Oracle 高级复制冲突解决机制

——多主体复制冲突解决机制

本文作者: eygle (eygle.com@gmail.com)

摘要：本文主要探讨多主体复制环境中的冲突解决机制，对高级复制的配置，冲突的产生和主要解决方案进行阐述。

1.1 环境说明

本例测试环境如下，主要包含三个测试站点，分别为：

编号	名称	操作系统	数据库版本
A	CONNER.EYGLE.COM	Red Hat Enterprise Linux AS release 3	9.2.0.4
B	AUTHAA.EYGLE.COM	Red Hat Enterprise Linux AS release 3	9.2.0.4
C	AVATAR.EYGLE.COM	SunOS 5.8 Generic_108528-23	9.2.0.4

1.2 复制概述

复制是在组成分布式数据库系统的多个数据库之间拷贝和维护数据库对象的过程，在一个站点上应用的改变被捕获并按照同步或者异步的方式传播至远程站点。高级复制完全集成于 Oracle 服务器中，是 Oracle 数据库的内建组件及功能。

复制使用分布式数据库技术在多个站点之间共享数据，但是复制数据库和分布式数据库并不完全相同。对于分布式数据库，数据在多个位置可用，但是对于特定的数据表，只存在于某一个位置。而对于复制，同样的数据在多个站点上同时存在。

高级复制的类型

高级复制包含以下几种类型:
1.多主体复制
2.物化视图复制
3.多主体和物化视图混合配置

本文主要介绍多主体复制及其冲突解决机制。

多主体复制(Multimaster replication)也称为 peer-to-peer 或者 n-way 复制，使用多个站点管理复制组和复制

对象。多主体复制中的每个站点都是一个主体站点，每个站点都和其他站点通信。

在多主体复制环境中，应用程序可以在任何站点更新复制表。Oracle 数据库通过一些列内在机制，维护各站点之间的数据一致性及数据完整性。

1.3 高级复制的配置

下面简要说明高级复制的配置过程。

首先需要确认正确安装了高级复制的数据库选件:

```
SQL> select value from v$option where parameter='Advanced replication';


VALUE
----------------------------------------------------------------
TRUE
```

1.3.1 设置相应的初始化参数

设置 global_names = true

```
SQL> alter system set global_names = true;


System altered.
```

设置 job_queue_processes = 10

```
SQL> alter system set job_queue_processes = 10;


System altered.
```

修改 global_name，本例统一使用域名：EYGLE.COM

使用命令:

```
alter database rename global_name to CONNER.EYGLE.COM;
```

统一域名，

```
SQL> alter database rename global_name to CONNER.EYGLE.COM;
```

```
Database altered.

SQL> select * from global_name;

GLOBAL_NAME
------------------------------------------------------------
CONNER.EYGLE.COM
```

逐一修改后三个站点分别为:
AUTHAA.EYGLE.COM
CONNER.EYGLE.COM
AVATAR.EYGLE.COM

1.3.2 创建用于存储元数据的表空间

也可以使用 user 表空间，或者创建独立的表空间，本例使用 USERS 表空间

```
create tablespace users datafile '/data1/oradata/user01.dbf'
size 200M
extent management local uniform size 128k
segment space management auto;
```

1.3.3 创建复制管理员用户

这一步骤需要在三个站点上分别执行

```
SQL> create user repadmin identified by repadmin default tablespace users;

User created.
```

1.3.4 给复制管理员授权

a.执行 GRANT_ADMIN_ANY_SCHEMA 过程授予 repadmin 以复制管理员的权限

```
SQL> execute dbms_repcat_admin.grant_admin_any_schema(username => '"REPADMIN"');

PL/SQL procedure successfully completed.
```

b.如果你想让 repadmin 能够为复制表创建物化试图日志，你可以授予 COMMENT ANY TABLE 和 LOCK ANY TABLE 的权限。

```
SQL> grant comment any table to repadmin;

Grant succeeded.

SQL> grant lock any table to repadmin;

Grant succeeded.
```

c.如果希望 repadmin 可以使用复制管理工具连接，那么需要授予 SELECT ANY DICTIONARY 的权限:

```
SQL> grant select any dictionary to repadmin;

Grant succeeded.
```

1.3.5 注册传播者(propagator)

传播者(propagator)负责把延迟事务传播到其他站点。

```
SQL> exec DBMS_DEFER_SYS.REGISTER_PROPAGATOR (username => 'repadmin');

PL/SQL procedure successfully completed.

SQL>
```

1.3.6 注册接收者(receiver)

接收者负责接收其他站点传播者发送来的延迟事务.

```
SQL> BEGIN
  2  DBMS_REPCAT_ADMIN.REGISTER_USER_REPGROUP (
  3  username => 'repadmin',privilege_type => 'receiver',list_of_gnames => NULL);
  4  END;
  5  /

PL/SQL procedure successfully completed.

SQL>
```

1.3.7 在主体站点定时清除

为了控制延迟事务队列的大小，我们需要定期清除成功完成的延迟事务。SCHEDULE_PURGE 过程自动完成清除过程。

我们需要以复制管理员身份运行这个过程.

```
CONNECT repadmin/repadmin
BEGIN
DBMS_DEFER_SYS.SCHEDULE_PURGE (
next_date => SYSDATE,
interval => 'SYSDATE + 1/24',
delay_seconds => 0);
END;
/

SQL> CONNECT repadmin/repadmin
Connected.
```

```
SQL> BEGIN
  2 DBMS_DEFER_SYS.SCHEDULE_PURGE (
  3 next_date => SYSDATE,
  4 interval => 'SYSDATE + 1/24',
  5 delay_seconds => 0);
  6 END;
  7 /

PL/SQL procedure successfully completed.

SQL>
```

1.3.8 创建 db link

使用如下脚本:

```
CONNECT SYSTEM/MMANAGER@CONNER
CREATE PUBLIC DATABASE LINK AUTHAA USING 'AUTHAA';
CREATE PUBLIC DATABASE LINK AVATAR USING 'AVATAR';
CONNECT repadmin/repadmin@CONNER
CREATE DATABASE LINK AUTHAA CONNECT TO repadmin IDENTIFIED BY repadmin;
CREATE DATABASE LINK AVATAR CONNECT TO repadmin IDENTIFIED BY repadmin;

CONNECT SYSTEM/MMANAGER@AUTHAA
CREATE PUBLIC DATABASE LINK CONNER USING 'CONNER';
CREATE PUBLIC DATABASE LINK AVATAR USING 'AVATAR';
CONNECT repadmin/repadmin@AUTHAA
CREATE DATABASE LINK CONNER CONNECT TO repadmin IDENTIFIED BY repadmin;
CREATE DATABASE LINK AVATAR CONNECT TO repadmin IDENTIFIED BY repadmin;

CONNECT SYSTEM/MMANAGER@AVATAR
```

```
CREATE PUBLIC DATABASE LINK CONNER USING 'CONNER';
CREATE PUBLIC DATABASE LINK AUTHAA USING 'AUTHAA';
CONNECT repadmin/repadmin@AVATAR
CREATE DATABASE LINK CONNER CONNECT TO repadmin IDENTIFIED BY repadmin;
CREATE DATABASE LINK AUTHAA CONNECT TO repadmin IDENTIFIED BY repadmin;
```

以下是其中一个平台上的执行过程:

```
[oracle@jumper admin]$ sqlplus /nolog

SQL*Plus: Release 9.2.0.4.0 - Production on Wed Jun 29 09:58:15 2005

Copyright (c) 1982, 2002, Oracle Corporation.  All rights reserved.

SQL> CONNECT SYSTEM/MMANAGER@CONNER
Connected.
SQL> CREATE PUBLIC DATABASE LINK AUTHAA USING 'AUTHAA';

Database link created.

SQL> CREATE PUBLIC DATABASE LINK AVATAR USING 'AVATAR';

Database link created.

SQL> CONNECT repadmin/repadmin@CONNER
Connected.
SQL> CREATE DATABASE LINK AUTHAA CONNECT TO repadmin IDENTIFIED BY repadmin;

Database link created.

SQL> CREATE DATABASE LINK AVATAR CONNECT TO repadmin IDENTIFIED BY repadmin;

Database link created.
```

```
SQL> select * from dual@avatar;

D
-
X

SQL> select * from dual@authaa;

D
-
X
```

1.3.9 创建定时任务 PUSH 延迟事务:

```
CONNECT repadmin/repadmin@conner

BEGIN
DBMS_DEFER_SYS.SCHEDULE_PUSH (
destination => 'authaa',
interval => 'SYSDATE + (1/144)',
next_date => SYSDATE,
parallelism => 1,
execution_seconds => 1500,
delay_seconds => 1200);
END;
/
BEGIN
DBMS_DEFER_SYS.SCHEDULE_PUSH (
destination => 'avatar',
interval => 'SYSDATE + (1/144)',
```

```
next_date => SYSDATE,
parallelism => 1,
execution_seconds => 1500,
delay_seconds => 1200);
END;
/
CONNECT repadmin/repadmin@authaa
BEGIN
DBMS_DEFER_SYS.SCHEDULE_PUSH (
destination => 'conner',
interval => 'SYSDATE + (1/144)',
next_date => SYSDATE,
parallelism => 1,
execution_seconds => 1500,
delay_seconds => 1200);
END;
/
BEGIN
DBMS_DEFER_SYS.SCHEDULE_PUSH (
destination => 'avatar',
interval => 'SYSDATE + (1/144)',
next_date => SYSDATE,
parallelism => 1,
execution_seconds => 1500,
delay_seconds => 1200);
END;
/
CONNECT repadmin/repadmin@avatar
BEGIN
DBMS_DEFER_SYS.SCHEDULE_PUSH (
destination => 'conner',
interval => 'SYSDATE + (1/144)',
next_date => SYSDATE,
```

```
parallelism => 1,
execution_seconds => 1500,
delay_seconds => 1200);
END;
/

BEGIN
DBMS_DEFER_SYS.SCHEDULE_PUSH (
destination => 'authaa',
interval => 'SYSDATE + (1/144)',
next_date => SYSDATE,
parallelism => 1,
execution_seconds => 1500,
delay_seconds => 1200);
END;
/
```

1.3.10复制对象

以 Scott 用户为示例对象

```
SQL> @?/rdbms/admin/utlsampl
Disconnected from Oracle9i Enterprise Edition Release 9.2.0.4.0 - Production
With the Partitioning option
JServer Release 9.2.0.4.0 - Production
[oracle@jumper admin]$ sqlplus scott/tiger

SQL*Plus: Release 9.2.0.4.0 - Production on Wed Jun 29 10:27:32 2005

Copyright (c) 1982, 2002, Oracle Corporation.  All rights reserved.
```

```
Connected to:
Oracle9i Enterprise Edition Release 9.2.0.4.0 - Production
With the Partitioning option
JServer Release 9.2.0.4.0 - Production

SQL> select * from tab;

TNAME                          TABTYPE  CLUSTERID
------------------------------ ------- ----------
BONUS                          TABLE
DEPT                           TABLE
EMP                            TABLE
SALGRADE                        TABLE



SQL> select constraint_name,constraint_type,table_name from USER_CONSTRAINTS;

CONSTRAINT_NAME                C TABLE_NAME
------------------------------ - ------------------------------
PK_DEPT                        P DEPT
PK_EMP                         P EMP
FK_DEPTNO                       R EMP
```

对于外键约束，需要单独复制

1.3.11创建复制组

```
CONNECT repadmin/repadmin@conner
BEGIN
DBMS_REPCAT.CREATE_MASTER_REPGROUP (
gname => 'rep_sct');
END;
```

```
/


SQL> CONNECT repadmin/repadmin@conner
Connected.
SQL> BEGIN
  2 DBMS_REPCAT.CREATE_MASTER_REPGROUP (
  3 gname => 'rep_sct');
  4 END;
  5 /

PL/SQL procedure successfully completed.

SQL>
```

如果遇到问题可以使用如下命令删除复制组:

```
exec dbms_repcat.drop_master_repgroup(gname=>'rep_sct',all_sites => true);


exec dbms_repcat.drop_master_repgroup(gname=>'rep_sct',all_sites => false);
```

1.3.12增加主体对象

```
BEGIN
DBMS_REPCAT.CREATE_MASTER_REPOBJECT (
gname => 'rep_sct',
type => 'TABLE',
oname => 'DEPT',
sname => 'scott',
use_existing_object => TRUE,
copy_rows => FALSE);
END;
/
```

```
BEGIN
DBMS_REPCAT.CREATE_MASTER_REPOBJECT (
gname => 'rep_sct',
type => 'TABLE',
oname => 'EMP',
sname => 'scott',
use_existing_object => TRUE,
copy_rows => FALSE);
END;
/
```

删除复制对象使用如下命令:

```
exec DBMS_REPCAT.DROP_MASTER_REPOBJECT(sname => 'scott',oname => 'dept',type => 'table');
exec DBMS_REPCAT.DROP_MASTER_REPOBJECT(sname => 'scott',oname => 'emp',type => 'table');
```

1.3.13增加主体站点

```
BEGIN
DBMS_REPCAT.ADD_MASTER_DATABASE (
gname => 'rep_sct',
master => 'authaa',
use_existing_objects => TRUE,
copy_rows => FALSE,
propagation_mode => 'ASYNCHRONOUS');
END;
/

BEGIN
DBMS_REPCAT.ADD_MASTER_DATABASE (
gname => 'rep_sct',
master => 'avatar',
```

```
use_existing_objects => TRUE,
copy_rows => FALSE,
propagation_mode => 'ASYNCHRONOUS');
END;
/
```

如果出现问题，可以使用如下命令删除主体站点:

```
exec dbms_repcat.remove_master_databases(gname => 'rep_sct',master_list => 'AUTHAA.EYGLE.COM');
exec dbms_repcat.remove_master_databases(gname => 'rep_sct',master_list => 'AVATAR.EYGLE.COM');
```

1.3.14产生复制支持

对复制对象产生复制支持，这一步骤实际上是生成内部的 PACKAGE，以捕获对象变化，从而生成可传播的任务。

```
BEGIN
DBMS_REPCAT.GENERATE_REPLICATION_SUPPORT (
sname => 'scott',
oname => 'DEPT',
type => 'TABLE',
min_communication => TRUE);
END;
/
BEGIN
DBMS_REPCAT.GENERATE_REPLICATION_SUPPORT (
sname => 'scott',
oname => 'EMP',
type => 'TABLE',
min_communication => TRUE);
END;
/
```

我们来查询一下产生复制支持以后，数据库自动生成了哪些内部对象：

```
SQL> select sname,oname,type,GENERATION_STATUS,OBJECT_COMMENT from dba_repobject;

SNAME ONAME     TYPE           GENERATION_STATUS OBJECT_COMMENT
----- ---------- --------------- ----------------- ------------------------------
SCOTT DEPT      TABLE          GENERATED
SCOTT DEPT$RP    PACKAGE                         SYSTEM-GENERATED: REPLICATION
SCOTT DEPT$RP    PACKAGE BODY                     SYSTEM-GENERATED: REPLICATION
SCOTT EMP       TABLE          GENERATED
SCOTT EMP$RP     PACKAGE                         SYSTEM-GENERATED: REPLICATION
SCOTT EMP$RP     PACKAGE BODY                     SYSTEM-GENERATED: REPLICATION

6 rows selected
```

