HZ BOOKS
华章科技

Oracle数据库领域传奇人物、前阿里B2B最高级别Oracle DBA吕海波（VAGE）10余年职业生涯的集大成之作

深入分析和挖掘Oracle数据库内核中的精髓与秘密，揭示了大量鲜为人知的原理和算法，并详细阐释了如何建立一套自己的调优排故模型

数据库技术丛书

Oracle Core Revealed

Oracle内核技术揭密

吕海波◎著

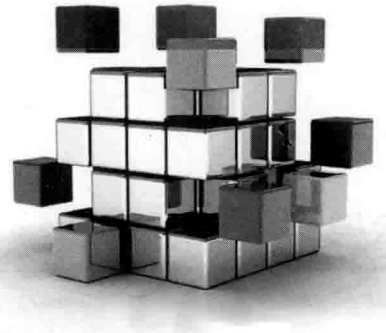
机械工业出版社
China Machine Press

数据库 技术丛书

Oracle Core Revealed

Oracle内核技术揭密

吕海波◎著



机械工业出版社
China Machine Press

图书在版编目（CIP）数据

Oracle 内核技术揭密 / 吕海波著 . —北京：机械工业出版社，2014.7
（数据库技术丛书）

ISBN 978-7-111-46931-5

I. O… II. 吕… III. 关系数据库系统 IV. TP311.138

中国版本图书馆 CIP 数据核字（2014）第 115943 号

Oracle 内核技术揭密

出版发行：机械工业出版社（北京市西城区百万庄大街 22 号 邮政编码：100037）
责任编辑：陈佳媛 责任校对：董纪丽
印 刷：三河市宏图印务有限公司印刷 版 次：2014 年 9 月第 1 版第 1 次印刷
开 本：186mm × 240mm 1/16 印 张：23
书 号：ISBN 978-7-111-46931-5 定 价：69.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88378991 88361066 投稿热线：（010）88379604
购书热线：（010）68326294 88379649 68995259 读者信箱：hzjsj@hzbook.com

Preface 前　言

美国有一句著名的谚语：如果上帝关闭了一扇门，他会为你打开一扇窗。美国还有一个有名的关于 Oracle 的笑话：上帝和埃里森的区别就是，上帝不认为自己是埃里森。

无论上帝怎么想，埃里森肯定认为自己是上帝，至少，是数据库界的上帝。这位数据库界的上帝所开创的著名的 Oracle 数据库软件是闭源的，对于想研究 Oracle 的 DBA 来说，相当于关上了一扇门。但同时 Oracle 中提供大量的 DUMP 命令，这又相当于为 DBA 打开了一扇窗。但现在，这扇窗正在慢慢关闭。

很早之前，有很多从 Oracle 公司流向外界的“内部资料”，对这些内部资料的研究甚至成为学习 Oracle 的一个专门分支：Oracle Internal。当时很多 DBA 都会在简历中加上一行：“精通 Oracle Internal”。当然，笔者也不例外。但是，近五六年来，不知是因为 Oracle 公司加强控制，还是因为众 DBA 研究热情下降，或者二者兼而有之，市面上所见的“内部资料”越来越少，特别是近两三年，基本已经绝迹。

从笔者来看，造成这种情况是“二者兼而有之”。“Oracle 有意控制”论并非空穴来风，伴随着 Oracle 数据库应用得越来越广泛，第三方维护市场的发展也是如火如荼。如果有了疑难问题只能找 Oracle 原厂的售后团队解决，那么第三方维护公司将很难与 Oracle 竞争，这是控制这块市场的最好方法。实现这一目标的捷径，当然就是控制非原厂 DBA 对 Oracle 数据库的了解程度。但是另一方面，Oracle 对 Oracle 技术社区的支持还是有些力度的。所以个人感觉是二者兼而有之，但愿我是“以小人之心，度君子之腹”。无论 Oracle 是否有意控制，现在 Internal 爱好者越来越少已是不争的事实，这与 Oracle 的闭源策略有很大关系。

