



Oracle 技术系列丛书

ORACLE®

AUTHORIZED ORACLE PRESS™—EXCLUSIVELY FROM OSBORNE

Oracle

高性能 SQL 调整

Oracle High-Performance SQL Tuning

(美) Donald K. Burleson 著 刘 砚 黄 春 等译
(Oracle Internal总编辑、12本数据库畅销书作者)



OFFICIAL • AUTHORIZED

Oracle Press

ONLY FROM OSBORNE



机械工业出版社
China Machine Press

OSBORNE Education

Oracle 技术系列丛书

Oracle 高性能 SQL 调整

(美) Donald K. Burleson 著

刘 砚 黄 春 等译



机械工业出版社
China Machine Press

本书由 Oracle 公司授权, 向读者详尽阐述如何调整 SQL 语句、查看内部执行计划和更改执行计划以提高语句性能。主要内容包括: 理解 SQL 调整在 Oracle 总体微调中的地位, 使用诸如内嵌视图和 BIF 扩展提高 Oracle SQL 性能, 确定并报告程序库缓存中的 SQL 语句, 调整 SQL 表访问、完整表扫描和平行查询, 运行 TKPROF 获得 SQL 跟踪报告, 使用 Oracle 线索为 Oracle SQL 语句更改执行计划, 使用 Oracle8i 优化器计划稳定性、基于成本的优化器和基于规则的优化器, 调整 SQL DML 语句、SQL 子查询和数据仓库 SQL, 调整带有临时表和索引的 SQL 语句, 使用 STATSPACK 诊断和优化系统性能。

本书内容丰富、分析透彻, 可供 Oracle 数据库管理员、技术开发人员参考。

Donald K. Burleson: Oracle High - Performance SQL Tuning (ISBN 0 - 07 - 219058 - 2) .

Copyright @ 2001 by The McGraw-Hill Companies, Inc.

Original language published by The McGraw-Hill Companies, Inc. All Rights reserved. No part of this publication may be reproduced or distributed in any means, or stored in a database or retrieval system, without the prior written permission of the publisher.

Simplified Chinese translation edition jointly published by McGraw-Hill Education (Asia) Co. and China Machine Press.

本书中文简体字版由美国麦格劳-希尔教育出版公司授权机械工业出版社出版, 未经出版者预先书面许可, 不得以任何方式复制或抄袭本书的任何部分。

本书封面贴有 McGraw-Hill 公司防伪标签, 无标签者不得销售。

版权所有, 侵权必究。

本书版权登记号: 图字: 01 - 2001 - 4780

图书在版编目 (CIP) 数据

Oracle 高性能 SQL 调整 / (美) 布尔勒森 (Burleson, D.K.) 著; 刘砚等译. - 北京: 机械工业出版社, 2002.3

(Oracle 技术系列丛书)

书名原文: Oracle High - Performance SQL Tuning

ISBN 7 - 111 - 09755 - 6

I. 0... II. ①布...②刘... III. ①关系数据库-数据库管理系统, Oracle②SQL 语言-程序设计 IV. TP311.138

中国版本图书馆 CIP 数据核字 (2001) 第 097303 号

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: 宋 宏 张鸿斌

北京第二外国语学院印刷厂印刷·新华书店北京发行所发行

2002 年 3 月第 1 版第 1 次印刷

787mm × 1092mm 1/16· 25.5 印张

印数: 0 001 - 4 000 册

定价: 43.00 元

凡购本书, 如有倒页、脱页、缺页, 由本社发行部调换

前 言

Oracle SQL 调整是所有 Oracle 调整过程中最重要的一部分, 进行恰当的 SQL 语句的调整能够在很大程度上提高整个 Oracle 数据库的性能。

不幸的是, 许多 Oracle 专业人员并不了解确保 SQL 优化调整的正确方法。

本书将对 Oracle SQL 进行广泛的指导, 旨在为必须书写 SQL 语句的所有 Oracle 专业人员提供有价值的帮助。本书将指导读者完成 SQL 调整的每一步, 同时为调整各种类型的 SQL 提供专家指导和技术帮助。

本书的主题包括 Oracle 优化器模式的详细讨论, 显示 SQL 执行计划的方法, 以及大量有关提高 SQL 性能的方法。本书选择了非常复杂并且技术含量很高的任务, 使用简单易懂的方式来解释每一个 SQL 调整的步骤。

因为 Oracle SQL 是一种“陈述式”的语言, 因此, 用户只需指定他们想要读取的字段, 与返回数据相关的表, 以及对返回数据的限制, 即可完成查询。由于 SQL 与生俱来的强大能力, 我们可以使用多种形式书写同一个查询, 而各种书写形式之间存在着很大程度的性能差异。本书将为你说明如何调整 SQL 语句, 查看执行计划, 以及如何更改执行计划以提高语句的性能。这是一个令人激动的主题, 因为它能够显著地提高所有 Oracle 数据库的性能。

本书使用经过证实的方法指定并调整不友好的 SQL 语句, 以获得优化的性能。本书提到的所有技能均使用标准的 Oracle 工具, 因此, 在以本书为指导调整 SQL 语句时, 你无需任何特殊产品。同时, 本书覆盖了从简单的 SELECT 语句到复杂的无关联子查询等 SQL 调整的所有方面。而且本书还包含了由作者提出的专家级技术, 这些技术能够在很大程度上提高 SQL 的性能。

随着 Oracle SQL 的演化和日趋复杂, 本书也将不断更新。欢迎你为将来的版本提出宝贵的意见, 你可以将任何有用信息发送至 [don @burleson.cc](mailto:don@burleson.cc)。

