HZ BOOKS
华章科技

资深Oracle技术专家十余年工作经验结晶，Oracle数据库性能优化领域的里程碑之作

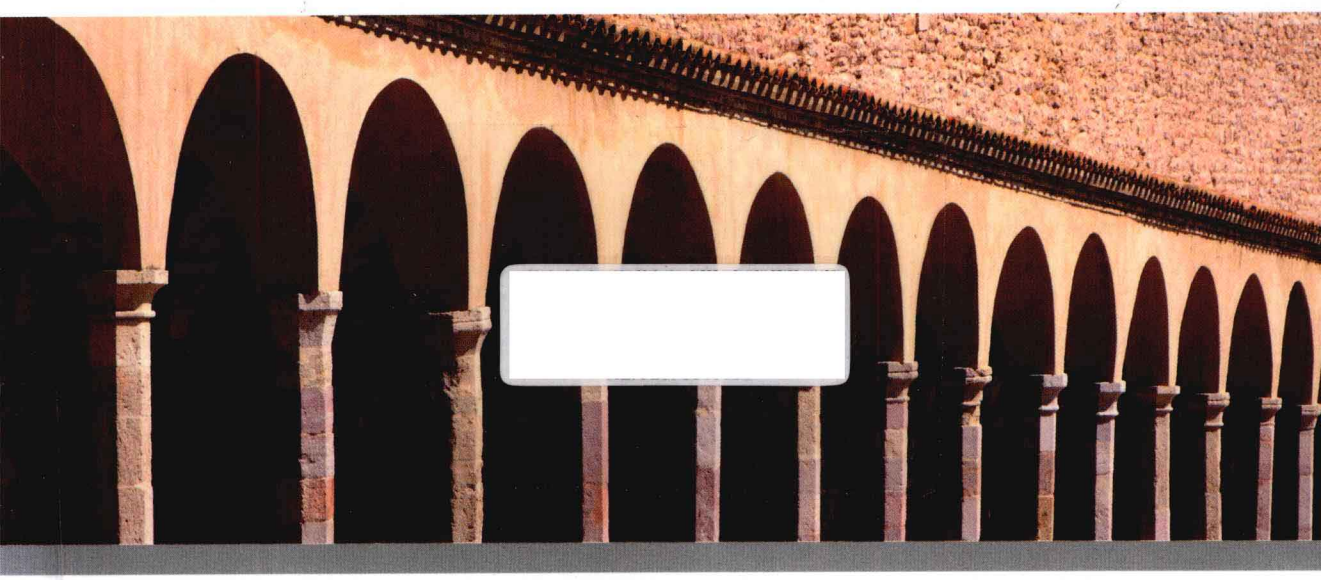
深入剖析Oracle SQL的优化与调优机制、核心技术和思想方法，包含大量技巧和最佳实践

