基于 Oracle 的邮件系统的存储优化

The Storage Optimization of A Mail System Based on Oracle

栾　杰　（石油化工科学研究院 100083）

摘要：本文分析了 Oracle 中大对象（LOB）的存储机理，探讨了 LOB 字段存储参数的正确配置，提供了多种监控方法，对基于 Oracle 的邮件系统进行了优化，使数据库存储空间的分配更加合理，提高了系统性能。

关键词：LOB　数据库碎片　行链接　优化

1　前言

本文以某邮件系统为例，讲述了 LOB 字段的存储机理、创建时需设定的参数及其对系统性能的影响，分析了实际应用中可能出现的若干问题，如数据库碎片、行链接等。本文还探讨了如何在创建表时正确配置 LOB 字段的存储参数，以及在后期维护过程中，如何监控包含 LOB 字段表的使用情况，为有效管理数据库的存储空间及性能调优提供依据。

对于邮件系统而言，为每封邮件分配的空间的大小相差很大，有的只需几十个字节，有的需要几千字节，如果附件很大，也可能达到几兆字节以上。同时，由于企业内部邮箱大小有限制，因此需要经常删除邮件，但已分配给邮件的空间却没有完全收回，这时就会造成大量碎片，虽然数据文件的物理尺寸不断增大，但其空间利用率却逐渐降低。另一方面，在部署应用程序初期，并不能准确地估计未来数据的增长情况，因此设置的一些参数可能不合理。虽然采用本地管理表空间的方式基本上可以消除一部分数据库碎片，但对于大对象数据而言，由于 LOB 字段存储的特殊性，仍然会产生行链接。数据库碎片和行链接都会增加数据库读的次数和磁盘查找时间，从而引起数据库性能下降。因此，经常监控 LOB 字段的存储情况并采取相应措施十分重要。

2　大对象存储机理

由于 LOB 数据类型可以存储小于 4KB 的数据，也可以存储远远大于 4KB 的数据，因此在存储方面有其独特之处。

Oracle 中 LOB 数据类型包括 BLOB、CLOB、NCLOB 以及 BFILE，其中 BFILE 为外部 LOB 类型，是指数据存储在数据库以外的操作系统的文件中，而 BLOB、CLOB 以及 NCLOB 为内部 LOB 数据类型。本文所涉及的邮件系统的 LOB 数据主要是邮件及其附件，采用的是 BLOB 数据类型，即二进制 LOB 数据类型，因此本文主要研究 BLOB 数据类型的存储。

创建包含 LOB 字段表时，Oracle 将同时创建两个段来容纳指定的列。这两个段分别为 LOBSEGMENT 和 LOBINDEX 类型。如果不指定存储参数，则将在与该表相同的表空间中按照默认的存储参数来创建这两个段，段的名称也是自动生成的。（注：Oracle9i 以后，LOB 索引的存储参数将被忽略，其名称是自动生成的，并与 LOB 字段存储在相同的表空间中。）例如，以下语句在创建表 mail 时为其指定了存储参数：

```
CREATE TABLE mail (pkey NOT NULL NUMBER, body BLOB)
    LOB (body)
    STORE AS lob_seg (
    TABLESPACE data
    CHUNK 8192
    ENABLE STORAGE IN ROW
    )
```

通过以下查询，可以获得由 Oracle 自动生成的 LOBSEGMENT 和 LOBINDEX 的信息：

```
SELECT segment_name,segment_type FROM user_extents;
```

如表 1 所示：

表 1　Oracle 为表 mail 创建的段

段的名称	段的类型
Mail	TABLE
SYS_IL0000025039C00010 $ $	LOBINDEX
SYS_LOB0000025039C00010 $ $	LOBSEGMENT

在创建包含 LOB 字段的表时需要设置的与存储有关的选项如下：

（1）CHUNK。CHUNK 是分配给 LOB 段的最小单元，通常是 Oracle 数据块的整数倍，其最小值为数据库块的大小，最大值为 32KB。数据库在读取或写入 LOB 数据时，也是以 CHUNK 为单位，而且一个 CHUNK 中只允许存放一行中的数据。例如，将 CHUNK 的值设置为 32KB，如果插入 33KB 的数据，则将占据 64KB 的空间，其中的 31KB 将会浪费掉。

设置合理的 CHUNK 值，不仅可以避免存储空间的浪费，而且由于 Oracle 对 LOB 字段的 I/O 操作也是以 CHUNK 为单位，因此可以改善 I/O 性能。CHUNK 的值应根据实际应用中 LOB 字段经常使用的平均长度来调节。

