# Oracle 的执行计划

## 优化定义

优化是选择最有效的执行计划来执行sql语句的过程,这是在处理任何数据的语句(select,insert,update Or delete)中的一个重要步骤.对Oracle来说,执行这样的语句有许多不同的方法,比如,将随以什么顺序访问那些表或索引的不同而不同.所使用的执行计划可以决定语句能执行的多块.Oracle中称之为优化器(Optimizer )的组件用来选择他认为最有效的

执行计划.

## 概念分析

### 共享SQL语句

**为了不重复解析相同的SQL语句(因为解析操作比较费资源，会导致性能下降)，在第一次解析之后，ORACLE将SQL语句及解析后得到的执行计划存放在内存中。**

**这块位于系统全局区域SGA(system global area)的共享池(shared buffer pool)中的内存可以被所有的数据库用户共享。因此，当你执行一个SQL语句(有时被称为一个游标)时，如果该语句和之前的执行过的某一语句完全相同，并且之前执行的该语句与其执行计划仍然在内存中存在，则ORACLE就不需要再进行分析，直接得到该语句的执行路径。ORACLE的这个功能大大地提高了SQL的执行性能并大大节省了内存的使用。使用这个功能的关键是将执行过的语句尽可能放到内存中，所以这要求有大的共享池(通过设置shared buffer pool参数值)和尽可能的使用绑定变量的方法执行SQL语句。**

### RowId

**rowid是一个伪列，既然是伪列，那么这个列就不是用户定义，而是系统自己给加上的。**

**对每个表都有一个rowid的伪列，但是表中并不物理存储ROWID列的值。不过你可以像使用其它列那样使用它，但是不能删除该列，也不能对该列的值进行修改、插入。一旦一行数据插入数据库，则rowid在该行的生命周期内是唯一的，即即使该行产生行迁移，行的rowid也不会改变**

### 为什么使用Rowid：

**rowid对访问一个表中的给定的行提供了最快的访问方法，通过ROWID可以直接定位到相应的数据块上，然后将其读到内存。我们创建一个索引时，该索引不但存储索引列的值，而且也存储索引值所对应的行的ROWID，这样我们通过索引快速找到相应行的ROWID后，通过该ROWID，就可以迅速将数据查询出来。这也就是我们使用索引查询时，速度比较快的原因。**

**在ORACLE8以前的版本中，ROWID由FILE 、BLOCK、ROW NUMBER构成。随着oracle8中对象概念的扩展，ROWID发生了变化，ROWID由OBJECT、FILE、BLOCK、ROW NUMBER构成。利用DBMS\_ROWID可以将rowid分解成上述的各部分，也可以将上述的各部分组成一个有效的rowid。**

### Recursive SQL概念

**为了执行用户发出的一个sql语句，Oracle必须执行一些额外的语句，我们将这些额外的语句称之为‘recursive calls’或‘recursive SQL statements’。如当一个DDL语句发出后，ORACLE总是隐含的发出一些recursive SQL语句，来修改数据字典信息，以便用户可以成功的执行该DDL语句。当需要的数据字典信息没有在共享内存中时，经常会发生Recursive calls，这些Recursive calls会将数据字典信息从硬盘读入内存中。用户不必关心这些recursive SQL语句的执行情况，在需要的时候，ORACLE会自动的在内部执行这些语句。当然DML语句与SELECT都可能引起recursive SQL。简单的说，我们可以将触发器视为recursive SQL。**

### Row Source(行源)

**用在查询中，由上一操作返回的符合条件的行的集合，即可以是表的全部行数据的集合；也可以是表的部分行数据的集合；也可以为对上2个row source进行连接操作(如join连接)后得到的行数据集合。**

### Predicate(谓词) ：

**一个查询中的WHERE限制条件**

### Driving Table(驱动表)：

**该表又称为外层表(OUTER TABLE)。这个概念用于嵌套与HASH连接中。如果该row source返回较多的行数据，则对所有的后续操作有负面影响。注意此处虽然翻译为驱动表，但实际上翻译为驱动行源(driving row source)更为确切。一般说来，是应用查询的限制条件后，返回较少行源的表作为驱动表，所以如果一个大表在WHERE条件有限制条件(如等值限制)，则该大表作为驱动表也是合适的，所以并不是只有较小的表可以作为驱动表，正确说法应该为应用查询的限制条件后，返回较少行源的表作为驱动表。**