Oracle High Performance SQL Engine

SQL Optimization and Tuning

Oracle高性能SQL引擎剖析

SQL优化与调优机制详解



黄玮◎著

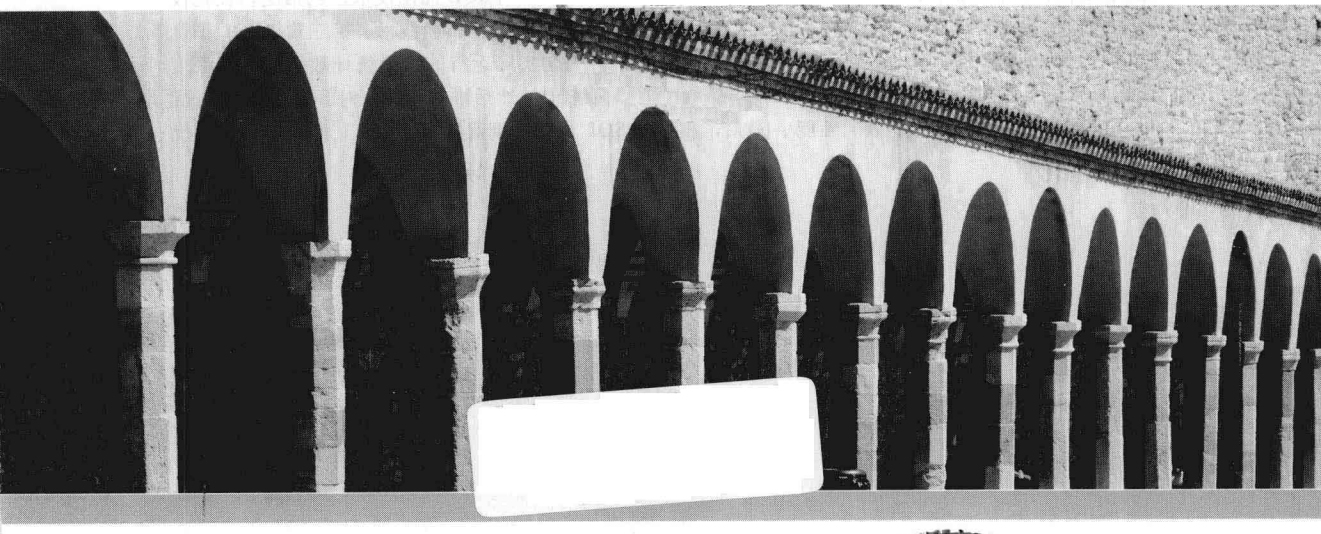
机械工业出版社
China Machine Press

数据库 技术丛书

Oracle High Performance SQL Engine
SQL Optimization and Tuning

Oracle高性能SQL引擎剖析

SQL优化与调优机制详解



黄玮◎著

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

Oracle 高性能 SQL 引擎剖析：SQL 优化与调优机制详解 / 黄玮著 . —北京：机械工业出版社，2013.1
（数据库技术丛书）

ISBN 978-7-111-40704-1

I. O… II. 黄 III. 关系数据库系统—数据库管理系统 IV. TP311.138

中国版本图书馆 CIP 数据核字（2012）第 291004 号

本书是 Oracle 数据库性能优化领域的里程碑之作，通过对 Oracle 高性能 SQL 引擎的深入剖析，深刻揭示了 Oracle SQL 的优化与调优机制、核心技术，以及性能优化的思想方法，是作者十余年工作经验与研究心得的结晶，包含大量技巧与最佳实践。

全书共分为三个部分。第一篇“执行计划”，首先详细讲解了各种执行计划的含义与操作，为后面的深入分析打下基础；然后重点讲解了执行计划在 SQL 语句执行的生命周期中所处的位置和作用，SQL 引擎如何生成执行计划以及如何获取 SQL 语句的执行计划，如何从各种数据源显示和查看已经生成执行计划等。第二篇“SQL 优化技术”，深入分析了 Oracle 的 SQL 优化技术，包括逻辑优化技术和物理优化技术，用大量示例详尽分析了 Oracle 中现有的各种查询转换技术。第三篇“SQL 调优技术”，深入剖析了 Oracle 提供的各项调优技术，首先对语句实际运行的性能统计数据进行了深度分析，介绍了各项统计数据是由什么操作导致的以及如何统计；然后讲解了如何对 SQL 语句进行优化，以及如何获得稳定、高效的性能；最后，根据对 SQL 优化及调优技术的分析，总结如何快速优化 SQL 的思路。

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：吴 怡
藁城市京瑞印刷有限公司印刷
2013 年 1 月第 1 版第 1 次印刷
186mm × 240mm • 29.5 印张
标准书号：ISBN 978-7-111-40704-1
定 价：89.00 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

序

——十年磨剑，一朝动世

我和黄玮的相识是从他的网站（HelloDBA.com）开始的，那时他很少出入论坛，但是坚持在自己的网站上发表一系列技术文章，我猜测 Hello DBA 是他向这个领域发出的第一声问候。

他在网站上发表的文章吸引了很多 Oracle 技术爱好者，大家在一起探讨学习，也因此而结识。黄玮还开发了一系列的小工具软件，流传范围极广，其中的 OraTracer 是学习 Oracle 内部原理的极好助手。

我在编辑《Oracle DBA 手记 3》一书时，曾经向黄玮索稿，他那时豪爽地说：你看好哪篇稿件？我来改。

于是，在那本书中，我收入了他 3 篇文章，在该书的序言中，我这样介绍他：

> 黄玮似乎是一位独行侠，居于东南之地，独树一帜，而与外界绝少往来，然每发一文必如春雷，惊动于世，其对 Oracle 内部原理研究之深入、剖析之透彻无不让人拍案叫绝。然黄玮之淡泊与隐忍使我们很少看到庐山真面目。他视一切喧嚣如无物，很少出入论坛，也极少发布技术之外的言论和探讨，只是按照自己的步调时不时放出一些让识货者视为珍宝的文章。感谢他慷慨付我以妙文，才有了这本书中他精彩的分享。