笔者早年也是“Oracle Internal”研究的爱好者，个人认为，对 Internal 的研究分为以下 3 个阶段：

- 好奇。
- 学以致用。
- 麻木。

好奇之心，人皆有之。对 IT 技术有兴趣的 DBA，在接触 Oracle 不久后，总会对

Oracle 内部算法产生强烈的好奇心：它是如何计算 HASH 值的？它是如何搜索 HASH 链表的？检查点到底完成了什么操作？如果你有这种好奇心，并且有强烈的探索欲望，恭喜你，你将有望成为高级 DBA。跟着你的好奇心去探索吧，在网上查找资料，再加上自己动手做测试，很快你就会对各种 Internal 有朦胧的了解。当出现一些等待事件、一些性能问题或故障时，你就可以从你所了解的原理出发，去分析问题了。这就是第二阶段——“学以致用”，到这一步，“高级 DBA”这个头衔就在向你招手了。

但是很快，你会发现，看不到 Oracle 的源代码，仅从各种 DUMP 结果中分析算法，无异于“隔靴搔痒”，有个最恰当的成语描述这种情况：雾里看花。你会发现，我们雾中看出来的“花”，在解决实际问题时，作用极为有限。一些疑难问题，还是无法理清头绪。这样的事情经历多了，就进入了“麻木”阶段。自然也会得出结论，对 Internal 的研究，现实意义有限，主要是满足好奇心。然后，有可能还会语重心长地告诫初学者，Internal 意义不大，浅尝辄止即可，不必浪费太多精力。当这样的情况越来越多后，人们的“研究”欲望自然会越来越低。所以，现在基本上已经很少有人再去研究 Internal 了。

反观开源领域，虽然源代码的改动并不是“想改你就改”那么简单，因为有各种各样的管理委员会控制，如果不能成为委员，改一个 Bug 都难，但由于可以看到源代码，只要下工夫钻研代码，想做到“明明白白我的心”并不难。这样一来，在遇到一些奇怪问题时，进行诊断、分析将更有依据。这其实是开源数据库在国内流行的主要原因之一。国内的 Committer 并不多，就算能读懂源码也不能修改，在这种情况下，开源除了“便宜”、成本低之外，还有什么优势呢？优势就是“用得明白”、“用得清楚”。闭源的 Oracle 虽然 Bug 更少、更稳定，但出了奇怪的问题后就很难解决。开源则不一样了。一边是 Oracle 的“雾里看花”，你看不到隐藏在暗处的原理；一边是开源领域的“明明白白我的心”，众多技术人员当然选择“弃暗投明”了。

而且近些年，由于前面说的两点原因，Oracle 这场雾更浓了，已经变成“雾霾”了。以前著名的内部资料 DSI 也止步于 Oracle 9i，鲜见 Oracle 10g 版本的，更别说 Oracle 11g 的了。外界 DBA 由于缺乏对原理的了解，很多基本操作都要依赖原厂工程师。比如 Exadata，如果没有原厂工程师协助，连安装都很难完成，更别谈运维了。Oracle 的大门从来就没有敞开过，现在，连“窗”也在逐渐关闭。

变革正在到来。在“门”、“窗”都没有的大环境下，或许可以选择把墙给凿了。凿穿墙壁，一样能看到 Internal。不过，工欲善其事，必先利其器。如果没有适当的工具，想要打开 Oracle 这样庞然大物般的软件无疑是以卵击石。幸运的是，现在凿墙利器已经有了，那就是“动态性能跟踪”语言，比如，Linux 下的 System Tap，Solaris 下的 DTrace，等等。经过笔者试用比较，Solaris 下的 DTrace 更适合用来研究 Oracle 原理。虽然我们只能在 Solaris 平台使用，但 Oracle 的原理在所有 OS 下是一样的（除了在 Windows 下略有不同）。在 Solaris 下研究出的原理，一样可以用于其他平台，可方便大家进行性能调优、故障诊断。笔者书中大多数 Oracle 内部原理的结论，都是使用 DTrace 加 MDB 分析而得出的。

