ITpub技术丛书

Broadview

国内第一本以作者10年的工作经验打造的
基于海量数据的数据库设计与优化的书籍



How to make your Oracle database fly

让Oracle跑得更快2

基于海量数据的数据库设计与优化

谭怀远 著

電子工業出版社

需要全本请访问 www.ertongbook.com

ITPUB 技术丛书

让 Oracle 跑得更快 2——
基于海量数据的数据库设计与优化

谭怀远　著

電子工業出版社
Publishing House of Electronics Industry
北京 · BEIJING

内 容 简 介

数据库设计，是最近几年才出现的技术领域，再早的时候，数据库是以一个黑盒的方式，附属到某个系统当中的，开发人员对它的关注非常少。

近年来，由于很多系统数据量呈几何级数激增，各种性能问题日益凸显出来，而这种性能问题绝大多数都落在了数据的载体——数据库身上，因此，人们越来越关注数据库的性能。而一个数据库性能的好坏，通常是在系统设计阶段就决定了的，于是，将数据库从系统设计中拿出来单独进行设计，变得越来越主流了。

这是一本以讨论海量数据环境下 Oracle 数据库设计与优化的书籍，也是作者 10 年来从事 Oracle 数据库工作的心得体会，是作者工作经验的结晶，这样的书籍并不多见。

本书通篇围绕着在海量数据环境下，如何构造一个高效的 Oracle 数据库这一核心，将许多相关技术融汇到这个核心话题当中，这些技术包括：分区、索引、数据库对象属性、并行技术、只读表空间、初始化参数、几种常见的数据库架构，以及在特定数据库架构下数据库的备份和恢复等相关技术。

本书适合于 Oracle DBA、开发人员、项目经理或者其他对数据库性能感兴趣的人员。

未经许可，不得以任何方式复制或抄袭本书之部分或全部内容。
版权所有，侵权必究。

图书在版编目（CIP）数据

让 Oracle 跑得更快. 2, 基于海量数据的数据库设计与优化 / 谭怀远著. —北京：电子工业出版社，2011.7
（ITPUB 技术丛书）
ISBN 978-7-121-13921-5

Ⅰ. ①让… Ⅱ. ①谭… Ⅲ. ①关系数据库－数据库管理系统，Oracle Ⅳ. ①TP311.138

中国版本图书馆 CIP 数据核字（2011）第 125501 号

策划编辑：张月萍
责任编辑：葛 娜
印 刷：北京中新伟业印刷有限公司
装 订：北京中新伟业印刷有限公司
出版发行：电子工业出版社
北京市海淀区万寿路 173 信箱 邮编：100036
开 本：787×980 1/16 印张：28.25 字数：721 千字
印 次：2011 年 7 月第 1 次印刷
印 数：4000 册 定价：69.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件到 dbqq@phei.com.cn。

服务热线：（010）88258888。

序

怀远君写的第一本书《让 Oracle 跑得更快——Oracle 10g 性能分析与优化思路》于 2010 年 8 月出版，曾经在 China-pub（中国互动网）和当当网的销量榜的前列占据了很长时间。在给该书写序的时候，曾经想抛砖引玉，鼓励勤于笔耕的怀远君继续在优化问题上深入下去。现在时间过去了快一年，新书《让 Oracle 跑得更快 2——基于海量数据的数据库设计与优化》也即将面世。

针对海量数据的数据库设计方面的书，在国内市面上并不多见，本书可以看成国内作者在这个领域里一个新的探索和尝试。随着近年来很多系统中数据量呈几何级数的暴增，对于存放这些数据的数据库的关注，被提到了一个新的高度，这完全是系统需求使然，而系统需求，即为用户需求，海量数据的数据库设计，不可避免地变成系统设计中的一个新领域。

纵观全书，可以视为作者第一本书的内容延续和技术的继续探讨，在第一本书里没有涉及的分区、索引、压缩、RAC、存储规划等内容，在本书里均进行了深入讨论。

关系数据库诞生已经有 30 多年时间，这几十年是数据大量增长、信息爆炸的年代，经过积累，很多应用的数据量已经非常可观。作为这些数据的载体、容器，数据库承受着日益增长的压力。如何面对海量数据带来的存储、查询速度、灾备等方面的压力，是每位数据库工程师所必须面对的课题。

本书的内容，对于处于一线面临海量数据压力的专业人员具有很好的参考价值；同时，对于处于系统设计、项目经理等职位的人员，也具有极好的参考价值，可以看做是作者的另一部心血力作。

