Asynchronous Change Data Capture Cookbook

April 2007

ORACLE

Change Data Capture is a generic term that is used to describe technology to capture incremental changes being applied to a data store. With the amount of data in data stores growing and growing Change Data Capture is key enabling functionality for data warehouses, specifically real-time or near real-time data warehouses.

In the context of the Oracle Database, Change Data Capture is database functionality that enables capturing incremental changes against an Oracle Database. Traditionally you would have to modify the source application in order to capture incremental changes. Oracle's Change Data Capture enables incremental change capture without making any changes to a source application.

Oracle Database 10*g* Release 2 introduces asynchronous distributed Change Data Capture functionality against Oracle Database 9*i* Release 2 and higher. This cookbook describes how to setup such an asynchronous Change Data Capture environment.

CONCEPTS AND TERMINOLOGY

From this point onwards the abbreviation CDC will be used to refer to the Oracle Database Change Data Capture feature.

Publishers and subscribers

CDC uses the concept of publishers and subscribers. A publisher is a database user that publishes captured change data. A subscriber is a database user that consumes the change data through a so-called subscription. For security reasons publisher and subscriber should not be the same database user. One publisher can support many subscribers.

CDC makes use of change tables and subscriber views. Changes are written to the change tables to get a scalable infrastructure for using the changes. Subscribers to the changes get their own subscriber views (database views) against the change tables so that they look at a consistent set of data. Subscribers can extend and purge their subscription windows, implicitly changing the data set that is visible through the database views. Data can be purged from the change tables if no subscribers include the data in their views anymore. CDC is controlled using calls to PL/SQL database packages that are part of the Oracle Database.

CDC implementation methods

CDC has different implementation methods:

- Synchronous CDC: with this implementation method you capture changes synchronously on the source database into change tables. This method uses internal database triggers to enable CDC. Capturing the change is part of the original transaction that introduces the change thus impacting the performance of the transaction.
- Asynchronous Autolog CDC: this implementation method requires a staging database separate from the source database. Asynchronous Autolog CDC uses the database's redo transport services to transport redo log information from the source database to the staging database[1]. Changes are captured at the staging database. The impact to the source system is minimal, but there is some latency between the original transaction and the change being captured.
- Asynchronous Hotlog CDC: this implementation method uses the online redo log files to capture changes. The original transaction performs changes to the tables, causing information to be written to the redo log files. A separate process mines the redo log files and captures the changes in the change tables. There is a small latency between the original transaction and the change being captured. For Asynchronous Hotlog CDC the change capture takes place on the same database.
- Asynchronous Distributed Hotlog CDC (see Figure 1): this implementation method requires a staging database in order to setup the change capture. As with the Asynchronous Hotlog CDC method the change capture is performed by a separate process on the source database. The change record is then propagated to the staging database where the change is written to the change tables. Oracle Streams provides the underlying infrastructure for this CDC method. There is some latency between the original transaction and the change being captured. Asynchronous Distributed Hotlog CDC requires an Enterprise Edition database for both source and staging databases.

1 Manually copying the redo flog files between the source and the staging database is also supported. You will have to explicitly assign the logs to the staging database in order to do accomplish the change capture. Please refer to the Oracle Streams Concepts and Administration documentation for the details.

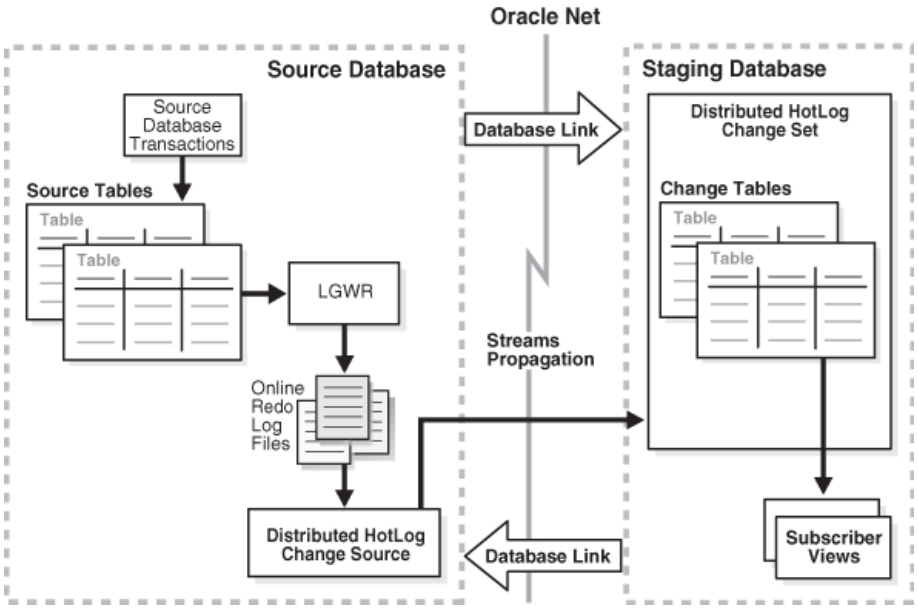

Figure 1: Asynchronous Distributed Hotlog architecture.

