了方便起见,建立了以下简单模型,和构造了部分测试数据:  
在某个业务受理子系统BSS中，  
--客户资料表  
create table customers  
(  
customer\_id number(8) not null, -- 客户标示  
city\_name varchar2(10) not null, -- 所在城市  
customer\_type char(2) not null, -- 客户类型  
  
...  
)  
create unique index PK\_customers on customers (customer\_id)  
由于某些原因，客户所在城市这个信息并不什么准确，但是在  
客户服务部的CRM子系统中，通过主动服务获取了部分客户20%的所在  
城市等准确信息，于是你将该部分信息提取至一张临时表中：  
create table tmp\_cust\_city  
(  
customer\_id number(8) not null,  
citye\_name varchar2(10) not null，  
customer\_type char(2) not null  
)

1) 最简单的形式  
--经确认customers表中所有customer\_id小于1000均为'北京'  
--1000以内的均是公司走向全国之前的本城市的老客户:)  
update customers  
set city\_name='北京'  
where customer\_id<1000

2) 两表(多表)关联update -- 仅在where字句中的连接  
--这次提取的数据都是VIP，且包括新增的,所以顺便更新客户类别  
update customers a -- 使用别名  
set customer\_type='01' --01 为vip，00为普通  
where exists (select 1   
from tmp\_cust\_city b  
where b.customer\_id=a.customer\_id  
)

3) 两表(多表)关联update -- 被修改值由另一个表运算而来  
update customers a -- 使用别名  
set city\_name=(select b.city\_name from tmp\_cust\_city b where b.customer\_id=a.customer\_id)  
where exists (select 1   
from tmp\_cust\_city b  
where b.customer\_id=a.customer\_id  
)  
-- update 超过2个值  
update customers a -- 使用别名  
set (city\_name,customer\_type)=(select b.city\_name,b.customer\_type   
from tmp\_cust\_city b   
where b.customer\_id=a.customer\_id)  
where exists (select 1   
from tmp\_cust\_city b  
where b.customer\_id=a.customer\_id  
)  
注意在这个语句中，  
=(select b.city\_name,b.customer\_type   
from tmp\_cust\_city b   
where b.customer\_id=a.customer\_id  
)  
与  
(select 1   
from tmp\_cust\_city b  
where b.customer\_id=a.customer\_id  
)  
是两个独立的子查询，查看执行计划可知，对b表/索引扫描了2篇；  
如果舍弃where条件，则默认对A表进行全表  
更新，但由于(select b.city\_name from tmp\_cust\_city b where  
where b.customer\_id=a.customer\_id)  
有可能不能提供"足够多"值，因为tmp\_cust\_city只是一部分客户的信息，  
所以报错（如果指定的列--city\_name可以为NULL则另当别论）：  
  
01407, 00000, "cannot update (%s) to NULL"  
// \*Cause:  
// \*Action:

一个替代的方法可以采用：  
update customers a -- 使用别名  
set city\_name=nvl((select b.city\_name from tmp\_cust\_city b   
where b.customer\_id=a.customer\_id),a.city\_name)  
或者  
set city\_name=nvl((select b.city\_name from tmp\_cust\_city b   
where b.customer\_id=a.customer\_id),'未知')  
-- 当然这不符合业务逻辑了

4) 上述3)在一些情况下，因为B表的纪录只有A表的20-30%的纪录数，  
考虑A表使用INDEX的情况，使用cursor也许会比关联update带来更好的性能：  
  
set serveroutput on

declare  
cursor city\_cur is  
select customer\_id,city\_name  
from tmp\_cust\_city  
order by customer\_id;  
begin  
for my\_cur in city\_cur loop

update customers  
set city\_name=my\_cur.city\_name  
where customer\_id=my\_cur.customer\_id;  
  
/\*\* 此处也可以单条/分批次提交，避免锁表情况 \*\*/  
-- if mod(city\_cur%rowcount,10000)=0 then  
-- dbms\_output.put\_line('----');  
-- commit;  
-- end if;  
end loop;  
end;