本书英文版书名: Oracle High-Performance SQL Tuning

英文版书号: ISBN 0-07-219058-2

原出版社网址: www.osborne.com

① 本书由刘砚、黄春等组织翻译, 万方工作室的全体同仁都参加了本书翻译、校正和输入等工作。具体参加本书翻译、录排、校对工作的其他人员为: 刘堃、牛奇、丁胜、葛丽、罗贤锋、罗浩杰、王宾浩、赵文伟、夏欣、李玲、孙楠、田小敏、龚丽娜、马志军、马秀丽、田小军、牛献忠、田蕴哲、金荣学、薛飞、叶品哲、邓燕、邢倩、王玉、李照军、刘思彬、钱凯斌、赵儒策、江南、李浩天、王凌、李锡林、张莉、范春、袁堡雷、邓笛涛、李林、聂宛、王笑天、李度飞、天鹏等。由于时间仓促, 且译者经验和水平有限, 译文难免有不妥之处, 恳请读者批评指正!

WUS0803

目 录

前言

第一部分 背景知识

第 1 章 SQL 简介.....	1	3.1.3 减少 SQL 解析的技术	40
1.1 SQL 的基本特征	1	3.2 生成执行计划	40
1.2 SQL 起源	3	3.2.1 表访问方式	41
1.2.1 SQL 模型.....	4	3.2.2 索引访问方式	43
1.2.2 select 操作	4	3.2.3 连接操作	45
1.2.3 project 操作	4	3.3 对 SQL 结果集进行排序	49
1.2.4 join 操作	5	3.4 结论	50
1.3 SQL 优化器	5	第 4 章 SQL 优化器简介.....	51
1.4 SQL 调整的目标	5	4.1 基本的优化器技术	51
1.5 作为 Oracle 调整步骤之一的 SQL 调整	6	4.1.1 基于规则的优化器	52
1.6 SQL 调整的障碍	8	4.1.2 基于成本的优化器	54
1.7 SQL 调整过程	9	4.2 优化器模式	56
1.7.1 定位使用频繁的 SQL 语句	10	4.2.1 rule 模式	56
1.7.2 调整 SQL 语句	11	4.2.2 choose 模式	56
1.7.3 调整持久化	11	4.2.3 first_rows 模式	57
1.8 SQL 调整的目标	12	4.2.4 all_rows 模式	57
1.9 SQL 调整工具箱	13	4.2.5 更快的执行速度与最小的资源 消耗	57
1.10 结论	18	4.3 基于规则优化的调整	58
第 2 章 Oracle SQL 扩展简介.....	19	4.3.1 更改基于规则的驱动表	59
2.1 Oracle 内部视图	19	4.3.2 如果基于规则的优化器没有使用正 确的索引	59
2.2 Oracle 内置函数	23	4.4 基于成本优化的调整	60
2.3 Oracle SQL 和面向对象扩展	26	4.4.1 调用基于成本的优化器	61
2.4 挑战关系原则——重复数据项列表	29	4.4.2 为 CBO 收集统计资料	62
2.4.1 varray 表	30	4.5 决定默认优化器模式	63
2.4.2 表嵌套	33	4.6 迁移到基于成本的优化器	64
2.4.3 SQL 对象扩展的性能	34	4.6.1 对开发人员进行重新培训	64
2.5 结论	34	4.6.2 选择基于成本的优化器的思想	65
第 3 章 理解 SQL 执行	36	4.7 结论	67
3.1 解析 SQL 语句	36	第 5 章 SQL 内部处理	68
3.1.1 重新书写查询	37	5.1 共享 SQL 区域和专用 SQL 区域	68
3.1.2 在 Oracle8i 和 Oracle9i 中使用 cursor_sharing	38	5.2 SQL 的 SGA 统计资料	69
		5.3 程序库缓存的内部情况	70
		5.3.1 程序库缓存中可以多次使用的	

SQL	70
5.3.2 监控程序库缓存不足率	71
5.3.3 使用 STATSPACK 监控程序库缓存 中的对象	72
5.3.4 程序库缓存的 STATSPACK 报告	73
5.4 调整 Oracle SQL 排序	74
5.5 确定程序库缓存中具有高影响力的 SQL 语句	79
5.6 关于程序库缓存中 SQL 的报告	83
5.6.1 使用带有 STATSPACK 的 access.sql 脚本	84
5.6.2 access.sql 报告	84
5.6.3 全表扫描报告	85
5.6.4 索引范围扫描报告	86
5.6.5 索引惟一性扫描报告	86
5.6.6 完全索引扫描报告	87
5.6.7 access.sql 报告的局限性	87
5.7 结论	88
第 6 章 调整 SQL 表访问	89
6.1 SQL 调整和全表扫描	89
6.1.1 决定全表扫描的阈值	90
6.1.2 查找全表扫描	90
6.1.3 优化器如何选择全表扫描	91
6.1.4 当 CBO 选择了错误的全表扫描	92
6.1.5 预防不必要的全表扫描	93
6.2 借助于索引的表访问	97
6.3 更改表访问方式	98
6.3.1 将索引由惟一索引更改为非惟一 索引	99
6.3.2 重新书写 SQL 语句以更改表访问 方法	100
6.4 重新排序表记录以减少输入输出	102
6.4.1 使用索引簇重新排序记录	104
6.4.2 使用 CTAS 重新排序记录	106
6.5 Oracle 存储参数和表访问性能	108
6.5.1 pctfree 存储参数	108
6.5.2 pctused 存储参数	109
6.5.3 freelists 存储参数	109
6.5.4 OPS 的 freelist groups 存储参数	110
6.5.5 存储参数规则总结	110
6.6 空闲列表管理和表访问性能	111