**在执行计划中，应该为靠上的那个row source，后面会给出具体说明。在我们后面的描述中，一般将该表称为连接操作的row source 1。**

### Probed Table(被探查表)：

**该表又称为内层表(INNER TABLE)。在我们从驱动表中得到具体一行的数据后，在该表中寻找符合连接条件的行。所以该表应当为大表(实际上应该为返回较大row source的表)且相应的列上应该有索引。在我们后面的描述中，**

**一般将该表称为连接操作的row source 2。**

### 组合索引(concatenated index)：

**由多个列构成的索引，如create index idx\_emp on emp(col1, col2, col3, ……)，则我们称idx\_emp索引为组合索引。在组合索引中有一个重要的概念：引导列(leading column)，在上面的例子中，col1列为引导列。当我们进行查询时可以使用”where col1 = ? ”，也可以使用”where col1 = ? and col2 = ?”，这样的限制条件都会使用索引，但是”where col2 = ? ”查询就不会使用该索引。所以限制条件中包含先导列时，该限制条件才会使用该组合索引。**

### 可选择性(selectivity)：

**比较一下列中唯一键的数量和表中的行数，就可以判断该列的可选择性。如果该列的”唯一键的数量/表中的行数”的比值越接近1，则该列的可选择性越高，该列就越适合创建索引，同样索引的可选择性也越高。在可选择性高的列上进行查询时，返回的数据就较少，比较适合使用索引查询。**

## 执行计划分析

* **为了执行语句，Oracle可能必须实现许多步骤。这些步骤中的每一步可能是从数据库中物理检索数据行，或者用某种方法准备数据行，供发出语句的用户使用。Oracle用来执行语句的这些步骤的组合被称之为执行计划**
* **执行计划是SQL优化中最为复杂也是最为关键的部分，只有知道了ORACLE在内部到底是如何执行该SQL语句后，才能知道优化器选择的执行计划是否为最优的。执行计划对于DBA来说，就象财务报表对于财务人员一样重要。所以我们面临的问题主要是：**

1. 如何得到执行计划；

2. 如何分析执行计划;

**从而找出影响性能的主要问题。**

* **如何得到执行计划**

显示下面sql语句的执行计划：

SELECT \*

FROM PAGE\_LAYOUT p , PAGE\_LAYOUT\_CONTENT pc

WHERE

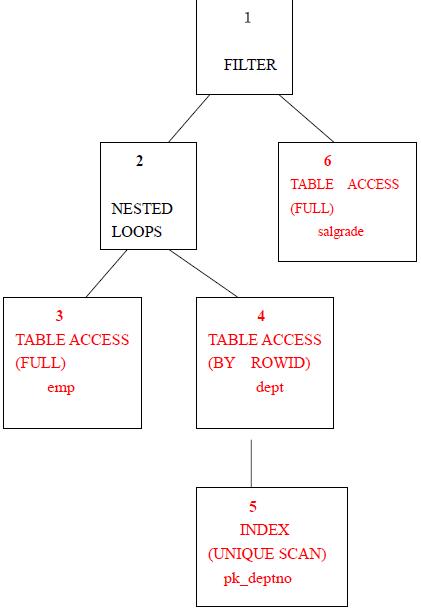
p."SID" = PC.PAGE\_LAYOUT\_SID

AND not EXISTS

(

SELECT \* FROM PAGE\_LAYOUT\_BRAND WHERE p."SID" BETWEEN 8 AND 20

)



上图是:

* + **SELECT ename, job, sal, dname**
  + **FROM emp, dept**
  + **WHERE emp.deptno = derpt.deptno**
  + **AND NOT EXISTS**
  + **( SELECT \***
  + **FROM salgrade**
  + **WHERE emp.sal BETWEEN losal AND hisal );**

的执行计划.

* **第３步和第６步分别的从EMP表和SALGRADE表读所有行。**
* **第５步在PK\_DEPTNO索引中查找由步骤３返回的每个DEPTNO值。它找出与DEPT表中相关联的那些行的ROWID。**
* **第４步从DEPT表中检索出ROWID为第５步返回的那些行。**
* **由黑色字框指出的步骤在行源上操作，如做2表之间的关联，排序，或过滤等操作。**
* **第２步实现嵌套的循环操作(相当于C语句中的嵌套循环)，接收从第３步和第４步来的行源，把来自第３步源的每一行与它第４步中相应的行连接在一起，返回结果行到第１步。**
* **第１步完成一个过滤器操作。它接收来自第２步和第６步的行源，消除掉第２步中来的，在第６步有相应行的那些行，并将来自第２步的剩下的行返回给发出语句的用户或应用**