（2）In – Line 与 Out – Of – Line 存储。如果 LOB 字段与行中其他列的数据存储在一起，则称为 In – Line 存储。此时，LOB 字段的长度小于 3964 个字节，存储在该表的段内，其对应的 LOBSEGMENT 和 LOBINDEX 为空。反之，如果 LOB 字段与行中其他列的数据存储在不同的位置，则称为 Out – Of – Line 存储。

（3）ENABLE STORAGE IN ROW 和 DISABLE STORAGE IN ROW。前者允许小于 3964 个字节的 LOB 字段存储在表的段内，当 LOB 字段大于 3964 个字节时自动存储在 LOB 段中；而后者是指不管 LOB 字段的长度如何，都不存储在表的段内，而是仅在表的行内存储 20 个字节的 LOB 定位符以及其他非 LOB 类型的数据，其数据存储在类型为 LOBSEGMENT 的段内。

如果设置为 DISABLE STORAGE IN ROW，Oracle 将首先读取 LOB 定位符，然后是 LOB 索引，最后才读取 LOB 段。因此，要进行三次 I/O 操作。但是对于大于 3964 字节的 LOB 数据来说，这是无法避免的。

In – Line 存储时，小于 Oracle 数据块的 LOB 数据会产生重做记录和回滚数据，因为它们是作为一般数据来处理的，而大于 Oracle 数据块的 LOB 数据则会发生行链接现象。Out – Of – Line 存储时，仅 LOB 定位符和对 LOBINDEX 的更改产生回滚数据，而存储在 LOBSEGMENT 段内的数据不会产生回滚数据，不过此时将产生大量重做记录，因为整个 CHUNK 中的所有数据都会产生重做记录[1]。例如，设置为 DISABLE STORAGE IN ROW CHUNK 8K，即使 LOB 字段仅更改了几个字节，这 8KB 的数据都会产生重做记录。

（4）LOB 段和 LOB 索引可以单独存储在与表所在的表空间不同的表空间中，从而可以减少 I/O 竞争。

LOB 字段的内部存储结构如图 1 所示，其中 LOB 定位符是一个指向 LOB 字段实际存放位置的指针，一般为 20 个字节。LOB 节点是用于记录属于某个 LOB 项的所有数据块的结构。

3　动态监控 LOB 字段的存储状态

为有效管理包含 LOB 字段的表的存储空间，需要动态监控其存储状态，如 LOB 字段的平均长度、该表每行的平均长度、Out – Of – Line 存储的行数、发生行链接的行数以及表空间的碎片情况。以下以某邮件系统为例，介绍了如何动态监控 LOB 字段存储状态的方法，并对结果进行了分析。

工作环境：Oracle 的版本为 8.1.7.0.0，服务器为 IBM/AIX RISC System/6000。Oracle 数据块的大小设置为 8KB，CHUNK 的值也设置为 8KB。

3.1　平均长度（以字节为单位）

通过以下语句可以监控 LOB 字段的平均长度：

```
SQL > SELECT AVG(dbms_lob.getlength(body)) avg_lob_len FROM mail;
Avg_lob_len
206287.546
```

通过以下语句可以获得表 mail 的存储信息：

```
SQL > ANALYZE TABLE mail COMPUTE STATISTICS;
SQL > SELECT avg_row_len, num_rows, blocks, empty_blocks FROM user_tables WHERE table_name = 'MAIL';
AVG_ROW_LEN NUM_ROWS BLOCKS EMPTY_BLOCKS
1009 77624 17324 6970
```

