基于 ORACLE 面向对象的管理信息系统

沈海燕　（铁道部科学研究院电子所　100081）

摘要：本文以沈东货运管理系统的开发为例，对基于 ORACLE 的面向对象(OOM)的管 理信息系统开发中的几个问题进行了研究和改进，从而使货运管理信息系 统的整体响应速度加快，系统性能有了明显的提高。

关键词：ORACLE　OOM　若干问题　研究和改进

1. 前言

本系统采用 ORACLE 数据库开发，两台 IBM RS6000/370 主机和上百台 WYSE 终端. 本系统是一个作业点繁多，数据来源范围广，数据类型复杂，数据量大，响应速度要求快，是一个大型面向对象的管理信息系统。

然而，在此系统的开发过程中，随着业务量，信息量的不断扩大，沈东 MIS 系统的性 能表现出存取速度和存储效率下降的趋势. 经过反复的研究实验以及 ORACLE 专家的指 导测试，从 ORACLE 内部结构入手，从具体应用设计出发等等，不断改进实体参数，优化 程序设计，调谐应用及内存分配，从而使性能达到了最佳状态。

2. 面向对象的框架结构实体参数的确定

在 ORACLE 中，对存于数据库中的部分数据库实体，例如表，聚集，表空间，索引，ROLLBACK 段等，他们所占空间情况与其创建时所存储参数的大小息息相关. 开发 人员要有足够的经验对实体的未来情况进行预测，估算出实体的初始空间，确认 缺省参数是否适用，是否需重新设置。否则，不恰当的参数可导致实体的区间数增 多而使数据块的非连性增大，而且可导致有较多的表链接现象发生，从而使访问 I/O 增多，加之沈东 MIS 系统中占用空间的这些实体数量较大，并且实体的参数没有合理设置，严重影响了数据库的性能，使开发工作出现了一些意想不到的障碍。

((示例 A))：货运子系统中，创建货票库基表实体。

```
create cluster c-hp-xzhph(hpid varchar2(12));——建聚集
create table t-hp-xzhpk ————————┐
      (hpid     varchar2(12),   │
       hph      varchar2(7),    │ 共由近 80 个
       ......                   │ 数据项组成
       jhyh     varchar2(11));—┘
cluster c-hp-xzhph(hpid); ———— 聚集
create index i-hpid on t-hp-xzhpk(hpid);————┐
create index i-hp-xzhpp on t-hp-xzhpk(hph, fzh)  │
create index i-jhyh on t-hp-xzhpk(jhyh)          │ 索引
.....                                            ┘
```

此实体生成后，数据库 t-hp-xzhpk(即基表)形成。此基表的特点是数据项 多，每条记录占用字节数长。可见，这是一个大型的基表。随着大量数据的不断 插入，即记录数的不断增多，渐渐暴露出存取速度慢，响应迟钝的毛病。到后来 甚至出现新记录不能插入的严重现象。PCTFREE, PCTUSED 是设置基表的两个重 要参数。还有 STORAGE 是为基表设定存储特征的，对大表该子句非常重要，对性 能有很大影响。本实体创建时，没有专门设置参数值，引用的是缺省参数值。

我们对本实体的缺省参数值改造如下。

```
create table t-hp-xzhpk
      (hpid varchar2(12),
      ......));
......
pctfree = 20 pctused = 35 ————┐ 新加
storage(initial 6144 next 6144 minextents 1  │ 参数
        maxextents = 20 pctincrease 60) ——┘ 设置
```

改造前缺省值：PCTFREE = 10, PCTUSED = 40. 即每块保留 10% 的自由空间用于 更新已有行，如果一个块使用空间小于 40%，该块为候选插入块。对绝大多数相对稳定的，记录不太长的基表是完全适用的。一般情况，对只作查询的表，将 PC TFREE 设为低值(<10)可增加满表搜索的性能，对综合查询子系统下的表，将 PC TFREE 设为 10，对插入较多的表，将 PCTUSED 设高一点(>40)可提高数据的存储 效率. 但具体到本例，即是

一个经常插入,修改较多,查询频繁的基表,又是一 个记录项很多的大表,所以在参数设置上要全面地综合地考虑多方因素,自由 空间 PCTFREE 要设大一点(>10),以便更新已有行;使用空间略微比缺省小一点 (<40),以便留出足够的空间插入记录。所以改设 PCTFREE=20,PCTUSED=35 更适 合此基表的特点。但 PCTFREE 与 PCTUSED 之和应小于 100,他两的合理设置可更 加有效的利用空间,对提高系统性能有重要影响。