我们抽取一个 Package（DEPT$RP）看一下具体内容：

```
create or replace package scott."DEPT$RP" as
  column_changed$$ RAW(1000);
  procedure rep_delete(
    "DEPTNO1_o" IN NUMBER,
    "DNAME2_o" IN VARCHAR2,
    "LOC3_o" IN VARCHAR2,
    site_name IN VARCHAR2,
    propagation_flag IN CHAR);
  procedure rep_delete(
    column_changed$ IN RAW,
    "DEPTNO1_o" IN NUMBER,
    "DNAME2_o" IN VARCHAR2,
    "LOC3_o" IN VARCHAR2,
    site_name IN VARCHAR2,
    propagation_flag IN CHAR);
  procedure rep_insert(
    "DEPTNO1_n" IN NUMBER,
```

```
   "DNAME2_n" IN VARCHAR2,
   "LOC3_n" IN VARCHAR2,
   site_name IN VARCHAR2,
   propagation_flag IN CHAR);
  procedure rep_update(
   "DEPTNO1_o" IN NUMBER,
   "DEPTNO1_n" IN NUMBER,
   "DNAME2_o" IN VARCHAR2,
   "DNAME2_n" IN VARCHAR2,
   "LOC3_o" IN VARCHAR2,
   "LOC3_n" IN VARCHAR2,
   site_name IN VARCHAR2,
   propagation_flag IN CHAR);
  procedure rep_update(
   column_changed$ IN RAW,
   "DEPTNO1_o" IN NUMBER,
   "DEPTNO1_n" IN NUMBER,
   "DNAME2_o" IN VARCHAR2,
   "DNAME2_n" IN VARCHAR2,
   "LOC3_o" IN VARCHAR2,
   "LOC3_n" IN VARCHAR2,
   site_name IN VARCHAR2,
   propagation_flag IN CHAR);
end "DEPT$RP";
```

我们看到 Oracle 通过系统 Package，跟踪 DELETE、UPDATE、INSERT 等操作。

我们再看一下 Package Body 的内容：

```

create or replace package body scott."DEPT$RP" as
  procedure rep_delete(
   "DEPTNO1_o" IN NUMBER,
   "DNAME2_o" IN VARCHAR2,
   "LOC3_o" IN VARCHAR2,
   site_name IN VARCHAR2,
```

```
  propagation_flag IN CHAR) is
begin
  rep_delete(
    NULL,
    "DEPTNO1_o",
    "DNAME2_o",
    "LOC3_o",
    site_name,
    propagation_flag);
end rep_delete;
procedure rep_delete(
  column_changed$ IN RAW,
  "DEPTNO1_o" IN NUMBER,
  "DNAME2_o" IN VARCHAR2,
  "LOC3_o" IN VARCHAR2,
  site_name IN VARCHAR2,
  propagation_flag IN CHAR) is
begin
  DBMS_REPCAT_INTERNAL_PACKAGE.CALL(
    'SCOTT','DEPT','REP_DELETE',6);
  DBMS_REPCAT_INTERNAL_PACKAGE.RAW_ARG(column_changed$);
  DBMS_REPCAT_INTERNAL_PACKAGE.NUMBER_ARG("DEPTNO1_o");
  DBMS_REPCAT_INTERNAL_PACKAGE.VARCHAR2_ARG("DNAME2_o");
  DBMS_REPCAT_INTERNAL_PACKAGE.VARCHAR2_ARG("LOC3_o");
  DBMS_REPCAT_INTERNAL_PACKAGE.VARCHAR2_ARG(site_name);
  DBMS_REPCAT_INTERNAL_PACKAGE.CHAR_ARG(propagation_flag);
end rep_delete;
procedure rep_insert(
  "DEPTNO1_n" IN NUMBER,
  "DNAME2_n" IN VARCHAR2,
  "LOC3_n" IN VARCHAR2,
  site_name IN VARCHAR2,
  propagation_flag IN CHAR) is
```

```
begin
  DBMS_REPCAT_INTERNAL_PACKAGE.CALL(
    'SCOTT','DEPT','REP_INSERT',5);
  DBMS_REPCAT_INTERNAL_PACKAGE.NUMBER_ARG("DEPTNO1_n");
  DBMS_REPCAT_INTERNAL_PACKAGE.VARCHAR2_ARG("DNAME2_n");
  DBMS_REPCAT_INTERNAL_PACKAGE.VARCHAR2_ARG("LOC3_n");
  DBMS_REPCAT_INTERNAL_PACKAGE.VARCHAR2_ARG(site_name);
  DBMS_REPCAT_INTERNAL_PACKAGE.CHAR_ARG(propagation_flag);
end rep_insert;
procedure rep_update(
  "DEPTNO1_o" IN NUMBER,
  "DEPTNO1_n" IN NUMBER,
  "DNAME2_o" IN VARCHAR2,
  "DNAME2_n" IN VARCHAR2,
  "LOC3_o" IN VARCHAR2,
  "LOC3_n" IN VARCHAR2,
  site_name IN VARCHAR2,
  propagation_flag IN CHAR) is
begin
  rep_update(
    NULL,
    "DEPTNO1_o",
    "DEPTNO1_n",
    "DNAME2_o",
    "DNAME2_n",
    "LOC3_o",
    "LOC3_n",
    site_name,
    propagation_flag);
end rep_update;
procedure rep_update(
  column_changed$ IN RAW,
  "DEPTNO1_o" IN NUMBER,
```

```
   "DEPTNO1_n" IN NUMBER,
   "DNAME2_o" IN VARCHAR2,
   "DNAME2_n" IN VARCHAR2,
   "LOC3_o" IN VARCHAR2,
   "LOC3_n" IN VARCHAR2,
   site_name IN VARCHAR2,
   propagation_flag IN CHAR) is
 begin
   DBMS_REPCAT_INTERNAL_PACKAGE.CALL(
     'SCOTT','DEPT','REP_UPDATE',9);
   DBMS_REPCAT_INTERNAL_PACKAGE.RAW_ARG(column_changed$);
   DBMS_REPCAT_INTERNAL_PACKAGE.NUMBER_ARG("DEPTNO1_o");
   DBMS_REPCAT_INTERNAL_PACKAGE.NUMBER_ARG("DEPTNO1_n");
   DBMS_REPCAT_INTERNAL_PACKAGE.VARCHAR2_ARG("DNAME2_o");
   DBMS_REPCAT_INTERNAL_PACKAGE.VARCHAR2_ARG("DNAME2_n");
   DBMS_REPCAT_INTERNAL_PACKAGE.VARCHAR2_ARG("LOC3_o");
   DBMS_REPCAT_INTERNAL_PACKAGE.VARCHAR2_ARG("LOC3_n");
   DBMS_REPCAT_INTERNAL_PACKAGE.VARCHAR2_ARG(site_name);
   DBMS_REPCAT_INTERNAL_PACKAGE.CHAR_ARG(propagation_flag);
 end rep_update;
end "DEPT$RP";
```

1.3.15状态检查

完成以上步骤，可以在主体站点分别检查复制对象及复制支持是否正确被创建.

```
SQL> select * from global_name;

GLOBAL_NAME
--------------------------------------------------------------------------------
AUTHAA.EYGLE.COM

SQL> select sname,oname,INTERNAL_PACKAGE_EXISTS,gname from dba_repobject where gname='REP_SCT'
```

```
  2  /

SNAME                          ONAME          I GNAME
------------------------------ ---------------- - ------------------------------
SCOTT                          DEPT           Y REP_SCT
SCOTT                          DEPT$RP          REP_SCT
SCOTT                          DEPT$RP          REP_SCT
SCOTT                          EMP            Y REP_SCT
SCOTT                          EMP$RP           REP_SCT
SCOTT                          EMP$RP           REP_SCT

6 rows selected.


SQL> select * from global_name;

GLOBAL_NAME
--------------------------------------------------------------------------------
AVATAR.EYGLE.COM

SQL> select sname,oname,INTERNAL_PACKAGE_EXISTS,gname from dba_repobject where gname='REP_SCT';

SNAME                          ONAME                          I GNAME
------------------------------ ------------------------------ - ------------------------------
SCOTT                          DEPT                         Y REP_SCT
SCOTT                          DEPT$RP                        REP_SCT
SCOTT                          DEPT$RP                        REP_SCT
SCOTT                          EMP                          Y REP_SCT
SCOTT                          EMP$RP                         REP_SCT
SCOTT                          EMP$RP                         REP_SCT

6 rows selected.
```

```
SQL> select * from global_name;

GLOBAL_NAME
--------------------------------------------------------------------------------
CONNER.EYGLE.COM

SQL> select sname,oname,INTERNAL_PACKAGE_EXISTS,gname from dba_repobject where gname='REP_SCT';

SNAME                          ONAME                          I GNAME
------------------------------ ------------------------------ - --------------------------
SCOTT                          DEPT                           Y REP_SCT
SCOTT                          DEPT$RP                          REP_SCT
SCOTT                          DEPT$RP                          REP_SCT
SCOTT                          EMP                            Y REP_SCT
SCOTT                          EMP$RP                           REP_SCT
SCOTT                          EMP$RP                           REP_SCT

6 rows selected.
```

1.3.16启动复制

```

BEGIN
DBMS_REPCAT.RESUME_MASTER_ACTIVITY (
gname => 'rep_sct');
END;
/

SQL> BEGIN
  2  DBMS_REPCAT.RESUME_MASTER_ACTIVITY (
```

```
3  gname => 'rep_sct');
4  END;
5  /
```

1.4 冲突解决

1.4.1 冲突概述

尽管你可能已经精心设计了你的应用以避免多主体站点间的冲突,但是你几乎不可能排除冲突的可能.高级复制技术的一个重要方面就是如何确保所有参与高级复制的站点之间数据一致。

当数据冲突出现时,你需要一个机制确保数据按照你的业务规则趋同。

高级复制允许我们根据业务逻辑定义一个冲突解决系统.Oracle 内建一些冲突解决方案用以解决冲突,如果这些冲突不足以应对你的特殊情况，那么你可以选择自定义冲突解决方案应对特殊情况。

在对复制表实施冲突解决方案之前，你应该分析数据以确定最可能产生冲突的情形。例如,静态数据例如员工编号会很少改变，但是一个员工的顾客分配可能经常改变，并可能由此产生冲突。

当你确定了冲突可能发生的情况以后，你需要决定如何解决冲突。是采用最后变更优先，还是站点优先?这是你需要考虑的问题。

1.4.2 复制冲突的类型

在复制环境中可能遇到以下几种类型的冲突:
1.更新冲突
更新冲突通常在多个站点同时或接近同时更新同一条记录时发生。
2.唯一性冲突
指复制的记录违反完整性约束，例如主键或者唯一键约束。
3.删除冲突
指不同站点同时或接近同时，一个站点删除记录，另外站点删除或者更新记录，那么随后的复制会因记录不存在无法删除或无法更新而产生冲突。

1.4.3 冲突的出现

我们看一下一个更新冲突的例子:

1.首先我们在三个站点同时(近似同时)更新同一记录

a.对于 conner 站点的操作

```
[oracle@jumper oracle]$ sqlplus scott/tiger@conner

SQL*Plus: Release 9.2.0.4.0 - Production on Fri Jul 1 10:36:55 2005

Copyright (c) 1982, 2002, Oracle Corporation.  All rights reserved.



Connected to:
Oracle9i Enterprise Edition Release 9.2.0.4.0 - Production
With the Partitioning option
JServer Release 9.2.0.4.0 - Production

SQL> select * from dept;

    DEPTNO DNAME         LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON

SQL> update dept set loc='BEIJING' where loc='BOSTON';

1 row updated.

SQL> commit;
```

```
Commit complete.

SQL> select * from dept;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BEIJING
```

b.对于 Authaa 站点的操作

```
[oracle@www167 oracle]$ sqlplus scott/tiger@authaa

SQL*Plus: Release 9.2.0.4.0 - Production on Fri Jul 1 10:33:10 2005

Copyright (c) 1982, 2002, Oracle Corporation.  All rights reserved.


Connected to:
Oracle9i Enterprise Edition Release 9.2.0.4.0 - Production
With the Partitioning option
JServer Release 9.2.0.4.0 - Production

SQL> update dept set loc='SHANGHAI' where loc='BOSTON';

1 row updated.

SQL> commit;

Commit complete.

SQL> select * from dept;
```

```
   DEPTNO DNAME         LOC
---------- -------------- -------------
       10 ACCOUNTING     NEW YORK
       20 RESEARCH       DALLAS
       30 SALES          CHICAGO
       40 OPERATIONS     SHANGHAI
```

3.对于 Avatar 站点的操作

```
$ sqlplus scott/tiger@avatar

SQL*Plus: Release 9.2.0.4.0 - Production on Fri Jul 1 10:37:23 2005

Copyright (c) 1982, 2002, Oracle Corporation.  All rights reserved.



Connected to:
Oracle9i Enterprise Edition Release 9.2.0.4.0 - 64bit Production
With the Partitioning, OLAP and Oracle Data Mining options
JServer Release 9.2.0.4.0 - Production

SQL> select * from dept;

   DEPTNO DNAME         LOC
---------- -------------- -------------
       10 ACCOUNTING     NEW YORK
       20 RESEARCH       DALLAS
       30 SALES          CHICAGO
       40 OPERATIONS     BOSTON

SQL> update dept set loc='GUANGZHOU' where loc='BOSTON';

1 row updated.
```

```
SQL> commit;

Commit complete.

SQL> select * from dept;

    DEPTNO DNAME         LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     GUANGZHOU
```

那么由于延迟事务，各站点的更新将会无法传播的任何其他站点，更新冲突出现。

2.检查错误

我们来看一下错误。

在 Conner 站点:

```
SQL> select * from deferror;

DEFERRED_TRAN_ID       ORIGIN_TRAN_DB          ORIGIN_TRAN_ID             CALLNO DESTINATION
START_TIME  ERROR_NUMBER ERROR_MSG                  RECEIVER
----------------------     ------------------------     ----------------------     ----------
-----------------------     -----------     ------------     ------------------------------
------------------------------
5.43.13195           AVATAR.EYGLE.COM 5.24.296499                0 CONNER.EYGLE.COM 2005-7-1
10       1403 ORA-01403: no data found      REPADMIN
```

在 Authaa 站点

```
SQL> select * from deferror;

DEFERRED_TRAN_ID       ORIGIN_TRAN_DB          ORIGIN_TRAN_ID             CALLNO DESTINATION
```

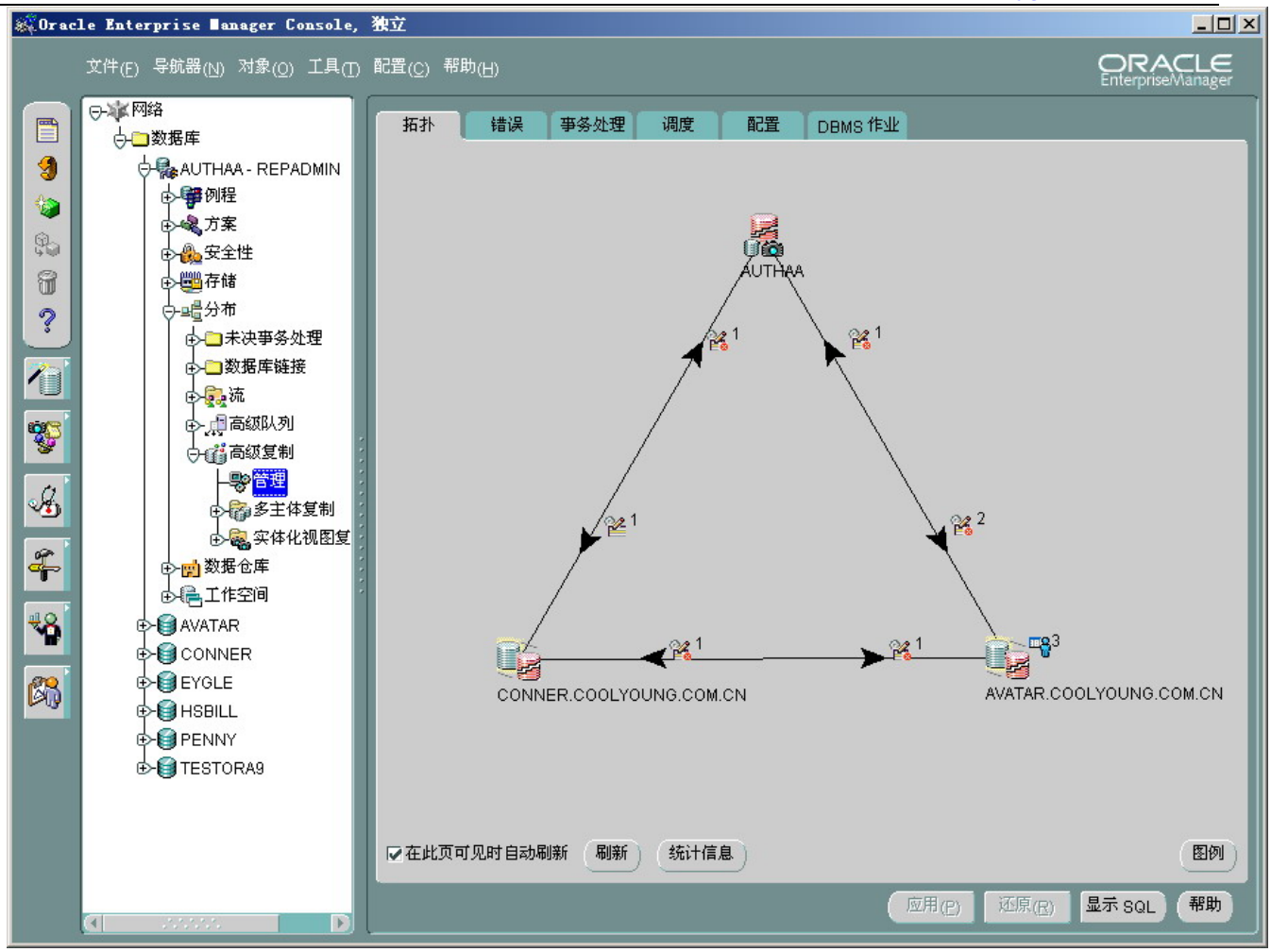
```
START_TIME  ERROR_NUMBER ERROR_MSG                    RECEIVER
----------------------     ------------------------     ----------------------     ----------
------------------------     -----------     ------------     ------------------------------
------------------------------
4.8.19521            CONNER.EYGLE.COM 3.8.14818                    0 AUTHAA.EYGLE.COM 2005-7-1
10        1403 ORA-01403: no data found      REPADMIN
5.44.19559           AVATAR.EYGLE.COM 5.24.296499                  0 AUTHAA.EYGLE.COM 2005-7-1
10        1403 ORA-01403: no data found      REPADMIN
```

