ORACLE 分布式数据库系统位置透明性的实现机制

邱晓奕 万映辉 （空军电讯工程学院 710077）

摘要：本文介绍了 ORACLE 分布式数据库系统位置透明性实现的四种机制，详细分析了这四种机制的原理及应用方法，并通过举例对每种机制的使用进行了扼要介绍。

关键词：ORACLE 分布式数据库 位置透明性 表 快照 触发器 快照日志 视图 同义词

分布式数据库系统的一个重要特征是它的存在对用户是完全透明的，用户能够象与单一集中式系统进行通信那样来开发程序和操作数据库。ORACLE 7 分布式数据库系统就实现了这种透明性，其分布式结构对系统管理员及用户都是相对透明的，ORACLE 7 使用以下四种机制实现了位置的透明性：表的同步复制，表的异步复制，视图，同义词。本文将详细阐述这几种机制的原理、功能及实现方法。

一、表的同步复制

ORACLE 利用表的同步复制机制将各结点经常要访问的数据在各结点都存放一个副本，此副本的存放对用户是透明的。网上任一数据库的主要数据一旦变更立即引起所有副本的数据更改。ORACLE 7 中使用了触发器来实现表的同步复制。

用触发器实现表的同步复制时，要求表具有如下的特征：

1. 表的每个副本都是可查询可修改的，主表的修改立即引起所有副本的修改。

2. 通过提交具有触发 UPDATE 语句的用户事务来完成远程副本的修改。当远程通信网络发生故障时，则不论是本地的还是远程的副本都将取消修改事务，故障排除后才作同步修改，从而保证副本的一致性。

当使用触发器进行表的同步复制时，应注意：

(1)确定被复制的表有主码存在。

(2)要给每个复制的表建立一个标志字段以标识该表是否修改过，从而防止触发器的无尽连锁触发。

(3)要与被复制的数据库建立必要的数据联接。

(4)在每个复制表上建立复制触发器。

下面举例来说明如何建立复制表的触发器。假设要在两个城市 S、N 作某公司的供货情况表 SUPPLY 的同步复制，要求 N 城市表的改变触发 S 城市表的改变。则要求在 N 城市至少建立三个触发器：插入记录触发器、修改记录触发器和删除记录触发器。为叙述简单，假设 SUPPLY 表有四个字段：DEPTNUM、PRODUCTNAME、PRICE、FLAG。程序代码如下：

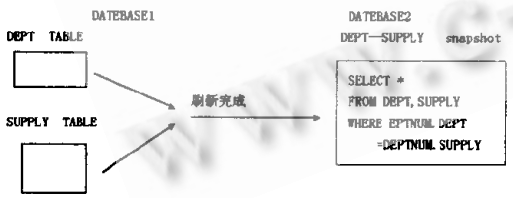
```
在城市 N：
/
    CREATE TRIGGER supply-insert / * 创建插入触发器 * /
  AFTER INSERT ON supply
    FOR EACH ROW
  WHEN (new. flag IS NULL) / * 标志位为 NULL，可将记录插入到 S 城市表中 * /
    BEGIN
      INSERT INTO fred. supply @S;
       VALUES (: new. deptnum, : new. product-name, :new. price, 'T');
  / * 插入一行远程记录，并设置标志位 * /
    END.
    /
  CREATE TRIGGER supply-update/ * 创建更改触发器 * /
    AFTER UPDATE ON supply
    FOR EACH ROW
  BEGIN
    IF NOT UPDATE('flag') THEN / * 标志为 NULL 则更改 S 城市的记录 * /
      UPDATE fred. supply @ S SET
        deptnum = :new. deptnum;
        deptname = :new. deptname;
        price = :new. price;
         flag = 'T';/ * 更改记录，并设置标志位 * /
```

```
    WHERE deptnum = :old.deptnum
    ENDIF;
    END.
/
    CREATE TRIGGER supply-delete / * * /
      AFTER DELETE ON supply
      FOR EACH ROW
    DECLARE
      mutating EXCETTION;
      PRAGMA exception-init (mutating, - 4091);
      BEGIN
        DELETE fred.supply @ S WHERE deptnum
= :old.deptnum;
        EXCETPION
        WHEN mutating THEN NULL;
    END.
```