6.6.1 空闲列表的链接和取消链接	112
6.6.2 减少空闲列表的重新链接	114
6.7 结论	114

第二部分 SQL 调整基础

第 7 章 Oracle SQL 调整的过程	115
7.1 SQL 调整的目标	115
7.2 SQL 调整步骤	116
7.3 第一步：确定具有高影响力的 SQL 语句	117
7.3.1 SQL 语句分级	117
7.3.2 确定使用频繁的 SQL 语句	118
7.3.3 使用 STATSPACK 指定具有高影 响力的 SQL	119
7.3.4 来自程序库缓存的 SQL 报告	122
7.3.5 使用第三方工具定位有问题的 SQL	126
7.4 第二步：抽取和解释 SQL 语句	128
7.5 第三步：调整 SQL 语句	129
7.6 SQL 调整实例	130
7.7 使用第三方工具快速进行 SQL 调整	132
7.8 结论	134
第 8 章 理解 Oracle SQL 工具	135
8.1 解释 SQL 语句	135
8.2 运行快速 SQL 跟踪	140
8.3 TKPROF 工具	141
8.3.1 为 SQL 跟踪设置环境	141
8.3.2 生成 SQL 跟踪文件	142
8.3.3 格式化跟踪文件	144
8.3.4 TKPROF 报告	145
8.4 Oracle 专家中心 SQL 分析报告	147
8.5 有关程序库缓存中所有 SQL 的报告	149
8.5.1 使用带有 STATSPACK 的 access.sql 脚本	150
8.5.2 access.sql 报告	151
8.6 结论	153
第 9 章 定位重要的 SQL 语句	154
9.1 建立实例范围的 SQL 基线	154
9.1.1 设置默认的 Optimizer_Mode	154
9.1.2 默认优化器模式持久化	155

9.1.3 提高全表扫描速度的实例范围	156	11.3 不必要的排序	188
参数	156	11.4 监控排序活动	188
9.1.4 其他影响 SQL 执行的初始化	156	11.5 结论	191
参数	156	第 12 章 使用 Oracle 提示进行调整	192
9.1.5 运行 SQL 基线测试	157	12.1 提示简介和历史	192
9.2 什么是重要的 SQL 语句	157	12.2 在 SQL 查询中指定提示	192
9.3 抽取需要调整的重要 SQL 语句的	158	12.3 优化器提示	193
技术	158	12.3.1 all_rows 提示	194
9.3.1 指定高频使用的 SQL 语句	158	12.3.2 rule 提示	194
9.3.2 抽取 SQL 语句的脚本	159	12.3.3 first_rows 提示	194
9.4 SQL 语句分级	160	12.4 表连接提示	194
9.4.1 查找不包含提示的新 SQL 语句	160	12.4.1 use_hash 提示	194
9.4.2 当使用优化器计划稳定性时查找新	163	12.4.2 use_merge 提示	196
SQL 语句	163	12.4.3 use_nl 提示	196
9.5 结论	163	12.4.4 star 提示	197
第 10 章 调整全表扫描和并行查询	164	12.5 表反连接提示	198
10.1 评估全表扫描的合法性	164	12.5.1 merge_aj 提示	198
10.2 查找要进行 Oracle 并行查询的表	165	12.5.2 hash_aj 提示	200
10.3 Oracle 并行查询简介	166	12.6 Index 提示	201
10.3.1 调用 Oracle 并行查询	166	12.6.1 Index 提示简介	201
10.3.2 Oracle 并行查询初始化参数	167	12.6.2 index_join 提示	202
10.3.3 设置最优化的并行度	168	12.6.3 and_equal 提示	202
10.3.4 使用并行查询提示	172	12.6.4 index_asc 提示	204
10.3.5 并行查询和表连接	173	12.6.5 no_index 提示	204
10.4 监控 Oracle 并行查询的使用	179	12.6.6 index_desc 提示	204
10.4.1 监控并行执行活动	179	12.6.7 index_combine 提示	204
10.4.2 使用 STATSPACK 监控 Oracle 并行	180	12.6.8 index_ffs 提示	206
查询	180	12.6.9 use_concat 提示	207
10.4.3 使用 V\$ 视图监控 Oracle 并行	181	12.7 Parallel 提示	209
查询	181	12.7.1 parallel 提示简介	209
10.4.4 并行查询和分布表	182	12.7.2 pq_distribute 提示	210
10.5 结论	183	12.7.3 noparallel 提示	211
第 11 章 优化 Oracle SQL 语句排序	185	12.8 表访问提示	211
11.1 有关 Oracle 排序的初始化参数	185	12.8.1 full 提示	211
11.1.1 sort_area_size 参数	186	12.8.2 簇表提示	211
11.1.2 sort_area_retained_size 参数	186	12.8.3 no_expand 提示	212
11.1.3 sort_multiblock_read_count 参数	187	12.8.4 nocache 提示	212
11.1.4 已废弃的 Oracle8 排序参数	187	12.8.5 ordered 提示	212
11.1.5 优化器模式和排序活动	187	12.8.6 ordered_predicates 提示	213
11.2 通过添加索引避免排序	187	12.8.7 push_subq 提示	213
		12.9 子查询中的提示	213