在 Avatar 站点:

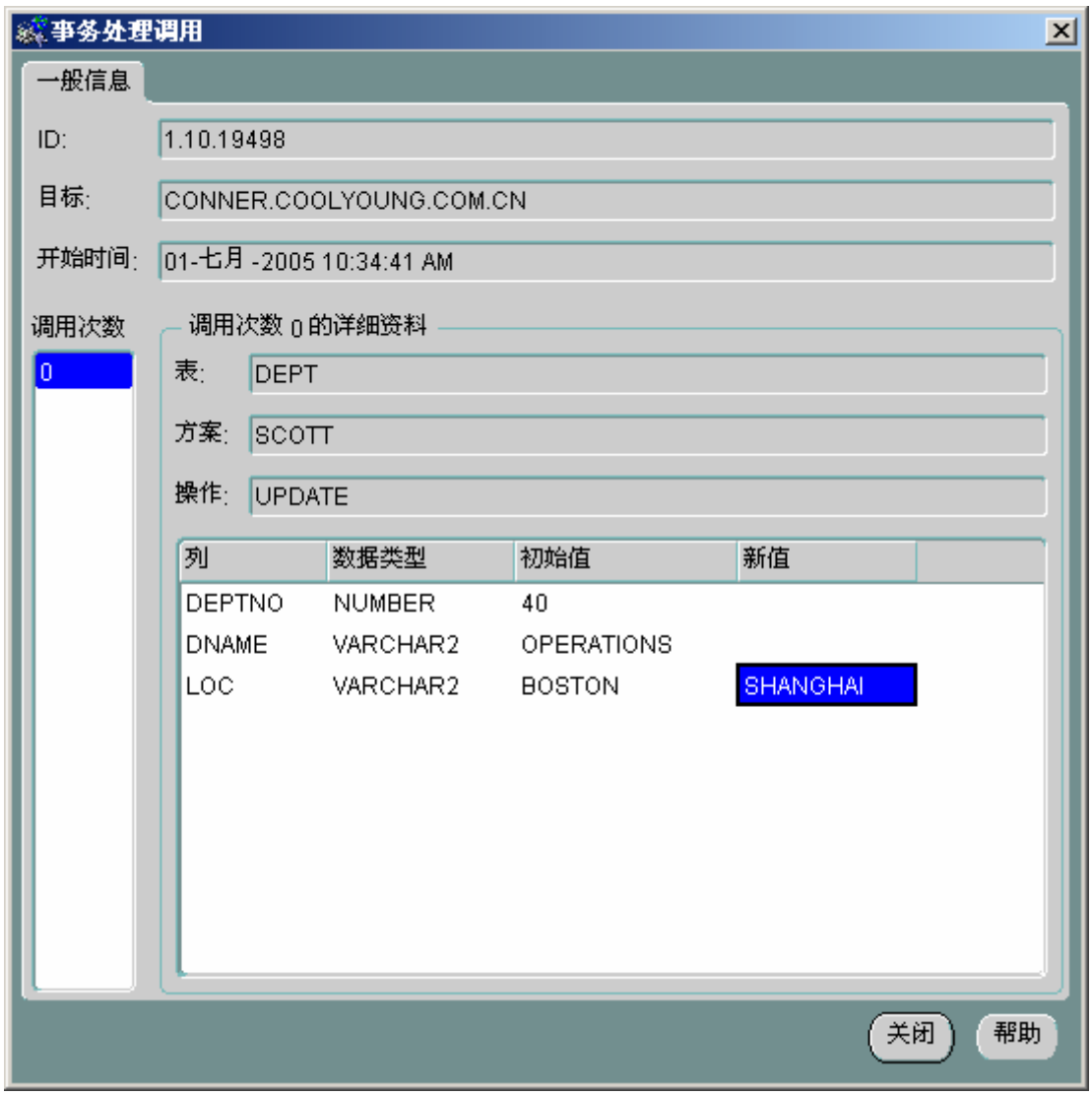
```
SQL> select * from deferror;

DEFERRED_TRAN_ID       ORIGIN_TRAN_DB          ORIGIN_TRAN_ID             CALLNO DESTINATION
START_TIME  ERROR_NUMBER ERROR_MSG                    RECEIVER
----------------------     ------------------------     ----------------------     ----------
------------------------     -----------     ------------     ------------------------------
------------------------------
4.19.294183          AUTHAA.EYGLE.COM 1.10.19498                   0 AVATAR.EYGLE.COM 2005-7-1
10        1403 ORA-01403: no data found      REPADMIN
6.11.291540          AUTHAA.EYGLE.COM 1.10.19498                   0 AVATAR.EYGLE.COM 2005-7-1
10         100 ORA-01403: no data found      REPADMIN
10.44.294783         CONNER.EYGLE.COM 3.8.14818                    0 AVATAR.EYGLE.COM 2005-7-1
10        1403 ORA-01403: no data found      REPADMIN
```

我们注意到出现错误 ORA-01403: no data found，更新冲突导致的问题已经出现。

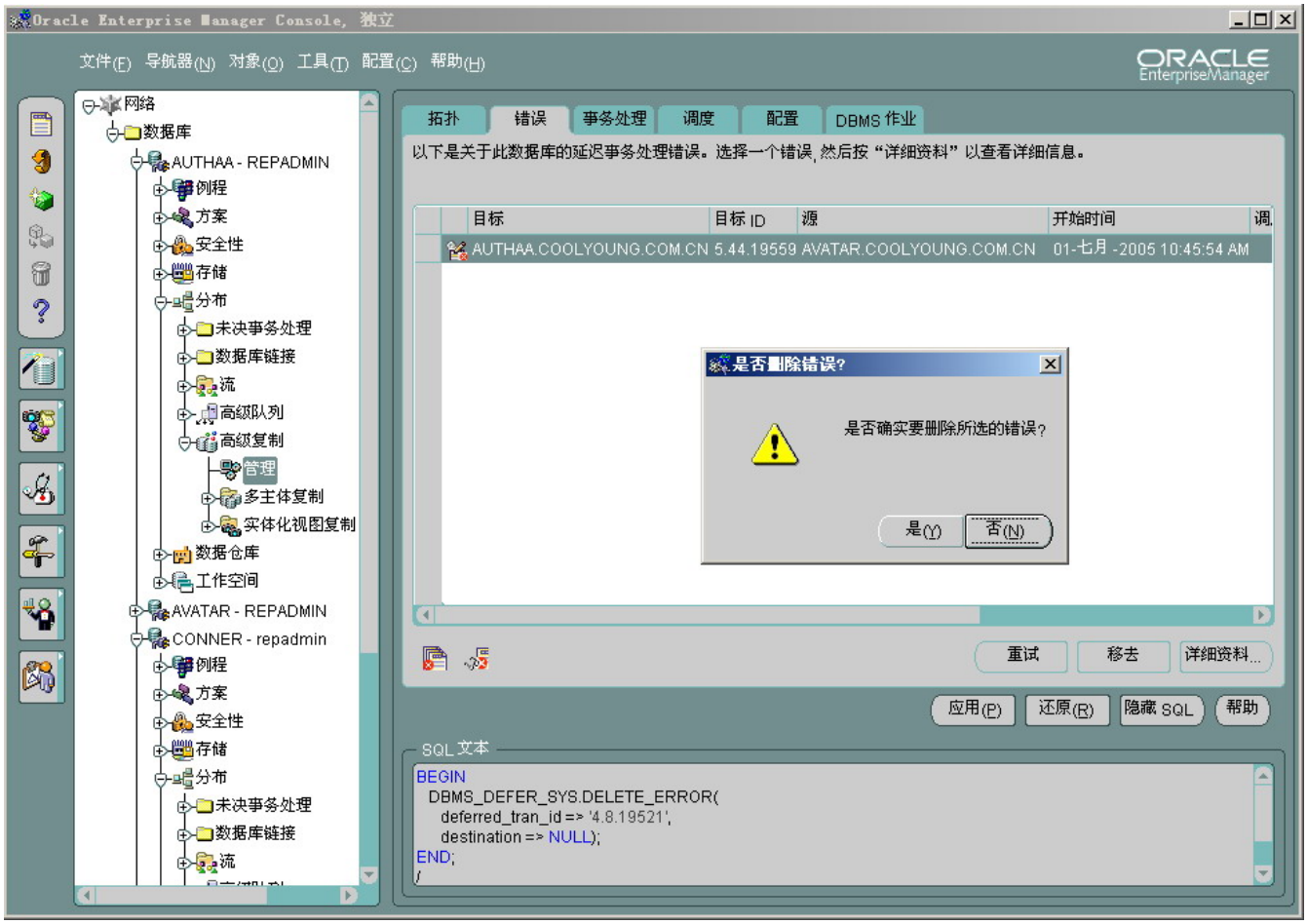
使用 OEM 工具我们可以很容易的看到各站点之间的故障情况。



我们也可以通过图形界面查看失败任务的具体信息：



我们可以在图形界面移除这些错误信息:



也可以通过手工方式来清除（这是通常建议采取的方式）：

在 Authaa 节点：

```
BEGIN
   DBMS_DEFER_SYS.DELETE_ERROR(
      deferred_tran_id => '4.8.19521',
      destination => NULL);
END;
/
BEGIN
   DBMS_DEFER_SYS.DELETE_ERROR(
      deferred_tran_id => '5.44.19559',
      destination => NULL);
END;
/
```

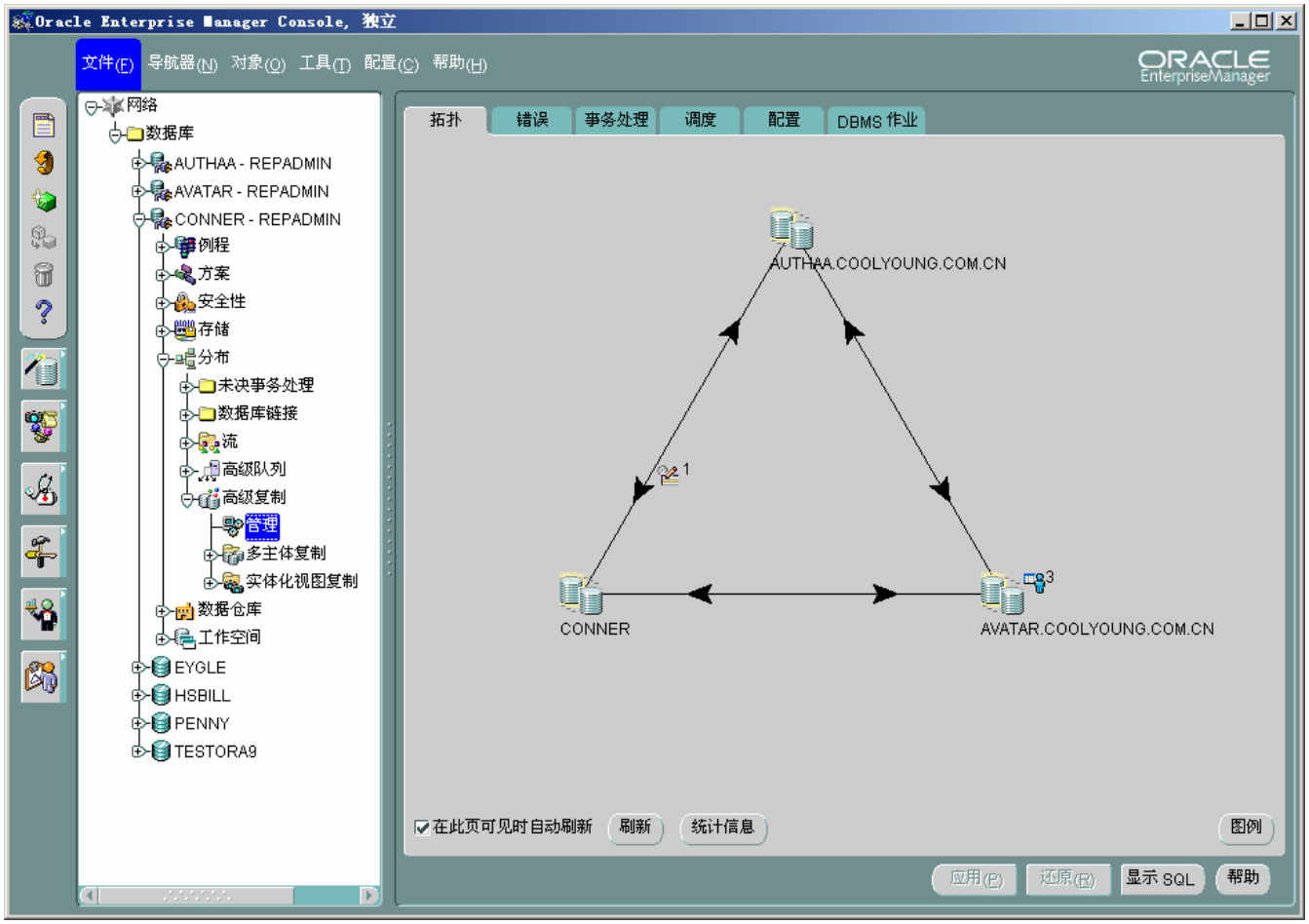
```
SQL> BEGIN
  2    DBMS_DEFER_SYS.DELETE_ERROR(
  3       deferred_tran_id => '4.8.19521',
  4       destination => NULL);
  5  END;
  6  /

PL/SQL procedure successfully completed

SQL> BEGIN
  2    DBMS_DEFER_SYS.DELETE_ERROR(
  3       deferred_tran_id => '5.44.19559',
  4       destination => NULL);
  5  END;
  6  /

PL/SQL procedure successfully completed

SQL> commit;

Commit complete

SQL> select * from deferror;

DEFERRED_TRAN_ID       ORIGIN_TRAN_DB          ORIGIN_TRAN_ID             CALLNO DESTINATION
START_TIME  ERROR_NUMBER ERROR_MSG                    RECEIVER
----------------------     ------------------------     ----------------------     ----------
------------------------     -----------     ------------     --------------------------------
------------------------------
```

