYSAUX 表空间会被自动创建，作为SYSTEM表空间的辅助表空间。之前创建并使用独立表空间的一些数据库组件，现在也开始使用 SYSAUX 表空间。

如果 SYSAUX 表空间不可用，则核心数据库功能将仍可操作。使用 SYSAUX 表空间的数据库功能可能会失败，或是功能受限。

如果配置不当，则此表空间中存储的数据量可能巨大，并随着时间推移而增长到无法管理的大小。有一些组件需要特别注意。

要检查哪些组件正占用空间：

```
SQL> select space_usage_kbytes, occupant_name, occupant_desc
from v$sysaux_occupants
order by 1 desc;
```

参考：
Note 329984.1 Usage and Storage Management of SYSAUX tablespace occupants SM/AWR, SM/ADVISOR, SM/OPTSTAT and SM/OTHER

6.3 本地管理表空间与字典管理表空间

从 Oracle 8i 起，Oracle 引入了本地管理表空间，但是是从 Oracle 9i 起它才变为默认设置。本地管理表空间，也称为 LMT，相对于数据字典管理表空间有一定的优势。

要验证哪个表空间为 Locally Managed（本地管理）或 Dictionary Managed（字典管理），您可以运行以下查询：

```
SQL> select tablespace_name, extent_management
from dba_tablespaces;
```

参考：
Note 93771.1 Introduction to Locally-Managed Tablespaces
Note 105120.1Advantages of Using Locally Managed vs Dictionary Managed Tablespaces

6.4 临时表空间

* 本地管理空间表将临时文件用于临时表空间，而字典管理表空间使用临时类型的表空间。当您运行较早的版本时（早于 Oracle 9i），一定要检查用于存储临时 segment 的表空间类型。默认情况下，所有表空间创建时都为 PERMANENT，因此您应确保专用于临时 segment 的表空间为 TEMPORARY 类型。

```
SQL> select tablespace_name, contents
from dba_tablespaces;

TABLESPACE_NAME CONTENTS
------------------------------ ---------
SYSTEM PERMANENT
USER_DATA PERMANENT
ROLLBACK_DATA PERMANENT
TEMPORARY_DATA TEMPORARY
```

* 确保数据库上的用户都分配了临时类型的表空间。以下查询列出了将永久表空间指定为其默认临时表空间的所有用户。

```
SQL> select u.username, t.tablespace_name
from dba_users u, dba_tablespaces t
where u.temporary_tablespace = t.tablespace_name
and t.contents <> 'TEMPORARY';
```

注意：用户 SYS 和 SYSTEM 会将 SYSTEM 表空间显示为它们的默认临时表空间。该值也可以改变，以防止 SYSTEM 表空间中产生碎片。

```
SQL> alter user SYSTEM temporary tablespace TEMP
```

*临时表空间中分配的空间是可以重复使用的。这是因为基于性能的考虑，以避免由于持续分配和取消分配 extent 和 segment所产生瓶颈。因此，当查看临时表空间中的可用空间时，可能会始终显示为已满的状态。以下查询可以列出关于临时 segment 使用情况的更多有用信息：

这将给出临时表空间的大小：

```
SQL> select tablespace_name, sum(bytes)/1024/1024 mb
from dba_temp_files
group by tablespace_name;
```

这将给出临时表空间的"高水位"（=单次使用的最大值）：

```
SQL> select tablespace_name, sum(bytes_cached)/1024/1024 mb
from v$temp_extent_pool
group by tablespace_name;
```

这将给出当前使用情况：

```
SQL> select ss.tablespace_name,
sum((ss.used_blocks*ts.blocksize))/1024/1024 mb
from gv$sort_segment ss, sys.ts$ ts
where ss.tablespace_name = ts.name
group by ss.tablespace_name;
```

6.5 表空间碎片

表空间碎片过多会对性能产生影响，特别是系统上正进行许多 Full Table Scans（全表扫描）时。碎片的另一个缺点是，当所有可用空间的总和远远超出您请求的空间时，您会得到空间不足的错误消息。

解决碎片的唯一办法是重新创建对象。从 Oracle8i 开始，您可以使用 "alter table .. move" 命令。在 Oracle8i 之前，您可以使用 export/import。

如果您需要对系统表空间消除碎片，则必须重建整个数据库，因为无法删除系统表空间。

参考：
Note 1020182.6 - SCRIPT to detect tablespace fragmentation

Note 1012431.6 - Common causes of Fragmentation
Note 147356.1 - How to Move Tables from One Tablespace to Another.

