SQL 调优

1 索引的使用

在数据库中，索引建立的目的就是为了避免对表的扫描，以此减少大量的 I/O 操作。但是索引也会增加系统负担。比如：创建、维护索引需要耗费系统时间；存储索引需要额外的物理空间；对表中的数据进行增加、删除、修改时，索引也需要动态的维护，这就降低了数据的维护速度。因此，索引不是越多越好，而是该保证在创建索引后系统的性能可以得到提高。

创建索引要遵循一定的规则：表的主键和外键必须要有索引；对经常与其他表进行连接的表的连接字段应该建立索引；经常出现在 where 子句中的字段应该建立索引；小字段应该建立索引。

```
SQL> create table emp
  2  (empno number(5) primary key,
  3  ename varchar2(20),
  4  job varchar2(10),
  5  sal number(8,2));

表已创建。

SQL> insert into emp
  2  values(1,'Tai','clerk',3000);

已创建 1 行。

SQL> insert into emp
  2  values(2,'Fiona','manager',8000);

已创建 1 行。

SQL> select * from emp;

     EMPNO ENAME                JOB                SAL
---------- -------------------- ---------- ----------
         1 Tai                  clerk            3000
         2 Fiona                manager          8000
```

在表 emp 上为其创建唯一索引 emp_nosal_index。执行语句 set autotrace traceonly 后，再执行其他语句，会列出该语句的执行计划，而不会真正的执行语句，大大减少了优化时间。因此，本例子采用它来分析 SQL 语句的执行计划。

```
SQL> set autotrace traceonly
```

```
SQL> create unique index emp_nosal_index on emp(empno,sal)
  2  /

索引已创建。
```

```
SQL> select empno,ename from emp where empno>0
  2  /


执行计划
----------------------------------------------------------
Plan hash value: 3956160932

--------------------------------------------------------------------------------
| Id  | Operation         | Name | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |      |     2 |    50 |     2   (0)| 00:00:01 |
|*  1 |  TABLE ACCESS FULL| EMP  |     2 |    50 |     2   (0)| 00:00:01 |
--------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   1 - filter("EMPNO">0)

Note
-----
   - dynamic sampling used for this statement


统计信息
----------------------------------------------------------
          5  recursive calls
          0  db block gets
         10  consistent gets
          0  physical reads
          0  redo size
        520  bytes sent via SQL*Net to client
        416  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          0  sorts (memory)
          0  sorts (disk)
          2  rows processed
```

```
SQL> set autotrace off
```

当实例结束时，需要执行语句 set autotrace off，否则继续执行查询时，该查询语句不会被真正执行，只是输出其执行计划和统计信息。

2 SQL 语法调优

调优 SQL 局域最主要的目的就是为了避免没有必要的全表扫描、避免没有效率的查询等。

(1) 避免在索引字段中使用 IS NULL 及 IS NOT NULL

在一些 where 子句中，由于编写了劣质的 SQL，即使某些列存在索引，系统在运行该

SQL 语句时也不能使用该索引，从而采用全表扫描，这就造成了相应速度的极大降低。

```
SQL> select * from emp where empno is not null
  2  /

执行计划
----------------------------------------------------------
Plan hash value: 3956160932

--------------------------------------------------------------------------------
| Id  | Operation         | Name | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |      |     2 |    36 |     2   (0)| 00:00:01 |
|   1 |  TABLE ACCESS FULL| EMP  |     2 |    36 |     2   (0)| 00:00:01 |
--------------------------------------------------------------------------------
```

由于已经为表 emp 的两个字段 empno 及 sal 建立了索引。但是从执行计划部分截图中可以看到，其查询计划将采用全表扫描。

这种条件可以将 NULL 值补成其他有意义的值，比如 1，具体如下：

```
SQL> select empno,ename from emp where empno =1
  2  /

执行计划
----------------------------------------------------------
Plan hash value: 3637378056

--------------------------------------------------------------------------------

-----------

| Id  | Operation                    | Name        | Rows  | Bytes | Cost (%CPU)|

 Time     |

--------------------------------------------------------------------------------

-----------

|   0 | SELECT STATEMENT             |             |     1 |     8 |     1   (0)|

 00:00:01 |

|   1 |  TABLE ACCESS BY INDEX ROWID| EMP         |     1 |     8 |     1   (0)|

 00:00:01 |

|*  2 |   INDEX UNIQUE SCAN          | SYS_C009525 |     1 |       |     0   (0)|

 00:00:01 |

--------------------------------------------------------------------------------
```

从上述截图可以看出，该语句的执行计划采用了索引，从而提高了响应速度。

(2) 尽可能不采用“！=”号

索引只能告诉有什么样的数据存在于彪悍总，因此如果采用“！=”号，将会做全表扫描，具体如下：

```
SQL> select empno,ename from emp where empno !=1
  2  /

执行计划
----------------------------------------------------------
Plan hash value: 3956160932

--------------------------------------------------------------------------------
| Id  | Operation         | Name | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |      |     1 |     8 |     2   (0)| 00:00:01 |
|*  1 |  TABLE ACCESS FULL| EMP  |     1 |     8 |     2   (0)| 00:00:01 |
--------------------------------------------------------------------------------
```

从上述语句的执行计划截图可以看到，它将采用全表扫描 TABLE ACCESS FULL，这样会降低系统的性能。