在 avatar 节点:

```
SQL> connect repadmin/repadmin@avatar
Connected to Oracle9i Enterprise Edition Release 9.2.0.4.0
Connected as repadmin

SQL>
SQL> BEGIN
  2    DBMS_DEFER_SYS.DELETE_ERROR(
  3       deferred_tran_id => '10.44.294783',
  4       destination => NULL);
  5  END;
  6  /

PL/SQL procedure successfully completed

SQL> BEGIN
  2    DBMS_DEFER_SYS.DELETE_ERROR(
  3       deferred_tran_id => '4.19.294183',
  4       destination => NULL);
  5  END;
  6  /

PL/SQL procedure successfully completed

SQL> BEGIN
  2    DBMS_DEFER_SYS.DELETE_ERROR(
  3       deferred_tran_id => '6.11.291540',
  4       destination => NULL);
  5  END;
  6  /

PL/SQL procedure successfully completed

SQL> commit;
```

```
Commit complete

SQL>

在 Conner 节点:
SQL> connect repadmin/repadmin@conner
Connected to Oracle9i Enterprise Edition Release 9.2.0.4.0
Connected as repadmin

SQL>
SQL> BEGIN
  2    DBMS_DEFER_SYS.DELETE_ERROR(
  3       deferred_tran_id => '5.43.13195',
  4       destination => NULL);
  5  END;
  6  /

PL/SQL procedure successfully completed

SQL> commit;

Commit complete

SQL>
```

此时我们看图形界面，各节点已经恢复正常，但是数据歧义已经出现：



1.4.4 数据冲突和事务顺序

1.冲突

在拥有三个或更多主体站点的复制环境中，可能会出现顺序冲突。

比如某个对主体站点 X 的传播因某种原因被阻塞，而更新继续被传播至其他站点。当传播重新开始时，站点 X 上的事务更新顺序可能已经不同于其他站点，由此，冲突可能出现。

缺省的，冲突的结果被记录在错误日志中，可以等以来事务执行以后，通过重新执行来解决这一冲突。

为了确保多个主体站点之间的数据一致，你必须选择一个冲突解决方案来保证多个站点之间的数据一致性。

我们看一下包含三个复制站点的环境中，因为事务顺序而产生的冲突：

这个复制环境中有 A,B,C 三个主站点（和我的测试例子类似），每个站点都设置了优先级，A 是 30，B 是 25，C 是 10，x 是 site-priority 冲突解决方案 column group 中的一列，接下来让我们看看数据的歧义是怎样因 order 而产生的：

Time	Action	Site A	Site B	Site C
1	所有站点上 x=2	2	2	2
2	Site A，更新 x=5	5	2	2
3	此时 Site C 故障 Down 掉	5	2	Down
4	Site A 把更新推向 B；Site A 和 Site B 同步，x=5,站点 c 仍然不可用，更新事务仍然在 Site A 的队列中保持	5	5	Down
5	Site C 恢复可用,x=2;Site A,Site B 此时 x=5	5	5	2
6	此时 Site B，更新 x=7	5	7	2
7	Site B 推事务到 Site A；站点 A，B 数据同步为 7；此时站点 C x=2	7	7	2
8	Site B 推事务至 Site C；此时站点 C 的旧值为 2，而站点 B 的更新认为旧值为 5；此时产生更新冲突；根据站点优先级，所有站点在 x=7 达成一致	7	7	7
9	此时 Site A 成功推 x=5 的更新至站点 C；而此时 x=7，并不是更新需要的旧值 x=2;而站点 A 的优先级(30)高于站点 C 的优先级(10)；Oracle 执行 x=5 的更新；因为顺序的冲突，站点数据不再一致	7	7	5

2.参考完整性

对于前面所说的，如果事务传播顺序发生改变，那么复制站点上就可能存在参考完整性冲突。比如客户关系系统，订单以来于客户信息，如果客户信息晚于订单信息被传递，那么复制站点上就可能出现参考完整性冲突。

1.4.5 常见的冲突解决方案

1.4.5.1 解决更新冲突

1.4.5.1.1 覆盖和丢弃解决方案

Overwrite 和 Discard 机制忽略或者源站点或者目的站点的数据，因此不保证多站点之间的数据聚集。这种方法被设计用于单主体站点和多物化视图站点，或者一些用户定义的通知工具。

Overwrite 机制使用源站点的新值替换目的站点的旧值；Discard 机制反之，忽略源站点的新值。

我们看一下这种冲突解决方案的设置:

1.挂起复制

```
connect repadmin/repadmin@conner


BEGIN
DBMS_REPCAT.SUSPEND_MASTER_ACTIVITY (
gname => 'REP_SCT');
END;
/
```

2.创建 COLUMN GROUP

```
BEGIN
DBMS_REPCAT.MAKE_COLUMN_GROUP (
sname => 'scott',
oname => 'dept',
column_group => 'dep_cg',
list_of_column_names => ' DNAME,LOC');
END;
/
```

3.为特定数据表指定冲突解决方案，本例创建 OVERWRITE 冲突解决方案

```
BEGIN
DBMS_REPCAT.ADD_UPDATE_RESOLUTION (
sname => 'scott',
oname => 'dept',
column_group => 'dep_cg',
sequence_no => 1,
method => 'OVERWRITE',
parameter_column_name => ' LOC');
END;
/
```

4.重新生成复制支持

```
BEGIN
DBMS_REPCAT.GENERATE_REPLICATION_SUPPORT (
sname => 'scott',
oname => 'dept',
type => 'TABLE',
min_communication => TRUE);
END;
/
```

5.重新激活复制

```
BEGIN
DBMS_REPCAT.RESUME_MASTER_ACTIVITY (
gname => 'rep_sct');
END;
/
```

操作日志如下:

```
SQL> BEGIN
  2  DBMS_REPCAT.MAKE_COLUMN_GROUP (
  3  sname => 'scott',
  4  oname => 'dept',
  5  column_group => 'dep_cg',
  6  list_of_column_names => ' DNAME,LOC');
  7  END;
  8  /

PL/SQL procedure successfully completed

SQL>
SQL> BEGIN
  2  DBMS_REPCAT.ADD_UPDATE_RESOLUTION (
  3  sname => 'scott',
```

```
  4  oname => 'dept',
  5  column_group => 'dep_cg',
  6  sequence_no => 1,
  7  method => 'OVERWRITE',
  8  parameter_column_name => 'LOC');
  9  END;
 10  /

PL/SQL procedure successfully completed

SQL>
SQL> BEGIN
  2  DBMS_REPCAT.GENERATE_REPLICATION_SUPPORT (
  3  sname => 'scott',
  4  oname => 'dept',
  5  type => 'TABLE',
  6  min_communication => TRUE);
  7  END;
  8  /

PL/SQL procedure successfully completed

SQL>
SQL> BEGIN
  2  DBMS_REPCAT.RESUME_MASTER_ACTIVITY (
  3  gname => 'rep_sct');
  4  END;
  5  /

PL/SQL procedure successfully completed

SQL>
```

再次更新:

```
SQL> connect scott/tiger@conner
Connected to Oracle9i Enterprise Edition Release 9.2.0.4.0
Connected as scott

SQL> select * from dept;

DEPTNO DNAME         LOC
------ -------------- -------------
    10 ACCOUNTING     NEW YORK
    20 RESEARCH       DALLAS
    30 SALES          CHICAGO
    40 OPERATIONS     BEIJING

SQL> update dept set loc='BOSTON' where deptno=40;

1 row updated

SQL> commit;

Commit complete

SQL>
SQL> connect scott/tiger@avatar
Connected to Oracle9i Enterprise Edition Release 9.2.0.4.0
Connected as scott

SQL> select * from dept;

DEPTNO DNAME         LOC
------ -------------- -------------
    10 ACCOUNTING     NEW YORK
```

```
   20 RESEARCH      DALLAS
   30 SALES         CHICAGO
   40 OPERATIONS    BOSTON

SQL> connect scott/tiger@authaa
Connected to Oracle9i Enterprise Edition Release 9.2.0.4.0
Connected as scott

SQL> select * from dept;

DEPTNO DNAME         LOC
------ -------------- -------------
   10 ACCOUNTING    NEW YORK
   20 RESEARCH      DALLAS
   30 SALES         CHICAGO
   40 OPERATIONS    BOSTON

SQL>
```

在 OVERWRITE 的冲突解决机制之下，更新得以完成。

1.4.5.1.2 Minimum 和 Maximum 冲突解决方案

最小值或最大值解决方案是指，当高级复制检测到某个列组（column group）的冲突，通过比较指定列由源站点传递的新值和目的站点的当前值来纠正数据冲突的方法。

在最大值或最小值解决方案下，如果指定列的新值**小于**或者**大于**当前值(依赖于采用的方法)，那么从源站点传来的新值应用于目标站点或者被丢弃。

下面我们来看一下测试步骤以及最大值或最小值解决方案的具体作用。

1.使用复制管理员身份登陆

```
CONNECT repadmin/repadmin@conner
```

2.挂起复制组
注意:在单主体复制环境中，你可能不需要挂起复制

```
BEGIN
DBMS_REPCAT.SUSPEND_MASTER_ACTIVITY (
gname => 'REP_SCT');
END;
/
```

3.创建 column group

```
BEGIN
DBMS_REPCAT.MAKE_COLUMN_GROUP (
sname => 'SCOTT',
oname => 'EMP',
column_group => 'job_minsal_cg',
list_of_column_names => 'SAL');
END;
/
```

4.使用 DBMS_REPCAT.ADD_UPDATE_RESOLUTION 过程定义冲突解决方案
本例创建一个最小值冲突解决方案

```
BEGIN
DBMS_REPCAT.ADD_UPDATE_RESOLUTION (
sname => 'SCOTT',
oname => 'EMP',
column_group => 'job_minsal_cg',
sequence_no => 1,
method => 'MINIMUM',
parameter_column_name => 'SAL');
END;
/
```

5.重新生成复制支持

```
BEGIN
DBMS_REPCAT.GENERATE_REPLICATION_SUPPORT (
sname => 'SCOTT',
oname => 'EMP',
type => 'TABLE',
min_communication => TRUE);
END;
/
```

6.重新启动复制

```
BEGIN
DBMS_REPCAT.RESUME_MASTER_ACTIVITY (
gname => 'REP_SCT');
END;
/
```

操作日志如下:

```
SQL> connect repadmin/repadmin@conner
Connected.

SQL> select sname,master,gname from dba_repgroup;

SNAME                     M GNAME
------------------------------ - ------------------------------
REP_SCT                    Y REP_SCT

SQL> BEGIN
  2  DBMS_REPCAT.SUSPEND_MASTER_ACTIVITY (
  3  gname => 'REP_SCT');
```

```
  4  END;
  5  /

PL/SQL procedure successfully completed.

SQL> BEGIN
  2  DBMS_REPCAT.MAKE_COLUMN_GROUP (
  3  sname => 'SCOTT',
  4  oname => 'EMP',
  5  column_group => 'job_minsal_cg',
  6  list_of_column_names => 'SAL');
  7  END;
  8  /

PL/SQL procedure successfully completed.

SQL> BEGIN
  2  DBMS_REPCAT.ADD_UPDATE_RESOLUTION (
  3  sname => 'SCOTT',
  4  oname => 'EMP',
  5  column_group => 'job_minsal_cg',
  6  sequence_no => 1,
  7  method => 'MINIMUM',
  8  parameter_column_name => 'SAL');
  9  END;
 10  /

PL/SQL procedure successfully completed.

SQL> BEGIN
  2  DBMS_REPCAT.GENERATE_REPLICATION_SUPPORT (
  3  sname => 'SCOTT',
  3  oname => 'EMP',
```

```
  4  type => 'TABLE',
  5  min_communication => TRUE);
  6  END;
  7  /

PL/SQL procedure successfully completed.

SQL> BEGIN
  2  DBMS_REPCAT.RESUME_MASTER_ACTIVITY (
  3  gname => 'REP_SCT');
  4  END;
  5  /

PL/SQL procedure successfully completed.

SQL>
```

我们看一下测试过程:

1.假设各站点之间存在数据歧义

```
SQL> select * from emp where empno=7369;

    EMPNO ENAME      JOB       MGR HIREDATE        SAL      COMM    DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
     7369 SMITH      CLERK       7902 17-DEC-80       600                    20

SQL> select * from emp@avatar where empno=7369;

    EMPNO ENAME      JOB            MGR HIREDATE        SAL      COMM    DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
     7369 SMITH      CLERK         7902 17-DEC-80       500                    20

SQL> select * from emp@authaa where empno=7369;
```

```
    EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
     7369 SMITH      CLERK          7902 17-DEC-80        500                    20
```

2.我们在 CONNER 站点执行更新

依据我们定义的冲突解决规则，我们看到，这个更新没有出现错误，而是依据最小值把更新传播到其他主体站点。

```
SQL> update emp set sal=200 where empno=7369;

1 row updated.

SQL> commit;

Commit complete.

SQL> select * from emp where empno=7369;

    EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
     7369 SMITH      CLERK          7902 17-DEC-80        200                    20


SQL> select * from emp@authaa where empno=7369;

    EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
     7369 SMITH      CLERK          7902 17-DEC-80        200                    20

SQL> select * from emp@avatar where empno=7369;

    EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
```

```
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
     7369 SMITH      CLERK          7902 17-DEC-80       200                20


SQL>
```

这就是最小值解决方案的意义，最大值解决方案与此类似，不再介绍。

1.4.5.1.3 Timestamp 冲突解决方案

最早时间戳(earliest timestamp)和最晚时间戳(latest timestamp)解决方案是最大值和最小值解决方案的一种变体。如果要使用时间戳解决方案，那么你必须在复制表中指定一个类型为 DATE 的字段。当应用程序更新列组(column group)中的任意列时，必须使用本地 SYSDATE 更新时间列。对于从其他站点传递来的改变，时间戳应该被置为从源站点而来的时间值。

时间戳解决方案需要两个条件:

1.计算机之间的时间需要同步

2.需要有时间戳用触发器自动记录时间

通过以下步骤我们建立时间戳解决方案:

1.使用复制管理员身份连接主体定义站点

```
CONNECT repadmin/repadmin@conner
```

2.挂起复制

```
BEGIN
DBMS_REPCAT.SUSPEND_MASTER_ACTIVITY (
gname => 'REP_SCT');
END;
/
```

3.定义时间戳字段

如果目标表不包含时间戳，那么需要增加一个时间字段用以记录插入和更新时的时间戳。

你需要使用 ALTER_MASTER_REPOBJECT 过程应用 DDL 语句到目标表。

```
BEGIN
DBMS_REPCAT.ALTER_MASTER_REPOBJECT (
sname => 'SCOTT',
oname => 'EMP',
type => 'TABLE',
ddl_text => 'ALTER TABLE scott.emp ADD (timestamp DATE)');
END;
/
```

4.重新生成复制支持

```
BEGIN
DBMS_REPCAT.GENERATE_REPLICATION_SUPPORT (
sname => 'SCOTT',
oname => 'EMP',
type => 'TABLE',
min_communication => TRUE);
END;
/
```

5.创建一个触发器

用以记录数据更新或插入的时间，这个时间用来实现时间戳冲突解决方案。

我们需要使用 DBMS_REPCAT.CREATE_MASTER_REPOBJECT 过程去创建 Trigger 并增加到主体组。

```
BEGIN
 DBMS_REPCAT.CREATE_MASTER_REPOBJECT (
  gname => 'REP_SCT',
  type => 'TRIGGER',
  oname => 'insert_time',
  sname => 'SCOTT',
  ddl_text => 'CREATE TRIGGER scott.insert_time
    BEFORE
        INSERT OR UPDATE ON scott.emp FOR EACH ROW
```

```
    BEGIN
        IF DBMS_REPUTIL.FROM_REMOTE = FALSE THEN
            :NEW.TIMESTAMP := SYSDATE;
        END IF;
    END;');
END;
/
```

创建触发器之后我们可以查询 dba_repobject，确认该触发器已经被加入到复制组中：

```
select sname,oname,type,status,id,gname from dba_repobject where type='TRIGGER';
```

6.创建列组(Column group)

