**深入解析B-Tree索引与Bitmap位图索引的锁代价**

通过以下实验，来验证Bitmap位图索引较之普通的B-Tree索引锁的“高昂代价”。位图索引会带来“位图段级锁”，实际使用过程一定要充分了解不同索引带来的锁代价情况。 1.两种索引类型的测试表 1） 在表t\_bitmap上创建位图索引 ora11g create table t\_bitmap (

通过以下实验，来验证Bitmap位图索引较之普通的B-Tree索引锁的“高昂代价”。位图索引会带来“位图段级锁”，实际使用过程一定要充分了解不同索引带来的锁代价情况。

**1.两种索引类型的测试表**  
1）在表t\_bitmap上创建位图索引

|  |  |
| --- | --- |
| 1  2  3  4 | ora11g> create table t\_bitmap (id number(10), name varchar2(10),sex varchar2(1));  Table created.  ora11g> create bitmap index t\_bitmap\_idx on t\_bitmap(sex);  Index created. |

2）在表t\_btree上创建普通B-Tree索引

|  |  |
| --- | --- |
| 1  2  3  4 | ora11g> create table t\_btree (id number(10), name varchar2(10), sex varchar2(1));  Table created.  ora11g> create index t\_btree\_idx on t\_btree(sex);  Index created. |

**2.每张表中初始化两条数据**  
注释：  
M - Male - 表示男孩；  
F - Femail - 表示女孩。  
**1）初始化数据t\_btree表数据**

|  |  |
| --- | --- |
| 1  2  3  4 | ora11g> insert into t\_btree values (1, 'Secoooler', 'M');  1 row created.  ora11g> insert into t\_btree values (2, 'Anna','F');  1 row created. |

**2）初始化数据t\_bitmap表数据**

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | ora11g> insert into t\_bitmap values (1, 'Secoooler', 'M');  1 row created.  ora11g> insert into t\_bitmap values (2, 'Anna','F');  1 row created.  ora11g> commit;  Commit complete. |

**3）查看初始化之后的结果**  
（1）t\_btree表中包含两条数据

|  |  |
| --- | --- |
| 1  2  3  4  5 | ora11g> select \* from t\_btree;          ID NAME       S  ---------- ---------- -           1 Secoooler  M           2 Anna       F |

（2）t\_bitmap表中包含两条数据

|  |  |
| --- | --- |
| 1  2  3  4  5 | ora11g> select \* from t\_bitmap;          ID NAME       S  ---------- ---------- -           1 Secoooler  M           2 Anna       F |

**3.在两个不同的session中，对具有普通B-Tree索引表t\_btree演示插入、修改和删除“男孩”数据**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | -- 第一个session中的插入后不要提交  ora11g> insert into t\_btree values (3, 'Andy', 'M');  1 row created.    -- 第二个session中插入同样的状态数据，可以  -- 看到，插入、修改和删除均能够成功完成  ora11g> insert into t\_btree values (4, 'Tutu', 'M');  1 row created.  ora11g> update t\_btree set sex='M' where id=2;  1 row updated.  ora11g> delete from t\_btree;  2 rows deleted. |

**4.在两个不同的session中，对具有Bitmap位图索引表t\_bitmap演示插入、修改和删除“男孩”数据**1）第一个session中的插入后不要提交

|  |  |
| --- | --- |
| 1  2 | ora11g> insert into t\_bitmap values (3, 'Andy', 'M');  1 row created. |

2）第二个session中对男孩数据进行处理，可以看到，只要操作信息中涉及到位图索引列的插入、修改和删除均无法完成！！  
（1）插入测试

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | --当插入数据涉及位图索引列“sex”字段时，是无法完成的。  ra11g> insert into t\_bitmap values (4, 'Tutu', 'M');  --问题出现了：出现了“锁等待”停滞不动的现象！  --当插入数据未涉及位图索引列“sex”字段时，是可以完成的。  ora11g> insert into t\_bitmap(id,name) values (4, 'Tutu');  1 row created.  ora11g> commit;  Commit complete. |

（2）更新测试

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | --此时第二个会话的测试数据内容如下。  ora11g> select \* from t\_bitmap;          ID NAME       S  ---------- ---------- -           1 Secoooler  M           2 Anna       F           4 Tutu    --当更新位图索引列“sex”字段值为“M”时，是无法完成的。  ora11g> update t\_bitmap set sex='M' where id=1;  1 row updated.  --此时成功，是因为第一行数据的sex值本身就是“M”。  ora11g> update t\_bitmap set sex='M' where id=2;  --问题出现了：出现了“锁等待”停滞不动的现象！  ora11g> update t\_bitmap set sex='M' where id=4;  --问题出现了：出现了“锁等待”停滞不动的现象！  --另外，特别注意一下，如果更新的列不是位图索引对应的列，将不会受位图  --段级索引锁的限制。如下所示。  ora11g> update t\_bitmap set name='Xu' where id=2;  1 row updated. |

（3）删除测试

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | --当删除的数据包含位图索引列“sex”字段值为“M”时，是无法完成的。  ora11g> delete from t\_bitmap where id=1;  --问题出现了：出现了“锁等待”停滞不动的现象！  --当删除表中的所有数据时，同样的道理，也是不能删除的。  ora11g> delete from t\_bitmap;  --问题出现了：出现了“锁等待”停滞不动的现象！ |

**5.小结**

本文以对数据本身冲击力最小的插入动作为例，演示了B-Tree和Bitmap索引的锁代价。对于B-Tree索引来说，插入动作不影响其他会话的DML操作；但是，对于Bitmap索引来说，由于是索引段级锁，会导致与操作列值相关的内容被锁定（文中提到的“M”信息）。进一步，对于更新动作来说，

产生上面现象的原因：

位图索引被存储为压缩的索引值，其中包含了一个范围内的ROWID，因此ORACLE必须针对一个给定值锁定所有范围内的ROWID，不支持行级别的锁定。

换一种描述方法：使用位图索引时，一个键指向多行（成百上千），如果更新一个位图索引键，会同时将其他行对应位图索引字段进行锁定！

较之B-Tree索引优点：位图以一种压缩格式存放，因此占用的磁盘空间比B-Tree索引要小得多

较之B-Tree索引缺点：这种锁定的代价很高，会导致一些DML语句出现“锁等待”，严重影响插入、更新和删除的效率，对于高并发的系统不适用。

位图索引使用原则：  
位图索引主要用于决策支持系统或静态数据，不支持索引行级锁定。  
位图索引最好用于低cardinality列（即列的唯一值除以行数为一个很小的值，接近零），例如上面的“性别”列，列值有“M”，“F”两种。在这个基本原则的基础上，要认真考虑包含位图索引的表的操作特点，如果是并发操作高的系统，不适合使用位图索引！