This cookbook only discusses the Asynchronous Distributed Hotlog CDC method which enables an Oracle Database 9*i* Release 2 database or higher version as a source database. Oracle Database 10*g* Release 2 is used as the staging database. The other CDC methods are either contained in one database or require the same Oracle Database 10*g* version for both source and staging.

In an asynchronous distributed CDC environment you require a publisher in both the source database and the staging database. Subscribers are only in the staging database. In order to use the underlying Oracle Streams functionality the publishers in both databases have to be granted the DBA role. For security reasons you should prevent users from connecting as the change data publishers.

Supplemental logging

By default the database only logs sufficient redo information to replay a transaction. Supplemental logging is a database concept to enable logging additional information to the redo log in order to make the logs easier to understand. If you want to use the change data, then your downstream process becomes much easier knowing the context of the change.

For example, if you change one non-key column in a table with one hundred columns, then Oracle Database will by default only log the changed column. If you want to use this change to update your data warehouse, then it will be difficult to identify the record that was changed. Rather than going through a resource-intensive query to figure out what record has changed, you can use supplemental logging in order to log additional column values, such as the primary or unique key values. Supplemental logging will have some, but typically minimal impact on the

performance of update transactions on the source database. Also, depending on the amount of updates relative to other DML, the redo log files will grow somewhat faster.

You can enable supplemental logging either on the database level (e.g. add supplemental logging for all primary keys), or for individual tables[2]. With a supplemental log group you can specify which column values you always want to be logged. Please refer to the Oracle Database documentation for more details.

ENVIRONMENT

The source system for this cookbook is an Oracle Database 9*i* Release 2, version 9.2.0.7, running on Redhat Enterprise Linux 3 (update 4), with database patch 4285404[3]. The source database instance is called CDCCBS, and runs in archive log mode. The starting point for this cookbook is that you have installed the database software and created a database. Of course you do not have to use the same database name, but then you will have to modify some statements in this cookbook accordingly.

Asynchronous distributed CDC requires at least version 9.2.0.6.

To verify the correct version of your database, use the following query (note that you have to run catpatch.sql after you applied a major patchset, and for 9*i*, even after you create a new database from a template):

```
SQL> select comp_name, version
  2  from dba_registry
  3  where comp_id = 'CATPROC';

COMP_NAME                      VERSION
------------------------------ -----------
Oracle9i Packages and Types    9.2.0.7.0
```

The staging system is an Oracle Database 10*g* Release 2 database, version 10.2.0.1, running on a different Redhat Enterprise Linux 3 system (update 6). The staging database is called CDCCBSTG. This cookbook assumes that you installed the database software and created a database.

Required source initialization parameters

A number of database initialization parameters are mandatory in order to setup asynchronous distributed hotlog CDC. The settings below are minimum settings.

For optimal performance and/or in heavily loaded environments you should increase values appropriately in order to meet your performance needs.

---

2 If you use Oracle Database 10*g* Release 2 as the source for your asynchronous distributed hotlog CDC environment, then supplemental logging will be automatically enabled for the primary key (or unique key) on your source table(s).

3 The scripts in this cookbook have also been verified using Oracle Database 10*g* 10.1.0.4 and 10.2.0.1 as source database without any additional patches.

On the 9.2 source database:

```
compatible='9.2.0.0.0'
global_names=TRUE
job_queue_processes=2
log_archive_dest_1='LOCATION=<full path>'
log_archive_format='%t_%s.dbf'
log_parallelism=1
open_links=4
open_links_per_instance=4
parallel_max_servers=3
processes=150
undo_retention=3600
```

Higher values for the initialization parameters are typically fine[4] [5]. Depending on the source application and the load on the application some of the values will already be much higher. To add CDC on an existing database you may have to adjust the values for these parameters. Refer to the Change Data Capture chapter in the Oracle Database Data Warehousing Guide documentation for the details.

You can verify the value for an initialization parameter using the "show parameter" feature in SQL Plus. For example, connect to the database as SYS (as SYSDBA), and type:

```
SQL> show parameter parallel
NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------
fast_start_parallel_rollback         string      LOW
log_parallelism                      integer     1
parallel_adaptive_multi_user         boolean     FALSE
parallel_automatic_tuning            boolean     FALSE
parallel_execution_message_size      integer     2148
parallel_instance_group              string
parallel_max_servers                 integer     5
parallel_min_percent                 integer     0
parallel_min_servers                 integer     0
parallel_server                      boolean     FALSE
parallel_server_instances            integer     1
parallel_threads_per_cpu             integer     2
recovery_parallelism                 integer     0
SQL>
```

To change parameter values, you can use the alter system command, for example (connect as SYS, as SYSDBA):

4 The compatibility parameter should match the database version you use to make optimal use of the database features underlying asynchronous distributed CDC.