ITPUB 创始人 tigefish

2011 年

目　　录

第一篇　分　　区

第 1 章　分区的渊源 …… 2

1.1　DELETE 与系统资源 …… 3
1.2　DELETE 与释放空间 …… 10

第 2 章　分区在海量数据库中的应用 …… 15

2.1　分区的种类 …… 16
2.1.1　范围分区 …… 16
2.1.2　哈希分区 …… 35
2.1.3　列表分区 …… 42
2.1.4　组合分区 …… 43
2.2　三种分区的比较 …… 46
2.2.1　范围分区的适用情况 …… 47
2.2.2　哈希分区的效率 …… 47
2.2.3　列表分区与范围分区各自的优势 …… 54
2.3　小结 …… 61

第二篇　索　　引

第 3 章　B 树索引 …… 64

3.1　B 树索引概述 …… 65
3.2　B 树索引的效率 …… 66
3.2.1　主键或者唯一性约束 …… 66
3.2.2　键值重复率低的字段比较适合使用 B 树索引 …… 77

需要全本请访问 www.ertongbook.com

第 4 章　位图索引 ····85
4.1　位图索引概述 ····86
4.2　什么时候使用位图索引 ····97
第 5 章　全文索引 ····109
5.1　全文索引概述 ····109
5.1.1　全文索引和普通索引的对比 ····110
5.1.2　全文索引的对象 ····114
5.1.3　全文索引在海量数据库中的应用 ····123
5.2　全文索引的空间 ····138
5.3　全文索引和 DML 操作 ····145
5.3.1　INSERT 操作 ····146
5.3.2　DELETE 操作 ····149
5.3.3　UPDATE 操作 ····150
5.4　CTX_REPORT 工具包 ····154
5.4.1　CTX_REPORT.DESCRIBE_INDEX ····158
5.4.2　DESCRIBE_POLICY 函数 ····161
5.4.3　CREATE_INDEX_SCRIPT 函数 ····162
5.4.4　CREATE_POLICY_SCRIPT 函数 ····164
5.4.5　INDEX_SIZE 函数 ····165
5.4.6　INDEX_STATS 存储过程 ····168
5.4.7　QUERY_LOG_SUMMARY 存储过程 ····170
5.5　小结 ····175

第三篇　对 象 属 性

第 6 章　对象属性概述 ····178
6.1　Oracle 数据库中的对象属性 ····178
6.2　对象属性和系统的关系 ····183
第 7 章　并行度 ····185
7.1　对象属性上的并行 ····186
7.2　使用 Hint 方式并行执行 ····189
7.3　索引上的并行度 ····193
第 8 章　数据压缩 ····198
8.1　数据压缩技术的应用 ····199
8.2　分区索引的压缩 ····203

8.3 数据压缩的优越性……205
8.3.1 节省空间……205
8.3.2 性能的提高……212
8.4 DML 操作和数据压缩……219

第 9 章 只读表空间……227

9.1 只读表空间与数据备份和恢复的关系……228
9.2 只读表空间对于数据库的启动和关闭的影响……231
9.3 只读表空间可以防止数据被意外删除和修改……237
9.4 只读表空间使表空间的管理更加方便……240

第 10 章 数据库对象的分析……245

10.1 CBO 和 RBO……246
10.2 分析和动态采样……251
10.3 对象分析的频度……254
10.3.1 数据入库后不再改变……255
10.3.2 数据存在表中且经常改变……271

第四篇 海量数据的数据库架构设计

第 11 章 RAC 架构……285

11.1 RAC 在海量数据库中的应用……293
11.2 RAC 架构之业务分割……295
11.3 RAC 架构之负载均衡……308
11.3.1 客户端的负载均衡……308
11.3.2 服务器端的负载均衡……313
11.4 RAC 架构之 FAILOVER……316

第 12 章 分布式数据库架构……319

12.1 分布式数据库的优越性……319
12.2 分布式数据库的数据处理……321
12.3 分布式数据库的字符集……323

第 13 章 Data Guard 架构……325

13.1 Data Guard 概述……326
13.2 Data Guard 的保护模式……329
13.2.1 最高数据保护模式……329
13.2.2 最高性能模式……329

13.2.3 最高可用性模式 ……330
13.3 Data Guard 和 RAC ……331
13.4 Data Guard 中 Standby 数据库的类型 ……332
13.4.1 物理 Standby 数据库 ……332
13.4.2 逻辑 Standby 数据库 ……333

