为什么有时Oracle数据库不用索引来查找数据？ PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/285/2021_2022__E4_B8_BA_E4_BB_80_E4_B9_88_E6_c67_285689.htm 当你运用SQL语言，向数据库发布一条查询语句时，ORACLE将伴随产生一个“执行计划”，也就是该语句将通过何种数据搜索方案执行，是通过全表扫描、还是通过索引搜寻等其它方式。搜索方案的选用与Oracle的优化器息息相关。 SQL语句的执行步骤 一条SQL语句的处理过程要经过以下几个步骤。 1 语法分析 分析语句的语法是否符合规范，衡量语句中各表达式的意义。 2 语义分析 检查语句中涉及的所有数据库对象是否存在，且用户有相应的权限。 3 视图转换 将涉及视图的查询语句转换为相应的对基表查询语句。 4 表达式转换 将复杂的SQL表达式转换为较简单的等效连接表达式。 5 选择优化器 不同的优化器一般产生不同的“执行计划” 6 选择连接方式 ORACLE有三种连接方式，对多表连接Oracle可选择适当的连接方式。 7 选择连接顺序 对多表连接Oracle选择哪一对表先连接，选择这两表中哪个表做为源数据表。 8 选择数据的搜索路径 根据以上条件选择合适的数据搜索路径，如是选用全表搜索还是利用索引或是其他的方式。 9 运行“执行计划” Oracle的优化器 Oracle有两种优化器：基于规则的优化器（RBO， Rule Based Optimizer），和基于代价的优化器（CBO， Cost Based Optimizer）。 RBO自ORACLE 6版以来被采用，有着一套严格的使用规则，只要你按照它去写SQL语句，无论数据表中的内容怎样，也不会影响到你的“执行计划”，也就是说对数据不“敏感”，Oracle公司已经不再发展这种技术了。

CBO自ORACLE 7版被引入，ORACLE自7版以来采用的许多新技术都是基于CBO的，如星型连接排列查询，哈希连接查询，和并行查询等。CBO计算各种可能“ 执行计划 ”的“ 代价 ”，即cost，从中选用cost最低的方案，作为实际运行方案。各“ 执行计划 ”的cost的计算根据，依赖于数据表中数据的统计分布，Oracle数据库本身对该统计分布并不清楚，须要分析表和相关的索引，才能搜集到CBO所需的数据。 一般而言，CBO所选择的“ 执行计划 ”都不会比RBO的“ 执行计划 ”差，而且相对而言，CBO对程序员的要求没有RBO那么苛刻，节省了程序员为了从多个可能的“ 执行计划 ”中选择一个最优的方案而花费的调试时间，但在某些场合下也会存在问题。 较典型的问题有：有时，表明明建有索引，但查询过程显然没有用到相关的索引，导致查询过程耗时漫长，占用资源巨大，问题到底出在哪儿呢？按照以下顺序查找，基本上能发现原因所在。 查找原因的步骤 首先，我们要确定数据库运行在何种优化模式下，相应的参数是：optimizer_mode。可在svrmgrl中运行“ show parameter optimizer_mode"来查看。Oracle V7以来缺省的设置应是"choose"，即如果对已分析的表查询的话选择CBO，否则选择RBO。如果该参数设为“ rule ”，则不论表是否分析过，一概选用RBO，除非在语句中用hint强制。 其次，检查被索引的列或组合索引的首列是否出现在PL/SQL语句的WHERE子句中，这是“ 执行计划 ”能用到相关索引的必要条件。 第三，看采用了哪种类型的连接方式。Oracle的共有Sort Merge Join（SMJ）、Hash Join（HJ）和Nested Loop Join（NL）。在两张表连接，且内表的目标列上建有索引时，只有Nested Loop才能有效地利用到该索引