12.10 结论	216	15.1 调用基于规则的优化	255
第 13 章 使用优化器计划稳定性进行 调整	217	15.2 使用 choose 作为默认优化模式时存在 的问题	256
13.1 存储框架简介	217	15.3 使用基于规则的默认优化器模式	256
13.1.1 优化器计划稳定性背后的思想	218	15.3.1 驱动表位置问题	257
13.1.2 在向 CBO 迁移时使用优化器计划 稳定性	219	15.3.2 驱动表和表基数	258
13.2 使用存储框架要做的准备	219	15.3.3 有关调整基于规则查询的提示	259
13.2.1 设置 use_stored_outlines 命令	220	15.4 基于规则的优化器无效的情况	260
13.2.2 检查重要的初始化参数	220	15.5 基于规则的优化器是最佳选择的 情况	260
13.2.3 创建框架包	220	15.6 结论	262
13.2.4 创建存储框架表空间	221	第 16 章 调整表连接	264
13.3 如何创建和修改存储框架	222	16.1 表连接类型	264
13.3.1 确定最快的执行计划	222	16.1.1 等连接	265
13.3.2 为最初的查询创建存储框架	224	16.1.2 外部连接	265
13.3.3 为带有提示的查询创建存储 框架	225	16.1.3 自连接	267
13.3.4 交换存储框架	227	16.1.4 反连接	269
13.4 管理存储框架	228	16.1.5 半连接	271
13.4.1 为存储框架使用字典视图和表	228	16.2 Oracle 表连接方法	274
13.4.2 使用框架包	230	16.2.1 嵌套循环连接	274
13.4.3 管理存储框架的分类	231	16.2.2 use_nl 提示	276
13.5 结论	234	16.2.3 散列连接	276
第 14 章 基于成本优化器的调整	235	16.2.4 排序合并连接	280
14.1 统计资料和基于成本的优化	235	16.2.5 星型连接	283
14.1.1 动态和静态 CBO 执行计划思想	235	16.3 评估表连接顺序	287
14.1.2 收集统计资料的频率	236	16.4 调整分布式 SQL 表连接	288
14.1.3 为 CBO 收集统计资料	238	16.4.1 总是为分布式连接使用 CBO	289
14.2 基于成本优化和 SQL 调整	239	16.4.2 察看分布式连接的执行计划	289
14.2.1 基于成本的表连接	240	16.4.3 分布式连接调整方针	291
14.2.2 调整带有子查询的基于成本的 SQL 语句	241	16.5 结论	292
14.2.3 调整复杂的布尔查询	242		
14.3 影响基于成本优化器行为的初始化 参数	248	第三部分 SQL 高级调整	
14.3.1 影响表连接的初始化参数	248	第 17 章 调整 SQL 的 DML 语句	293
14.3.2 影响 CBO 索引行为的初始化 参数	253	17.1 Oracle 存储参数和 DML 性能	293
14.4 结论	253	17.1.1 pctfree 存储参数	294
第 15 章 基于规则优化器的调整	255	17.1.2 pctused 存储参数	294
		17.1.3 freelists 参数和 DML 性能	295
		17.1.4 OPS 的 freelist 组存储参数	295
		17.1.5 存储参数规则的小结	295
		17.2 空闲列表管理和 DML 性能	296

17.2.1 链接到空闲列表和从空闲列表断 开链接	297	查询	345
17.2.2 减少空闲列表重新链接操作	299	19.5.4 使用 NOT EXISTS 运算符的非关联 子查询	346
17.3 长数据列和 DML 行为	299	19.6 调整带有不相等条件的子查询	347
17.4 根据平均记录长度来设置 pctfree 和 pctused 参数	301	19.6.1 在子查询中使用 ALL 和 ANY 子句	347
17.5 缓冲区忙等待以及 DML 冲突	302	19.6.2 在子查询中使用不相等条件	349
17.5.1 利用 STATSPACK 发现 DML 等待 冲突	303	19.7 用提示来提高子查询的执行速度	351
17.5.2 用 STATSPACK 确定缓冲区忙 等待	305	19.8 结论	352
17.6 SQL update 语句、子查询和并行 DML ..	312	第 20 章 用索引调整 SQL	353
17.7 DML 执行时约束的开销	314	20.1 把握进行索引的时机	353
17.8 用 DML 维持索引所需的开销	315	20.1.1 提取 SQL 语句	354
17.9 使用 PL/SQL 批量插入来提高 SQL 插入 数据的速度	316	20.1.2 添加索引	355
17.10 结论	317	20.1.3 衡量全表扫描的合法性	355
第 18 章 用临时表调整 SQL	319	20.1.4 衡量带有新索引的查询的执行 时间	358
18.1 与字典视图一起使用 CTAS	319	20.1.5 预测整个数据库的受益情况	358
18.2 结论	328	20.2 通过添加索引来消除排序	358
第 19 章 SQL 子查询的调整	329	20.3 不必要的排序	358
19.1 Oracle 子查询的基本知识	329	20.4 索引谓词	359
19.2 关联与非关联子查询	331	20.5 B 树索引中可供选择的索引	363
19.2.1 数据量规模和子查询的问题	332	20.5.1 位图索引	363
19.2.2 子查询的执行时间	332	20.5.2 基于函数的索引	363
19.2.3 子查询执行的基本特征	335	20.5.3 反主键索引与 SQL 性能	363
19.3 子查询的自动 SQL 转换	335	20.6 带有 IN 条件的查询的索引用法	364
19.4 调整具有 IN 和 EXISTS 子句的子查询 ..	335	20.7 结论	365
19.4.1 使用 IN 子句的非关联子查询	335	第 21 章 数据仓库的 SQL 调整	366
19.4.2 带有 IN 子句的关联子查询	340	21.1 大型数据表连接调整	366
19.4.3 带有 EXISTS 子句的非关联子 查询	340	21.1.1 ordered 提示	367
19.4.4 使用 EXISTS 子句的关联子查询	340	21.1.2 order_predicates 提示	367
19.5 调整带有 NOT IN 和 NOT EXISTS 子句 的子查询	342	21.2 Oracle 分区与 Oracle SQL 调整	368
19.5.1 使用 NOT IN 运算符的非关联子 查询	342	21.2.1 分区和 SQL 数据表连接	368
19.5.2 带有 NOT IN 运算符的关联子 查询	344	21.2.2 分区与 SQL 执行速度	368
19.5.3 带有 NOT EXISTS 运算符的关联子 查询	345	21.3 Oracle 的并行查询和 Oracle SQL 调整 ..	369
		21.3.1 并行查询的 int.ora 参数	369
		21.3.2 设置最优的并行度	370
		21.3.3 使用并行查询提示	371
		21.4 数据仓库查询的 Oracle 优化	373
		21.4.1 optimizer_search_limit 参数	374
		21.4.2 optimizer_max_permutations 参数	374