第 14 章 存储架构的设计（ASM+SAN） ……335

14.1 ASM ……336
14.1.1 方便存储管理 ……337
14.1.2 ASM 的一个管理工具——asmcmd ……345
14.1.3 存储的高可用性 ……348
14.1.4 提升存储性能 ……350
14.2 SAN ……352

第 15 章 海量数据的备份和恢复 ……354

15.1 传统的数据备份方案 ……354
15.1.1 通过数据导出方式备份数据 ……355
15.1.2 数据库冷备份 ……355
15.1.3 数据库热备份 ……356
15.2 海量数据的数据库备份和恢复方案 ……357
15.2.1 使用分布式数据库 ……357
15.2.2 以传递表空间的方式对表空间进行备份和恢复 ……358
15.2.3 对加载数据进行备份 ……368

第五篇 初始化参数

第 16 章 内存相关的参数 ……373

16.1 SGA_TARGET ……373
16.2 PGA_AGGREGATE_TARGET ……379

第 17 章 I/O 相关的参数 ……382

17.1 DB_FILE_MULTIBLOCK_READ_COUNT ……383
17.2 和数据库 I/O 及数据文件相关的其他 4 个参数 ……388
17.2.1 DB_WRITER_PROCESSES ……388
17.2.2 DISK_ASYNCH_IO ……389
17.2.3 DBWR_IO_SLAVES ……389
17.2.4 DB_FILES ……390

第 18 章　优化器相关的参数……391
18.1　CURSOR_SHARING……391
18.2　OPTIMIZER_DYNAMIC_SAMPLING……394
18.3　OPTIMIZER_MODE……398
18.3.1　ALL_ROWS……398
18.3.2　FIRST_ROWS_n……398
18.3.3　FIRST_ROWS……399
第 19 章　其他的参数……400
19.1　并行相关的参数……400
19.1.1　在对象的属性中设置并行……401
19.1.2　在 Hint 中使用并行……401
19.2　分布式数据库相关的参数……404
附录 A　Oracle 字符集……406
附录 B　SHOW_SPACE 存储过程……415
附录 C　一个典型的 OLAP 系统数据库的设计方案……419

需要全本请访问 www.ertongbook.com

开 篇 的 话

一年前的这个时候，我也是在写一本书，书名叫《让 Oracle 跑得更快——Oracle 10g 性能分析与优化思路》。

那是一本讲 Oracle 性能优化的书，书是在 2010 年 8 月初上市的，销量出乎意料的好，曾经一度在中国互动出版网（china-pub.com）上排在计算机类书的销售第一位，随后在当当网的数据库类书中也长时间占据着销售排名的首位。

现在的这本书可以看作是《让 Oracle 跑得更快——Oracle 10g 性能分析与优化思路》的姊妹篇，它继承了上一本书的核心内容，就是写数据库的性能；同时，也保持了上一本书的写作风格，就是用一种思考和启发的方式来写作。

《让 Oracle 跑得更快——Oracle 10g 性能分析与优化思路》主要是以知识点作为切入点来讨论 Oracle 数据库的性能分析和优化，比如并行技术、执行计划、优化器、AWR 报告等；而本书就显得更加具体和有针对性，它主要是讨论在海量数据的情况下，数据库的设计和优化相关的话题。

这样，本书和《让 Oracle 跑得更快——Oracle 10g 性能分析与优化思路》的内容叠加起来，就基本上形成了一套比较完整的关于数据库性能优化方面的书籍。

为什么一定要写关于数据库性能方面的书籍呢？

我一直都持有这样一种观点：现在很多的系统中，数据库的功能正在从数据存储工具的角色慢慢转变成数据处理器的角色。

近年来，用户对系统的性能要求变得越来越高，这和用户对系统的依赖性变得越来越强直接相关；而这种性能实际上很多时候就是指数据库的性能，因为数据库是数据的最终载体和数据分析、处理的机器（大多数时候）。

于是，数据库的性能问题就日益凸显出来，无论是项目负责人、开发人员还是 DBA，都不可回避地要面对这个问题，唯一的区别是各自考虑的层面不同而已。

记得上一本书上市不久，ITPUB 的创始人 Tigerfish 先生就鼓励我再写一本关于数据库设计方面的书，他的原话是这样的：

当 DBA 从烦琐的日常工作脱身出来举目远望的时候，再往前的一片田野便是架构问题，最好的、最彻底的能一劳永逸的优化，往往从架构设计开始。期待怀远君将来的新作，可以在这片更广阔的天地里驰骋。