### 实现执行计划的步骤

**执行计划中的步骤不是按照它们编号的顺序来实现的：Oracle首先实现上图树结构图形里作为叶子出现的那些步骤(例如步骤3、5、6)。由每一步返回的行称为它下一步骤的行源。然后Oracle实现父步骤**

**Oracle以下列顺序实现这些步骤**

* **首先，Oracle实现步骤３，并一行一行地将结果行返回给第２步**
* **对第３步返回的每一行，Oracle实现这些步骤**
* **Oracle实现步骤５，并将结果ROWID返回给第４步**

**Oracle实现步骤４，并将结果行返回给第２步。**

**Oracle实现步骤２，将接受来自第３步的一行和来自第４步的一行，并返回给第１步一行。**

**Oracle实现步骤６，如果有结果行的话，将它返回给第１步**

**实现步骤１，如果从步骤６返回行，Oracle将来自第２步的行返回给发出SQL语句的用户。**

## 优化器选择

* **对于以上的操作过程，可以使用first\_rows作为优化目**

**标以便于实现快速响应用户的请求。**

* **有些父步骤在它们被实现之前需要来自子步骤的所有行**

**。对这样的父步骤，直到所有行从子步骤返回之前**

**Oracle不能实现该父步骤。这样的父步骤包括排序，排**

**序一合并的连接，组功能和总计。对于这样的操作，可**

**以用all\_rows作为优化目标，使该中类型的操作耗费的**

**资源最少。**

**有时语句执行时，并不是象上面说的那样一步一步有先有后的进行，而是可能并行运行，如在实际环境中，3、5、4步可能并行运行，以便取得更好的效率。从上面的树型图上，是很难看出各个操作执行的先后顺序，而通过ORACLE生成的另一种形式的执行计划，则可以很容易的看出哪个操作先执行，哪个后执行，这样的执行计划是我们真正需要的，后面会给出详细说明。现在先来看一些预备知识**

## 访问路径—access path

**优化器在形成执行计划时需要做的一个重要选择是如何从数据库查询出需要的数据。对于SQL语句存取的任何表中的任何行，可能存在许多存取路径(存取方法)，通过它们可以定位和查询出需要的数据。优化器选择其中自认为是最优化的路径。**

**在物理层，oracle读取数据，一次读取的最小单位为数据库块(由多个连续的操作系统块组成)，一次读取的最大值由操作系统一次I/O的最大值与multiblock参数共同决定，所以即使只需要一行数据，也是将该行所在的数据库块读入内存。**

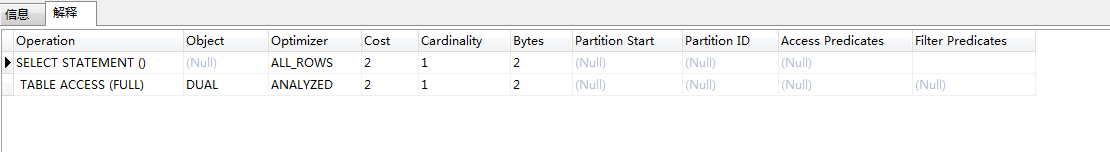
* **逻辑上，oracle用如下存取方法访问数据：**
* **1 全表扫描（Full Table Scans, FTS）**
* **2 通过ROWID的表存取（Table Access by ROWID或rowid lookup）**
* **3 索引扫描（Index Scan或index lookup）**

### 全表扫描

* **为实现全表扫描，Oracle读取表中所有的行，并检查每一行是否满足语句的WHERE限制条件。Oracle顺序读取分配给表的每个数据块，直到读到表的最高水线处(high water mark, HWM，标识表的最后一个数据块)。一个多块读操作可以使一次I/O能读取多块数据块(db\_block\_multiblock\_read\_count参数设定)，而非只读取一个数据块，这极大的减少了I/O总次数，提高了系统的吞吐量，所以利用多块读的方法可以高效实现全表扫描，而且只有在全表扫描的情况下才能使用多块读操作。在这种访问模式下，每个数据块只被读一次。由于HWM标识最后一块被读入的数据，而delete操作不影响HWM值，所以一个表的所有数据被delete后，其全表扫描的时间不会有改善，一般我们需要使用truncate命令来使HWM值归为0。幸运的是oracle 10G后，可以人工收缩HWM的值。**
* **由FTS模式读入的数据被放到高速缓存的Least Recently Used (LRU)列表的尾部，这样可以使其快速交换出内存，从而不使内存重要的数据被交换出内存。**
* **使用FTS的前提：在较大的表上不建议使用全表扫描，除非取出数据的比较多，超过总量的5% -- 10%，或你想使用并行查询功能时。**