其中，AVG_ROW_LEN 为 In – Line 存储的数据的平均长度，NUM_ROWS 为该表的总行数。由此可以看

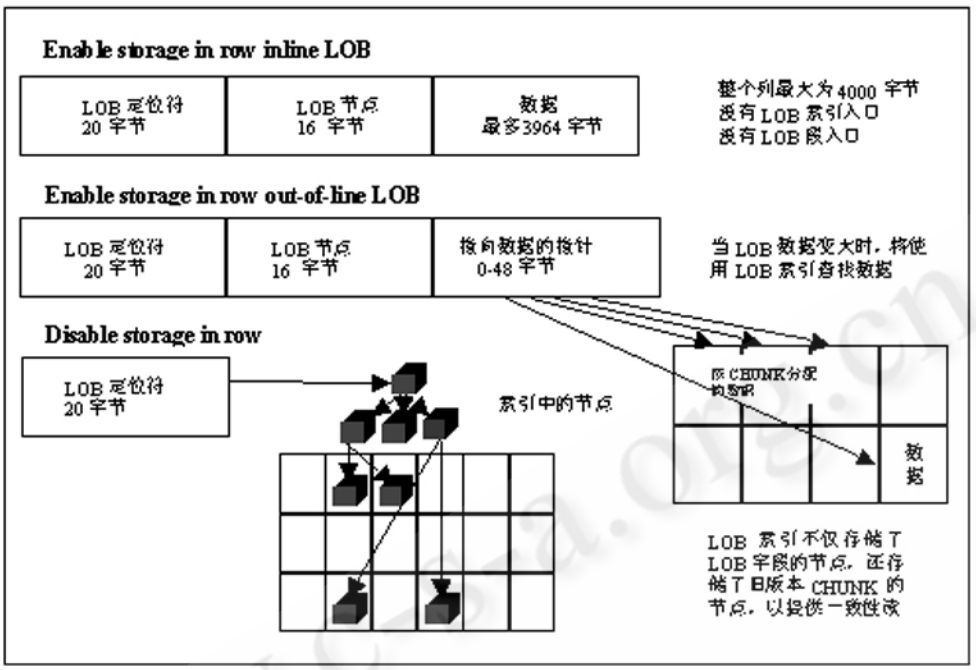

图 1　LOB 字段的内部存储结构

出，In – Line 存储的数据的平均长度并不大，仅 1KB 左右，而数据块的大小为 8KB，足以容纳这些数据。另外，LOB 字段的平均长度为 200KB 左右，而 CHUNK 值为 8KB，设置得较小。

我们也可以使用 Show_Space 存储过程来查看表及其 LOB 字段的存储情况。Thomas Kyte【2】对 DBMS_SPACE 包进行了包装，编写了 Show_Space 存储过程，可以用来显示表的存储信息。

```
SQL > EXEC Show_Space('MAIL');
Total Blocks. . . . . 24295
Unused Blocks. . . . . 6970
```

从输出结果中可以看出，Unused Blocks 与查询表 user_tables 得出的 EMPTY_BLOCKS 是一致的，表示位于高水位标志（High Water Mark）以上的数据块数，即从未使用过的数据块数。Total Blocks 代表该表中曾经使用过的数据库块的数目，即高水位标志 HWM。原则上，HWM 随着数据量的增加而增大，却不会随着数据量的减少而降低，即使将表中所有数据删除 HWM 也不会降低，因此就会形成碎片。由于在全表扫描时，Oracle 将读取 HWM 以下的所有数据块，包括空的数据块，因此增加了额外的 I/O 负担。

以下语句可以获得 mail 表中 LOB 字段的存储信息：

```
SQL > SELECT table_name, segment_name FROM user_lobs;
TABLE_NAME SEGMENT_NAME
MAIL SYS_LOB0000025039C00010$$
SQL > EXEC Show_Space('SYS_LOB0000025039C00010$$','USER_NAME','LOB');
Total Bytes. . . . . 25801678848
SQL > SELECT SUM(dbms_lob.getlength(body)) actual_total_len FROM mail;
Actual_total_len. . . . . 1.6107E+10
```

从以上数据可以看出，LOB 字段的实际长度（Actual_total_len）总共约为 16GB，而占据的空间（Total Bytes）约为 26GB。由此可见该表的存储空间分配不合理，利用率很低，应进行必要调整。

3.2 Out – Of – Line 存储的行数和平均长度

在创建 mail 表时，由于设置为 ENABLE STORAGE IN ROW，因此有必要查看 Out – Of – Line 存储的行数及其平均长度，以确定存储参数设置是否合理。通过以下语句可以获得 Out – Of – Line 存储的行数及其平均长度（以字节为单位）：

```
SQL > SELECT count(*) FROM mail WHERE dbms_lob.getlength(body) >3964;
COUNT(*)
44757
SQL > SELECT AVG(dbms_lob.getlength(body))
```

avg_lob_len FROM mail WHERE dbms_lob. getlength (body) >3964;

Avg_lob_len

354920.285

大于 3964 字节的数据就会存储在专门的 LOB 段中,也就是 Out – Of – Line 存储。从 3.1 节的查询中已知该表的总行数(NUM_ROWS),由此得出 Out – Of – Line 存储的行所占的比例大约 60% 。由于 Out – Of – Line 存储的行数很多,并且其平均长度约为 355KB,因此可以将其设置为 DISABLE STORAGE IN ROW。

3.3 行链接

当一行数据的长度超过数据块的大小时,就会发生行链接,此时产生的数据库碎片属于表的碎片【3】。对于包含 LOB 字段表,出现行链接的现象是很难避免的,但是可以进行监控,在必要时重建该表。我们可以通过以下查询语句来获得行链接的数目:

