**如何格式化不属于任何段的损坏块 (Doc ID 1526163.1)**

|  |  |
| --- | --- |
| [To Bottom](https://support.oracle.com/epmos/faces/DocumentDisplay?_afrLoop=254644673164587&parent=SrDetailText&sourceId=3-11146276381&id=1526163.1&_afrWindowMode=0&_adf.ctrl-state=qnhjjen5x_508) |  |



|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **文档内容**   |  |  | | --- | --- | |  | [症状](https://support.oracle.com/epmos/faces/DocumentDisplay?_afrLoop=254644673164587&parent=SrDetailText&sourceId=3-11146276381&id=1526163.1&_afrWindowMode=0&_adf.ctrl-state=qnhjjen5x_508#SYMPTOM) | |  | [原因](https://support.oracle.com/epmos/faces/DocumentDisplay?_afrLoop=254644673164587&parent=SrDetailText&sourceId=3-11146276381&id=1526163.1&_afrWindowMode=0&_adf.ctrl-state=qnhjjen5x_508#CAUSE) |  |  |  |  | | --- | --- | --- | |  | [解决方案](https://support.oracle.com/epmos/faces/DocumentDisplay?_afrLoop=254644673164587&parent=SrDetailText&sourceId=3-11146276381&id=1526163.1&_afrWindowMode=0&_adf.ctrl-state=qnhjjen5x_508#FIX) | | |  | [参考](https://support.oracle.com/epmos/faces/DocumentDisplay?_afrLoop=254644673164587&parent=SrDetailText&sourceId=3-11146276381&id=1526163.1&_afrWindowMode=0&_adf.ctrl-state=qnhjjen5x_508#REF) |       **适用于:**  Oracle Database - Enterprise Edition - 版本 8.1.7.4 到 11.2.0.1.0 [发行版 8.1.7 到 11.2]  本文档所含信息适用于所有平台  **症状**  1. Rman 备份失败，显示 ORA-19566 错误，且被报告的坏块不属于任何对象  2. Dbverify 显示存在坏块  3. 坏块不属于任何对象  **原因**  在重新使用和重新格式化坏块之前，RMAN 和 DBV 仍会一直报告坏块。  **解决方案**  免责声明：本说明中给出的步骤并不总是保证问题解决。  解决此类问题的一个可能的方法如下。请注意，它不能保证问题解决，但它已知解决了几种案例的问题。如果在某个数据文件中报告有很多块损坏，请在第六步提示输入 blocknumber 中输入已经报告的最高的坏块号码。  当 一个对象被重新创建，已经分配给它（甚至是损坏的块）的块将变成空闲空间（free space）。他们在那里等待重新分配到一个需要额外空间的对象。一旦它们被重新分配到用于一个对象上新的extent，只有当任何DML操作使用到这些 块（即使损坏的块，处在空闲空间，现在需要被分配），他们将在DML操作更改、使用这些块之前，被重新格式化掉。    需要注意的是extent的简单的分配不格式化块。第7步分配的extent和第8步执行DML操作，使用在第七步中分配的块，这样的坏块才被重用和格式化。    在本文档中，我们尝试手动重新格式化坏块。    **第 1 步 - 确定损坏的数据文件**  损坏可以被报告在应用层，如DBV和RMAN，或alert.log。  例如，您可以在您的RMAN备份过程中获得以下信息：  RMAN-03009: failure of backup command on nm4501 channel at 04/29/2005 09:44:41  ORA-19566: exceeded limit of 0 corrupt blocks for file E:\xxxx\test.ORA.    坏块位于文件 E:\xxxx\test.ORA 中。      **第 2 步 - 在受影响的数据文件上运行 DBV/Rman 验证并检查坏块**  在报告坏块的数据文件上运行 dbverify。  # dbv userid={system/password} file={full path filename} logfile={output filename}  检查{输出文件名}结果  示例输出：  DBVERIFY: Release 9.2.0.3.0 - Production on Thu Aug 25 11:15:54 2005  Copyright (c) 1982, 2002, Oracle Corporation. All rights reserved.    