21.5 结论	376	23.1 在 SQL 中使用 like 和 case 子句	388
第 22 章 用 STATSPACK 进行 SQL 调整 ...	377	23.1.1 在 Oracle 查询中使用 like 子句	388
22.1 STATSPACK 和 SQL 数据收集	378	23.1.2 用 case 语句合并多重扫描	390
22.2 用 STATSPACK 对 Oracle SQL 排序进行 调整	378	23.2 调整带有 BIF 的 SQL 语句	391
22.3 用 STATSPACK 跟踪 SQL	384	23.2.1 BIF 与字符数据类型一起使用	392
22.3.1 SQL 快照阈值	384	23.2.2 BIF 与日期数据类型一起使用	392
22.3.2 STATSPACK SQL 前 10 位报表	385	23.2.3 使用 substr BIF	395
22.4 结论	387	23.2.4 Oracle9i 中对 BIF 的改进	396
第 23 章 调整使用内置函数和特殊运算 符的 SQL 语句	388	23.3 结论	397
		23.4 本书的总结	397

第一部分 背景知识

第 1 章 SQL 简介

SQL 是结构化查询语言 (Structured Query Language) 的缩写。但不幸的是, SQL 不是结构化的, SQL 也不是仅用来查询的, 而且就其本身而论, SQL 也不是一种语言, 因为 SQL 是嵌在其他语言例如 C 或 COBOL 中使用的。尽管存在这样命名的错误, SQL 仍然成为了对关系数据库的主要访问方式。

本章将介绍 Oracle SQL 的特征, 并对本书将要通篇使用的方法的基本内容进行阐述。在本章中, 我们将讨论下列主题。

- **SQL 的基本特征** 对 SQL 与导航数据库查询语言进行比较。
- **SQL 起源** 阐述 SQL 演化成为数据库访问实际标准的过程。
- **SQL 优化器** 对 SQL 优化的过程作简短的介绍。
- **SQL 调整的目标** 介绍 SQL 调整的总体目标。
- **作为 Oracle 调整步骤之一的 SQL 调整** 讨论 SQL 调整在总体调整模型中的地位。
- **SQL 调整的障碍** 讨论在试图对 Oracle SQL 进行调整时遇到的问题。
- **SQL 调整的过程** 讨论对单个 SQL 语句进行调整的一般目标。
- **SQL 调整工具箱** 介绍我们在本书中将要通篇使用的用来检查 SQL 语句以便于调整的工具箱。

1.1 SQL 的基本特征

SQL 标准提案最初是设计用来代替在现存数据库中使用的令人头痛的导航语言的。在 20 世纪 60 年代, IBM IMS 数据库是惟一的大型商业数据库管理系统。与其他基于关系模型的数据库不同, IMS 是带有内部指针结构的层次数据库, 这些内部指针结构用于在数据库记录之间进行导航。

导航数据库访问工具要求程序员使用跟踪指针的方法在数据结构中进行导航。下面是在流行的 IDMS 数据库中进行查询的实例, IDMS 是早期的 CODASYL 网络数据库。

```
MOVE 'JONES' TO CUST-DESC.  
OBTAIN CALC CUSTOMER.  
MOVE CUSTOMER-ADDRESS TO OUT-REC.  
WRITE OUT-REC.
```

```

FIND FIRST ORDER WITHIN CUSTOMER-ORDER.
  PERFORM ORDER-LOOP UNTIL END-OF-SET.
*****
ORDER-LOOP.
*****
  OBTAIN FIRST ORDER-LINE-REC WITHIN ORDER-LINE.
  PERFORM ORDER-LINE-LOOP UNTIL END-OF-SET.
  FIND NEXT ORDER WITHIN CUSTOMER-ORDER.
*****
ORDER-LINE-LOOP.
*****
  OBTAIN NEXT ORDER-LINE-REC WITHIN ORDER-LINE.
  MOVE QUANTITY-ORDERED TO OUT-REC.
  WRITE OUT-REC.
  OBTAIN OWNER WITHIN ORDER-LINE-PRODUCT
  MOVE PRODUCT-NAME TO OUT-REC.
  WRITE OUT-REC.