。SMJ即使相关列上建有索引，最多只能因索引的存在，避免数据排序过程。HJ由于须做HASH运算，索引的存在对数据查询速度几乎没有影响。 第四，看连接顺序是否允许使用相关索引。假设表emp的deptno列上有索引，表dept的列deptno上无索引，WHERE语句有emp.deptno=dept.deptno条件。在做NL连接时，emp做为外表，先被访问，由于连接机制原因，外表的数据访问方式是全表扫描，emp.deptno上的索引显然是用不上，最多在其上做索引全扫描或索引快速全扫描。 第五，是否用到系统数据字典表或视图。由于系统数据字典表都未被分析过，可能导致极差的“ 执行计划 ”。但是不要擅自对数据字典表做分析，否则可能导致死锁，或系统性能下降。 第六，索引列是否函数的参数。如是，索引在查询时用不上。 第七，是否存在潜在的数据类型转换。如将字符型数据与数值型数据比较，Oracle会自动将字符型用to_number()函数进行转换，从而导致第六种现象的发生。第八，是否为表和相关的索引搜集足够的统计数据。对数据经常有增、删、改的表最好定期对表和索引进行分析，可用SQL语句“ analyze table xxxx compute statistics for all indexes."。Oracle掌握了充分反映实际的统计数据，才有可能做出正确的选择。 第九，索引列的选择性不高。 我们假设典型情况，有表emp，共有一百万行数据，但其中的emp.deptno列，数据只有4种不同的值，如10、20、30、40。虽然emp数据行有很多，ORACLE缺省认定表中列的值是在所有数据行均匀分布的，也就是说每种deptno值各有25万数据行与之对应。假设SQL搜索条件DEPTNO=10，利用deptno列上的索引进行数据搜索效率，往往不比全表扫描的高，Oracle理所当然对索引

“ 视而不见 ”，认为该索引的选择性不高。 但我们考虑另一种情况，如果一百万数据行实际不是在4种deptno值间平均分配，其中有99万行对应着值10，5000行对应值20，3000行对应值30，2000行对应值40。在这种数据分布图案中对除值为10外的其它deptno值搜索时，毫无疑问，如果索引能被应用，那么效率会高出很多。我们可以采用对该索引列进行单独分析，或用analyze语句对该列建立直方图，对该列搜集足够的统计数据，使Oracle在搜索选择性较高的值能用上索引。 第十，索引列值是否可为空（NULL）。如果索引列值可以是空值，在SQL语句中那些需要返回NULL值的操作，将不会用到索引，如COUNT（*），而是用全表扫描。这是因为索引中存储值不能为全空。 第十一，看是否有用到并行查询（PQO）。并行查询将不会用到索引。 第十二，看PL/SQL语句中是否有用到bind变量。由于数据库不知道bind变量具体是什么值，在做非相等连接时，如“ ”,“ like ”等。Oracle将引用缺省值，在某些情况下会对执行计划造成影响。 如果从以上几个方面都查不出原因的话，我们只好用采用在语句中加hint的方式强制Oracle使用最优的“ 执行计划 ”。 hint采用注释的方式，有行注释和段注释两种方式。 如我们想要用到A表的IND_COL1索引的话，可采用以下方式：“ SELECT /* INDEX（A IND_COL1）*/ * FROM A WHERE COL1 = XXX." 注意，注释符必须跟在SELECT之后，且注释中的“ ”要紧跟着注释起始符“ /* ”或“ -- ”，否则hint就被认为是一般注释，对PL/SQL语句的执行不产生任何影响。 两种有效的跟踪调试方法 Oracle提供了两种有效的工具来跟踪调试PL/SQL语句的执行计划。 一种是EXPLAIN TABLE方式。用户必须首先在

自己的模式（SCHEMA）下，建立PLAN_TABLE表，执行计划的每一步骤都将记录在该表中，建表SQL脚本为在${Oracle_HOME}/rdbms/admin/下的utlxplan.sql。 打开SQL*PLUS，输入“ SET AUTOTRACE ON ”，然后运行待调试的SQL语句。在给出查询结果后，Oracle将显示相应的“ 执行计划 ”，包括优化器类型、执行代价、连接方式、连接顺序、数据搜索路径以及相应的连续读、物理读等资源代价。 如果我们不能确定需要跟踪的具体SQL语句，比如某个应用使用一段时间后，响应速度忽然变慢。我们这时可以利用Oracle提供的另一个有力工具TKPROF，对应用的执行过程全程跟踪。 我们要先在系统视图V$session中，可根据USERID或MACHINE，查出相应的SID和SERIAL#。 以SYS或其他有执行DBMS_SYSTEM程序包的用户连接数据库，执行“ EXECUTE DBMS_SYSTEM.SET_SQL_TRACE_IN_session（SID，SERIAL#，TRUE）；”。 然后运行应用程序，这时在服务器端，数据库参数“ USER_DUMP_DEST ”指示的目录下，会生成ora__xxxx.trc文件，其中xxxx为被跟踪应用的操作系统进程号。 应用程序执行完成后，用命令tkprof对该文件进行分析。命令示例：“ tkprof tracefile outputfile explain=userid/password"。在操作系统Oracle用户下，键入“ tkprof ”，会有详细的命令帮助。分析后的输出文件outputfile中，有每一条PL/SQL语句的“ 执行计划 ”、CPU占用、物理读次数、逻辑读次数、执行时长等重要信息。根据输出文件的信息，我们可以很快发现应用中哪条PL/SQL语句是问题的症结所在。