二、表的异步复制(快照)

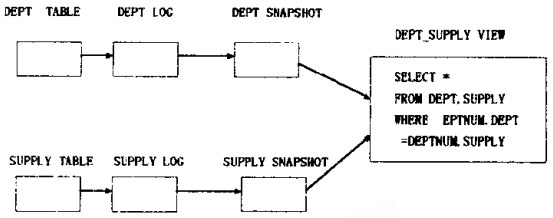
快照(SNAPSHOT)是只读的复制表,它可以是表的全部的副本,也可以是表的一个子集的副本。快照可以通过对一个或多个主表或视图的分布查询来定义。当主表被修改时快照并不修改,为保证快照与主表的一致性,可以根据需求定期自动或手动地刷新快照。由于快照的本质是只读的记录集,所以当表被频繁查询时才建立表的快照并分布于数据库的其他结点上,当用户从分布式系统的多结点上查询主表数据时通过快照会使查询速度大大提高。建立快照可以使用两种方法:如图示

方法 A:



方法 A,快照直接建立在两个表上,当两个表修改之后,重新运行 SQL 命令,即可刷新快照。但这是建立在两个表上的复杂快照,它的每次刷新都是完全刷新,所须的系统开销相对较大。

方法 B:



方法 B 中,先建立两个表的快照日志(SNAPSHOT LOG),快照日志中记录本表的刷新记录,相应的快照也只根据快照日志的内容作部分刷新,反映到快照视图上的总是刷新的记录。方法 B 建立的是简单快照,但建立快照日志要占用系统空间。对于一个表,只能建立一个快照日志,快照日志的建立用如下代码:

```
/
CREATE SNAPSHOT LOG ON SUPPLY / * 建立
SUPPLY 表的快照日志 * /
TABLESPACE user;
STORAGE ( INITIAL 10K NEXT 10K PCTIN-
CREASE 50);
```

ORACLE 系统中使用快照机制(异步表复制机制),可实现多副本与主表的一致性,同时也使主表的物理位置透明于用户。从而很好地实现了位置透明性问题。

三、视图、同义词

本地视图(LOCAL VIEW)在分布式数据库系统中可为本地及远程的表提供位置透明性。如本地表 DEPT,远程表 SUPPLY,为使着两张表的位置对用户透明,可在本地数据库中建立一个视图 SALE:

```
CREATE VIEW SALE AS
SELECT deptnum, productname, price, deptaddress
FROM COMPANY. DEPT, PS. SUPPLY @ N/ *
DEPT 表存放于 COMPANY 库中,SUPPLY 表存于 N 城
市的 PS 库中 * /
WHERE DEPT.deptnum = SUPPLY.deptnum;
```

当用户用语句:

```
SELECT * FROM SALE;
```

访问视图时,不知道也不必知道数据物理上存放于何处以及多表的检索是如何实现的,并且查询语句的写法也于直接从两张表中检索的语句写法有很大的不同。所以视图也很好地隐藏了数据的物理存放,实现了位置的透明性。

(下转 50 页)

(上接31页)

同义词无论是在分布式或非分布式环境中都是非常有用的,它可将原基本对象在分布式系统中的物理位置隐藏起来。若基本对象改名了或改变了结构,只要改变同义词的定义,而引用同义词的应用则不必进行任何修改。另外,使用同义词还可使分布式数据库系统中的用户的编程量减少。

以例示之,假定一分布式数据库系统中存在这样一个全局表名:

PS.SUPPLY @ N / * SUPPLY 表是存放于地点 N 中的 PS 库中 * /

若使用同义词定义:

CREATE PUBLIC SYNONYM SUPPLY FOR PS·SUPPLY @ N

则用户对 PS.SUPPLY @ N 的查询就可用同义词 SUPPLY 来替代,即使此表移到别的数据库中,比如移到 HR 库中,应用也不必改变,只需改变同义词定义:

CREATE PUBLIC SYNONYM SUPPLY FOR HR·SUPPLY @ N

这样,从用户的角度看,数据的物理位置是透明的。

(来稿时间:1998 年 1 月)