```

这里我们看到在数据记录之间进行查询导航、记录访问、指针查找以及在指针之间移动的时候，都需要用到指针的数值（如图 1-1 所示）。这里的关键在于，此种类型的数据库查询要求操作者对数据库的内部结构有一定的了解才能够获得所需的数据。

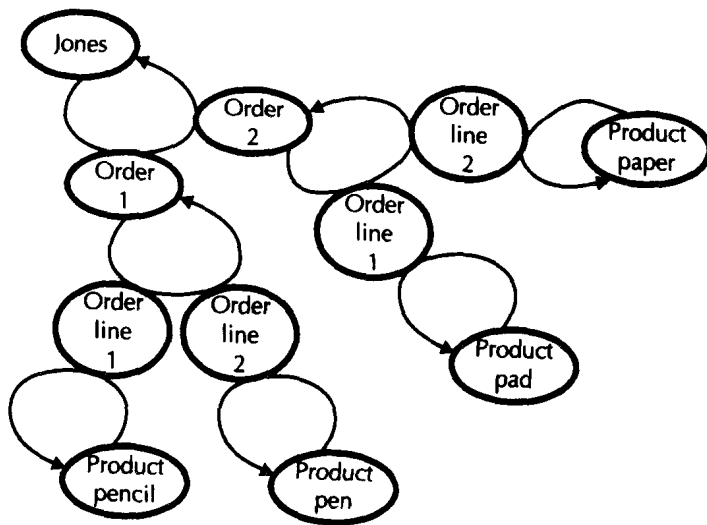


图 1-1 导航数据库查询

在 SQL 中相同的语句却有着非常不同的语法和功能。与导航数据库访问语言不同，SQL 的设计目标是，使你能够仅仅获得想要显示的确定的字段、包含所需数据的表和表的连接条件。

```

select
  customer_address,
  product_name,
  quantity_ordered
from

```

```
customer c,  
order o,  
order_line l,  
product  
where  
  customer_name = 'JONES'  
and  
  c.cust_nbr = o.cust_nbr  
and  
  o.order_nbr = l.order_nbr  
and  
  l.product_nbr = p.product_nbr  
;
```

我们还将在本章的后面对 SQL 的基本结构做进一步的讨论。

尽管在大多数的时候 SQL 是同关系数据库联系在一起的，但是值得一提的是它在处理非关系数据库时也非常流行，IDMS 网络数据库开发者在创建了 SQL 引擎后将他们的产品重新命名为 IDMS/R，并且现在有几个面向对象的数据库系统提供 SQL 前端，这些 SQL 前端使他们的数据库看起来像是关系数据库。

1.2 SQL 起源

在 1970 年，IBM 公司的 Edgar Codd 博士和 Chris Date 开发了用于数据存储的关系模型。在这个模型中，数据以一种叫作“关系”或者“表”的简单的线性结构存储。这个关系模型在其他早期模型的基础上所做的最好的改进就是它更加简单。这个关系模型不再要求用户知道一系列的导航数据操作语言命令，而是引入了一种叫做 SQL 的陈述式语言来简化对数据的访问和操作。

在 Codd 和 Date 的模型中，表就是由“记录”和“字段”组成的二维数组。记录叫做“元组”（tuples，在英文中与 couples 押韵），字段叫做“属性”。表通常包含做为该表主键的一个或几个字段。在它们的关系数据库模型中，表与表之间是相互独立的，这与层次模型和网络模型不同，在层次模型和网络模型中，表与表之间是指针连接的。

关系数据库模型在原有的层次数据库和网络数据库的基础上做了如下的改进：

- **简单性** 由记录和字段组成的表的概念非常简单易懂。最终用户使用简单的数据模型。在层次数据库和网络数据库中使用的复杂的网络图表在关系数据库中不再使用。
- **数据独立性** 数据独立性是指在修改数据结构（在这里是指表）的时候不会影响到现存的程序。这个功能的存在归功于表和表之间不存在硬链接。操作者可以向表中添加字段，可以向数据库添加表，也可以在对表的结构做较小的甚至是不做改动的情況下添加新的数据关系。关系数据库比层次数据库和网络数据库为操作者提供了更高的数据独立性。
- **陈述式的数据访问** SQL 用户指定他们所需的数据，然后由嵌入的 SQL（一种过程语言）来决定如何获取这些数据。在访问关系数据库时，用户告知系统取得数据的条件。系统随之取出符合 SQL 语句中选择条件的数据。数据导航对于终端用户或程序员来说是不可见的，这一点与 CODASYL DML 语言不同，在 CODASYL DML 语言中程序员必须

知道访问路径的细节。

在市场中，陈述式的数据访问能力比关系数据库的内部存储内容更具吸引力，SQL 渐渐成为关系模型的代名词。

1.2.1 SQL 模型

第一个 SQL 模型有三类功能：定义、操作和授权。

- **定义** 是指执行对象创建、对象清除和对象修改功能的数据定义语言（data definition language, DDL）。
- **操作** 是指执行选择、插入、更新和删除功能的数据操作语言（data manipulation language, DML）。
- **授权** 是指授权和取消授权控制的机制。

在操作功能中，我们可以看到 SQL 的三个操作：select、project 和 join。这三个简单的操作定义了 SQL 的全部功能。

1.2.2 select 操作

select 操作通过过滤掉不需要记录的方法减小了表的长度。通过在 where 语句中指定条件的方式，用户可以从结果数据集中过滤掉不需要的记录，如图 1-2 所示。总之，select 操作在纵向上缩小了结果数据集。