#### 例子

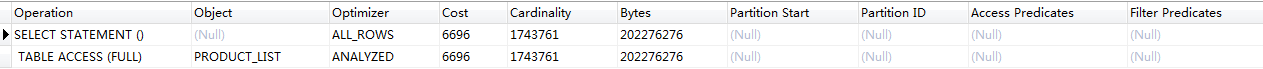
SELECT \* FROM dual;



#### 实验

全表扫描单品表(上品)—大表

SELECT \* FROM PRODUCT\_LIST p

**

耗时:

74 s

### 通过Rowid存取

* **行的ROWID指出了该行所在的数据文件、数据块以及行在该块中的位置，所以通过ROWID来存取数据可以快速定位到目标数据上，是Oracle存取单行数据的最快方法。**
* **为通过ROWID存取表，Oracle 首先要获取被选择行的ROWID，或从语句的WHERE子句中得到，或者通过表的一个或多个索引的索引扫描得到。Oracle然后以得到的ROWID为依据定位每个被选择的行。**
* **此存取方法不会用到多块读操作，一次I/O只能读取一个数据块。我们经常在执行计划中看到该存取方法，如通过索引查询数据。**

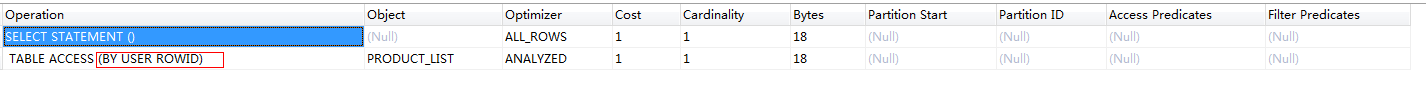
#### 例子

SELECT p."SID",ROWID as r

FROM PRODUCT\_LIST p WHERE

p.ROWID=

'AAAsydAATAALu8WAAh'



耗时:3 ms

### 索引扫描

* **通过index查找到数据对应的rowid值(对于非唯一索引可能返回多个rowid值)，然后根据rowid直接从表中得到具体的数据。**
* **一个rowid唯一的表示一行数据，该行对应的数据块是通过一次i/o得到的，该次i/o只会读取一个数据库块。**
* **在索引中，除了存储每个索引的值外，索引还存储具有此值的行对应的ROWID值。索引扫由2步组成：**

**(1) 扫描索引得到对应的rowid值。**

**(2) 通过找到的rowid从表中读出具体的数据。**

**每步都是单独的一次I/O，但对于索引，由于经常使用，绝大多数都**

**已经CACHE到内存中，所以第1步的I/O经常是逻辑I/O，即数据可以**

**从内存中得到。但是对于第2步来说，如果表比较大，则其数据不可**

**能全在内存中，则其I/O很有可能是物理I/O，这是一个机械操作，相**

**对逻辑I/O来说，极其费时间。所以如果对大表进行索引扫描，取出**

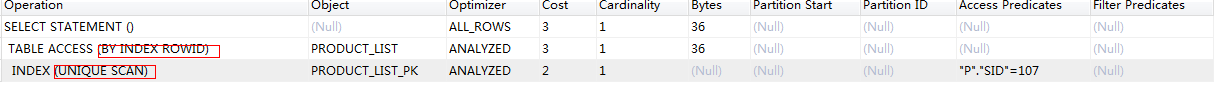
**的数据如果大于总量的5% -- 10%，使用索引扫描会效率下降很多。**

#### 例子

SELECT p."SID",p.PRODUCT\_NAME,p.PRO\_SKU

FROM PRODUCT\_LIST p WHERE

p."SID" = 107



注意:Table Access by rowid 这表面不是通过FTS存取路径访问数据,而是通过index lookup存取路径访问数据的,此例中的rowid是通过索引查找sid列的值得到的,这种方式是Index Unique Scan。

耗时: 2 ms

**根据索引的类型与where限制条件的不同，有4种类型的索引扫描**

#### 索引唯一扫描(index unique scan)

**通过唯一索引查找一个数值经常返回单个ROWID。如果该唯一索引有多个列组成(即组合索引)，则至少要有组合索引的引导列参与到该查询中，如创建一个索引：create index idx\_test on emp(ename, deptno, loc)。则select ename from emp where ename = ‘JACK’ and deptno = ‘DEV’语句可以使用该索引。如果该语句只返回一行，则存取方法称为索引唯一扫描。而select ename from emp where deptno = ‘DEV’语句则不会使用该索引，因为where子句种没有引导列。如果存在UNIQUE 或PRIMARY KEY 约束（它保证了语句只存取单行），Oracle经常实现唯一性扫描。**

##### 例子