Oracle数据库执行计划的一些基本概念（1） PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/143/2021_2022_Oracle_E6_95_B0_E6_c102_143556.htm 一．相关的概念 Rowid的概念：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，但是不能删除改列，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。 Recursive SQL概念：有时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。用户不比关心这些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(0selectivity)：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。 二．oracle访问数据的存取方法 1) 全表扫描（Full Table Scans, FTS） 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)，而不是只读取一个数据块，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十分高效地实现全表扫描，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。 使用FTS的前提条件：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。 使用全表扫描的例子： SQL> explain plan for 0select * from dual.Query Plan----------SELECT STATEMENT[CHOOSE] Cost=TABLE ACCESS FULL DUAL2) 通过ROWID的表存取（Table Access by ROWID或rowid lookup） 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。 这种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。我们会经常在执行计划中看到该存取方法，如通过索引查询数据。 使用ROWID存取的方法： SQL> explain plan for 0select * from dept where rowid = AAAAyGAADAAAAAATAAF.Query Plan----------SELECT STATEMENT

[CHOOSE] Cost=1TABLE ACCESS BY ROWID DEPT [ANALYZED]3）索引扫描（Index Scan或index lookup） 我们先通过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，这种查找方式称为索引扫描或索引查找(index lookup)。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，在此情况下该次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。索引扫描可以由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，但是对于索引，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，所以其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，是极其费时间的。所以如果多大表进行索引扫描，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。如下列所示： SQL> explain plan for 0select empno, ename from emp where empno=10.Query Plan----------SELECT STATEMENT [CHOOSE] Cost=1TABLE ACCESS BY ROWID EMP [ANALYZED]INDEX UNIQUE SCAN EMP_I1但是如果查询的数据能全在索引中找到，就可以避免进行第2步操作，避免了不必要的I/O，此时即使通过索引扫描取出的数据比较多，效率还是很高的 SQL> explain plan for 0select empno from emp where empno=10.-- 只查询empno列值Query

Plan----------SELECT STATEMENT [CHOOSE] Cost=1INDEX UNIQUE SCAN EMP_I1 100