正如我之前描述的，由于他的文章内容非常深入，加上后期只以英文著述，这使得能够真正理解其文章价值的人愈加稀少，我一直深以为憾事。《Oracle DBA 手记 3》收入他的文章，也是为了让更多的读者可以从他的分享中受益。

这一次收到黄玮的书稿，异常惊喜，SQL 优化与调优技术是一个复杂的主题，其核心技术 CBO 算法在不同版本中一直在演进，甚至在某些小版本中，也可能出现很大的算法改变，所以要想讲清楚成本这件事是相当困难的，而黄玮是这个方向的绝佳人选。

书中的部分内容之前已经在他的网站上阅读，这次通读书稿，掩卷回思，我认为本书大致可以分为两个部分：

- 第一部分介绍了 SQL 执行计划的各个组件的含义和作用，这一部分可以作为所有 Oracle 技术爱好者的参考书，通过查阅学习去逐步理解 Oracle SQL 的执行计划；
- 第二部分是优化器成本的计算，作者通过大量的运算去推演验证 Oracle 的 CBO 法则，这一部分内容相对艰深，需要反复阅读去理解。

黄玮是个实证派，而研究优化器算法，也唯有实证派才能够胜任，他通过大量测试进行推演，从而破解 CBO 计算法则，这不仅需要深入的计算机知识，也需要极大的耐心和毅力，这满纸的数字对于读者来说，可能会成为过眼云烟，而对于作者来说，却是锱铢必较的艰苦之旅。

我非常敬佩作者将这些知识带给我们，帮助我们破解了很多 Oracle 的技术秘密，要知道这些内容未见于任何已知的文档，也可能根本就没有这样的文档，写就这些内容，足以证明作者对于数据库技术的狂热和执着。我曾经和 Oracle 公司的开发人员交流，他们提到，由于 CBO 的复杂性和不断修正完善，在内部维护一份实时的文档也是极为困难的。

而如今，我们能够从黄玮的研究中分享成果，实在是一大幸事。

最后我还想提醒读者朋友们，本书中的部分算法推演可能并不绝对精确（这和 Oracle 的算法取舍相关），而且很多内容和数据库版本有关，作者只是提供了方法和过程，读者学习后就能够自行去开始自己的探索之旅。

我非常幸运能够先睹为快地通读了本书，也希望读者们能够认识到本书的价值，为自己的知识积累补充重要的篇章。

盖国强

云和恩墨创始人，Oracle ACE 总监，ITPUB 论坛超级版主

前　言

作为一个数据库管理系统来说，Oracle 不仅具备为用户存储和管理海量数据的能力，还能够解析用户提交结构化查询语言（Structured Query Language，SQL）的请求，高效和快速地响应请求。为了保证语句的查询效率，Oracle 开发了许多技术，从各个方面提供支持，例如，物理设计、内存管理和 SQL 语句本身的自动优化与调优。

各种基于数据库的系统中，绝大多数功能都需要通过数据库管理系统查询和操作数据，因而后台的性能对系统整体性能的影响是相当关键的。而要实现对数据的管理与查询，程序需要通过 SQL 语句对数据库发起相应的请求。也就是说，SQL 语句的性能成为影响系统性能至关重要的因素。

Oracle 数据库作为目前市场占有量最大的关系型数据库管理系统，拥有成千上万的专利技术保证数据库系统的性能。而对于应用系统性能影响重大的 SQL 引擎，Oracle 更是提供了强大的技术保障，不仅采用了主流数据库系统当中优化效果最好的优化模式——基于代价的优化模式，还引入了数量庞大的、有别于其他数据库系统的专利优化技术。这些优化技术使得 Oracle 在 SQL 优化和调优方面独树一帜。

然而，尽管 Oracle 可能已经在内部帮助我们完成了 99% 的优化工作，但剩余的 1% 仍然可能成为导致性能下降的决定因素。因此，在 Oracle 数据库系统当中，发生性能问题的例子还是层出不穷。并且，以作者的个人经历来看，这些性能问题绝大多数是由 SQL 语句引起的。因此，深入理解和掌握 Oracle 的 SQL 语句优化和调优技术，是开发人员和数据库管理员都应掌握的，也是开发出性能高效的数据库系统的必要前提。

本书结合作者多年来对 Oracle 的 SQL 内部优化机制的研究以及 SQL 调优的经验，深入分析了 Oracle 的各项 SQL 语句的优化与调优技术，侧重于对这些技术的内部机制的介绍，目的是帮助读者更加深入地了解和消化这些技术，使读者在面对和解决由 SQL 引起的性能问题时，能透彻地看清问题的本质，迅速找到有效的解决方法。