使用 DBMS_REPCAT.MAKE_COLUMN_GROUP 为目标表创建列组

```
BEGIN
DBMS_REPCAT.MAKE_COLUMN_GROUP (
sname => 'SCOTT',
oname => 'EMP',
column_group => 'emp_timestamp_cg',
list_of_column_names => 'EMPNO,MGR,timestamp');
END;
/
```

7.定义冲突解决方案

使用 DBMS_REPCAT.ADD_UPDATE_RESOLUTION 为指定表定义冲突解决方案。本例定义了一个最晚时间戳解决方案(LATEST TIMESTAMP).

```
BEGIN
DBMS_REPCAT.ADD_UPDATE_RESOLUTION (
sname => 'SCOTT',
oname => 'EMP',
column_group => 'emp_timestamp_cg',
sequence_no => 1,
method => 'LATEST TIMESTAMP',
```

```
parameter_column_name => 'timestamp');
END;
/
```

8.重新生成复制支持

```
BEGIN
DBMS_REPCAT.GENERATE_REPLICATION_SUPPORT (
sname => 'SCOTT',
oname => 'EMP',
type => 'TABLE',
min_communication => TRUE);
END;
/
```

9.重新启动复制

```
BEGIN
DBMS_REPCAT.RESUME_MASTER_ACTIVITY (
gname => 'REP_SCT');
END;
/
```

以下是实施过程:

```
SQL> connect repadmin/repadmin@conner
Connected.

SQL> BEGIN
  2  DBMS_REPCAT.SUSPEND_MASTER_ACTIVITY (
  3  gname => 'REP_SCT');
  4  END;
  5  /
```

```
PL/SQL procedure successfully completed.

SQL> BEGIN
  2  DBMS_REPCAT.ALTER_MASTER_REPOBJECT (
  3  sname => 'SCOTT',
  4  oname => 'EMP',
  5  type => 'TABLE',
  6  ddl_text => 'ALTER TABLE scott.emp ADD (timestamp DATE)');
  7  END;
  8  /

PL/SQL procedure successfully completed.

SQL>
SQL> BEGIN
  2  DBMS_REPCAT.GENERATE_REPLICATION_SUPPORT (
  3  sname => 'SCOTT',
  4  oname => 'EMP',
  5  type => 'TABLE',
  6  min_communication => TRUE);
  7  END;
  8  /

PL/SQL procedure successfully completed.



SQL> BEGIN
  2   DBMS_REPCAT.CREATE_MASTER_REPOBJECT (
  3    gname => 'REP_SCT',
  4    type => 'TRIGGER',
  5    oname => 'insert_time',
  6    sname => 'SCOTT',
  7    ddl_text => 'CREATE TRIGGER scott.insert_time
```

```
 8        BEFORE
 9              INSERT OR UPDATE ON scott.emp FOR EACH ROW
10        BEGIN
11              IF DBMS_REPUTIL.FROM_REMOTE = FALSE THEN
12                    :NEW.TIMESTAMP := SYSDATE;
13              END IF;
14        END;');
15  END;
16  /

PL/SQL procedure successfully completed.

SQL> select sname,oname,type,status,id,gname from dba_repobject where type='TRIGGER';

SNAME     ONAME                         TYPE           STATUS           ID GNAME
---------- ------------------------------ ---------------- ---------- ----------
SCOTT     INSERT_TIME                 TRIGGER        VALID          7645 REP_SCT
SQL> BEGIN
  2  DBMS_REPCAT.MAKE_COLUMN_GROUP (
  3  sname => 'SCOTT',
  4  oname => 'EMP',
  5  column_group => 'emp_timestamp_cg',
  6  list_of_column_names => 'EMPNO,MGR,timestamp');
  7  END;
  8  /

PL/SQL procedure successfully completed.



SQL> BEGIN
  2  DBMS_REPCAT.ADD_UPDATE_RESOLUTION (
  3  sname => 'SCOTT',
  4  oname => 'EMP',
```

```
  5  column_group => 'emp_timestamp_cg',
  6  sequence_no => 1,
  7  method => 'LATEST TIMESTAMP',
  8  parameter_column_name => 'timestamp');
  9  END;
 10  /

PL/SQL procedure successfully completed.



SQL> BEGIN
  2  DBMS_REPCAT.GENERATE_REPLICATION_SUPPORT (
  3  sname => 'SCOTT',
  4  oname => 'EMP',
  5  type => 'TABLE',
  6  min_communication => TRUE);
  7  END;
  8  /

PL/SQL procedure successfully completed.



SQL> BEGIN
  2  DBMS_REPCAT.RESUME_MASTER_ACTIVITY (
  3  gname => 'REP_SCT');
  4  END;
  5  /

PL/SQL procedure successfully completed.

SQL>
```

我们测试一下此方案在实际应用中的效果:

1.登陆 conner 站点

```
SQL> connect scott/tiger@conner
Connected.
SQL> select * from emp where empno=7369;

    EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO TIMESTAMP
--------- ---------- --------- --------- --------- ---------- ---------- ---------- ---------
     7369 SMITH      CLERK          7902 17-DEC-80        200                    20

SQL> alter session set nls_date_format='yyyy-mm-dd hh24:mi:ss';

Session altered.

SQL> @a
SQL> update emp set mgr=&mgrno where empno=7369;
Enter value for mgrno: 7698
old   1: update emp set mgr=&mgrno where empno=7369
new   1: update emp set mgr=7698 where empno=7369

1 row updated.

SQL> commit;

Commit complete.

SQL> select * from emp where empno=7369;

    EMPNO ENAME      JOB              MGR HIREDATE                     SAL       COMM     DEPTNO
TIMESTAMP
--------- ---------- --------- --------- ------------------- ---------- ---------- ----------
-------------------
     7369 SMITH      CLERK           7698 1980-12-17 00:00:00        200                    20
2005-07-10 23:04:47
```

```
SQL> select * from emp where empno=7369;

    EMPNO ENAME      JOB              MGR HIREDATE                 SAL       COMM     DEPTNO
TIMESTAMP
---------- ---------- --------- ---------- ------------------ ---------- ---------- ----------
-------------------
     7369 SMITH      CLERK           7698 1980-12-17 00:00:00        200                 20
2005-07-10 23:04:47

SQL> select * from emp where empno=7369;

    EMPNO ENAME      JOB              MGR HIREDATE                 SAL       COMM     DEPTNO
TIMESTAMP
---------- ---------- --------- ---------- ------------------ ---------- ---------- ----------
-------------------
     7369 SMITH      CLERK           7566 1980-12-17 00:00:00        200                 20
2005-07-10 23:06:52

SQL>
```

2.在 authaa 站点

```
SQL> connect scott/tiger@authaa
Connected.

SQL> set echo on
SQL> select * from emp where empno=7369;

    EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO TIMESTAMP
---------- ---------- --------- ---------- --------- ---------- ---------- ---------- ---------
     7369 SMITH      CLERK           7902 17-DEC-80        200                    20
```

```
SQL> alter session set nls_date_format='yyyy-mm-dd hh24:mi:ss';

Session altered.

SQL> @a
SQL> update emp set mgr=&mgrno where empno=7369;
Enter value for mgrno: 7839
old   1: update emp set mgr=&mgrno where empno=7369
new   1: update emp set mgr=7839 where empno=7369

1 row updated.

SQL> commit;

Commit complete.

SQL> select * from emp where empno=7369;

    EMPNO ENAME      JOB              MGR HIREDATE                 SAL       COMM     DEPTNO
TIMESTAMP
---------- ---------- --------- ---------- ------------------- ---------- ---------- ----------
-------------------
     7369 SMITH      CLERK           7839 1980-12-17 00:00:00        200                    20
2005-07-10 23:01:17

SQL> select * from emp where empno=7369;

    EMPNO ENAME      JOB              MGR HIREDATE                 SAL       COMM     DEPTNO
TIMESTAMP
---------- ---------- --------- ---------- ------------------- ---------- ---------- ----------
-------------------
     7369 SMITH      CLERK           7698 1980-12-17 00:00:00        200                    20
2005-07-10 23:04:47
```

```
SQL> select * from emp where empno=7369;

    EMPNO ENAME      JOB              MGR HIREDATE                 SAL       COMM     DEPTNO
TIMESTAMP
---------- ---------- --------- ---------- ------------------ ---------- ---------- ----------
-------------------
     7369 SMITH      CLERK           7566 1980-12-17 00:00:00        200                 20
2005-07-10 23:06:52

SQL>
```

3.在 avatar 站点

```
SQL> connect scott/tiger@avatar
Connected.
SQL> set echo on
SQL> select * from emp where empno=7369;

    EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO TIMESTAMP
---------- ---------- --------- ---------- --------- ---------- ---------- ---------- ---------
     7369 SMITH      CLERK          7902 17-DEC-80        200                    20

SQL> alter session set nls_date_format='yyyy-mm-dd hh24:mi:ss';

Session altered.

SQL> @a
SQL> update emp set mgr=&mgrno where empno=7369;
Enter value for mgrno: 7566
old   1: update emp set mgr=&mgrno where empno=7369
new   1: update emp set mgr=7566 where empno=7369

1 row updated.
```

```
SQL> commit;

Commit complete.

SQL> select * from emp where empno=7369;

     EMPNO ENAME      JOB              MGR HIREDATE                   SAL       COMM     DEPTNO
TIMESTAMP
---------- ---------- --------- ---------- ------------------- ---------- ---------- ----------
-------------------
      7369 SMITH      CLERK           7566 1980-12-17 00:00:00        200                    20
2005-07-10 23:06:52

SQL> select * from emp where empno=7369;

     EMPNO ENAME      JOB              MGR HIREDATE                   SAL       COMM     DEPTNO
TIMESTAMP
---------- ---------- --------- ---------- ------------------- ---------- ---------- ----------
-------------------
      7369 SMITH      CLERK           7566 1980-12-17 00:00:00        200                    20
2005-07-10 23:06:52

SQL> select * from emp where empno=7369;

     EMPNO ENAME      JOB              MGR HIREDATE                   SAL       COMM     DEPTNO
TIMESTAMP
---------- ---------- --------- ---------- ------------------- ---------- ---------- ----------
-------------------
      7369 SMITH      CLERK           7566 1980-12-17 00:00:00        200                    20
2005-07-10 23:06:52

SQL>
```

注意观察以上结果变化，我们会发现，数据最终同步为 7566，也就是 avatar 站点上的数据，因其拥有最晚时间戳为：2005-07-10 23:06:52。
在三次查询观察中，我们注意到 authaa 站点有一次同步为 7698，也就是 conner 站点的数据，因为 conner 与 authaa 相比具有更晚的时间戳。
这是正常的，也就是事务顺序和数据变化有关。

1.4.5.1.4 Priority Groups 冲突解决方案

优先组冲突解决方案

优先组允许你为特定列每个可能的值指定优先级。如果 Oracle 检测到冲突，将使用高优先级的数据更新低优先级的数据。

1.使用复制管理员登陆

```
CONNECT repadmin/repadmin@conner
```

2.挂起复制

```
BEGIN
DBMS_REPCAT.SUSPEND_MASTER_ACTIVITY (
gname => 'REP_SCT');
END;
/
```

3.定义字段

```
BEGIN
DBMS_REPCAT.MAKE_COLUMN_GROUP (
sname => 'scott',
oname => 'emp',
column_group => 'emp_priority_cg',
list_of_column_names => 'MGR,HIREDATE,SAL,JOB');
END;
/
```

4.创建优先级组

```
BEGIN
DBMS_REPCAT.DEFINE_PRIORITY_GROUP (
gname => 'rep_sct',
pgroup => 'job_pg',
datatype => 'VARCHAR2');
END;
/
```

5.定义优先值

DBMS_REPCAT.ADD_PRIORITY_<datatype>过程有几个不同版本可以使用，对于支持的数据类型，分别可以使用如 NUMBER, VARCHAR2 等。
执行这个过程定义所有可能值。

```
BEGIN
DBMS_REPCAT.ADD_PRIORITY_VARCHAR2(
gname => 'rep_sct',
pgroup => 'job_pg',
value => 'PRESIDENT',
priority => 100);
END;
/
BEGIN
DBMS_REPCAT.ADD_PRIORITY_VARCHAR2(
gname => 'rep_sct',
pgroup => 'job_pg',
value => 'MANAGER',
priority => 80);
END;
/
BEGIN
DBMS_REPCAT.ADD_PRIORITY_VARCHAR2(
```

```
gname => 'rep_sct',
pgroup => 'job_pg',
value => 'ANALYST',
priority => 60);
END;
/
BEGIN
DBMS_REPCAT.ADD_PRIORITY_VARCHAR2(
gname => 'rep_sct',
pgroup => 'job_pg',
value => 'SALESMAN',
priority => 40);
END;
/
BEGIN
DBMS_REPCAT.ADD_PRIORITY_VARCHAR2(
gname => 'rep_sct',
pgroup => 'job_pg',
value => 'CLERK',
priority => 20);
END;
/
```

6. 增加 PRIORITY GROUP 解决方案

```
BEGIN
DBMS_REPCAT.ADD_UPDATE_RESOLUTION (
sname => 'SCOTT',
oname => 'EMP',
column_group => 'emp_priority_cg',
sequence_no => 1,
method => 'PRIORITY GROUP',
parameter_column_name => 'JOB',
```

```
priority_group => 'job_pg');
END;
/
```

7.重新生成复制支持

```
BEGIN
DBMS_REPCAT.GENERATE_REPLICATION_SUPPORT (
sname => 'SCOTT',
oname => 'EMP',
type => 'TABLE',
min_communication => TRUE);
END;
/
```

8.激活复制

```
BEGIN
DBMS_REPCAT.RESUME_MASTER_ACTIVITY (
gname => 'REP_SCT');
END;
/
```

以下是具体操作过程：

```
SQL> BEGIN
  2  DBMS_REPCAT.SUSPEND_MASTER_ACTIVITY (
  3  gname => 'REP_SCT');
  4  END;
  5  /

PL/SQL procedure successfully completed

SQL>
SQL> BEGIN
```

```
  2  DBMS_REPCAT.MAKE_COLUMN_GROUP (
  3  sname => 'scott',
  4  oname => 'emp',
  5  column_group => 'emp_priority_cg',
  6  list_of_column_names => 'MGR,HIREDATE,SAL,JOB');
  7  END;
  8  /

PL/SQL procedure successfully completed

SQL>
SQL> BEGIN
  2  DBMS_REPCAT.DEFINE_PRIORITY_GROUP (
  3  gname => 'rep_sct',
  4  pgroup => 'job_pg',
  5  datatype => 'VARCHAR2');
  6  END;
  7  /

PL/SQL procedure successfully completed

SQL>
SQL> BEGIN
  2  DBMS_REPCAT.ADD_PRIORITY_VARCHAR2(
  3  gname => 'rep_sct',
  4  pgroup => 'job_pg',
  5  value => 'PRESIDENT',
  6  priority => 100);
  7  END;
  8  /

PL/SQL procedure successfully completed
```

```
SQL>
SQL> BEGIN
  2  DBMS_REPCAT.ADD_PRIORITY_VARCHAR2(
  3  gname => 'rep_sct',
  4  pgroup => 'job_pg',
  5  value => 'MANAGER',
  6  priority => 80);
  7  END;
  8  /

PL/SQL procedure successfully completed

SQL>
SQL> BEGIN
  2  DBMS_REPCAT.ADD_PRIORITY_VARCHAR2(
  3  gname => 'rep_sct',
  4  pgroup => 'job_pg',
  5  value => 'ANALYST',
  6  priority => 60);
  7  END;
  8  /

PL/SQL procedure successfully completed

SQL>
SQL> BEGIN
  2  DBMS_REPCAT.ADD_PRIORITY_VARCHAR2(
  3  gname => 'rep_sct',
  4  pgroup => 'job_pg',
  5  value => 'SALESMAN',
  6  priority => 40);
  7  END;
  8  /
```

```
PL/SQL procedure successfully completed

SQL> BEGIN
  2  DBMS_REPCAT.ADD_PRIORITY_VARCHAR2(
  3  gname => 'rep_sct',
  4  pgroup => 'job_pg',
  5  value => 'CLERK',
  6  priority => 20);
  7  END;
  8  /

PL/SQL procedure successfully completed

SQL>
SQL> BEGIN
  2  DBMS_REPCAT.ADD_UPDATE_RESOLUTION (
  3  sname => 'SCOTT',
  4  oname => 'EMP',
  5  column_group => 'emp_priority_cg',
  6  sequence_no => 1,
  7  method => 'PRIORITY GROUP',
  8  parameter_column_name => 'JOB',
  9  priority_group => 'job_pg');
 10  END;
 11  /

PL/SQL procedure successfully completed

SQL>
SQL> BEGIN
  2  DBMS_REPCAT.GENERATE_REPLICATION_SUPPORT (
  3  sname => 'SCOTT',
```

```
  4  oname => 'EMP',
  5  type => 'TABLE',
  6  min_communication => TRUE);
  7  END;
  8  /

PL/SQL procedure successfully completed

SQL> BEGIN
  2  DBMS_REPCAT.RESUME_MASTER_ACTIVITY (
  3  gname => 'REP_SCT');
  4  END;
  5  /

PL/SQL procedure successfully completed

SQL>
```