在之后的沟通中，他又提及了几次，这种友好的鼓励变成了我开始考虑写这本书的原动力。

Tigerfish 先生说得没错，真正的一劳永逸地解决问题的方法，就是在架构设计上。

一个好的架构，会使一个系统充满了可用性和扩展性；反之，就会后患无穷，开发人员要付出大量的后续工作，来修补这种架构缺陷导致的后遗症。

坦白地说，关于数据库的架构问题，特别是在海量数据情况下的数据库架构设计，我认为是一件不太容易的事情。

一方面，它和业务息息相关，需要考虑很多业务需求。

另一方面，数据系统架构设计，就其本身而言，在很多地方又有很大的独立性，比如：

- 分布式数据库架构
- RAC 架构
- Data Guard 架构
- 存储架构

最初想写很多内容，除了上面提及的数据库本身的架构设计外，还考虑对几种典型的系统，比如 OLTP、OLAP 的数据库架构问题进行讨论，甚至包括中间件架构、网络架构等。这看起来真是一个十分庞大的工程！

最终我放弃了这么庞大的内容，决定专心来写关于海量数据的数据库架构设计，这主要考虑到以下几个原因：

（1）我本人从事海量数据的数据库 DBA 工作超过 10 年，对这一块比较熟悉。

（2）数据库设计对于海量数据的数据库来说更有意义，更为重要。

当数据量大到一定程度时，很多看似不是问题的问题都变成了问题。

这样的数据库要解决数据的装载、数据的存储、数据的处理、数据的备份和恢复，这在海量数据的情况下，并非一件容易的事情（实际上，是一件很麻烦的事情）。

（3）数据库的数据越来越多，数据越来越受到重视。

随着信息化的进程越来越快，以前不是海量数据的数据库，现在也变成了海量数据的数据库；以前没有太关注的数据，现在也被严重关注。

基于以上三点原因，我决定只写海量数据的数据库设计；而对于一些 OLTP 相关的技术，书中有提到，但没有做过多的介绍，比如并发、绑定变量和 OLTP 相关的初始化参数都简单带过，这样让本书的内容看起来更有针对性一些。

实际上，如果要比较 OLTP 和 OLAP 数据库设计在系统设计中的重要程度，我认为对后者的关注更应该多一些，主要原因是：

（1）在 OLTP 数据库层面，相对简单，因为它的数据量显得不是那么“大”，很多常规的方法都可以使用，并且还有很多现成的技术都可以用。

（2）对于 OLTP 系统，即使数据量相对较大，但是 SQL 操作相对简单，通常来说，SQL 执行都是一个很快的过程。

这可以用一个小例子来说明。

对于 OLTP 系统的数据库，SQL 语句大多是这样的：

```
select *
from
 test where object_id=100
```

这些 SQL 语句的特点是，它们的谓词条件主要是这样的：

```
where col=…
```

而这些字段上都创建了索引，同时索引键值的重复率非常低（比如银行账户、股票交易代码），甚至很多时候这些字段都是主键（不允许有键值重复）。

这样的特性，使得 OLTP 系统数据库中的 SQL 执行通常都非常快。比如，下面就是这条 SQL 语句的执行结果，我们看到执行时间只能用毫秒衡量。

```
call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.00          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        2      0.00       0.00          0          9          0           4
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        4      0.00       0.00          0          9          0           4
```

```
Misses in library cache during parse: 0
Optimizer mode: ALL_ROWS
Parsing user id: 55

Rows     Row Source Operation
-------  ---------------------------------------------------
      4  TABLE ACCESS BY INDEX ROWID TEST (cr=9 pr=0 pw=0 time=143 us)
      4   INDEX RANGE SCAN TEST_IDX (cr=4 pr=0 pw=0 time=289 us)(object id 51366)

********************************************************************************
```

而在 OLAP 系统数据库中，通常运行着一些类似于下面的这种 SQL 语句，它们的主要作用是进行数据分析和聚合处理。

```
select object_type,count(*)
from
 test group by object_type
```

OLAP 系统数据库的数据量非常大，它们消耗的系统资源要远远大于 OLTP 系统中 SQL 消耗的系统资源。下面是上面 SQL 语句的执行情况。注意：这只是演示的例子，实际上，这些 SQL 语句的执行时间要远远长于这个时间。

```
call     count       cpu    elapsed       disk      query    current        rows
------- ------  -------- ---------- ---------- ---------- ----------  ----------
Parse        1      0.00       0.00          0          0          0           0
Execute      1      0.00       0.00          0          0          0           0
Fetch        4      0.21       0.22          0       3550          1          39
------- ------  -------- ---------- ---------- ---------- ----------  ----------
total        6      2.21       2.32          0       3550          1          39

Misses in library cache during parse: 0
Optimizer mode: ALL_ROWS
Parsing user id: 55

Rows     Row Source Operation
-------  ---------------------------------------------------
     39  HASH GROUP BY (cr=3550 pr=0 pw=0 time=220962 us)
 198980   TABLE ACCESS FULL TEST (cr=3550 pr=0 pw=0 time=597066 us)

********************************************************************************
```

