ORACLE 的查询优化

杨晓强　朱卫东　（北方交通大学计算中心　100044）

摘要：本文主要讨论了 ORACLE 访问数据的两种方法和其优化器工作原理，并在此基础上介绍了如何对 SQL 语句进行优化。

关键词：优化　索引　表扫描　索引扫描

我们在使用数据库编程时，经常会遇到大量的数据库查询语句，对于同样一个目的，可以编写出不同的 SQL 语句，但不同的 SQL 语句会使数据库的响应速度大相径庭。据统计大约 90％的性能问题是由于应用开发程序员或用户使用了不恰当的查询语句而使系统的响应速度减慢，因此提高书写 SQL 语句的质量、优化查询语句对软件性能的提高有很大的作用，在实际运行当中，也能节省一大笔开销。

一、什么时候应该考虑优化

经验规则说明了在下列情况下应该考虑查询优化，如果 SQL 语句

·有表连接

·有子查询

·访问的表在 PC 机或局域网上超过 500 行、在超级小型机上超过 10000 行、在大型 机上超过 30000 行。

·有排序操作且在 PC 机或局域网上超过 200 行、在超级小型机上超过 500 行、在大 型机上超过 1000 行，如果 SQL 语句包含下列中的任何一个，ORACLE 将对数据排序。

·ORDER BY、· GROUP BY、·UNION 、·DISTINCT、·某些表连接。

SQL 语句的好坏往往同实际运行系统的数据库结构、记录数量等具体情况有关，我们首先应当对 ORACLE 的数据库管理系统的基本规律有一些了解，这样才能对查询进行优化时行之有据。

二、ORACLE 访问数据的两种方法

ORACLE 访问表中的数据时，选择下面两种方法之一：

·索引扫描，使用索引访问数据

·表扫描，读表中所有页

当 ORACLE 使用表扫描时，ORCALE 直接读表中所有的页，从中直接找出所有符合条件的记录；当 ORACLE 使用索引扫描访问数据时，它根据索引值在索引中找到满足该值的行所对应的 ROWID。然后根据 ROWID 直接找到所需要的行。当对一个表进行查询时，如果返回行数占全表总行数的 10％至 15％时，使用索引将能极大地优化查询的性能。一般考虑当返回的行数占全表总行数的 25％以下时，就应该建立索引，下面是建立索引的一些基本规则。

·唯一值较多的字段应建立索引，而当唯一值较少时，例如'性别'只有'男'和 '女'两种值，对于这种字段，就不应建立索引，以免降低访问数据的速度。

·最大值或最小值经常被查询的字段应建立索引。

·经常作为多个表连接的字段，对这种字段也应建立索引以提高表连接速度。

·查询命中率较高的字段。

·建立复合索引时，应将唯一值较多的字段放在前面，而唯一值较少的放在后面，当有多 个字段唯一值数目大致相同时，应将需经常查询的字段放在前面。

当查询必须访问表中 40％以上的行时，表扫描将比索引扫描更高效。因为通过索引访问数据，ORACLE 对每行将需要完成两次物理上的 I/O 操作：

·第一次 I/O 操作是读索引，并取得 ROWID。

·第二次 I/O 操作是根据 ROWID 访问基表上的行。

四、ORACLE 优化器工作原理

由于 ORACLE 是关系型数据库，SQL 语句是面向结果而不是面向过程的查询语言，所以它有一个基于成本的优化器，为一个及时的查询提供一个最佳的执行策略，这个执行策略就是执行这个查询所需的一系列步骤。相

同的一条 SQL 语句可以有许多不同的执行策略，优化器将估算出全部执行方法中时间最少的也就是成本最低的那一种方法，一般来说，最为重要的就是使用什么索引和采取何种表连接方式。而所有优化的进行都是基于用户所使用的语句中的 WHERE 子句。除非访问表中大部分的行，否则应给 ORACLE 使用索引的机会，这一机会可在下面两种情况下出现：

·WHERE 子句涉及的列上存在索引。

·使用如图 1 所示的判定(WHERE 子句的搜索变量)编写 WHERE 子句，而不使用图 2 所示 的判定类型。

下面的一些因素也可能影响 ORACLE 是否使用索引：

```
ORACLE 在下列断言中考虑使用索引：
COLUMN BETWEEN Low-value AND High-value
COLUMN IN (list-of-value)
COLUMN LIKE
COLUMN <, >, = <, > = , =
COLUMN <, >, = (SUBQUERY)
COLUMN = X OR COLUMN1 = Y
COLUMN = X AND COLUMN = Y
```

图 1

```
ORACLE 在下列断言中不使用索引
COLUMN IS NULL
COLUMN NOT NULL
COLUMN NOT BETWEEN Lowvalue AND Highvalue
COLUMN NOT IN ()
COLUMN NOT LIKE
COLUMN = FUNCTIONNAME()
COLUMN ! =
COLUMN = X OR COLUMN = Y CONNECT BY
SELECT DISTINCT
GROUP BY
FUNCTIONNAME(COLUMN) =
```