在 STORAGE 的设置中,首先要估算实体的初始空间。比如对作大量数据录 入的表,将 INITIAL 值设的适中,NEXT 和 PCTINCREASE 设的相对较大,对作更新频繁的表,将 INITIAL 设的大些,NEXT 设的小些。本例的基表即要经常录入, 又要时常更新, 所以在参数设置上要权衡考虑,将 INITIAL 设的比缺省大多点,NEXT 和 PCTINCREASE 设的比缺省略微大一点。尽量使数据块的非连性最小,减 少表链接现象发生。可见,STORAGE 参数的正确设置即能降低存取数据库中数 据的时间,又能减少对附加空间的动态分配,对改善系统性能有着关键作用。

在开发中,由于经验不足,建基表时往往都用缺省参数值, 忽略考虑参数的合 理性。一旦出现了性能方面的问题,便束手无策。如果具备了一定的经验,在一开始建实体时,先考虑一下缺省参数的合理性,是否需重新设置。这样才能避免潜伏的 隐患对系统性能的不利影响。在沈东 MIS 系统开发中,经过对实体参数的合理改造,使系统性能有了明显改善,收到了良好的效果。

3. **建立索引(INDEX)的基本原则**

创建索引是提高检索效率最有效的方法之一。开发中,有些开发人员以此为由 动不动就建索引。熟不知索引又是以空间为代价来获取存储效率的提高,并且索引仅当表的容量较大时(占用较多存储空间)才能显示出查询速度的提高, 一般当表 的容量较小时,有没有索引影响不大。不合理的索引大量存在时, 势必降低数据库的处理性能。当出现迟缓的响应速度时,开发人员往往对系统的性能大加责骂, 实 际上"真凶"有时就是自己。那么,是不是大容量的表就可以建索引? 实践经验表明,也不能一概而论,还应统筹考虑特殊因素,才能顾全大局的正确地建立索引。

创建索引时主要遵循下述原则:

(1)在连接多个表时,最好用索引;

(2)对只读的表建较多索引,对更新频繁的表作较少索引,且可将 PCTFREE 的值 设的较大些,如 PCTFRER=30,目的是减少更新数据时带来的开销,对单一的 小表不作索引;

(3)最好在加载数据后再建索引,这可减少索引更新的次数和重新维护索引列 的时间;

(4)建立并置索引时要正确选定列的次序,通常把预计最常用的列放在最前面。

总之,不负责任地,盲目地,毫无选择地建索引并不可取。正确地建立索引不但 会极大提高查询速度,而且也极大提高系统的综合性能。沈东 MIS 系统的开发实践 证实,由于开发人员众多,基表众多,不合理索引的大量存在,是降低系统性能的重 要因素之一。

4. **聚集(CLUSTER)的合理用法**

聚集是表的一种存储方法, 他把单独组织的但逻辑上常在一起查询的基表在 物理上邻近存放,这样可减少数据的搜索时间,提高性能。聚集多个表可改进连接 性能,但会降低单表完全查找,降低插入语句和修改聚集码的 UPDATE 语句的性能。他同建索引一样,有时是以一些牺牲为代价的。同样要综合考虑。否则,事于愿违。

货票库基表是一个数据更新频繁的大表,为了适应表的变化,ORACLE 要花掉 一定时间对 CLUSTER 维护,况且又是多个基表的聚集,消耗的时间就更多,同时降低 了单表完全查询。所以本例最好只把最频繁连接的几个基表作 CLUSTER,基表数目 一定要尽可能的少,本例可删除后 7,8 个聚集关系。

有些开发者就以回避的态度,索性不建 CLUSTER,说:"不就搜索运行的时间慢 点吗!"。这种不求最佳的回避方法同样降低系统性能。一般情况,把经常做连接的 较稳定的基表建 CLUSTER,能极大提高查询速度,收到极佳效果。在沈东 MIS 系统开 发中,经过对建 CLUSTER 的合理性改造后,不但响应速度大大提高,并且系统的综合 性能有了很大改善。同时积累了宝贵的经验。