但由于笔者精力有限，而且部分结论还要对 Oracle 的核心代码反汇编才能得出，这将耗费更多精力，因此难免个别结论分析有误，如果各位读者朋友在阅读时发现有误之处，敬请告知，笔者不胜感谢。笔者的 BLOG 地址为：www.mythdata.com，有问题大家可以到该博客留言探讨。

DTrace 跟踪，再加上调试工具 MDB，笔者称为 DBA 中新的领域："调试 Oracle"。调试技术的引入，加上我们对 Oracle 的理解，可以让我们把 Oracle 原理看得更加清晰，可以达到与阅读开源代码同样的效果。这如同吹散"雾霾"的北风，让我们不再"雾里看花"。对原理把握得更加清晰，也使得调优、故障诊断更加精准。"Change"，是这两年最流行的词汇。"我们一直身处变革之中而不自知"。变革正在悄然进入 DBA 领域，传统的 Oracle 运维日渐式微，这已然是不争事实。众多处于"麻木"阶段的 DBA 纷纷转行。"调试 Oracle"领域的出现，将是一条新的发展之路。希望本书能给大家提供一种新的思路。

目　录 *Contents*

第 1 章 Chapter 1

存储结构

存储结构，其实就是表空间、数据文件中的空间组织和使用形式，也许有人会认为存储结构不过是基础知识，相对简单，其实这里面隐藏了很多 Oracle 的秘密，如果不注意挖掘，将无法把 Oracle 提供的性能特性完全发挥出来，为我们所用。本章将会抽丝剥茧地为大家全面介绍表空间、数据文件的知识点，除此以外，在本章的最后，还会提供一个精彩的实际案例，帮助大家进一步理解相关知识点，但希望大家不要急着直接学习案例，这样没有太大意义。那么，下面就让我们来扮演一次福尔摩斯，亲手揭开存储结构的谜团吧！

1.1 区：表空间中的基本单位

区，Extent，逻辑上连续的空间。它是表空间中空间分配的基本单位。如果在某表空间中创建一个表，哪怕只插入一行，这个表至少也会占一个区。

具体来讲，在 Oracle 10g 中，如果创建一个新表，初始至少为这个表分配一个区。而在 Oracle 11.2.0.3 以上版本中，创建新表时默认一个区都不会分配，也就是说，这个表此时不占存储空间。只有在向表中插入第一行数据时，才会默认为表分配第一个区。

无论是 Oracle 10g 还是 Oracle 11GR2，如果表原有区中的空间用完了，Oracle 就会默认为表一次分配一个区的空间。

可以通过 DBA_EXTENTS 数据字典视图查看表所属区。假设有一个表 Table1，想要查询它所在的区，可通过如下方式：

```
select extent_id, file_id, block_id, blocks from dba_extents where
segment_name='TABLE1' order by extent_id;
```

上面语句中，每个列的意义都很简单，这里不再介绍，如有问题，可以查看 Oracle 联机文档 Oracle Database Reference 中的视图介绍。

说了这么多，大家会不会有一个疑问：既然区这么重要，是空间分配的基本单位，那么，区的大小是如何定义的呢？

Oracle 专门设定了两种类型的表空间：统一区大小表空间和系统管理区大小表空间。区的大小就是由这两种表空间决定的。下面，先从统一区大小讲起。

1.1.1 统一区大小表空间和区的使用规则

统一区大小的表空间理解起来很简单，顾名思义，就是创建表空间时，设定区大小为一个统一的值。如下命令创建一个区大小为 1MB 的表空间：

```
create tablespace tbs_ts1 datafile '/u01/Disk1/tbs_ts1_01.dbf' size 50m uniform size 1m;
```