图 2

·表的行数，行数越多使用索引的机会越大。

·可用索引类型。非唯一索引不利于性能的提高，而唯一的索引极有利于性能的提高。

五、使用索引优化 SQL 语句

在一般情况下，优化查询中我们要充分利用索引，对于有索引字段的表操作或涉及表连接的操作应尽可能的使用索引字段，下面举例特别说明，均指采用索引扫描效率更高的情况。

例如：

假设表 TELUSER 的 TELNUM 字段建立了索引，那么将：

SELECT USERID FROM TELUSER WHERE SUBSTR(TELNUM, 1, 2) = '28';①

改写成：

SELECT USERID FROM TELUSER WHERE TELNUM LIKE '28%'; ②

将能极大的提高执行效率。因为语句①只能使用的表扫描的方法来访问数据，而语句②正确地使用了索引，数据访问采用索引扫描的方法。而且我们在编写 SQL 语句时，还应注意适当的加一些冗余的搜索参数，以给优化器以更多的选择条件。另外，在判定中应避免使用不兼容的数据类型，数据类型不兼容可能使得优化器无法执行一些本来可以进行的优化操作。

六、正确地使用复合索引

一个查询要想正确地使用复合索引，必须在 WHERE 字句中引用构成复合索引的第一个字段或包括第一个字段在内的多个字段，假设索引 CALL-INDEX 由(TELNUM, CALLDATE)两个字段组成，那么下面的语句①、②都正确地使用了复合索引，查找数据采用的方法是索引扫描：

SELECT * FROM CALLRECORD WHERE CALLDAY = '98 - APR - 1' AND TELNUM = '63240500'; ①

CALLRECORD WHERE TELNUM = '63240500'; ②

SELECT * FROM CALLRECORD WHERE CALLDAY = '98 - APR - 1'; ③

语句③由于没有引用索引 CALL-INDEX 的 TELNUM 字段，所以不能使用复合索引，查找数据采用了表扫描的方法，所以效率低一些。

七、优化 OR 条件

OR 子句的效率非常低，原因是 ORACLE 只能按其

中的各个条件分别选出一个元组集，再求这些元组集的并，并是开销大的操作而且在 OR 连接的条件中只要有一个条件无合适存取路径就必须采用表扫描来处理。一般我们可以通过将其转化为几个相应的子查询的并来进行优化，因为在子查询中可以充分利用可以优化的条件进行优化。

假定在 UNITID 和 UNITNAME 上存在索引：

```
SELECT * FROM UNIT WHERE UNITID = '11110000' OR UNITNAME = '北方交大';
```

对上面的 SQL 语句重新编码为：

```
SELECT * FROM UNIT WHERE UNITID = '11110000'
UNION
SELECT * FROM UNIT WHERE UNITNAME = '北方交大' AND
UNITID ! = '11110000';
```

其执行效率将高得多，而且经验表明首先指定最明确的索引判定效率将高的多，这将允许 ORACLE 最小化不等条件的检查数目。

八、优化相关子查询

相关子查询是指对一个表进行操作时，根据该表以外的其他表中的数据来限定从该表中取出的数据。其中的查询条件依赖外层查询中的某个值，在处理时首先查找外层的某值，然后根据该值去处理内层的查询，它需要反复求值，效率很低，要尽量避免使用相关子查询以提高效率。

例如将：

```
SELECT * FROM UNIT WHERE UNITID =
(SELECT UNIT.UNITID FROM UNIT, UNITFEE WHERE
UNIT.UNITID = UNITFEE.UNITID AND UNITFEE.UNITFEE > 900)
```

改写为：

```
SELECT * FROM UNIT , UNITFEE
WHERE UNIT. UNITID = UNITFEE. UNITID AND UNITFEE.UNIFEE>900
```

九、强制表扫描

前面已经提到，当查询必须访问表中 40% 以上的行时，表扫描将比索引扫描效率更高。因为不能直接指明 ORACLE 是否使用索引，因此有必要学习如何编写强制 ORACLE 使用表扫描的判定。

如果要制止 ORACLE 使用索引，在 WHERE 子句中做一个假的算术表达式即可，通常的做法是在数字列上加一个零或者在字符列上加上一个 NULL 串。

例如：

将 `SELECT * FROM UNIT WHERE UNITID = '11110000';`

改为：`SELECT * FROM UNIT WHERE UNITID || '' = '11110000';`

在字符列上串联上一个 NULL 串。

将 `SELECT * FROM UNITFEE WHERE UNITFEE>10.00;`

改为：`SELECT * FROM UNITFEE WHERE UNITFEE + 0 > 10.00;`

在数字列上加上一个零。

十、NULL 值的索引访问

从前面我们已经了解到，对于含有 NULL 或 NOT NULL 条件的判定 ORACLE 不选择索引。有时，我们可以编写查询"某些值"的判定查找包含 NOT NULL 的行。

例如：

将 `SELECT * FROM TELUSER WHERE UNITID IS NOT NULL;`

改为：`SELECT * FROM TELUSER WHERE UNITID > = '0';`

使用上面的技术应该注意：如果大多数记录都包含 UNITID 值(非 NULL)，那么第一个查询(强制表扫描)将比第二个查询(索引扫描)高效得多。

我们还可以通过建立永久性表连接、建立视图、建立簇等方法来提高 ORACLE 的查询效率，这已超出了本文的讨论范围，有兴趣的读者可以参看[1]，[2]。

参考文献

[1] Jonathan S. sayles . ORACLE SQL * PLUS 教程. 周立等译. 学苑出版社.1996.

[2] George Koch &Robert Muller . ORACLE 7 使用与参考大全. 欣力等译. 清华大学出版社.1995.

[3] 刘智峰等. "基于 ORACLE 特性的 SQL 语句分析优化". 计算机研究与发展.790 - 794 页，1997,10.

[4] 萨师煊，王珊. 数据库系统概论. 高等教育出版社.1995.

(来稿时间：1998 年 5 月)