ASSM管理（BMB段管理）的内部机理 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/143/2021_2022_ASSM_E7_AE_A1_E7_90_86_c102_143297.htm 在920以前，表的剩余空间的管理与分配都是由链接列表freelist来完成的，因为freelist存在串行的问题因此容易引起往往容易引起段头的争用与空间的浪费（其实这一点并不明显），最主要的还是因为需要DBA花费大量的精力去管理这些争用并监控表的空间利用。 自动段空间管理（ASSM），它首次出现在Oracle920里。有了ASSM，链接列表freelist被位图所取代，它是一个二进制的数组，能够迅速有效地管理存储扩展和剩余区块（free block），因此能够改善分段存储本质，ASSM表空间上创建的段还有另外一个称呼叫Bitmap Managed Segments（BMB 段）。 让我们看看位图freelist是如何实现的。我会从使用区段空间管理自动参数创建tablespace开始： create tablespace demo datafile /ora01/oem/demo01.dbf size 5m EXTENT MANAGEMENT LOCAL -- Turn on LMT SEGMENT SPACE MANAGEMENT AUTO -- Turn on ASSM. 一旦你定义好了tablespace，那么表和索引就能够使用各种方法很容易地被移动到新的tablespace里，带有ASSM的本地管理tablespace会略掉任何为PCTUSED、NEXT和FREELISTS所指定的值。 当表格或者索引被分配到这个tablespace以后，用于独立对象的PCTUSED的值会被忽略，而Oracle9i会使用位图数组来自动地管理tablespace里表格和索引的freelist。对于在LMT的tablespace内部创建的表格和索引而言，这个NEXT扩展子句是过时的，因为由本地管理的tablespace会管理它们。但是，INITIAL参数仍然是需要的，

因为Oracle不可能提前知道初始表格加载的大小。对于ASSM而言，INITIAL最小的值是三个块。 新的管理机制用位图来跟踪或管理每个分配到对象的块，每个块有多少剩余空间根据位图的状态来确定，如>75%,50%-75%,25%-50%和 使用ASSM的一个巨大优势是，位图freelist肯定能够减轻缓冲区忙等待（buffer busy wait）的负担，这个问题在Oracle9i以前的版本里曾是一个严重的问题 在没有多个freelist的时候，每个Oracle表格和索引在表格的头部都曾有一个数据块，用来管理对象所使用的剩余区块，并为任何SQL插入声明所创建的新数据行提供数据块。当数据缓冲内的数据块由于被另一个DML事务处理锁定而无法使用的时候，缓冲区忙等待就会发生。当你需要将多个任务插入到同一个表格里的时候，这些任务就被强制等待，而同时Oracle会在同时分派剩余的区块，一次一个。 有了ASSM之后，Oracle宣称显著地提高了DML并发操作的性能，因为（同一个）位图的不同部分可以被同时使用，这样就消除了寻找剩余空间的串行化。根据Oracle的测试结果，使用位图freelist会消除所有分段头部（对资源）的争夺，还能获得超快的并发插入操作 尽管ASSM显示出了令人激动的特性并能够简化Oracle DBA的工作，但是Oracle9i的位图分段管理还是有一些局限性的： 一旦DBA被分配之后，它就无法控制tablespace内部的独立表格和索引的存储行为。大型对象不能够使用ASSM，而且必须为包含有LOB数据类型的表格创建分离的tablespace。 你不能够使用ASSM创建临时的tablespace。这是由排序时临时分段的短暂特性所决定的。只有本地管理的tablespace才能够使用位图分段管理。 使用超高容量的DML（例如INSERT、UPDATE和DELETE等）的时

候可能会出现性能上的问题。 1、我们先创建一个本地管理的表空间，采用段自动管理方式 create tablespace demo datafile /ora01/oem/demo01.dbf size 50m EXTENT MANAGEMENT LOCAL --一定是本地管理 SEGMENT SPACE MANAGEMENT AUTO. --ASSM管理的标志 2、创建同样一个表 SQL> create table demotab (x number) tablespace demo storage (initial 1000K). Table created 我们指定初试区间大小是1000K SQL> 0select t.table_name,t.initial_extent,t.next_extent,t.pct_free,t.pct_used from user_tables t where t.table_name = DEMOTAB. TABLE_NAME INITIAL_EXTENT NEXT_EXTENT PCT_FREE PCT_USED ------------------------------ -------------- ---------- ---------- ---------- DEMOTAB 1024000 10 可以看到，NEXT_EXTENT与PCT_USED都为空。 3、执行该过程，检查表的初始状态 SQL> exec show_space(demotab). Total Blocks.....128 Total Bytes.....1048576 Unused Blocks.....125 Unused Bytes.....1024000 Last Used Ext FileId.....7 Last Used Ext BlockId.....8 Last Used Block.....3 从这里我们能看到一些该表的特性，其中最引人注意的就是表头了，占用了三个块的大小（128-125） 另外一个注意的地方就是该表从第8个块开始，但是实际上这里是错误的，应当是从第9个块开始，文件头占用了64K的空间等于8个块。 我们从dba_extent中也能看到这样的信息，实际上是从第9个块开始的。 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com