tbs_ts1 表空间包含一个 50MB 的数据文件，区大小为 1MB。在此表空间中创建一个测试表：table1，观察一下区大小。

```
SQL> create table table1(id int,name varchar2(20)) tablespace tbs_ts1;
Table created.

SQL> insert into table1values(1,'VAGE');

1 row created.

SQL> commit;

Commit complete.

SQL> select extent_id, file_id, block_id, blocks from dba_extents where
segment_name='TABLE1' order by extent_id;
EXTENT_ID    FILE_ID   BLOCK_ID     BLOCKS
-------------------- --------------------------
0            4         128          128
```

可以看到，table1 表目前只包含一个区，它从 4 号文件的第 128 号块开始，大小为 128 个块。笔者这里的块大小为 8KB，128 个块，正好就是 1MB。

从上面的结果可以看到，表 table1 从 4 号文件的 128 号块开始占用空间。从 128 ~ 257 号块是 table1 的第一个区，那么，0 ~ 127 号块又是干什么用的呢？

事实上，每个文件的前 128 个块，都是文件头，被 Oracle 留用了。在 Oracle 10g 中是 0 至 8 号块被 Oracle 留用。而从 Oracle 11GR2 开始，一下就留用 128 个块，真是大手笔，不是吗？

这一部分文件头又分两部分，其中 0 号、1 号块是真正的文件头，2 ~ 127 号块是位图块。而在 Oracle10g 中，2 ~ 8 号块则是位图块。

这个位图块又是干什么用的呢？

很容易理解，是用来记录表空间中区的分配情况的。位图块中的每一个二进制位对应一个区是否被分配给某个表、索引等对象。如果第一个二进制位为 0 说明表空间中第一个区未分配，如果为 1 说明已分配；第二个二进制位对应第二个区，以此类推，如图 1-1 所示。

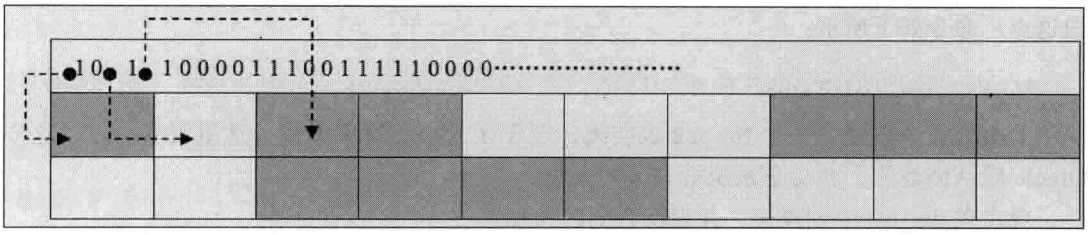


图 1-1　位图块示意图

在图 1-1 中，上面的二进制位就是位图块中的数据，下面表格表示区，其中灰色区是已分配区，白色区是未分配区。第 1 个区对应的二进制位是 1，代表此区已分配，第 2 个区对应的二进制位是 0，表示此区未分配，等等。

位图块又分两部分，其中第一个位图块又被当作位图段头，可以在 DUMP 文件中找到 Oracle 对此块类型的说明：Bitmapped File Space Header。从第二个位图块也就是 3 号块开始，就是真正的位图数据了，DUMP 文件中这些块的类型说明为：Bitmapped File Space Bitmap。

关于位图块的 DUMP 格式描述，大家可以在网上找一下，此处不赘述。下面讨论一个网上较少讨论的话题：当要分配区时，Oracle 如何在位图块中搜索可用区。

大家可以考虑一个问题，如果块大小为 8KB，0 号、1 号块是文件头，2 号块是位图头，在 Oracle 10g 中，3 ~ 8 号块是位图数据块，共 6 个位图块，大小是 48K 字节，每个字节 8 个二进制位，一共 393216 个二进制位。每个二进制位对应一个区，一共就 393216 个区。如果是 Oracle 11GR2，这个数字又要多出好多倍。那么，如果某个表要在数据文件中分配一个新区，Oracle 如何在这 30 多万个二进制位中，确定哪个二进制位对应的区可以分配给表呢？