查询一下定义结果：

```
SQL> select GNAME,SNAME,PRIORITY_GROUP,DATA_TYPE,VARCHAR2_VALUE,PRIORITY from dba_reppriority;

GNAME      SNAME      PRIORITY_GROUP                DATA_TYPE VARCHAR2_VALUE    PRIORITY
---------- ---------- ------------------------------ --------- --------------- ----------
REP_SCT    REP_SCT    JOB_PG                        VARCHAR2  PRESIDENT            100
REP_SCT    REP_SCT    JOB_PG                        VARCHAR2  MANAGER               80
REP_SCT    REP_SCT    JOB_PG                        VARCHAR2  ANALYST               60
REP_SCT    REP_SCT    JOB_PG                        VARCHAR2  SALESMAN              40
REP_SCT    REP_SCT    JOB_PG                        VARCHAR2  CLERK                 20
```

我们看一下此方案的具体作用：

```
SQL> connect scott/tiger@authaa
Connected.
SQL> set echo on
```

```
SQL> select * from emp where empno=7934;

    EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
     7934 MILLER     CLERK          7782 23-JAN-82      1300                    10

SQL> @a
SQL> update emp set job='&newjob' where empno=7934;
Enter value for newjob: MANAGER
old   1: update emp set job='&newjob' where empno=7934
new   1: update emp set job='MANAGER' where empno=7934

1 row updated.

SQL> commit;

Commit complete.

SQL> select * from emp where empno=7934;

    EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
     7934 MILLER     MANAGER        7782 23-JAN-82      1300                    10


SQL> connect scott/tiger@avatar
Connected.

SQL> select * from emp where empno=7934;

    EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
     7934 MILLER     CLERK          7782 23-JAN-82      1300                    10
```

```
SQL> @a
Enter value for newjob: SALESMAN
old   1: update emp set job='&newjob' where empno=7934
new   1: update emp set job='SALESMAN' where empno=7934

1 row updated.



Commit complete.

SQL> select * from emp where empno=7934;

    EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
     7934 MILLER     SALESMAN       7782 23-JAN-82      1300                  10

SQL> /

    EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
     7934 MILLER     MANAGER        7782 23-JAN-82      1300                  10
```

1.4.5.1.5 Site Priority 冲突解决方案

站点优先级(Site Priority)冲突解决方案

站点优先级解决方案是优先组解决方案的一种特殊形式。因此，许多站点优先级过程和优先组过程非常类似。

和优先组解决方案按照域值来定义不同，站点优先级解决方案按照定义站点确定优先级。

例如，如果存在两个站点，定义 conner.eygle.com 的优先级高于 authaa.eygle.com 的优先级，那么当发生冲突时，conner.eygle.com 的值被接受。

1.使用复制管理员身份连接

```
CONNECT repadmin/repadmin@conner
```

2.挂起复制

```
BEGIN
DBMS_REPCAT.SUSPEND_MASTER_ACTIVITY (
gname => 'REP_SCT');
END;
/
```

3.增加站点列以存储站点值

```
BEGIN
DBMS_REPCAT.ALTER_MASTER_REPOBJECT (
sname => 'SCOTT',
oname => 'EMP',
type => 'TABLE',
ddl_text => 'ALTER TABLE scott.emp ADD (site VARCHAR2(30))');
END;
/
```

4.重新生成复制支持

```
BEGIN
DBMS_REPCAT.GENERATE_REPLICATION_SUPPORT (
sname => 'scott',
oname => 'emp',
type => 'TABLE',
min_communication => TRUE);
END;
/
```

5.创建触发器

当更新或插入时，记录站点的 global name。

```
BEGIN
DBMS_REPCAT.CREATE_MASTER_REPOBJECT (
gname => 'rep_sct',
type => 'TRIGGER',
oname => 'insert_site',
sname => 'scott',
ddl_text => 'CREATE TRIGGER scott.insert_site
BEFORE
INSERT OR UPDATE ON scott.emp FOR EACH ROW
BEGIN
IF DBMS_REPUTIL.FROM_REMOTE = FALSE THEN
SELECT global_name INTO :NEW.SITE FROM GLOBAL_NAME;
END IF;
END;');
END;
/
```

6.创建列组

注意，对于站点优先级解决方案，新增加的 site 列需要包含在列组中。

```
BEGIN
DBMS_REPCAT.MAKE_COLUMN_GROUP (
sname => 'scott',
oname => 'emp',
column_group => 'ename_sitepriority_cg',
list_of_column_names => 'ename,site');
END;
/
```

7.创建站点优先组

```
BEGIN
```

```
DBMS_REPCAT.DEFINE_SITE_PRIORITY (
gname => 'rep_sct',
name => 'ename_sitepriority_pg');
END;
/
```

8.定义站点优先级

```
BEGIN
DBMS_REPCAT.ADD_SITE_PRIORITY_SITE (
gname => 'rep_sct',
name => 'ename_sitepriority_pg',
site => 'CONNER.EYGLE.COM',
priority => 100);
END;
/
BEGIN
DBMS_REPCAT.ADD_SITE_PRIORITY_SITE (
gname => 'rep_sct',
name => 'ename_sitepriority_pg',
site => 'AUTHAA.EYGLE.COM',
priority => 50);
END;
/
BEGIN
DBMS_REPCAT.ADD_SITE_PRIORITY_SITE (
gname => 'rep_sct',
name => 'ename_sitepriority_pg',
site => 'AVATAR.EYGLE.COM',
priority => 25);
END;
/
```

9.创建冲突解决方案

```
BEGIN
DBMS_REPCAT.ADD_UPDATE_RESOLUTION (
sname => 'scott',
oname => 'emp',
column_group => 'ename_sitepriority_cg',
sequence_no => 1,
method => 'SITE PRIORITY',
parameter_column_name => 'site',
priority_group => 'ename_sitepriority_pg');
END;
/
```

10.生成复制支持

```
BEGIN
DBMS_REPCAT.GENERATE_REPLICATION_SUPPORT (
sname => 'scott',
oname => 'emp',
type => 'TABLE',
min_communication => TRUE);
END;
/
```

11.激活复制

```
BEGIN
DBMS_REPCAT.RESUME_MASTER_ACTIVITY (
GNAME => 'rep_sct');
END;
/
```

以下是在主体定义站点的操作日志：

```
SQL> BEGIN
  2  DBMS_REPCAT.SUSPEND_MASTER_ACTIVITY (
```

```
  3  gname => 'REP_SCT');
  4  END;
  5  /

PL/SQL procedure successfully completed

SQL>
SQL> BEGIN
  2  DBMS_REPCAT.ALTER_MASTER_REPOBJECT (
  3  sname => 'SCOTT',
  4  oname => 'EMP',
  5  type => 'TABLE',
  6  ddl_text => 'ALTER TABLE scott.emp ADD (site VARCHAR2(30))');
  7  END;
  8  /

PL/SQL procedure successfully completed

SQL>
SQL> BEGIN
  2  DBMS_REPCAT.GENERATE_REPLICATION_SUPPORT (
  3  sname => 'scott',
  4  oname => 'emp',
  5  type => 'TABLE',
  6  min_communication => TRUE);
  7  END;
  8  /

PL/SQL procedure successfully completed

SQL>
SQL> BEGIN
  2  DBMS_REPCAT.CREATE_MASTER_REPOBJECT (
```

```
  3  gname => 'rep_sct',
  4  type => 'TRIGGER',
  5  oname => 'insert_site',
  6  sname => 'scott',
  7  ddl_text => 'CREATE TRIGGER scott.insert_site
  8  BEFORE
  9  INSERT OR UPDATE ON scott.emp FOR EACH ROW
 10  BEGIN
 11  IF DBMS_REPUTIL.FROM_REMOTE = FALSE THEN
 12  SELECT global_name INTO :NEW.SITE FROM GLOBAL_NAME;
 13  END IF;
 14  END;');
 15  END;
 16  /

PL/SQL procedure successfully completed

SQL>
SQL> BEGIN
  2  DBMS_REPCAT.MAKE_COLUMN_GROUP (
  3  sname => 'scott',
  4  oname => 'emp',
  5  column_group => 'ename_sitepriority_cg',
  6  list_of_column_names => 'ename,site');
  7  END;
  8  /

PL/SQL procedure successfully completed

SQL>
SQL> BEGIN
  2  DBMS_REPCAT.DEFINE_SITE_PRIORITY (
  3  gname => 'rep_sct',
```

```
  4  name => 'ename_sitepriority_pg');
  5  END;
  6  /

PL/SQL procedure successfully completed

SQL>
SQL> BEGIN
  2  DBMS_REPCAT.ADD_SITE_PRIORITY_SITE (
  3  gname => 'rep_sct',
  4  name => 'ename_sitepriority_pg',
  5  site => 'CONNER.EYGLE.COM',
  6  priority => 100);
  7  END;
  8  /

PL/SQL procedure successfully completed

SQL> BEGIN
  2  DBMS_REPCAT.ADD_SITE_PRIORITY_SITE (
  3  gname => 'rep_sct',
  4  name => 'ename_sitepriority_pg',
  5  site => 'AUTHAA.EYGLE.COM',
  6  priority => 50);
  7  END;
  8  /

PL/SQL procedure successfully completed

SQL> BEGIN
  2  DBMS_REPCAT.ADD_SITE_PRIORITY_SITE (
  3  gname => 'rep_sct',
  4  name => 'ename_sitepriority_pg',
```

```
  5  site => 'AVATAR.EYGLE.COM',
  6  priority => 25);
  7  END;
  8  /

PL/SQL procedure successfully completed

SQL>
SQL> BEGIN
  2  DBMS_REPCAT.ADD_UPDATE_RESOLUTION (
  3  sname => 'scott',
  4  oname => 'emp',
  5  column_group => 'ename_sitepriority_cg',
  6  sequence_no => 1,
  7  method => 'SITE PRIORITY',
  8  parameter_column_name => 'site',
  9  priority_group => 'ename_sitepriority_pg');
 10  END;
 11  /

PL/SQL procedure successfully completed

SQL>
SQL> BEGIN
  2  DBMS_REPCAT.GENERATE_REPLICATION_SUPPORT (
  3  sname => 'scott',
  4  oname => 'emp',
  5  type => 'TABLE',
  6  min_communication => TRUE);
  7  END;
  8  /

PL/SQL procedure successfully completed
```

```
SQL>
SQL> BEGIN
  2  DBMS_REPCAT.RESUME_MASTER_ACTIVITY (
  3  GNAME => 'rep_sct');
  4  END;
  5  /

PL/SQL procedure successfully completed

SQL>
```

下面看一下站点优先级解决方案的作用:

在 conner 站点:

```
SQL> connect scott/tiger@conner
Connected.
SQL> set echo on
SQL> select * from emp where empno=7839;

    EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO SITE
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
     7839 KING       PRESIDENT          17-NOV-81       5000                    10

SQL> update emp set ename='&newname' where empno=7839;
Enter value for newname: EYGLE.COM
old   1: update emp set ename='&newname' where empno=7839
new   1: update emp set ename='EYGLE.COM' where empno=7839

1 row updated.

SQL> commit;
```

```
Commit complete.

SQL> select * from emp where empno=7839;

    EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO SITE
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
------------------------------
     7839 EYGLE.COM   PRESIDENT               17-NOV-81        5000                       10
CONNER.EYGLE.COM

SQL>
```

在 authaa 站点:

```
SQL> connect scott/tiger@authaa
Connected.
SQL> set echo on
SQL> select * from emp where empno=7839;

    EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO SITE
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
------------------------------
     7839 KING      PRESIDENT          17-NOV-81      5000                10

SQL> update emp set ename='&newname' where empno=7839;
Enter value for newname: EYGLEe
old   1: update emp set ename='&newname' where empno=7839
new   1: update emp set ename='EYGLEe' where empno=7839

1 row updated.

SQL> commit;

Commit complete.
```

```
SQL> select * from emp where empno=7839;

    EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO SITE
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
------------------------------
     7839 EYGLEe     PRESIDENT           17-NOV-81       5000                    10 AUTHAA.EYGLE.COM


SQL> /

    EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO SITE
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
------------------------------
     7839  EYGLE.COM   PRESIDENT               17-NOV-81        5000                         10
CONNER.EYGLE.COM

SQL>
```

在 avatar 站点:

```
SQL> connect scott/tiger@avatar
Connected.
SQL> set echo on
SQL> select * from emp where empno=7839;

    EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO SITE
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
------------------------------
     7839 KING       PRESIDENT           17-NOV-81       5000                    10

SQL> update emp set ename='&newname' where empno=7839;
Enter value for newname: EYGLE
old   1: update emp set ename='&newname' where empno=7839
```

```
new   1: update emp set ename='EYGLE' where empno=7839

1 row updated.

SQL> commit;

Commit complete.

SQL> select * from emp where empno=7839;

    EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO SITE
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
------------------------------
     7839 EYGLE      PRESIDENT           17-NOV-81       5000                    10 AVATAR.EYGLE.COM

SQL> /

    EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO SITE
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
------------------------------
     7839  EYGLE.COM   PRESIDENT               17-NOV-81        5000                         10
CONNER.EYGLE.COM

SQL>
```

我们注意到，由于站点 conner 具有最高的优先级，最后数据都统一到 EYGLE.COM 上来。这就是站点优先的作用。

1.4.5.2唯一性冲突解决方案

在一个复制环境中，经常在 Insert 的过程中，你可能会遇到唯一性约束冲突。如果你的商业规则允许你删除重复行，你可以使用 Oracle 预定义的冲突解决方法来解决这一类的冲突。

然而，更为常见的是，你可能想要去修改冲突值以避免冲突，修改冲突值可以确保你不会丢失重要数据。Oracle 预定义的唯一性冲突解决方案可以通过增加站点名称或序列使得冲突值变得唯一。

伴随唯一性冲突解决方案提供的还有另外一个通知工具。冲突信息被修改从而得以插入相应数据表，但是 DBA 应该被通知以分析冲突并最终决定是删除该记录还是合并或者保留。

为了满足测试的需要，我们给 emp 表增加一个 email 字段，并建立唯一性索引，加入复制组。

以下操作需要在三个站点上执行，保证数据结构的一致性.

```
SQL> connect scott/tiger@conner
Connected.
SQL> ALTER TABLE emp ADD CONSTRAINT emp_email_uk UNIQUE (email);

Table altered.

SQL> select index_name,table_name from user_indexes;

INDEX_NAME                     TABLE_NAME
------------------------------ ------------------------------
EMP_EMAIL_UK                    EMP
PK_DEPT                        DEPT
PK_EMP                         EMP
```

赋予初值(可选):

```
SQL>  update scott.emp set email=ename||'@itpub.net';

14 rows updated.

SQL> commit;
```

```
Commit complete.
```