SQL 优化是指在语句运行之前，由 SQL 引擎对语句进行解析，采用各项技术对查询进

行优化，找到其最佳的执行路径，即执行计划；SQL 调优则是对已经存在执行计划的语句进行进一步调整，使其运行性能更加接近性能指标的要求，达到性能改进的目的。SQL 优化与 SQL 调优是在改善 SQL 语句性能的过程中，两个不同阶段实施的技术与方法。SQL 优化，基本上是由优化器根据当前环境和数据实施的内部优化措施；SQL 调优，则是通过对语句、物理对象以及优化环境的干预，使得优化器能够选择到最优的 SQL 执行方式。因此，这两个方面的技术内容是相辅相成的。

在 RBO 时期（9i 之前），由于优化技术的限制，使得 SQL 语句往往不能获得最佳的执行计划，从而导致性能低下，需要依赖于开发人员或数据库管理员的经验和技能，对语句进行人工干预，从而调整其性能。可以说，这个时期的优化技术是有限的，并且优化与调优过程比较独立。进入 CBO 时代后，统计数据（Statistics Data）为优化器采用更加智能和复杂的优化技术提供了有力的数据保障，越来越多的基于代价的优化技术被应用到了 Oracle 的 SQL 引擎当中。同时，这也使得自动和智能化的优化技术成为可能。在 10g、11g 当中，Oracle 引入了多项新的优化技术。这些优化技术不仅仅能针对当前的运行结果进行调优，而且能对优化环境和物理设计等影响优化效果的因素进行深入检查，提供短期和长期的各种调优建议。并且，部分调优结果还可以在语句的下一个执行周期中影响其优化方法。这就使得优化与调优技术更加紧密地结合在一起，为 SQL 语句提供一个稳定、高效的运行性能。

简单地说，我们在调整 SQL 语句性能时，需要灵活地使用 SQL 调优技术，使得 SQL 语句在优化阶段能够获得真正最佳性能的执行计划。图 0-1 描述了 SQL 优化与 SQL 调优在改进性能过程中所处的位置以及相互作用的方式。希望读者可以先通过该图体会到这两方面技术之间的关系，在后续的阅读和学习过程中能清楚地知道各种技术在提高性能过程中的作用。

无论是 SQL 优化还是调优，它们的核心内容都是执行计划（Execution Plan）。许多相关技术，例如查询转换、SQL 调用配置（SQL Profile）都是围绕执行计划这一要素展开的。本书将从执行计划开始，逐步剖析 Oracle 的 SQL 优化与调优的相关技术。

本书分为三篇、共八章。

第一篇解释什么是 SQL 语句的执行计划。主要内容包括：执行计划在 SQL 语句执行的生命周期中所处的位置和作用；SQL 引擎如何生成执行计划以及如何手工生成一条语句的执行计划；如何从各种数据源显示和查看已经生成的执行计划。本篇的重点部分则是向读者解释如何读懂执行计划，包含执行计划结构解析、各种数据所代表的含义、执行计划各种操作的含义和示例，以及执行计划与内部函数之间的本质关系。

第二篇深入分析 Oracle 的 SQL 优化技术。总体上来说，SQL 优化技术可以分为两类：逻辑优化技术和物理优化技术。其中，逻辑优化主要是指查询转换技术。本篇当中详尽地分析了 10g、11g 中现有的各种查询转换技术，并给出实际示例帮助读者理解这些技术。而物理优化则指的是优化器通过代价估算来选择最佳的执行计划。优化器要正确估算执行计划及其操作的代价，则需要准确的统计数据的支持。因此本书在分析优化器的代价估算方法之前，先分析 Oracle 如何收集、统计系统和对象的统计数据。然后，结合作者推导出的各种代价估算公式，演示了各种情形下的代价计算方法。