（3）对于数据备份，OLTP 系统数据库可以使用 Rman 增量的方式备份，也可以使用 Data Guard 来解决数据安全问题。

（4）对于 OLTP 系统的高并发，可以考虑使用中间件层间来解决。

（5）要提高 OLTP 数据库的处理效率，可以使用处理能力高的服务器。

（6）要提高 OLTP 数据库的数据访问速度，可以考虑使用内存数据库。

……

总之，对于 OLTP 系统，在数据库层面找到一个合适的解决方案，还不是一件非常困难和费神的事情。

反过来看 OLAP 系统数据库（或者数据仓库系统数据库），由于数据量太大，很多问题都变得很难，比如，对于一个 1000TB 数据的数据库，可能下面的问题都会使你非常头痛：

- 如何规划数据存储？
- 如何管理这些数据，包括备份和恢复数据？
- 如何优化查询？
- 如何实现容灾，保证业务不中断？

可以说，上面的每一条都非常困难，都需要全局地考虑用户需求、业务流程等一系列因素，通过各方权衡找到一个最终的解决方案。

上面谈的内容，就是我想写这本书的目的。和我的上一本书一样，实际上是在讨论对问题的一些思路，它可能没有提供一个完整、具体的解决方案，但是可以帮助启发读者的思维，提供一些参考和帮助。

我也看了《让 Oracle 跑得更快——Oracle 10g 性能分析与优化思路》的一些书评，发现一些读者反映书中的内容写得不够深入，在这里我顺便做一些解释：

我一直认为，对于一个技术，最重要的不是这个技术本身的机制，而是什么时候使用这个技术。

这看起来就像一种本末的关系，比如，可以使用很多示例来说明强制变量绑定和变量窥视（bind peeking）机制，一直到我们明白这个过程的每个环节中 Oracle 做了什么。

但是当我们抬起头来接受别人的质询：

“我们这个系统需要强制变量绑定吗？”

时，我们真的能立刻做出回答吗？

我们知道系统的并发情况吗？

我们知道每天用户的主要工作是什么吗？

我们知道系统是属于 OLAP 还是 OLTP，抑或是二者的一个混合体？

对技术精益求精是技术人员的一种学习方式，但绝不等同于不加选择盲目地沉迷技术，技术终归是为系统服务的，让系统保持健康和高效才是做技术的最终目的。

“授人以鱼，不如授人以渔。”

本书虽然谈不上是“渔”，但是这是作者的写作思路，我希望通过本书，把自己 10 多年来的 Oracle DBA 经验和大家分享，如果能够使读者有所收益，那将是我最高兴的事情！

第一篇 分区

基于海量数据的数据库设计与优化

分区技术是基于海量数据的数据库系统（比如数据仓库或者 OLAP 系统）中最重要的一个技术，我很难想象，在一个存储着上 TB 或者几十 TB 的数据库中，数据表不使用分区的情况。

当我想写一本关于海量数据的数据库设计方面的书时，第一个念头想到的就是分区。

第1章 分区的渊源

基于海量数据的
数据库设计与优化

2003 年的一件事情，我至今记忆犹新，那是和 Oracle 分区相关的一件事情，那时候我带领着一个 DBA 团队管理着几套大型系统的数据库服务器。我们经常会在上午的时候接到用户投诉，投诉的内容几乎都是一样的，就是查询缓慢，但并不阻塞，只不过时间比正常时要慢十几倍，平常半分钟返回结果的查询，此时就变成了十几分钟。

经过对数据库进行分析，发现在这个时间段的数据库资源使用非常严重，CPU 的使用率差不多到了 80%，这种情况是不正常的。当时我们管理的数据库的压力并不太大，于是通过进一步分析，找到了问题所在，数据库中运行着大量这样的 SQL 语句：

```
delete from t_name where j_time<to_date('2003-04-01','yyyy-mm-dd')
```

差不多有 10 几条这样的 SQL 语句。

这是一个最简单不过的按照条件删除数据的操作，通过和开发单位联系，知道他们有一个 job，在凌晨 2 点开始执行，这个任务就是删除 1 个月之前的数据，以便释放空间。

需要全本请访问 www.ertongbook.com