DBVERIFY - Verification starting : FILE = E:\xxxx\test.ORA  **Page 48740 is marked corrupt** \*\*\*  Corrupt block relative dba: 0x01c0be64 (file 7, block 48740)  Bad check value found during dbv:  Data in bad block -  type: 0 format: 2 rdba: 0x0000be64  last change scn: 0x0000.00000000 seq: 0x1 flg: 0x05  consistency value in tail: 0x00000001  check value in block header: 0xb964, computed block checksum: 0x2a5a  spare1: 0x0, spare2: 0x0, spare3: 0x0  \*\*\*  DBVERIFY - Verification complete    Total Pages Examined : 64000  Total Pages Processed (Data) : 0  Total Pages Failing (Data) : 0  Total Pages Processed (Index): 1751  Total Pages Failing (Index): 0  Total Pages Processed (Other): 45  Total Pages Processed (Seg) : 0  Total Pages Failing (Seg) : 0  Total Pages Empty : 62203  Total Pages Marked Corrupt : 1  请注意，在数据文件 7 中，报告 **块 48740** 损坏。    或者 在RMAN中  对于整个数据库  Rman> backup validate check logical database ;  For specific datafile  Rman> backup validate check logical datafile <fileno> ;  Once done query    SQL>Select \* from v$database\_block\_corruption ;   \*\*如果在第 2 步中 v$database\_block\_corruption 报告的块数量很多，接下来最好使用第 4 步而不是第 3 步。    **第 3 步 - 检查块是否是某个对象的一部分 - 适用于损坏的块号小的情况**  查询 dba\_extents，确认坏块是否属于任何对象。  SQL> select segment\_name, segment\_type, owner         from dba\_extents        where file\_id = <Absolute file number>          and <corrupted block number> between block\_id              and block\_id + blocks -1;    如果块不属于任何对象，查询dba\_free\_space 确认坏块是否属于数据文件的可用空间。  SQL> Select \* from dba\_free\_space where file\_id= <Absolute file number>       and <corrupted block number> between block\_id and block\_id + blocks -1;    **第 4 步 - 查找受影响的块并验证其是否属于任何段的一种比较好的方法是使用 RMAN，这种方法既好用又便捷。**  如果在第 2 步中已经运行了 rman 验证，请直接转到下面给出的 sqlplus 脚本，以确认对象。    $ rman target / nocatalog  or  $ rman target sys/ nocatalog    run {  allocate channel d1 type disk;  allocate channel d2 type disk;  ------------------------------------------------------------------------  --  multiple channels may be allocated for parallelizing purposes  --  depends: RMAN - Min ( MAXOPENFILES , FILESPERSET )  --  Defaults: MAXOPENFILES =8, FILESPERSET =64  ------------------------------------------------------------------------  allocate channel dn type disk;  backup check logical validate database;    注意：如果RDBMS是小于11g并且处于非归档模式下，则数据库必须处于 mounted 模式      \*\*\* 在进行进一步操作之前，\*必须\*运行并完成 RMAN 命令“backup check logical validate database”。  \*\*\* 此命令完成（基于文件）后将会填充“v$database\_block\_corruption”视图。  \*\*\* 如果未完成，在接下来的步骤中您就有可能得到无效/不完整的信息。    **运行以下 sql 查询，以确定块是位于可用空间中还是已占用空间中**  set lines 200 pages 10000  col segment\_name format a30  SELECT e.owner, e.segment\_type, e.segment\_name, e.partition\_name, c.file#  , greatest(e.block\_id, c.block#) corr\_start\_block#  , least(e.block\_id+e.blocks-1, c.block#+c.blocks-1) corr\_end\_block#  , least(e.block\_id+e.blocks-1, c.block#+c.blocks-1)  - greatest(e.block\_id, c.block#) + 1 blocks\_corrupted  , null description  FROM dba\_extents e, v$database\_block\_corruption c  WHERE e.file\_id = c.file#  AND e.block\_id <= c.block# + c.blocks - 1  AND e.block\_id + e.blocks - 1 >= c.block#  UNION  SELECT s.owner, s.segment\_type, s.segment\_name, s.partition\_name, c.file#  , header\_block corr\_start\_block#  , header\_block corr\_end\_block#  , 1 blocks\_corrupted  , 'Segment Header' description  FROM dba\_segments s, v$database\_block\_corruption c  WHERE s.header\_file = c.file#  AND s.header\_block between c.block# and c.block# + c.blocks - 1  UNION  SELECT null owner, null segment\_type, null segment\_name, null partition\_name, c.file#  , greatest(f.block\_id, c.block#) corr\_start\_block#  , least(f.block\_id+f.blocks-1, c.block#+c.blocks-1) corr\_end\_block#  , least(f.block\_id+f.blocks-1, c.block#+c.blocks-1)  - greatest(f.block\_id, c.block#) + 1 blocks\_corrupted  , 'Free Block' description  FROM dba\_free\_space f, v$database\_block\_corruption c  WHERE f.file\_id = c.file#  AND f.block\_id <= c.block# + c.blocks - 1  AND f.block\_id + f.blocks - 1 >= c.block#  ORDER BY file#, corr\_start\_block#;      **第 5 步 - 以非 SYS 或 SYSTEM（用户）的用户身份创建一个虚拟表**  SQL> connect scott/password    在包含出现坏块的数据文件的表空间中创建虚拟表，并使用 nologging 选项，以防止生成 redo记录：  SQL> create table s (         n number,         c varchar2(4000)       ) nologging tablespace <tablespace name having the corrupt block> ;    不同的存储参数可以被用来适应特定的环境。  我们使用 PCTFREE 99以加快该块的格式化    确认表是被创建在正确的表空间中，通过查询 user\_segments：  SQL> select segment\_name,tablespace\_name from user\_segments        where segment\_name='S' ;    **请注意，在11gR2中，由于延迟段创建概念，从上面提到的 user\_segments 查询可能不会报告。在这种情况下，查询 USER\_TABLES**    SQL> select segment\_name,tablespace\_name from user\_tables where segment\_name='S' ;    **第 6 步 - 在虚拟表上创建触发器，一旦重新使用坏块，该触发器便会引发异常**  以 sysdba 身份连接，并创建以下触发器：  请注意，在出现文件号提示时，输入相关文件号（v$datafile 中的 rfile# 值）  CREATE OR REPLACE TRIGGER corrupt\_trigger    AFTER INSERT ON scott.s    REFERENCING OLD AS p\_old NEW AS new\_p    FOR EACH ROW  DECLARE    corrupt EXCEPTION;  BEGIN    IF (dbms\_rowid.rowid\_block\_number(:new\_p.rowid)=&blocknumber)   and (dbms\_rowid.rowid\_relative\_fno(:new\_p.rowid)=&filenumber) THEN       RAISE corrupt;    END IF;  EXCEPTION    WHEN corrupt THEN       RAISE\_APPLICATION\_ERROR(-20000, 'Corrupt block has been formatted');  END;  /  出现块编号提示时，输入坏块的块编号。  出现文件号提示时，输入损坏的数据文件的相关文件号（v$datafile 中的 rfile# 值）。  **第 7 步 - 为受影响的数据文件中的表分配空间。**  请注意：  1）如果这是一个ASSM表空间，你可能需要重复此步骤数次。也就是说，创建多个表和分配多个 extent。  并周期性地查看 dba\_extents，以确保空闲空间现在分配给一个虚拟表。  这是因为ASSM将自动确定下一个 extent 的大小    2）建议确保数据文件 AUTOEXTEND 置为 OFF，以防止其增长。  首先通过查询 dba\_free\_space 查找 extent 大小  SQL> Select BYTES from dba\_free\_space where file\_id=<file no> and <corrupt block no> between block\_id and block\_id + blocks -1;  BYTES  ---------------- ---------- ---------- ---------- ---------- ------------   65536  在本例中，它的大小是 64K。