SQL > Analyze table mail compute statistics;

SQL > SELECT chain_cnt FROM user_tables WHERE table_name = 'MAIL';

CHAIN_CNT

7159

由此可见该表存在大量行链接,Oracle 需要扫描多个数据块来检索一行的数据,因此 I/O 性能就会降低。

3.4 表空间的碎片

我们知道,段是由范围组成。由于自由范围碎片的存在,也会造成段内碎片的产生。FSFI(自由空间碎片索引)可以直观地说明表空间中自由范围的碎片情况。

通过以下语句可以获得表空间的 FSFI 值:

SQL > SELECT tablespace _ name, sqrt (max (blocks) / sum (blocks)) *

(100/sqrt(sqrt(count(blocks)))) FSFI FROM dba _free_space

GROUP BY tablespace_name ORDER BY 1;

TABLESPACE_NAME FSFI

DATA 16

FSFI 的值一般介于 0 到 100 之间,通常情况下,如果该值低于 30,说明自由空间的可用性较差,必须对该表空间进行调整。从以上输出结果中可知,LOB 字段所在的表空间 DATA 的 FSFI 为 16,存在大量碎片,必须对其进行调整。

4 优化策略

Oracle 数据库的性能调优是一项系统工程,涉及的方面很多,其中磁盘的 I/O 对数据库的性能有较大影响。对于包含 LOB 字段的表来说,如何合理设置 LOB 字段的存储参数,从而减少磁盘 I/O 次数,提高数据库响应时间,这一点至关重要。另一方面,由于表空间碎片和行链接都会导致在数据访问时需要更多的磁盘 I/O,因此也必须对此进行调整。

4.1 合理设置存储参数

重建包含 LOB 字段的表并设置合适的 CHUNK 值,同时根据实际的 LOB 字段的长度以及 In – Line 存储和 Out – Of – Line 存储的比例来确定是设置为 ENABLE STORAGE IN ROW 还是设置为 DISABLE STORAGE IN ROW。如果大部分 LOB 字段小于 3964 字节,则应选择 ENABLE STORAGE IN ROW。如果大部分 LOB 字段远大于 3964 字节,则应选择 DISABLE STORAGE IN ROW,并尽量设置大一点的 CHUNK 值,这样 Oracle 每次 I/O 操作读取的数据就多一些,所需的整体 I/O 次数就会少一些,从而可以提高整体性能。在创建表时,应将 LOB 字段单独存储在一个表空间中,并将该表空间的数据块设置大一点,以便提高性能【4】。

4.2 消除表空间碎片和行链接

如果采用字典管理表空间的方式,则可以先用 EXPORT 实用程序将整个表导出,再使用 TRUNCATE 命令删除表中的所有行,最后再用 IMPORT 实用程序导入。

如果采用本地管理表空间的方式,此时使用 EXP/IMP 效果并不明显,但可以通过删除表,然后重建表并重新加载数据来回收空间;或通过使用 ALTER TABLE MOVE 命令把表移动到一个不同的表空间中来回收空间。这两种处理方式都必须脱机进行。例如,可以使用以下语句将该表转移到其他表空间:

ALTER TABLE mail MOVE tablespace new _ tablespace;

不过这需要重建索引,否则该表的索引将不可用,

(下转第 79 页)

(上接第 75 页)

因此在转移前应确定该表所有的索引。而且,对于较大的表或者数据量很大的情况,在转移时将消耗大量资源,并且时间很长,因此应在系统空闲时执行此操作。Oracle 也可以将由字典管理的表空间转换为由本地管理的表空间,即通过 DBMS_SPACE_ADMIN 包的 Tablespace_Migrate_To_Local 过程来实现。但是这种转换并不是完全转换,而且并没有消除碎片,所以建议还是使用 ALTER TABLE MOVE 命令来消除浪费的空间。

5 结论

在优化包含 LOB 字段的表时,难点主要是 LOB 字段的长度变化很大,要同时满足较大和较小的数据需求,需要对数据进行监控和分析,合理设置各个存储选项。另外,对于 LOB 字段来说,行链接是很难避免的,需要定期监控 LOB 字段的存储情况,并采取相应措施进行重组。

参考文献

1 V. Jegraj,LOB Performance Guidelines, Oracle Write Paper, May 2004。

2 Thomas Kyte,URL:http://asktom.oracle.com/。

3 熊曾刚等,基于 Oracle9i 关系数据库性能优化策略,计算机系统应用,2004,02(68 –69)。

4 王海涛、鹿凡,Oracle9i 性能调整,清华大学出版社,2004。