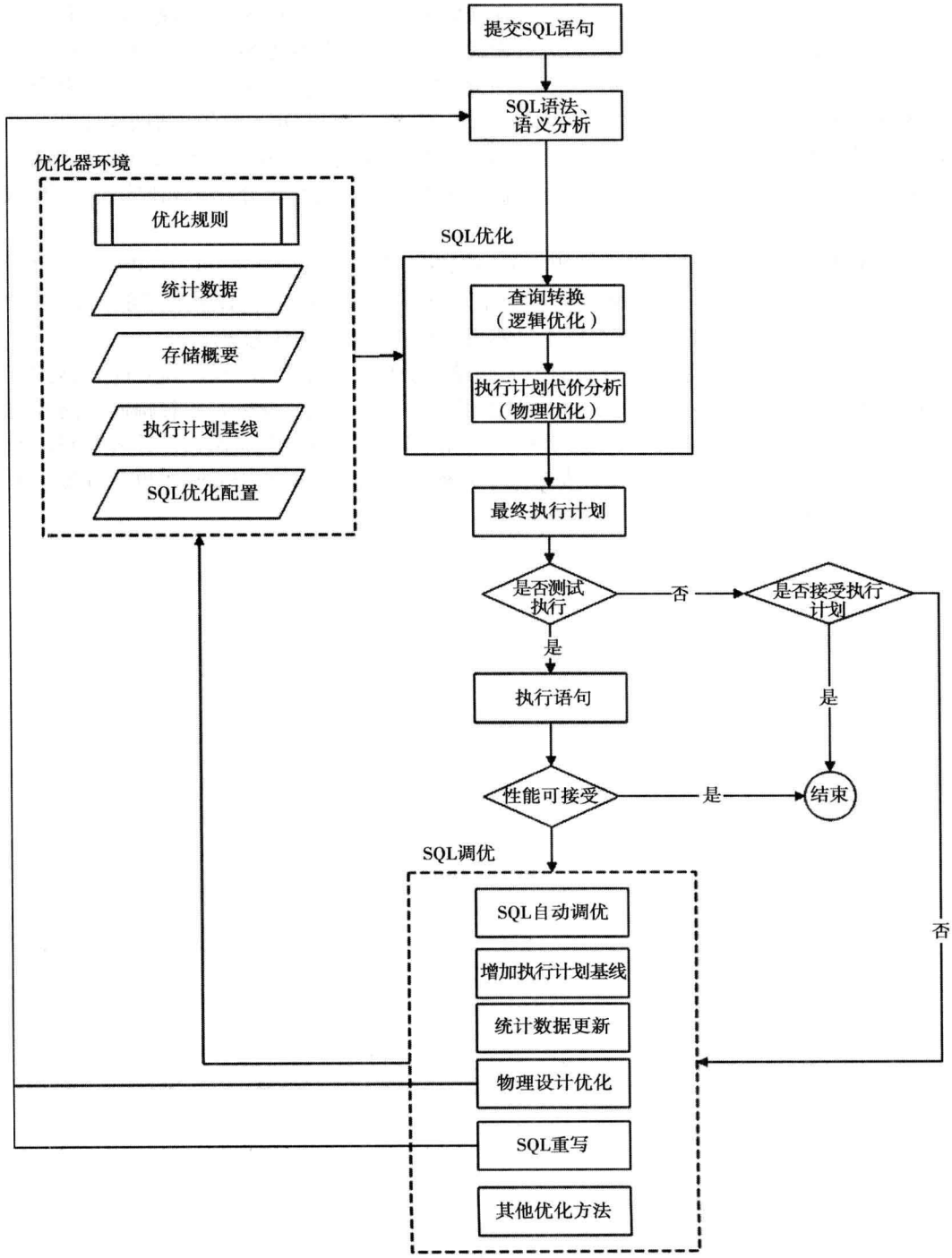


图 0-1　SQL 性能改进过程

第三篇详细介绍和分析 Oracle 提供的各项调优技术。在进行 SQL 调优时，尤其是对已经产生了实际的性能问题的 SQL 语句进行调优时，我们需要参考 SQL 语句的实际运行的性能统计数据。因此，本篇先对语句实际运行的性能统计数据进行了深度分析，向读者介绍了各项统计数据是由什么操作导致的以及如何统计。其次，介绍 Oracle 的各项调优技术，深入分析这些技术如何对 SQL 语句进行优化，以及如何使用这些技术帮助 SQL 语句获得稳定、高效的性能。最后，依据对 SQL 优化及调优技术的分析，向读者介绍了如何快速优化 SQL 的思路。

在本书中有不少演示代码用于解释相关知识点。读者请执行脚本 00_01_Prepare.sql（进入 Oracle 11g 中再执行 00_01_Prepare_11g.sql）准备演示环境。本书所介绍的内容将基于 10g 和 11g 的版本特性，其中有部分脚本只能用于相应版本，请注意提示。

www.HelloDBA.com 是作者专属的个人网站，是作者分享个人经验和心得的重要平台。本书中使用的所有代码脚本都会放在本网站上供读者免费下载。此外，因篇幅问题而删减的附录“SQL 提示的含义和示例”也将以电子文档的形式提供读者免费阅读。读者在阅读本书过程中，如有任何疑问，可以给作者发送电子邮件：fuyuncat@gmail.com，也可以关注作者的微博（http://www.weibo.com/u/1407854870）。