因此，按照以下方法分配 extent：  SQL> alter table scott.s  allocate extent (DATAFILE 'E:\xxxx\test.ORA' SIZE 64K);    如果在这个数据文件中有多个64K的空闲 extent，则可能需要使用这个循环：  BEGIN  for i in 1..1000000 loop  EXECUTE IMMEDIATE 'alter table scott.s allocate extent (DATAFILE '||'''E:\xxxx\test.ORA''' ||'SIZE 64K) ';  end loop;  end ;  /    继续分配空间直到坏块成为 scott.s 的一部分 — 使用以下查询进行检查：  SQL> select segment\_name, segment\_type, owner         from dba\_extents        where file\_id = <Absolute file number>          and <corrupt block number> between block\_id              and block\_id + blocks -1 ;      **第 8 步 - 向虚拟表中插入数据以格式化块**  示例代码（取决于表空间的大小，循环的次数可能发生变化）：  BEGIN  FOR i IN 1..1000000000 LOOP  INSERT /\*+ APPEND \*/ INTO scott.s select i, lpad('REFORMAT',3092, 'R') from dual;  commit ;  END LOOP;  END;     或者  BEGIN    FOR i IN 1..1000000000 LOOP      INSERT INTO scott.s VALUES(i,'x');    END LOOP;  END;  /   或使用以下包含 2 个循环的代码：  Begin    FOR i IN 1..1000000000 loop      for j IN 1..1000 loop        Insert into scott.s VALUES(i,'x');      end loop;      commit;    END LOOP;  END;    每向表中插入一行就会触发触发器，且一旦向坏块中插入第一行数据，就会产成 ORA-20000 异常。  **第 9 步 - 通过运行 DBV 和 Rman 备份确定数据文件中的坏块情况**   运行 dbverify 或 RMAN 在损坏的数据文件（或整个数据库）上再次验证。它不会显示块损坏。     确保你做两次手动日志切换或检查点，使得在内存中的信息写入到磁盘。   RMAN 备份不会报告此块上的任何错误。    运行真实的备份操作之前，你可以在数据文件上重新运行 RMAN validate 命令，并检查 v$database\_block\_corruption 不再显示块被标记为损坏。  **对于数据库版本 <=10gR2**   Rman> Backup validate check logical datafile <fileno>,<fileno> ;  **对于数据库版本>=11gR1**   Rman> Backup validate check logical datafile <fileno> ;   **或**   Rman> validate datafile <fileno> block <blockno reported corrupt>, <blockno reported corrupt>  ;   一旦完成     SQL> SELECT \* FROM V $ DATABASE\_BLOCK\_CORRUPTION;    **第 10 步 - 删除第 4 步中创建的虚拟表**    SQL> DROP TABLE scott.s ;    如果版本为 10gr1 及以上，同时使用purge选项以清空回收站  **第 11 步 – 执行手动日志切换和检查点**    执行两次日志切换和检查点，以便将在内存中格式化的块写入到磁盘并使 dbverify 不再报告错误  SQL>Alter system switch logfile ;  --> Do this couple of time    SQL>Alter system checkpoint ;    **第 12 步 - 删除第 6 步中创建的触发器**  SQL> DROP triggercorrupt\_trigger ;    重要通知：  [Bug 7381632](https://support.oracle.com/epmos/faces/BugDisplay?parent=DOCUMENT&sourceId=1526163.1&id=7381632) - ORA-1578 Free corrupt blocks may not be reformatted when Flashback is enabled    =======================================  **Use RMAN to format corrupt data block which is not part of any object (Doc ID 1459778.1)**    If RMAN reads it, it will fail with error ORA-19566 due to the corrupt block which means we cannot use this procedure. Instead we'd have to use below document:  Document [336133.1](https://support.oracle.com/epmos/faces/DocumentDisplay?parent=DOCUMENT&sourceId=1459778.1&id=336133.1) How to Format Corrupted Block Not Part of Any Segment  or  中文  **如何格式化不属于任何段的损坏块 (Doc ID 1526163.1)** |