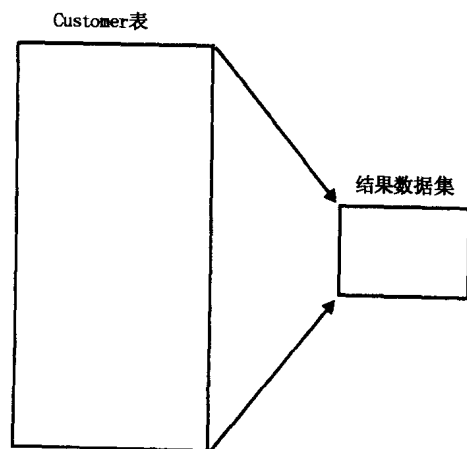


图 1-2 select 过滤器减少了返回记录的数目

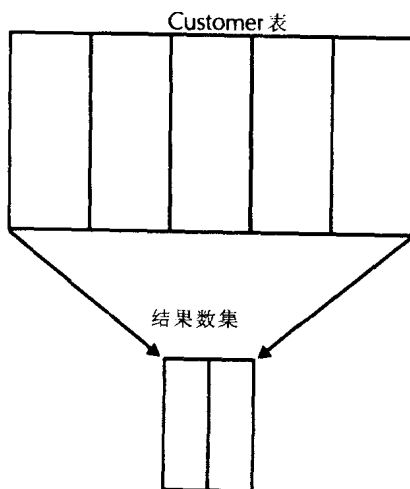


图 1-3 project 操作将去掉不需要的字段

1.2.3 project 操作

正如 select 操作减少了记录的数目一样，project 操作将减少字段的数目。在 SQL 选择语句

中指定的字段名称将决定显示哪些字段，如图 1-3 所示。总之，project 操作在横向上缩小了结果数据集。

1.2.4 join 操作

如图 1-4 所示，join 操作用来在拥有公共字段的两个或多个独立表之间建立关联。在 join 操作中，两个或多个独立的表根据公共字段的值进行合并。

在这个简单的框架中，我们可以看到在 SQL 中的查询是一种“陈述空间”（state - space）类型的查询。也就是说，书写查询语句的操作者不需要考虑数据的导航路径。SQL 优化器在内部对数据表的导航路径进行处理。

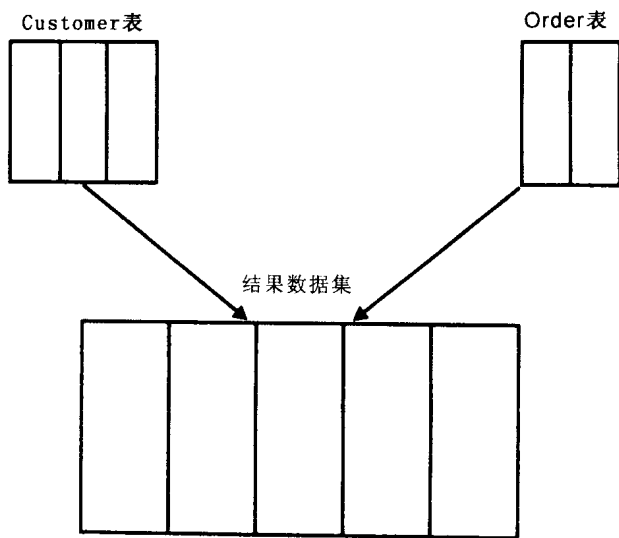


图 1-4 join 操作

1.3 SQL 优化器

在最完美的情况下，SQL 优化器应该能够产生最有效的方法服务于数据查询。如果 SQL 优化器拥有有关查询中包含的所有表的全部信息（例如，记录的数目、某字段中值的分配），那么这个优化器应该总是能够选择访问数据的最佳执行计划。

而在现实的情况下，SQL 优化器需要人工的辅助。Oracle DBA 要能够确保 SQL 优化器可以获得所有的对象统计资料，同时 SQL 优化的极端复杂性使得任何供应商都很难提供总是选择最佳数据访问路径的优化器。因此，我们需要像这样的一本书来了解如何帮助优化器选择最有效的数据访问路径。

1.4 SQL 调整的目标

SQL 调整是 Oracle 调整活动中最花费时间而且最具挑战性的操作。与 SGA（只需执行一次）调整不同，SQL 调整是持续进行的过程。每一个 SQL 语句都必须进行单独的调整，而且这些调整必须使用提示或新的 Oracle8i 优化器计划稳定性来实现。有关使用优化器计划稳定性的详尽信息，请参阅本书第 13 章。

另外，SQL 调整对 Oracle 性能有着直接的正面影响。如果 Oracle 优化器选择了非优化的执行计划，人工调整通常可以使查询速度翻 3 倍。对于那些每天都要执行上千次的查询来说，这样做的结果将是大大提高数据库的性能，同时还可以延长数据库服务器上硬件的寿命。以下是 SQL 调整目标的总结列表：

- **确保所有的 SQL 语句在执行之前得到“认可”** 请注意书写 SQL 语句的程序员的目标同进行 SQL 调整的操作者的目标有很大的不同。程序员考虑的是以最快的速度得到要查询的正确结果，这样就使得从执行的角度来看通常存在一些并不优化的 SQL 语句。

- **创建有益于所有的 SQL 操作的 Oracle 统计资料和索引** Oracle SQL 优化器并不是存在于真空中的, Oracle DBA 的目标是创建恰当的索引和对象分析方法, 从而确保所有的 SQL 功能都能以优化的方式执行。
- **为所有的 SQL 锁定执行计划** 添加新的索引有时会造成对一系列的 SQL 语句进行更改, 而 SQL 调整的主要目标就是使得这些调整的更改持久化。对于任何 SQL 语句来说, 仅存在一个优化的执行计划, 一旦这个执行计划确定以后, 执行计划持久化就成为 Oracle DBA 的目标, 这一过程是通过向查询中添加提示或是使用优化器计划稳定性来实现的。

在本书中, 我们将把这一目标作为中心主题进行讨论。

1.5 作为 Oracle 调整步骤之一的 SQL 调整

SQL 调整是 Oracle 调整中不可缺少的组成部分, 而且它将对 Oracle 的性能产生直接的影响。然而, 在没有执行先决条件操作以前, 请不要直接进行 SQL 调整, 尽管这样做很有诱惑力。