5) 关联update的一个特例以及性能再探讨  
在oracle的update语句语法中，除了可以update表之外，也可以是视图，所以有以下1个特例：  
update (select a.city\_name,b.city\_name as new\_name  
from customers a,  
tmp\_cust\_city b  
where b.customer\_id=a.customer\_id  
)  
set city\_name=new\_name  
这样能避免对B表或其索引的2次扫描，但前提是 A(customer\_id) b(customer\_id)必需是unique index  
或primary key。否则报错：  
  
01779, 00000, "cannot modify a column which maps to a non key-preserved table"  
// \*Cause: An attempt was made to insert or update columns of a join view which  
// map to a non-key-preserved table.  
// \*Action: Modify the underlying base tables directly.

6）oracle另一个常见错误  
回到3)情况,由于某些原因，tmp\_cust\_city customer\_id 不是唯一index/primary key  
update customers a -- 使用别名  
set city\_name=(select b.city\_name from tmp\_cust\_city b where b.customer\_id=a.customer\_id)  
where exists (select 1   
from tmp\_cust\_city b  
where b.customer\_id=a.customer\_id  
)  
当对于一个给定的a.customer\_id  
(select b.city\_name from tmp\_cust\_city b where b.customer\_id=a.customer\_id)  
返回多余1条的情况，则会报如下错误：  
  
01427, 00000, "single-row subquery returns more than one row"  
// \*Cause:  
// \*Action:

一个比较简单近似于不负责任的做法是  
update customers a -- 使用别名  
set city\_name=(select b.city\_name from tmp\_cust\_city b where b.customer\_id=a.customer\_id)

如何理解 01427 错误，在一个很复杂的多表连接update的语句，经常因考虑不周，出现这个错误，  
仍已上述例子来描述，一个比较简便的方法就是将A表代入 值表达式 中,使用group by 和  
having 字句查看重复的纪录  
(select b.customer\_id,b.city\_name,count(\*)  
from tmp\_cust\_city b,customers a   
where b.customer\_id=a.customer\_id  
group by b.customer\_id,b.city\_name  
having count(\*)>=2  
)

作者Blog：<http://blog.csdn.net/aceplus/>  
相关文章  
ORACLE UPDATE 语句语法与性能分析的一点看法   
移动的微软移动的Windows   
[收藏]数据库设计中的反规范技术探讨   
Oracle9i与SYBASE ASE12.5相比的几个不足   
如何在已装Windowns XP机器上再装Windows 2000   
  
对该文的评论   
aceplus ( 2005-07-04)   
/\*\*\*\* 改正1个错误 \*\*\*\*/

一个比较简单近似于不负责任的做法是  
update customers a -- 使用别名  
set city\_name=(select b.city\_name from tmp\_cust\_city b   
where b.customer\_id=  
a.customer\_id and rownum=1)

如何理解 01427 错误，在一个很复杂的多表连接update的语句，经常因考虑不周，出现这个错误，  
仍已上述例子来描述，一个比较简便的方法就是将A表代入 值表达式 中,使用group by 和  
having 字句查看重复的纪录  
(select b.customer\_id,b.city\_name,count(\*)  
from tmp\_cust\_city b,customers a   
where b.customer\_id=a.customer\_id  
group by b.customer\_id,b.city\_name  
having count(\*)>=2  
)  
  
aceplus ( 2005-07-04)   
/\*\*\*\* 改正1个错误 \*\*\*\*/

一个比较简单近似于不负责任的做法是  
update customers a -- 使用别名  
set city\_name=(select b.city\_name from tmp\_cust\_city b   
where b.customer\_id=  
a.customer\_id rownum=1)

如何理解 01427 错误，在一个很复杂的多表连接update的语句，经常因考虑不周，出现这个错误，  
仍已上述例子来描述，一个比较简便的方法就是将A表代入 值表达式 中,使用group by 和  
having 字句查看重复的纪录  
(select b.customer\_id,b.city\_name,count(\*)  
from tmp\_cust\_city b,customers a   
where b.customer\_id=a.customer\_id  
group by b.customer\_id,b.city\_name  
having count(\*)>=2  
)