目　录

第一篇
执行计划

执行计划是指示 Oracle 如何获取和过滤数据、产生最终结果集，是影响 SQL 语句执行性能的关键因素。我们在深入了解执行计划之前，首先需要知道执行计划是在什么时候产生的，以及如何让 SQL 引擎为语句生成执行计划。

在深入了解执行计划之前，我们先了解 SQL 语句的处理执行过程。当一条语句提交到 Oracle 后，SQL 引擎会分为三个步骤对其处理和执行：解析（Parse）、执行（Execute）和获取（Fetch），分别由 SQL 引擎的不同组件完成。SQL 引擎的组件如图 1-1 所示。

1. SQL 编译器（SQL Compiler）

将语句编译到一个共享游标中。SQL 编译器由解析器（Parser）、查询优化器（Query Optimizer）和行源生成器（Row Source Generator）组成。

- 解析器（Parser）——执行对 SQL 语句的语法、语义分析，将查询中的视图展开、划分为小的查询块。
- 查询优化器（Query Optimizer）——为语句生成一组可能被使用的执行计划，估算出每个执行计划的代价，并调用计划生成器（Plan Generator）生成计划，比较计划的代价，最终选择选择一个代价最小的计划。查询优化器由查询转换器（Query Transform）、代价估算器（Estimator）和计划生成器（Plan Generator）组成。

注意，上述优化器实际上指的是基于代价的优化器（Cost Based Optimizer，CBO），CBO 也是当前采用的所有优化和调优技术的核心基础。

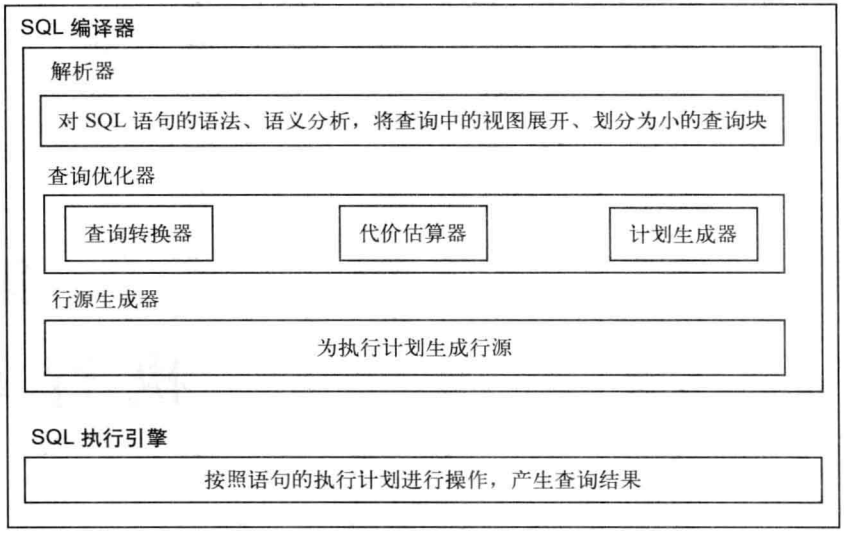


图 1-1 SQL 引擎结构及其组件示意图

查询转换器（Query Transformer）——查询转换器决定是否重写用户的查询（包括视图合并、子查询反嵌套），以生成更好的查询计划。

代价估算器（Estimator）——代价估算器使用统计数据来估算操作的选择率（Selectivity）、返回数据集的势（Cardinality）和代价，并最终估算出整个执行计划的代价。

计划生成器（Plan Generator）——计划生成器会考虑可能的访问路径（Access Path）、关联方法和关联顺序，生成不同的执行计划，让查询优化器从这些计划中选择出代价最小的一个计划。

❑ 行源生成器（Row Source Generator）——行源生成器从优化器接收到优化的执行计划后，为该计划生成行源（Row Source）。行源是一个可被迭代控制的结构体，它能以迭代方式处理一组数据行、并生成一组数据行。

2. SQL 执行引擎（SQL Execution Engine）

SQL 执行引擎依照语句的执行计划进行操作，产生查询结果。在每一个操作中，SQL 执行引擎会以迭代方式执行行源、生成数据行。

提示：当 Oracle 引入一些新的优化技术时，会出现一些新的组件，例如，SQL 进化管理器（SPM）、SQL 性能分析器（SPA）等，这些组件会与 SQL 引擎的组件融合，提供更好的优化和调优方法。