Oracle 调整的目的非常明确。正如你在图 1-5 中所看到的, 所有的 Oracle 数据库都是以层次结构进行组织的, 同时它包含影响子组件的全局组件。

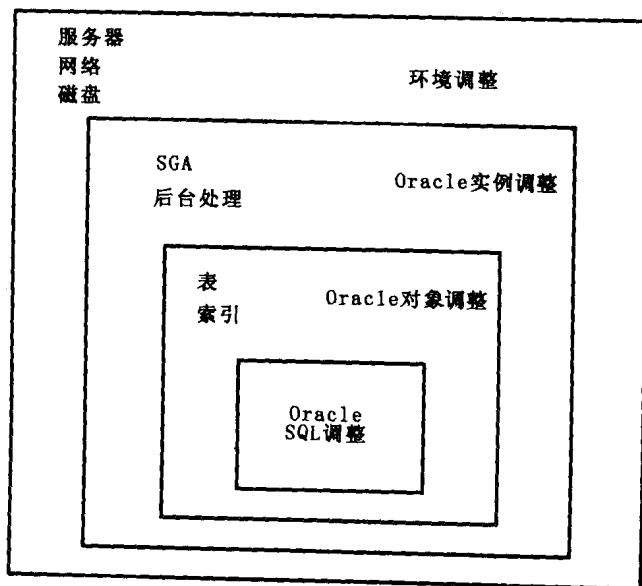


图 1-5 Oracle 调整层次结构

这种层次结构非常重要, 因为任何在高层进行的错误调整将会影响到位于它下面层次中的所有操作, 以下是对 Oracle 调整层次结构的详细说明:

- **环境调整** 环境调整是对数据库服务器、网络以及磁盘输入输出子系统进行调整。只有对这些组件进行调整之后, Oracle DBA 才能够开始对 Oracle 数据库进行调整。
- **数据库服务器调整** Oracle 数据库服务器调整是进行所有 Oracle 调整的先决条件。如果数据库服务器的 RAM 或 CPU 不足, 那么不管多少 Oracle 调整都无法解决这个问题。

- **网络调整** 必须对网络底层进行调整以确保在网络协议层不存在数据包传输问题。有关对 Oracle 网络通信进行调整的详尽信息，请参阅本书第 7 章。
- **磁盘调整** 对磁盘输入输出子系统进行调整是 Oracle 调整中绝对的先决条件。磁盘输入输出瓶颈和磁盘访问问题都必须在 Oracle 调整开始之前解决。
- **实例调整** 在进行了外部环境的调整以后，下一个步骤就是对 Oracle 实例进行调整。这包括对 SGA 内存区和 Oracle 后台处理行为进行调整。这一步骤对有关 SQL 调整所做的操作是，对数据库中的所有 SQL 的默认优化器模式 (optimizer_mode) 进行设置。有关 Oracle optimizer_mode 的详尽信息，请参阅本书的第 14 章和第 15 章。
- **对象调整** 对实例进行调整之后，我们要对每一个 Oracle 对象进行调整从而优化性能。这一步骤包括对所有的存储参数进行正确的设置，尤其是对影响输入输出的参数进行设置。pctfree、pctused 和 freelists 参数的设置都会对 SQL 性能产生重要的影响。
- **SQL 调整** 在完成了所有的常规调整之后，你就已经为进行 SQL 调整做好了准备。这一步骤包括定义高频使用的 SQL 语句，对语句进行调整，以及确保优化的执行计划持久化。

数据库设计和 SQL 性能

注意，我们有意遗漏了影响 SQL 速度的最重要的因素之一，那就是原始数据结构的设计，如图 1-6 所示。Oracle 表中数据标准化的程度和计划冗余的多少都会对查询的速度产生巨大的影响。你将在本书后面学到，通过向表中添加冗余的字段（即反标准化），可以避免昂贵的 SQL 连接，而且能够提高性能。

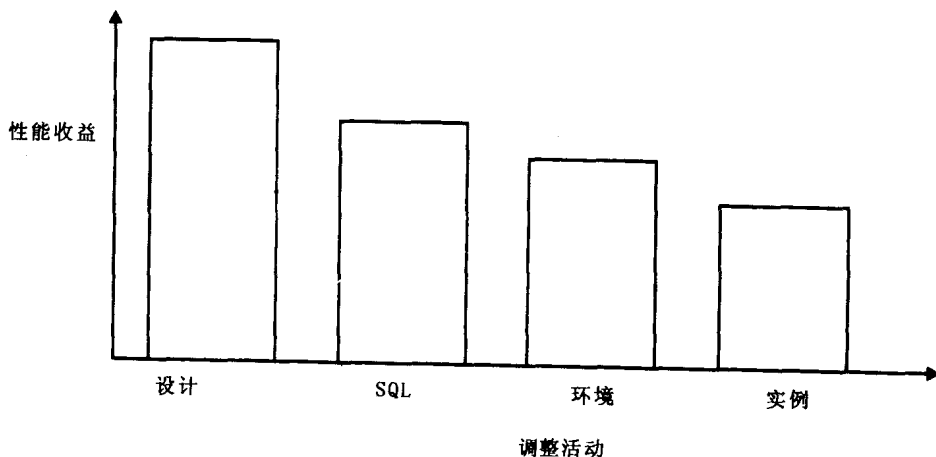


图 1-6 不同调整操作产生的相应性能收益

我们已经跳过了对这个方面的讨论，因为一旦应用程序进入了产品环境以后，Oracle DBA 对表设计进行更改几乎是不可能的。然而，我们将会在后面的章节中讨论对字段进行复制的方法，告诉你如何通过从一个表向另一个表复制数据字段的方法来避免使用 SQL 表连接（如图 1-7 所示），并且告诉你如何提供触发器来保持复制的数据项的更新。