增加复制对象:

```
SQL> BEGIN
  2  DBMS_REPCAT.CREATE_MASTER_REPOBJECT (
  3  gname => 'rep_sct',
  4  type => 'TABLE',
  5  oname => 'EMP',
  6  sname => 'scott',
  7  use_existing_object => TRUE,
  8  copy_rows => FALSE);
  9  END;
 10  /

PL/SQL procedure successfully completed

SQL> BEGIN
  2  DBMS_REPCAT.CREATE_MASTER_REPOBJECT (
  3  gname => 'rep_sct',
  4  type => 'INDEX',
  5  oname => 'emp_email_uk',
  6  sname => 'scott',
  7  use_existing_object => TRUE,
  8  copy_rows => FALSE);
  9  END;
 10  /

PL/SQL procedure successfully completed
```

产生复制支持:

```
SQL>
SQL> BEGIN
```

```
  2  DBMS_REPCAT.GENERATE_REPLICATION_SUPPORT (
  3  sname => 'scott',
  4  oname => 'emp',
  5  type => 'TABLE',
  6  min_communication => TRUE);
  7  END;
  8  /

PL/SQL procedure successfully completed
```

至此，准备工作完成。

以下是测试步骤:

1 使用复制管理员身份连接

```
CONNECT repadmin/repadmin@conner
```

2.挂起复制

```
BEGIN
DBMS_REPCAT.SUSPEND_MASTER_ACTIVITY (
gname => 'REP_SCT');
END;
/
```

3.创建数据表存储通知工具传递的信息

```

BEGIN
DBMS_REPCAT.EXECUTE_DDL (
gname => 'REP_SCT',
ddl_text => 'CREATE TABLE scott.conf_report (
line        NUMBER(2),
txt         VARCHAR2(80),
timestamp   DATE,
table_name  VARCHAR2(30),
```

```
table_owner     VARCHAR2(30),
conflict_type   VARCHAR2(7))');
END;
/
```

4.使用步骤 3 中通知表的用户属主连接

```
CONNECT scott/tgier@conner
```

5.创建 Package 传递错误通知至通知表

```
CREATE OR REPLACE PACKAGE notify AS
FUNCTION emp_unique_violation (
email IN OUT VARCHAR2,
discard_new_values IN OUT BOOLEAN)
RETURN BOOLEAN;
END notify;
/
```

```
CREATE OR REPLACE PACKAGE BODY notify AS
TYPE message_table IS TABLE OF VARCHAR2(80) INDEX BY BINARY_INTEGER;
PROCEDURE report_conflict(conflict_report IN MESSAGE_TABLE,
report_length IN NUMBER,
conflict_time IN DATE,
conflict_table IN VARCHAR2,
table_owner IN VARCHAR2,
conflict_type IN VARCHAR2) IS
BEGIN
FOR idx IN 1..report_length LOOP
BEGIN
INSERT INTO scott.conf_report
(line, txt, timestamp, table_name, table_owner, conflict_type)
VALUES (idx, SUBSTR(conflict_report(idx),1,80), conflict_time,
conflict_table, table_owner, conflict_type);
```

```
EXCEPTION WHEN others THEN NULL;
END;
END LOOP;
END report_conflict;
FUNCTION emp_unique_violation(email IN OUT VARCHAR2,
discard_new_values IN OUT BOOLEAN)
RETURN BOOLEAN IS
local_node VARCHAR2(128);
conf_report MESSAGE_TABLE;
conf_time DATE := SYSDATE;
BEGIN
BEGIN
SELECT global_name INTO local_node FROM global_name;
EXCEPTION WHEN others THEN local_node := '?';
END;
conf_report(1) := 'UNIQUENESS CONFLICT DETECTED IN EMP ON ' ||
TO_CHAR(conf_time, 'MM-DD-YYYY HH24:MI:SS');
conf_report(2) := ' AT NODE ' || local_node;
conf_report(3) := 'ATTEMPTING TO RESOLVE CONFLICT USING' ||
' APPEND SITE NAME METHOD';
conf_report(4) := 'EMAIL: ' || email;
conf_report(5) := NULL;
report_conflict(conf_report,5,conf_time,'emp','scott','UNIQUE');
discard_new_values := FALSE;
RETURN FALSE;
END emp_unique_violation;
END notify;
/
```

6.使用复制管理员连接

```
CONNECT repadmin/repadmin@conner
```

7.复制步骤 5 中创建的 Package 至所有的主体站点

```
BEGIN
DBMS_REPCAT.CREATE_MASTER_REPOBJECT (
gname => 'rep_sct',
type => 'PACKAGE',
oname => 'notify',
sname => 'scott');
END;
/
BEGIN
DBMS_REPCAT.CREATE_MASTER_REPOBJECT (
gname => 'rep_sct',
type => 'PACKAGE BODY',
oname => 'notify',
sname => 'scott');
END;
/
```

8.增加通知工具

```
BEGIN
DBMS_REPCAT.ADD_UNIQUE_RESOLUTION(
sname => 'scott',
oname => 'emp',
constraint_name => 'emp_email_uk',
sequence_no => 1,
method => 'USER FUNCTION',
comment => 'Notify DBA',
parameter_column_name => 'email',
function_name => 'scott.notify.emp_unique_violation');
END;
/
```

9.增加冲突解决方案

```
BEGIN
DBMS_REPCAT.ADD_UNIQUE_RESOLUTION(
sname => 'scott',
oname => 'emp',
constraint_name => 'emp_email_uk',
sequence_no => 2,
method => 'APPEND SITE NAME',
parameter_column_name => 'email');
END;
/
```

10.重新生成复制支持

```
BEGIN
DBMS_REPCAT.GENERATE_REPLICATION_SUPPORT (
sname => 'scott',
oname => 'emp',
type => 'TABLE',
min_communication => TRUE);
END;
/
```

11.激活复制

```
BEGIN
DBMS_REPCAT.RESUME_MASTER_ACTIVITY (
gname => 'rep_sct');
END;
/
```

下面是执行过程:

```
SQL> connect repadmin/repadmin@conner
Connected.
```

```
SQL> BEGIN
  2  DBMS_REPCAT.SUSPEND_MASTER_ACTIVITY (
  3  gname => 'REP_SCT');
  4  END;
  5  /

PL/SQL procedure successfully completed.



SQL> BEGIN
  2  DBMS_REPCAT.EXECUTE_DDL (
  3  gname => 'REP_SCT',
  4  ddl_text => 'CREATE TABLE scott.conf_report (
  5  line       NUMBER(2),
  6  txt        VARCHAR2(80),
  7  timestamp  DATE,
  8  table_name VARCHAR2(30),
  9  table_owner    VARCHAR2(30),
 10  conflict_type  VARCHAR2(7))');
 11  END;
 12  /

PL/SQL procedure successfully completed

SQL> CONNECT scott/tiger@conner
Connected.



SQL> CREATE OR REPLACE PACKAGE notify AS
  2  FUNCTION emp_unique_violation (
  3  email IN OUT VARCHAR2,
  4  discard_new_values IN OUT BOOLEAN)
```

```
  5  RETURN BOOLEAN;
  6  END notify;
  7  /

Package created

SQL> CREATE OR REPLACE PACKAGE BODY notify AS
  2  TYPE message_table IS TABLE OF VARCHAR2(80) INDEX BY BINARY_INTEGER;
  3  PROCEDURE report_conflict(conflict_report IN MESSAGE_TABLE,
  4  report_length IN NUMBER,
  5  conflict_time IN DATE,
  6  conflict_table IN VARCHAR2,
  7  table_owner IN VARCHAR2,
  8  conflict_type IN VARCHAR2) IS
  9  BEGIN
 10  FOR idx IN 1..report_length LOOP
 11  BEGIN
 12  INSERT INTO scott.conf_report
 13  (line, txt, timestamp, table_name, table_owner, conflict_type)
 14  VALUES (idx, SUBSTR(conflict_report(idx),1,80), conflict_time,
 15  conflict_table, table_owner, conflict_type);
 16  EXCEPTION WHEN others THEN NULL;
 17  END;
 18  END LOOP;
 19  END report_conflict;
 20  FUNCTION emp_unique_violation(email IN OUT VARCHAR2,
 21  discard_new_values IN OUT BOOLEAN)
 22  RETURN BOOLEAN IS
 23  local_node VARCHAR2(128);
 24  conf_report MESSAGE_TABLE;
 25  conf_time DATE := SYSDATE;
 26  BEGIN
 27  BEGIN
```

```
 28  SELECT global_name INTO local_node FROM global_name;
 29  EXCEPTION WHEN others THEN local_node := '?';
 30  END;
 31  conf_report(1) := 'UNIQUENESS CONFLICT DETECTED IN EMP ON ' ||
 32  TO_CHAR(conf_time, 'MM-DD-YYYY HH24:MI:SS');
 33  conf_report(2) := ' AT NODE ' || local_node;
 34  conf_report(3) := 'ATTEMPTING TO RESOLVE CONFLICT USING' ||
 35  ' APPEND SITE NAME METHOD';
 36  conf_report(4) := 'EMAIL: ' || email;
 37  conf_report(5) := NULL;
 38  report_conflict(conf_report,5,conf_time,'emp','scott','UNIQUE');
 39  discard_new_values := FALSE;
 40  RETURN FALSE;
 41  END emp_unique_violation;
 42  END notify;
 43  /

Package body created

SQL> CONNECT repadmin/repadmin@conner
Connected to Oracle9i Enterprise Edition Release 9.2.0.4.0
Connected as repadmin

SQL>
SQL> BEGIN
  2  DBMS_REPCAT.CREATE_MASTER_REPOBJECT (
  3  gname => 'rep_sct',
  4  type => 'PACKAGE',
  5  oname => 'notify',
  6  sname => 'scott');
  7  END;
  8  /
```

```
PL/SQL procedure successfully completed

SQL> BEGIN
  2  DBMS_REPCAT.CREATE_MASTER_REPOBJECT (
  3  gname => 'rep_sct',
  4  type => 'PACKAGE BODY',
  5  oname => 'notify',
  6  sname => 'scott');
  7  END;
  8  /

PL/SQL procedure successfully completed

SQL>
SQL> BEGIN
  2  DBMS_REPCAT.ADD_UNIQUE_RESOLUTION(
  3  sname => 'scott',
  4  oname => 'emp',
  5  constraint_name => 'emp_email_uk',
  6  sequence_no => 1,
  7  method => 'USER FUNCTION',
  8  comment => 'Notify DBA',
  9  parameter_column_name => 'email',
 10  function_name => 'scott.notify.emp_unique_violation');
 11  END;
 12  /

PL/SQL procedure successfully completed

SQL>
SQL> BEGIN
  2  DBMS_REPCAT.ADD_UNIQUE_RESOLUTION(
  3  sname => 'scott',
```

```
  4  oname => 'emp',
  5  constraint_name => 'emp_email_uk',
  6  sequence_no => 2,
  7  method => 'APPEND SITE NAME',
  8  parameter_column_name => 'email');
  9  END;
 10  /

PL/SQL procedure successfully completed

SQL>

SQL> BEGIN
  2  DBMS_REPCAT.GENERATE_REPLICATION_SUPPORT (
  3  sname => 'scott',
  4  oname => 'emp',
  5  type => 'TABLE',
  6  min_communication => TRUE);
  7  END;
  8  /

PL/SQL procedure successfully completed

SQL>
SQL> BEGIN
  2  DBMS_REPCAT.RESUME_MASTER_ACTIVITY (
  3  gname => 'rep_sct');
  4  END;
  5  /

PL/SQL procedure successfully completed

SQL>
```

下面让我们来看一下唯一性冲突解决方案的作用:
在 authaa 站点:

```
SQL> connect scott/tiger@authaa
Connected.
SQL> set echo on
SQL> @a
SQL> set linesize 190
SQL> select * from emp where empno=&&empno;
Enter value for empno: 7902
old   1: select * from emp where empno=&&empno
new   1: select * from emp where empno=7902

    EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO EMAIL
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
-----------------------
     7902 FORD       ANALYST         7566 03-DEC-81       3000                    20 FORD@itpub.net

SQL> update emp set email='eygle@itpub.net' where empno=&empno;
old   1: update emp set email='eygle@itpub.net' where empno=&empno
new   1: update emp set email='eygle@itpub.net' where empno=7902

1 row updated.

SQL> commit;

Commit complete.

SQL> select * from emp where empno=&empno;
old   1: select * from emp where empno=&empno
new   1: select * from emp where empno=7902

    EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO EMAIL
```

```
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
-------------------------
     7902 FORD      ANALYST        7566 03-DEC-81      3000                20 eygle@itpub.net

SQL> select * from emp where empno=7934;

    EMPNO ENAME     JOB            MGR HIREDATE        SAL      COMM    DEPTNO EMAIL
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
------------------------
     7934 MILLER      MANAGER           7782 23-JAN-82        1300                      10
eygle@itpub.net.AVATAR
```

在 avatar 站点:

```
SQL> connect scott/tiger@avatar
Connected.
SQL> set echo on
SQL> @a
SQL> set linesize 190
SQL> select * from emp where empno=&&empno;
Enter value for empno: 7934
old   1: select * from emp where empno=&&empno
new   1: select * from emp where empno=7934

    EMPNO ENAME     JOB            MGR HIREDATE        SAL      COMM    DEPTNO EMAIL
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
--------------------------
     7934 MILLER     MANAGER        7782 23-JAN-82      1300                10 MILLER@itpub.net

SQL> update emp set email='eygle@itpub.net' where empno=&empno;
old   1: update emp set email='eygle@itpub.net' where empno=&empno
new   1: update emp set email='eygle@itpub.net' where empno=7934
```

```
1 row updated.

SQL> commit;

Commit complete.

SQL> select * from emp where empno=&empno;
old   1: select * from emp where empno=&empno
new   1: select * from emp where empno=7934

    EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO EMAIL
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
     7934 MILLER     MANAGER         7782 23-JAN-82       1300                 10 eygle@itpub.net



SQL> select * from emp where empno=7902;

    EMPNO ENAME      JOB              MGR HIREDATE         SAL    COMM     DEPTNO EMAIL
---------- ---------- --------- ---------- --------- ----------- ------ ----------
     7902 FORD       ANALYST         7566 03-DEC-81       3000           20 eygle@itpub.net.AUTHAA
```

我们看到在此解决方案下，更改被赋予新值记录在表中，数据得以保存，剩下来的就是根据实际情况来解决具体问题了。

查询通知表可以获得冲突日志:

```
SQL> select * from scott.conf_report@avatar;

LINE TXT                                                          TIMESTAMP   TABLE_NAME
TABLE_OWNER    CONFLICT_TYPE
---- ------------------------------------------------------------------------------
   1 UNIQUENESS CONFLICT DETECTED IN EMP ON 07-11-2005 17:57:02              2005-7-11 1 emp
scott          UNIQUE
   2  AT NODE AVATAR.EYGLE.COM                                              2005-7-11 1 emp
scott          UNIQUE
   3 ATTEMPTING TO RESOLVE CONFLICT USING APPEND SITE NAME METHOD            2005-7-11 1 emp
```

```
scott         UNIQUE
   4 EMAIL: eygle@itpub.net                                                2005-7-11 1 emp
scott         UNIQUE
   5                                                          2005-7-11 1 emp        scott
UNIQUE
```

1.4.5.3创建冲突解决方案避免删除冲突

对于更新冲突，有两个值可以比较，可以用以解决冲突；然而简单的删除操作使我们之前讨论的相关的解决方案变得无效，一旦数据删除之后，无可比较，处理冲突往往变得复杂。

最好的应对删除冲突的方法是，在删除时，通过标记一行为删除而不是真正的删除它；然后定期的清理这些标记为删除的数据。

因为你不是物理的删除记录，那么你的变更可以在所以站点上传播并聚合，当冲突发生时，数据仍然存在可供比较。当你确定数据已经聚合，不存在问题，那么就可以调用相应的过程，把这些标记为删除的记录清除掉。

那么注意，当你开发前端应用时就要注意，在查询中排除这些标记过的记录，以免给用户带来歧义。简单的，你可以通过类似的语句排除这些已经"被删除"的记录:

```
SELECT * FROM hr.locations WHERE remove_date IS NULL;
```