第 1 章　生成和显示执行计划

任何一条 SQL 语句要正确运行并返回结果，SQL 执行引擎都必须获得一个相应的执行计划。当缓存当中找不到与当前环境相匹配的执行计划时，SQL 编译器会解析和生成一个相应的执行计划。已经生成的执行计划会驻留在缓存当中，直至其失效或者被清出缓存。

如果想要生成和显示一条语句的执行计划，方法有多种。大致上分为两类：从内存或者历史数据中读取曾经执行语句的执行计划；使用 Explain Plan 命令解析语句后，从表 PLAN_TABLE 获得生成的执行计划。

在本章中，我们将会了解到以下内容：

- 在 Oracle 中，SQL 语句如何生成执行计划。
- 如何获取和显示 SQL 语句的执行计划。

1.1　生成执行计划

在 Oracle 中，任何一条语句在解析过程中都会生成一个唯一的数值标识，即 SQL_ID。而同一条语句，在解析过程中，可能会因为执行环境的改变（例如某些优化参数被改变）而生成多个版本的游标，不同的游标会有不同的执行计划。每个游标都会按顺序赋予一个序列号，即 CHILD_NUMBER，一条语句生成的第一个游标的 CHILD_NUMBER 为 0；相应的，Oracle 会为每个执行计划生成一个哈希值以作区分。而多个不同版本的游标，其执行计划可能会相同，也可能不同。

因此，我们可以知道，一条合法的 SQL 语句在执行过后，在内存中最少会有一个执行计划与其游标相对应。当前实例内存（Library Cache）中的执行计划可以通过视图 V$SQL_PLAN 读取（RAC 环境中，可以通过视图 gv$sql_plan 查看 RAC 当中其他实例上的执行计划）。在启用了自动负载知识库（Automatic Workload Repository，AWR，10g 及以后版本），Oracle 会将内存中的执行计划存储在历史数据当中，我们可以通过查询语句或者 Oracle 提供的包 DBMS_XPLAN 从历史数据中读取。此外，从 10g 开始，Oracle 还提供一个自动优化工具 DBMS_SQLTUNE 对单个或一组语句进行自动优化，它可以在一段时间内捕捉内存中语句和执行计划来生成一组 SQL 集（或者称 SQL 调优集，SQL Tuning Set），我们同样可以从 SQL 集中读取和显示语句的执行计划。在 11g 当中，Oracle 又引入了 SQL 执行计划管理（SQL Plan Management）的特性，可以将语句的一个或多个执行计划存储在一个执行计划基线（Plan Baseline）当中，我们同样可以读取基于执行计划基线生成的计划。

提示：AWR 的历史数据、执行计划基线都是有保存期限的，可以通过相关参数设置。

除了通过执行 SQL 让 Oracle 处理引擎在内存中生成执行计划外，我们还可以通过命令 Explain Plan 让优化器仅对 SQL 语句进行解释，生成查询计划。由于语句并不会实际执行，

因此它可以含有没有赋值的绑定变量。

执行 Explain Plan 命令后，Oracle 会将解释生成的查询计划插入表 SYS.PLAN_TABLE$（10g 之前，表名为 PLAN_TABLE；10g 之后，通过公共同义词 PLAN_TABLE 指向 SYS.PLAN_TABLE$）中。我们就可以通过查询语句或者 Oracle 提供的包 DBMS_XPLAN 从该表中读取查询计划。注意，通过 Explain Plan 解释出来的查询计划不会被缓存到内存中以便在语句执行时重用，我们在缓存当中看到的是类似“explain plan for <SQL>”的形式。

要注意的是，如果要解析语句的执行计划，用户必须拥有语句中对象及其依赖对象的权限。如果语句中存在视图，用户必须有对视图依赖表的查询权限。例如，当一个用户 A 基于表 T 创建了一个视图 V，并将视图的查询权限赋予了用户 B，那么用户 B 仅能通过视图查询表的数据，但无法直接调用 Explain Plan 命令解析基于该视图的查询的执行计划。

1.2 显示执行计划

我们现在知道，有三个途径可以获取查询计划：v$sql_plan、dba_hist_sql_plan 和 PLAN_TABLE。如果需要读取一条 SQL 语句的执行计划，就需要知道该条语句的 SQL_ID，如果该语句存在多个游标或者执行计划，则还需要知道游标的 CHILD_NUMBER 或计划的哈希值（可选）。而无论我们通过哪个途径来获取执行计划，显示方式主要是两种：语句查询和包 DBMS_XPLAN 显示。