5. **查询应用程序的优化**

在沈东 MIS 系统试运行过程中, 出现过系统性能严重下降的问题,经过对典型 应用的分析,发现应用系统中频繁使用的 SQL 查询程序不够优化,存在大量的查询操作,且查询结果需动态刷新,而每个查询都需几十秒钟,甚至几分钟,操作十分频 繁。在查询期间,CPU 的 IDLE%=0,CPU 全部被占用,导致系统整体响应速度变慢。可见 ,优化查询应用程序,减轻 CPU 负载是一项刻不容缓的工作。下面列举几种典型的低 效查询例子,优化后减轻了 CPU 的占用时间,执行速度提高了,系统性

能大大改善。

((示例 C)):查询程序中,游标 CURSOR 嵌套使用,双重循环。

```
cursor c1 is select zd1 from tab1 .... (定义游标 C1)
cursor c2 is select zd2 from tab2 (定义游标 C2,
  where tj = c1.tj; (并以游标 C1 为条件)
loop fetch c1 ; (外层循环)
  open c2;
  ......
  loop fetch c2 ; (内层循环)
  select zd3 from tab3 ....
  .......
  endloop
  ......
  endloop;
```

此种查询方法,使用了嵌套 CURSOR(游标)和双重循环。表面看上去编程结 构清析,但 CPU 时间全部被占用。我们做如下改进。

```
create view st as select ... from tab1, tab2(把两个基表 tab1, tab2 共创一个视图 st)
  where tab1.cid = tab2.cid;
cursor c is select zd1, zd2 from st..... (对视图 st 定义游标 C)
loop fetch c;
  ......
end loop;
```

这样避免嵌套 CURSOR 和多重循环后,查询时间由优化前的 1 分多钟减为 6 秒 钟左右。可见在编写查询程序时,只注意表面编程结构的流畅还不够,还应注意分析哪种编程风格占用 CPU 时间最少,效果最佳。具体到本例,就是要避免游标 (CURSOR) 多级嵌套或和 CURSOR 嵌套效果类似的编程弊习。

((示例 D)):反复的全表查询。

```
cursor c1 is select distinct xw, ... from xxk; (游标指向不同箱位) loop fetch c1;
exit when c1%notfound; (下面是统计四种箱型在不同货位的个数)
select count(*) from xxk where xx = '05' and xw = c1.xw...(全表查询)
select count(*) from xxk where xx = '10' and xw = c1.xw...(全表查询)
select count(*) from xxk where xx = '20' and xw = c1.xw...(全表查询)
select count(*) from xxk where xx = '40' and xw = c1.xw...(全表查询)
  ......
endloop;
```

此例假如游标(CURSOR)循环 100 次,每循环 1 次就做 4 次全表查询,假如此 表有 500 条记录,4 种箱型就查询:500 乘 4 = 2000 次,循环结束就查询:2000 乘 100 = 200000 次。游标长时间的占用内存,CPU 也得不到缓解。我们做如下改进:

```
cursor c1 is
  select count(*) from xxk where xx = '05' and xw = c1.xw...
  union
  select count(*) from xxk where xx = '10' and xw = c1.xw...
  union
  select count(*) from xxk where xx = '20' and xw = c1.xw...
  union
  select count(*) from xxk where xx = '40' and xw = c1.xw...
  loop fetch c1;
  .......
endloop;
```

使用集合(UNION)定义游标后,查询时间由 1 分多钟下降为 5 秒。避免了 反复的全表查询,CPU 占用时间少了,极大提高了查询速度。

从以上两个优化查询示例不难看出,同样的程序,不同的编程风格 ,有时会带 来对系统性能有着重要影响的两种截然不同的结果。还有很多例子没有在此一一列举。总之,对那些执行时间较长的程序(>5 秒),可考虑是否有替代的实现方法,从 而减少 CPU 的使用效率,节省升级 CPU 所需的费用。这就要求我们在编程中勤思考, 善总结,力求完美的编程风格,使程序最优化,这样才能避免系统性能的不断下降。

参考文献

[1] ORACLE 数据库系统,清华大学出版社 1996

[2] ORACLE 数据库管理及应用开发,清华大学出版社 1996

[3] ORACLE7 技术手册,北方交通大学自动化所 科学出版社 1996

(来稿时间:1998 年 3 月)