7. 对象

7.1 Extent 的数量

尽管过度扩展对象的性能影响不是很大，但是很多过度扩展对象积聚起来确实会影响性能。以下查询将列出分配的 extent 超过了指定最小量的所有对象。将 <--minext--> 值改为实际数值，通常，对于 extent 个数超过100或200的对象，可以通过使用更大的 extent 来重建：

```
SQL> select owner, segment_type, segment_name, tablespace_name,
count(blocks), SUM(bytes/1024) "BYTES K", SUM(blocks)
from dba_extents
where owner NOT IN ('SYS','SYSTEM')
group by owner, segment_type, segment_name, tablespace_name
having count(*) > <--minext-->>
order by segment_type, segment_name;
```

7.2 下一个 extent

非常重要的是，segment 可以增长，因此在需要时它们可以分配下一个 extent。如果表空间中没有足够的可用空间，则无法分配下一个 extent，且对象无法增长。以下查询返回了所有无法分配其下一个 extent 的 segment：

```
select s.owner, s.segment_name, s.segment_type,
s.tablespace_name, s.next_extent
from dba_segments s
where s.next_extent > (select MAX(f.bytes)
from dba_free_space f
where f.tablespace_name = s.tablespace_name);
```

请注意，如果表空间中有许多碎片，则此查询可能会返回仍然可以增长的对象。上述查询基于可以表空间中的最大可用块。如果 有许多这样彼此相连的"较小"可用块，则 Oracle 将合并这些块以提供 extent 分配。

因此，需要改写 Note 1020182.6 'SCRIPT to detect tablespace fragmentation' 中的脚本以将每个对象的下一个 extent 与表空间中的"连续"字节（表 space_temp）进行比较。

7.3 索引

基本不需要重建索引，更多时候建议您选择合并索引。有关完整说明，请参阅以下文章：

参考：
Note 989093.1: Index Rebuild, the Need vs the Implications
Note 989186.1: Script to investigate a b-tree index structure

8. AUTO 和 MANUAL undo

从 Oracle 9i 开始，我们引入了一种管理前镜像的新方式。之前，这是通过 RollBack Segment 进行的，或称为 manual undo（手动 undo）。当参数 UNDO_MANAGEMENT 设置为 AUTO 时，使用 automatic undo（自动 undo）。未设置或设置为 MANUAL 时，我们使用"过去"的 rollback segment 机制。尽管当前版本中，两个版本都可用，我们建议使用自动 undo。

8.1 AUTO UNDO

AUM（自动 undo 管理，Automatic Undo Management）几乎不需要配置。您基本上只需要定义将前映像保持可用的时间量。这是通过参数 UNDO_RETENTION 控制的，以秒为单位定义。因此，值 900 表示 15 分钟。

一定要意识到，如果 undo 表空间中存在空间压力时，我们不保证前镜像一定会保留这么长时间。

因此，以下公式可用于计算最佳 undo 表空间大小：

Note 262066.1: How To Size UNDO Tablespace For Automatic Undo Management

从 Oracle 10g 开始，您可以选择使用 GUARANTEE 选项，以确保在定义的 undo_retention 时间之前，undo 信息不会被覆盖。

Note 311615.1: Oracle 10G new feature - Automatic Undo Retention Tuning

8.2 MANUAL UNDO

* 损坏的 rollback segment 会阻止实例打开数据库。仅当知道 rollback segment 名称时，我们才有可能采取更正操作。因此，请在init.ora 中参数 "rollback_segments" 中列出所有的 rollback segment。