1 使用复制管理员身份连接

```
CONNECT repadmin/repadmin@conner
```

2.挂起复制

```
BEGIN
DBMS_REPCAT.SUSPEND_MASTER_ACTIVITY (
gname => 'REP_SCT');
END;
/
```

3.向复制表中添加一列

用以标记删除记录，建议采用时间戳(timestamp)标记删除的记录。

```
BEGIN
DBMS_REPCAT.ALTER_MASTER_REPOBJECT (
sname => 'scott',
oname => 'emp',
type => 'TABLE',
ddl_text => 'ALTER TABLE scott.emp ADD (remove_date DATE)');
END;
/
```

4.重新生成复制支持

```
BEGIN
DBMS_REPCAT.GENERATE_REPLICATION_SUPPORT (
sname => 'scott',
oname => 'emp',
type => 'TABLE',
min_communication => TRUE);
END;
/
```

5.创建一个 Package 用以清楚标记过的记录

```
BEGIN
DBMS_REPCAT.CREATE_MASTER_REPOBJECT (
gname => 'rep_sct',
type => 'PACKAGE',
oname => 'purge',
sname => 'scott',
ddl_text => 'CREATE OR REPLACE PACKAGE scott.purge AS
PROCEDURE remove_emp(purge_date DATE);
END;');
END;
```

```
/


BEGIN
DBMS_REPCAT.CREATE_MASTER_REPOBJECT (
gname => 'rep_sct',
type => 'PACKAGE BODY',
oname => 'purge',
sname => 'scott',
ddl_text => 'CREATE OR REPLACE PACKAGE BODY scott.purge AS
PROCEDURE remove_emp(purge_date IN DATE) IS
BEGIN
DBMS_REPUTIL.REPLICATION_OFF;
LOCK TABLE scott.emp IN EXCLUSIVE MODE;
DELETE scott.emp WHERE remove_date IS NOT NULL
AND remove_date < purge_date;
DBMS_REPUTIL.REPLICATION_ON;
EXCEPTION WHEN others THEN
DBMS_REPUTIL.REPLICATION_ON;
END;
END;');
END;
/
```

6.为 Package 和 Package Body 产生复制支持

```
BEGIN
DBMS_REPCAT.GENERATE_REPLICATION_SUPPORT (
sname => 'scott',
oname => 'purge',
type => 'PACKAGE',
min_communication => TRUE);
END;
/
```

```
BEGIN
DBMS_REPCAT.GENERATE_REPLICATION_SUPPORT (
sname => 'scott',
oname => 'purge',
type => 'PACKAGE BODY',
min_communication => TRUE);
END;
/
```

7.手工执行

在独立的终端窗口,在其他主体站点手工 push 管理请求，你可能需要执行 DO_DEFERRED_REPCAT_ADMIN 过程数次，因为一些管理请求有多步操作。

```
BEGIN
DBMS_REPCAT.DO_DEFERRED_REPCAT_ADMIN (
gname => 'rep_sct',
all_sites => FALSE);
END;
/
```

8.激活复制

```
BEGIN
DBMS_REPCAT.RESUME_MASTER_ACTIVITY (
gname => 'rep_sct');
END;
/
```

下面是具体操作日志:

```
[oracle@jumper oracle]$ sqlplus /nolog


SQL*Plus: Release 9.2.0.4.0 - Production on Mon Jul 11 22:21:26 2005
```

```
Copyright (c) 1982, 2002, Oracle Corporation.  All rights reserved.


SQL> CONNECT repadmin/repadmin@conner
Connected.
SQL> set echo on
SQL> BEGIN
  2  DBMS_REPCAT.SUSPEND_MASTER_ACTIVITY (
  3  gname => 'REP_SCT');
  4  END;
  5  /

PL/SQL procedure successfully completed.

SQL>
SQL> BEGIN
  2  DBMS_REPCAT.ALTER_MASTER_REPOBJECT (
  3  sname => 'scott',
  4  oname => 'emp',
  5  type => 'TABLE',
  6  ddl_text => 'ALTER TABLE scott.emp ADD (remove_date DATE)');
  7  END;
  8  /

PL/SQL procedure successfully completed.

SQL> BEGIN
  2  DBMS_REPCAT.GENERATE_REPLICATION_SUPPORT (
  3  sname => 'scott',
  4  oname => 'emp',
  5  type => 'TABLE',
  6  min_communication => TRUE);
  7  END;
  8  /
```

```
PL/SQL procedure successfully completed.

SQL>
SQL> BEGIN
  2  DBMS_REPCAT.CREATE_MASTER_REPOBJECT (
  3  gname => 'rep_sct',
  4  type => 'PACKAGE',
  5  oname => 'purge',
  6  sname => 'scott',
  7  ddl_text => 'CREATE OR REPLACE PACKAGE scott.purge AS
  8  PROCEDURE remove_emp(purge_date DATE);
  9  END;');
 10  END;
 11  /

PL/SQL procedure successfully completed.

SQL>
SQL> BEGIN
  2  DBMS_REPCAT.CREATE_MASTER_REPOBJECT (
  3  gname => 'rep_sct',
  4  type => 'PACKAGE BODY',
  5  oname => 'purge',
  6  sname => 'scott',
  7  ddl_text => 'CREATE OR REPLACE PACKAGE BODY scott.purge AS
  8  PROCEDURE remove_emp(purge_date IN DATE) IS
  9  BEGIN
 10  DBMS_REPUTIL.REPLICATION_OFF;
 11  LOCK TABLE scott.emp IN EXCLUSIVE MODE;
 12  DELETE scott.emp WHERE remove_date IS NOT NULL
 13  AND remove_date < purge_date;
 14  DBMS_REPUTIL.REPLICATION_ON;
```

```
 15  EXCEPTION WHEN others THEN
 16  DBMS_REPUTIL.REPLICATION_ON;
 17  END;
 18  END;');
 19  END;
 20  /

PL/SQL procedure successfully completed.

SQL> BEGIN
  2  DBMS_REPCAT.GENERATE_REPLICATION_SUPPORT (
  3  sname => 'scott',
  4  oname => 'purge',
  5  type => 'PACKAGE',
  6  min_communication => TRUE);
  7  END;
  8  /

PL/SQL procedure successfully completed.

SQL> BEGIN
  2  DBMS_REPCAT.GENERATE_REPLICATION_SUPPORT (
  3  sname => 'scott',
  4  oname => 'purge',
  5  type => 'PACKAGE BODY',
  6  min_communication => TRUE);
  7  END;
  8  /

PL/SQL procedure successfully completed.

SQL> BEGIN
  2  DBMS_REPCAT.DO_DEFERRED_REPCAT_ADMIN (
```

```
  3  gname => 'rep_sct',
  4  all_sites => FALSE);
  5  END;
  6  /

PL/SQL procedure successfully completed.

SQL> BEGIN
  2  DBMS_REPCAT.RESUME_MASTER_ACTIVITY (
  3  gname => 'rep_sct');
  4  END;
  5  /

PL/SQL procedure successfully completed.

SQL>
```

在此解决方案之下，常规的删除操作可以转变为 Mark 记录，也就是记录 remove_date,然后通过 Package 定期清除这些标记过的记录。

这实际上已经完全转变为用户代码的编写和处理过程，如果我们有自己的冲突处理逻辑需要实现，只需要按照同样的步骤处理即可。到最后，一切都只不过是代码的编写过程。

1.5 常用命令

1.5.1 如何挂起或激活复制组

在高级复制的维护中，很多时候都需要挂起复制才能进行具体操作，可以使用如下命令挂起复制操作：

```
BEGIN
DBMS_REPCAT.SUSPEND_MASTER_ACTIVITY (
gname => 'REP_SCT');
END;
```

```
/
```

使用如下命令激活复制组：

```
BEGIN
DBMS_REPCAT.RESUME_MASTER_ACTIVITY (
gname => 'REP_SCT');
END;
/
```

1.5.2 如何删除复制对象

首先需要挂起复制组，然后执行如下命令：

```
SQL> exec DBMS_REPCAT.DROP_MASTER_REPOBJECT(sname => 'scott',oname => 'emp',type => 'table');

PL/SQL procedure successfully completed
```

1.5.3 查询列组及相应列

```
SQL> col sname for a10
SQL> col group_comment for a10
SQL> select * from DBA_REPCOLUMN_GROUP;

SNAME      ONAME                          GROUP_NAME                     GROUP_COMM
---------- ------------------------------ ------------------------------ ----------
SCOTT      EMP                            EMP_TIMESTAMP_CG
SCOTT      DEPT                           DEP_CG
SCOTT      EMP                            JOB_MINSAL_CG
SCOTT      EMP                            EMP_PRIORITY_CG

SQL> col column_name for a10
SQL> select * from DBA_REPGROUPED_COLUMN;
```

```
SNAME      ONAME                          GROUP_NAME                     COLUMN_NAM
---------- ------------------------------ ------------------------------ ----------
SCOTT      DEPT                           DEP_CG                         DNAME
SCOTT      DEPT                           DEP_CG                         LOC
SCOTT      EMP                            EMP_PRIORITY_CG                HIREDATE
SCOTT      EMP                            EMP_PRIORITY_CG                JOB
SCOTT      EMP                            EMP_TIMESTAMP_CG               EMPNO
SCOTT      EMP                            EMP_TIMESTAMP_CG               MGR
SCOTT      EMP                            EMP_TIMESTAMP_CG               TIMESTAMP
SCOTT      EMP                            JOB_MINSAL_CG                  SAL

8 rows selected
```

1.5.4 查询已定义的冲突解决方案

```
SQL> select * from dba_repconflict;

SNAME      ONAME                          CONFLICT_TYPE REFERENCE_NAME
---------- ------------------------------ ------------- ------------------------------
SCOTT      EMP                            UPDATE        EMP_TIMESTAMP_CG
SCOTT      DEPT                           UPDATE        DEP_CG
SCOTT      EMP                            UPDATE        JOB_MINSAL_CG
```

1.5.5 如何删除已定义的冲突解决方案

首先需要挂起复制组：

```
SQL> BEGIN
  2  DBMS_REPCAT.SUSPEND_MASTER_ACTIVITY (
  3  gname => 'REP_SCT');
  4  END;
  5  /
```

```
PL/SQL procedure successfully completed
```

```
SQL> exec dbms_repcat.drop_update_resolution(sname => 'scott',oname => 'emp',column_group =>
'EMP_TIMESTAMP_CG',sequence_no => 1)

PL/SQL procedure successfully completed

SQL> select * from dba_repconflict;

SNAME      ONAME            CONFLICT_TYPE REFERENCE_NAME
---------- ---------------- ------------- ------------------------------
SCOTT      DEPT             UPDATE        DEP_CG
SCOTT      EMP              UPDATE        JOB_MINSAL_CG

SQL> exec dbms_repcat.drop_update_resolution(sname => 'scott',oname => 'emp',column_group =>
'JOB_MINSAL_CG',sequence_no => 1)

PL/SQL procedure successfully completed

SQL> select * from dba_repconflict;

SNAME      ONAME                        CONFLICT_TYPE REFERENCE_NAME
---------- ---------------------------- ------------- ------------------------------
SCOTT      DEPT                         UPDATE        DEP_CG

SQL>
```

1.5.6 如何删除列组

如果想删除列组，首先需要删除包含列组的冲突解决方案，否则无法删除。

```
SQL> select * from dba_repcolumn_group;

SNAME      ONAME                          GROUP_NAME                     GROUP_COMM
---------- ------------------------------ ------------------------------ ----------
SCOTT      EMP                            EMP_TIMESTAMP_CG
SCOTT      DEPT                           DEP_CG
SCOTT      EMP                            JOB_MINSAL_CG
SCOTT      EMP                            EMP_PRIORITY_CG

SQL>  exec  dbms_repcat.drop_column_group(sname  =>  'scott',oname  =>  'emp',column_group  =>
'EMP_TIMESTAMP_CG')

PL/SQL procedure successfully completed

SQL>  exec  dbms_repcat.drop_column_group(sname  =>  'scott',oname  =>  'emp',column_group  =>
'EMP_PRIORITY_CG')

PL/SQL procedure successfully completed

SQL>  exec  dbms_repcat.drop_column_group(sname  =>  'scott',oname  =>  'emp',column_group  =>
'JOB_MINSAL_CG')

PL/SQL procedure successfully completed

SQL> select * from dba_repcolumn_group;

SNAME      ONAME                          GROUP_NAME                     GROUP_COMM
---------- ------------------------------ ------------------------------ ----------
SCOTT      DEPT                           DEP_CG
```

此后需要重新生成复制支持，然后启动复制：

```
SQL> BEGIN
  2  DBMS_REPCAT.GENERATE_REPLICATION_SUPPORT (
  3  sname => 'SCOTT',
  4  oname => 'EMP',
  5  type => 'TABLE',
  6  min_communication => TRUE);
  7  END;
  8  /

PL/SQL procedure successfully completed

SQL> select * from dba_repcolumn_group@authaa;

SNAME      ONAME                          GROUP_NAME                     GROUP_COMM
---------- ------------------------------ ------------------------------ ----------
SCOTT      DEPT                           DEP_CG
SCOTT      EMP                            EMP_PRIORITY_CG
SCOTT      EMP                            EMP_TIMESTAMP_CG
SCOTT      EMP                            JOB_MINSAL_CG

SQL>
SQL> BEGIN
  2  DBMS_REPCAT.RESUME_MASTER_ACTIVITY (
  3  gname => 'REP_SCT');
  4  END;
  5  /

PL/SQL procedure successfully completed

SQL> select * from dba_repcolumn_group@authaa;

SNAME      ONAME                          GROUP_NAME                     GROUP_COMM
---------- ------------------------------ ------------------------------ ----------
```

```
SCOTT     DEPT                      DEP_CG

SQL>
```

1.5.7 如何修改表的结构定义

首先挂起复制：

```
SQL> BEGIN
  2  DBMS_REPCAT.SUSPEND_MASTER_ACTIVITY (
  3  gname => 'REP_SCT');
  4  END;
  5  /

PL/SQL procedure successfully completed
```

通过 DBMS_REPCAT.ALTER_MASTER_REPOBJECT 进行表结构修改:

```

SQL> BEGIN
  2  DBMS_REPCAT.ALTER_MASTER_REPOBJECT (
  3  sname => 'SCOTT',
  4  oname => 'EMP',
  5  type => 'TABLE',
  6  ddl_text => 'ALTER TABLE scott.emp modify (site VARCHAR2(30))');
  7  END;
  8  /

PL/SQL procedure successfully completed
```

重新生成复制支持：

```

SQL> BEGIN
```

```
  2  DBMS_REPCAT.GENERATE_REPLICATION_SUPPORT (
  3  sname => 'scott',
  4  oname => 'emp',
  5  type => 'TABLE',
  6  min_communication => TRUE);
  7  END;
  8  /

PL/SQL procedure successfully completed
```

激活复制：

```
SQL> BEGIN
  2  DBMS_REPCAT.RESUME_MASTER_ACTIVITY (
  3  GNAME => 'rep_sct');
  4  END;
  5  /

PL/SQL procedure successfully completed

SQL>
```

作者简介：

盖国强，网名 eygle

Itpub Oracle 管理版版主，Itpub 论坛超级版主，曾任 ITPUB MS 版版主。
CSDN eMag Oracle 电子杂志主编。

曾任职于某国家大型企业，服务于烟草行业,开发过基于 Oracle 数据库的大型 ERP 系统，属国家信息产业部重点工程。同时负责 Oracle 数据库管理及优化，并为多家烟草企业提供 Oracle 数据库管理、优化及技术支持。

目前任职于北京某电信增值业务系统提供商企业，首席 DBA，负责数据库业务。管理全国 30 多个数据库系统。项目经验丰富，曾设计规划及支持中国联通增值业务等大型数据库系统。

实践经验丰富，长于数据库诊断、性能调整与 SQL 优化等。对于 Oracle 内部技术具有深入研究。

高级培训讲师，培训经验丰富，曾主讲 itpub dba 培训及 itpub 高级性能调整等主要课程。
《Oracle 数据库 DBA 专题技术精粹》、《Oracle 数据库性能优化》两书的主编及主要作者。

你可以在 http://www.eygle.com 上找到关于作者的更多信息。