Oracle 用的方法其实很简单，在位图块中，找一个标记位，如果 0~2 号区被占用了，标记位的值为 3；如果 3~4 号区又被占用了，标记位增加为 5。假设此时 2 号区被释放了，标记位变为 2。

如果需要分配新的区，从这个标记位处开始查找即可。假设目前标记位值为 5，有进程需要 4 个未分配区，Oracle 就从 5 号区开始向下查找。

需要注意的是，如果开启了闪回 Drop，而且 Drop 了表，那么，区并不会被释放，因此标记位不会下降。因为 Drop 只是改名，而并不会真正删除表。

1.1.2 系统管理区大小

刚才介绍了统一区大小，并且测试了区的分配规则。在系统管理区中，区的分配规则是一样的，但是，区大小的设定不再由人为决定，Oracle 会根据表大小自动设置。

Oracle 如何设定区大小呢？只需要创建一个表空间，并进行很简单的测试，就能搞明白这点。命令如下所示：

```
create tablespace tbs_ts2 datafile '/u01/Disk1/tbs_ts2_01.dbf' size 50m reuse;
```

上面的命令创建了一个 tbs_ts2 表空间，至于区大小的管理方式，这里没有指定，这样 Oracle 默认创建的，就会是系统管理区的大小。

现在在 tbs_ts2 中创建表，并观察它的区大小。

```
SQL> create table table_lhb1(id int,name varchar2(20)) tablespace tbs_ts2;
Table created.

SQL> select extent_id, file_id, block_id, blocks from dba_extents where segment_
name='TABLE_LHB1' order by extent_id;

 EXTENT_ID    FILE_ID   BLOCK_ID     BLOCKS
---------- ---------- ---------- ----------
         0          5        128          8
```

可以看到 TABLE_LHB1 目前有一个区，大小只有 8 个块，也就是 64KB。插入一些数据，将表撑大点再观察。

```
SQL> insert into table_lhb1 select rownum,'aaa' from dba_objects;

12650 rows created.

SQL> select extent_id, file_id, block_id, blocks from dba_extents where segment_
name='TABLE_LHB1' order by extent_id;

 EXTENT_ID    FILE_ID   BLOCK_ID     BLOCKS
---------- ---------- ---------- ----------
         0          5        128          8
         1          5        136          8
         2          5        144          8
         3          5        152          8
```

上面向表中插入了 12650 行，表目前涉及 4 个区，每个区都是 8 个块 64KB。提交后继续插入，命令如下：

```
SQL> insert into table_lhb1 select * from table_lhb1;

12650 rows created.

SQL> insert into table_lhb1 select * from table_lhb1;
```

```
25300 rows created.

SQL> insert into table_lhb1 select * from table_lhb1;

50600 rows created.

SQL> insert into table_lhb1 select * from table_lhb1;

101200 rows created.

SQL> commit;

Commit complete.

SQL> select extent_id, file_id, block_id, blocks from dba_extents where segment_
name='TABLE_LHB1' order by extent_id;

 EXTENT_ID    FILE_ID   BLOCK_ID     BLOCKS
---------- ---------- ---------- ----------
         0          5        128          8
         1          5        136          8
         2          5        144          8
         3          5        152          8
         4          5        160          8
         5          5        168          8
         6          5        176          8
         7          5        184          8
         8          5        192          8
         9          5        200          8
        10          5        208          8
        11          5        216          8
        12          5        224          8
        13          5        232          8
        14          5        240          8
        15          5        248          8
        16          5        256        128
        17          5        384        128

18 rows selected.
```