* rollback segment 太小或不足可能会对数据库的行为有严重影响。因此，有些问题必须要考虑到。以下查询会显示在线 rollback segment 是否足够，或 rollback segment 是否太小。

```
SQL> select d.segment_name, d.tablespace_name, s.waits, s.shrinks,
s.wraps, s.status
from v$rollstat s, dba_rollback_segs d
```

```
where s.usn = d.segment_id
order by 1;

SEGMENT_NAME TABLESPACE_NAME WAITS SHRINKS WRAPS STATUS
------------ ---------------- ----- --------- --------- --------
RB1 ROLLBACK_DATA 1 0 160 ONLINE
RB2 ROLLBACK_DATA 31 1 149 ONLINE
SYSTEM SYSTEM 0 0 0 ONLINE
```

WAITS 表示在哪些 rollback segment header上有等待。通常，通过添加 rollback segment可以减少这类争用。

如果 SHRINKS 非零，则说明该特定 rollback segment 设置了参数 OPTIMAL，或者 DBA显式执行了命令来缩小该 rollback segment。SHRINKS 表示因为事务扩展 rollback segment 超出了 OPTIMAL 大小而随后缩小 rollback segment 的次数。如果该值过高，则 OPTIMAL 大小的值以及 rollback segment 的总大小应该增加（minextents 或 extent 大小本身可以增加，这主要取决于 WRAPS 列的值）。

WRAPS 列说明 rollback segment 切换到另一 extent 以操作事务的次数。如果该值巨大，则需要增加 rollback segment 的 extent 大小。

参考：
Note 62005.1 Creating, Optimizing, and Understanding Rollback Segments

9. 内存管理

这一章与版本密切相关。根据正在运行的版本，可用的选项会有所不同。一直以来，Oracle 投入了大量的时间和精力来为最终用户更加有效、透明地管理内存。因此，我们建议尽量使用自动功能。

9.1 Oracle 9i 之前的版本

不同的内存组件 (SGA & PGA)的大小需要在数据库启动时定义。这些值是静态的。因此，如果一个内存组件大小过低，则需要重新启动数据库使更改生效。本文档不讨论如何确定不同组件的最优值或最佳值，这已远远超出本文档的范围。但是，这些版本中常被误用的一个参数是 sort_area_size。

init.ora 中的参数 "sort_area_size" 定义了可用于排序的内存量。应当仔细选择该值，因为这是 User Global Area (UGA) 的一部分，因此将分别为每个用户进行分配。

如果有许多用户同时对数据库执行大量排序操作，则系统可能会耗尽内存。

例：您的 sort_area_size 为 1Mb，数据库上同时有 200 个用户。尽管内存是动态分配的，但是可能会分配至 200Mb，从而会导致系统上产生大量交换。

9.2 Oracle 9i

从 Oracle 9i 开始，我们使用以下参数：

workarea_size_policy = [AUTO | MANUAL]
pga_aggregate_target = <value>

这允许您为 PGA 内存定义 1 个 pool，可以在各个会话之间共享。
如果您经常收到 ORA-4030 错误，则表明该值指定的过低.

9.3 Oracle 10g

在 10g 中，引入了 Automatic Shared Memory Management（自动共享内存管理，ASMM）。自动共享内存管理功能，通过将参数 SGA_TARGET 设置为非零值来启用。

此功能的优势在于，您可以在不同组件之间共享内存资源。
资源可以根据需要由 Oracle 自动进行分配和取消分配。

Automatic PGA Memory（自动 PGA 内存）管理仍可通过参数 "workarea_size_policy" 和 "pga_aggregate_target" 进行使用。

9.4 Oracle 11g

在 11g 中，引入 Automatic Memory Management（自动内存管理，AMM）。通过使用两个参数，MEMORY_MAX_TARGET 和 MEMORY_TARGET，可以启用
PGA 和 SGA 的自动调整。

参考：
Note 443746.1 Automatic Memory Management(AMM) on 11g

9.5 Oracle 12c

通过查询数据库的某些视图,你可以在PDB中监控SGA的内存使用状况

Note 1516229.1 How to Monitor SGA Memory on Pluggable Database

10. 日志和跟踪

10.1 告警日志

数据库的告警日志是按时间顺序写入的。始终会添加数据，因此该文件大小会增长到非常巨大。应定期清除或截断该文件，因为大型告警日志会占用不必要的磁盘空间，从而会降低 OS 向文件写入的速度。