1.2.1 通过查询语句显示计划

通过查询语句从一些视图里读出执行计划并作格式化输出的方法都非常相似，这里以 v$sql_plan 视图为例，示例程序见代码清单 1-1。

代码清单 1-1 显示执行计划（查询语句）

```
HELLODBA.COM>col "Query Plan_Table" format a30
--  提示：SQL_ID 可以从视图 v$sql_text 和 dba_hist_sqltext（或 stats$sqltext）等视图中查询获得。
HELLODBA.COM>select id,lpad(' ', 2*(level-1))||operation||' '||options||' '||
  2         object_name||' '||decode(id, 0, 'Cost='||cost) "Query Plan_Table"
  3  from v$sql_plan
  4  start with id = 0
  5         and sql_id = 'dq7gjn1yrpcyz'
  6         and plan_hash_value = 616708042
  7  connect by prior id = parent_id
  8         and sql_id = 'dq7gjn1yrpcyz'
  9         and plan_hash_value = 616708042;

        ID Query Plan_Table
---------- ------------------------------
         0 SELECT STATEMENT   Cost=2
         1   TABLE ACCESS FULL T_USERS
```

1.2.2 通过包 DBMS_XPLAN 显示计划

这个包可以根据我们选择的函数以及输入的参数来格式化显示相关的执行计划，在我们

随后的内容中，主要会使用（也推荐读者使用）该工具显示执行计划。

DBMS_XPLAN 含有 5 个函数用于输出格式化的执行计划，display、display_cursor、display_awr、display_sqlset 和 display_sql_plan_baseline，分别用于显示 Explain Plan 命令解释的计划、内存中的执行计划、AWR 历史数据中的计划、SQL 优化集中语句的计划、执行计划基线（关于 SQL 优化集和执行计划基线，我们会在后面第 7 章中具体介绍）。它们都是管道化表函数（Pipelined Table Function），返回的结果是一个系统自定义的集合数据类型 dbms_xplan_type_table。我们可以通过表函数（Table）进行映射后进行查询。

1.2.2.1 DISPLAY

DISPLAY 函数用于显示存储在 PLAN_TABLE 中的执行计划，或与 PLAN_TABLE 拥有相同结构的表中的执行计划。此外，如果从视图 v$sql_plan_statistics_all 可以获得该执行计划的相关统计数据，DISPLAY 也可以格式化输出这些数据。

参数描述：

- TABLE_NAME：存储查询计划的表名（不区分大小写），默认值为 PLAN_TABLE。
- STATEMENT_ID：SQL 语句 ID。在 PLAN_TABLE 中，每条语句的执行计划都会有一个唯一的 ID 来标识。这个 ID 可以在执行 Explain Plan 命令时，通过 Set Statement_id 子句来指定。如果输入为 NULL，则会获取最近一条被解释的语句。
- FORMAT：输出格式。在 DISPLAY 函数中，有以下预定义的格式（模板）可供选择：
  - 'BASIC'：基本格式。输出的内容最少，仅仅输出查询计划中每个操作的 ID、名称和选项以及操作的对象名。
  - 'TYPICAL'：典型格式。输出的内容是我们进行语句调优时大多数情况下所需要的信息。除了基本格式中的内容外，还会输出优化器估算出的每个操作的记录行数、字节数、代价和时间，以及相关的提示信息（如远程 SQL、优化器建议等）。如果存在谓词（Predicate）条件，还会输出每个操作中的过滤（Filter）条件和访问（Access）条件。此外，如果查询涉及分区表，还会输出分区裁剪信息；如果查询涉及并行查询，还会输出并行操作的相关信息（如表队列信息、并行查询分布方式等）。这种格式是默认格式。
  - 'SERIAL'：串行执行格式。这种格式和典型格式的输出内容基本一致，不同之处在于，对并行查询，它不会输出相关的并行内容。
  - 'ALL'：完全格式。输出的内容相对完整。除了典型格式的内容以外，还会输出字段投射信息和别名信息。

除了这些预定义的格式外，用户还可以通过在格式化字符串中添加或者屏蔽一些关键词进行细化输出。每一个细化选项代表了输出内容中的单个信息（可能是执行计划表中的一个列，也可能是一个附加信息）。在 DISPLAY 函数中，以下细化控制选项可供选择：

- ROWS：优化器估算出的记录行数；
- BYTES：优化器估算出的字节数；