多次插入后，表已经占了很多空间，我们来看一下现在区的使用情况。0 ~ 15 号区，大小为 8 个块 64KB，从第 16 号区开始，区大小变为了 128 个块 1MB 大小。也就是说，表的大小小于 1MB 时，表的每个区都是 64KB，当表的大小超过 1MB，再分配新区时，区的大小将是 1MB。读者可以继续测试，当表进一步变大，区大小将会变成 8MB，表继续扩大，更大的区也有，这里就不测了。

可见，在系统管理区大小表空间中，区的大小随表的增大而增大。

到这里，我们已经发现了系统管理区大小的秘密。很简单吧？有时候探索 Oracle 也不

是件复杂的事情，只要多动手就行了。我在做培训的时候，也常跟学员讲：探索的过程，就是熟悉 Oracle 的过程。你探索出的结果不一定很有用，但探索的过程，会加深对 Oracle 的熟悉程度，这才是研究、探索 Oracle 的真正目的。研究 Oracle，很多时候结果不是目的，过程更有意义。

我们已经了解了统一区大小和系统管理区大小的不同，那么，什么时候使用统一区大小，什么时候使用系统管理区大小呢？

从空间的利用率上讲，小区节省空间，大区可能会浪费空间。比如，当区大小是 10MB 时，为一个表分配了一个 10MB 的区，哪怕它只使用了这 10MB 中的 1 个字节，这 10MB 空间也完全属于这个表了，其他表无法再使用这部分空间。从这个角度上讲，小区的空间利用率无疑是高的。

但从性能角度上讲，对于随机访问，大区、小区没有影响。但对于全表扫描这样的操作，大区又是更合适的。因为连续空间更多，可以减少磁头在区间的定位。

在系统管理区大小的方式下，当表比较小时，区也比较小，当表大时，区也随之变大，这种方式无疑可以在空间的利用率、全扫描的性能之间找到一种平衡。因此建议大多数情况下，都可以采用系统管理区大小的方式。除非有某个表，已明确地知道它会很大，为了保证全扫描的性能，直接建一个统一区大小，并且区比较大的表空间，以便将表存放其中。

如果使用统一区大小，几百 KB 甚至 1MB 的区，都有点小。其实可以参考系统管理型的表空间，当段的大小超过 64MB 时，区大小为 8MB。在使用统一区大小时，也可以将所有区都固定为 8MB。

我见到过很多数据库都使用统一区大小，而且其大小为 1MB。原因是在大部分的操作系统中，一次 I/O 操作的最大的读、写数据量是 1MB。即使使用 8MB 的区，一个区也必须分 8 次进行 I/O 操作，超过 1MB 的区大小，并不能减少 I/O 操作的次数。

但是，我们要考虑一点，8MB 的区连续的空间更多。读取 8MB 内的第 1MB 和第 2MB 数据虽然必须要分两次 I/O 操作，但这两次 I/O 操作很可能是连续 I/O，因为第 1MB 和第 2MB 数据有可能是相连的。如果区大小仅为 1MB，虽然读取表的第 1 区和第 2 区也是两次 I/O 操作，但这两次 I/O 操作很可能不相连，是随机 I/O 操作。连续 I/O 操作的性能当然比随机 I/O 操作的要高。

因此，出于全表扫描性能的考虑，即使使用统一区大小，大点的区（如 8MB 大小）是很合适的选择。

还有一个问题，不知道大家有没考虑到。这个问题涉及统一区大小表空间的位图块。每个二进制位对应一个区的使用情况，这是没问题的，但系统管理区大小呢？就比如刚才创建的 TABLE_LHB1 表，前 16 个区大小为 64KB，之后的区大小为 1MB。区的大小不同，如何用二进制位来反映区的使用情况呢？

Oracle 的处理方法是这样的，以 64KB（也就是 8 个块）为准，每个二进制位对应 64KB。1MB 的区，对应 16 个二进制位。每分配一个 1MB 的区，Oracle 将对应的 16 个二