11g 之前的版本：

```
SQL> show parameter background_dump_dest

NAME TYPE VALUE
------------------------------ -------- ---------------------------------
background_dump_dest string D:\Oradata\Admin\PROD\Trace\BDump
```

11g 和更高版本：

```
SQL> show parameter diagnostic_dest

NAME TYPE VALUE
------------------------------ -------- ---------------------------------
diagnostic_dest string /oracle/admin/<SID>
```

10.2 Max_dump_file_size

Oracle Server 进程会生成特定错误或冲突的跟踪文件。这些跟踪文件可用于进一步分析问题。init.ora 参数 "max_dump_file_size" 限制了这些跟踪文件的大小。该参数值应指定为操作系统块大小的整数倍。
应确保磁盘空间可以处理指定的最大大小，否则，应更改该值。

```
SQL> show parameter max_dump_file_size

NAME TYPE VALUE
---------------------------------- ------- ----------------------
max_dump_file_size integer 10240
```

10.3 用户和核心转储大小参数

参数 "user_dump_dest" 和 "core_dump_dest" 可以包含许多跟踪信息。
一定要定期清除此目录，因为此目录会占用大量磁盘空间。

注意：从 Oracle 11g 开始，该位置由参数 "diagnostic_dest" 控制。

参考：
Note 564989.1 How To Truncate a Background Trace File Without Bouncing the Database

10.4 审计文件

默认情况下，每次以 SYS 或 SYSDBA 进行的连接，都会记录在操作系统文件中。
存放位置是通过参数 "audit_file_dest" 进行控制的。如果未设置该参数，则位置默认为 $ORACLE_HOME/rdbms/audit。
随着时间推移，该位置可能会包含很多审计信息，从而占用大量空间。

11. 高级健康状况检查

前面的章节已经说明了要检查的基本项目，从而避免常见的数据库警示。在这一部分中，您可以参考几篇文章，了解如何进行更深入的分析和监控。这些文章主要说明数据字典完整性 (Data Dictionary Integrity) 和数据库结构验证。

Oracle 11g 之前的版本：

Note 456468.1 - Identify Data Dictionary Inconsistency
NOTE 136697.1 - "hcheck8i.sql" script to check for known problems in Oracle8i,Oracle9i, and Oracle10g

Oracle 11g

Note 466920.1: 11g New Feature Health monitor

参考

NOTE:456468.1 - Identify Data Dictionary Inconsistency
NOTE:112011.1 - ALERT: RESIZE or AUTOEXTEND can "Over-size" Datafiles and Corrupt the Dictionary
NOTE:1012431.6 - Overview of Database Fragmentation in Oracle 7
NOTE:249664.1 - Pfile vs SPfile
NOTE:115424.1 - How to Rename or Move Datafiles and Logfiles
NOTE:329984.1 - Usage and Storage Management of SYSAUX tablespace occupants SM/AWR, SM/ADVISOR, SM/OPTSTAT and SM/OTHER
NOTE:443746.1 - Automatic Memory Management (AMM) on 11g & 12c

NOTE:62005.1 - Creating, Optimizing, and Understanding Rollback Segments

NOTE:69739.1 - How to Turn Archiving ON and OFF in Oracle RDBMS
NOTE:311615.1 - Oracle 10G new feature - Automatic Undo Retention Tuning
NOTE:262472.1 - 10g: BIGFILE Type Tablespaces Versus SMALLFILE Type
NOTE:122555.1 - Determine How Much Disk Space is Needed for the Archive Files
NOTE:136697.1 - hcheck.sql - Script to Check for Known Problems in Oracle8i, Oracle9i, Oracle10g, Oracle 11g and Oracle 12c
NOTE:147356.1 - How to Move Tables from One Tablespace to Another.
NOTE:93771.1 - Introduction to Locally-Managed Tablespaces
NOTE:1020182.6 - Script to Detect Tablespace Fragmentation
NOTE:102995.1 - Maintenance of Online Redo Log Groups and Members
NOTE:105120.1 - Advantages of Using Locally Managed vs Dictionary Managed Tablespaces
NOTE:564989.1 - How To Truncate a Background Trace File Without Bouncing the Database
NOTE:262066.1 - How To Size UNDO Tablespace For Automatic Undo Management

Didn't find what you are looking for?