5 For Oracle Database 9*i* log_parallelism has to be set to 1; this parameter no long exists in Oracle Database 10*g*.

```
alter system set log_parallelism=1 scope=spfile ;
```

Note that due to the scope=spfile portion of the command you have to restart the database to implement the new setting. If you use a pfile, then you can modify the pfile and restart the database pointing to the updated pfile (tip: backup the old pfile).

Your source database has to run in archive log mode. If you are currently not running in archive log mode, then make sure you set the required archive log initialization parameters as specified before, for example (connect as SYS, as SYSDBA):

```
alter system set log_archive_dest_1=
'LOCATION=/private/oracle/oradata/cdccbs/archive' scope=spfile
;

alter system set log_archive_format='%t_%s.dbf' scope=spfile ;
```

You then have to shutdown the database in order to enable archivelog mode:

```
shutdown immediate

startup mount

alter database archivelog ;

alter database open ;
```

Verify to make sure you run in archivelog mode:

```
alter system switch logfile ;
```

Check the archivelog folder on the file system to make sure a logfile was just archived. Do not continue until you successfully run in archivelog mode on the source database.

Required staging initialization parameters

On the Oracle Database 10*g* Release 2 staging database you need at least the following initialization settings:

```
compatible='10.2.0.1.0'
global_names=TRUE
open_links=4
open_links_per_instance=4
parallel_max_servers=2
processes=150
java_pool_size=50M
streams_pool_size=50M
```

Depending on the type of application(s) you run on the staging database you should increase the values of the above parameters. Refer to the Change Data Capture chapter in the Oracle Database Data Warehousing Guide for the details.

Connection settings

In order to use CDC, the databases have to be able to communicate with each other over a database link. The database links use TNS connect information to communicate with other databases.

For both the source and the staging environment, edit the file <oracle database home>/network/admin/tnsnames.ora and add an entry for the other database. Alternatively, run the Oracle Net Configuration Assistant (netca) and add a Local Net Service Name configuration.

For example, on the source, add the connect information for the staging database:

```
CDCCBSTG =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)
        (HOST = <host name>)(PORT = <listener port staging db>)
      )
    )
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = cdccbstg)
    )
  )
```

Verify the connect information you specified by either using SQL Plus (out of the same Oracle home) using a connection to the remote database as SYSTEM, or use the test button in the Oracle Net Configuration Assistant.

Tip: you may have to verify the value for NAMES.DEFAULT_DOMAIN setting in <oracle database home>/network/admin/sqlnet.ora. You may have to include the value for this parameter in the naming of your tnsnames.ora entry. Refer to the Oracle Database Net Services Reference for details if you cannot get this step to work.

SET UP THE DATABASE USERS

On the source database, you will create two database users: (1) the user who owns the source tables, CDC_SOURCE, and (2) the publisher on the source database, CDC_SOURCE_PUB. On the staging database you will also create two database users: (1) the publisher on the staging database, CDC_STG_PUB, and (2) the subscriber on the staging database, CDC_STG_USER.

Source Oracle Database 9*i* Release 2

Connect to the source database as SYS, and execute the following statements in order to create the users with the right privileges (note that you should choose your own secret passwords):

```
create user cdc_source
identified by cdc_source
default tablespace users
temporary tablespace temp ;

grant connect, resource, select any table to cdc_source ;
```

Note that the select any table privilege is not required, but will in this case be taken advantage of by using create table as select * from.

```
create user cdc_source_pub
identified by cdc_source_pub
default tablespace users
temporary tablespace temp
quota unlimited on system
quota unlimited on users ;

/* For Oracle Database 10g as source database, also perform:

alter user cdc_source_pub quota unlimited on sysaux ;

*/

grant create session, create table, create database link,
select_catalog_role, execute_catalog_role, dba to
cdc_source_pub ;

grant execute on dbms_aqadm to cdc_source_pub ;

grant execute on dbms_capture_adm to cdc_source_pub ;

grant execute on dbms_apply_adm to cdc_source_pub ;

grant execute on dbms_propagation_adm to cdc_source_pub ;

grant execute on dbms_streams_adm to cdc_source_pub ;
```

```
begin

  dbms_rule_adm.grant_system_privilege (
    privilege => DBMS_RULE_ADM.CREATE_RULE_SET_OBJ,
    grantee => 'cdc_source_pub',
    grant_option => FALSE);

  dbms_rule_adm.grant_system_privilege (
    privilege => DBMS_RULE_ADM.CREATE_RULE_OBJ,
    grantee => 'cdc_source_pub',
    grant_option => FALSE);

END;
/
```

Make sure you use a default tablespace for your database users in order to prevent database objects from being created in the SYSTEM (or on Oracle Database 10*g*, the SYSAUX) tablespace.

Staging Oracle Database 10*g* Release 2

Connect to the staging database as SYS, and execute the following statements in order to create the users with the necessary privileges (note again that you should use your own secret passwords):

```
create user cdc_stg_pub
identified by cdc_stg_pub
default tablespace users
temporary tablespace temp
quota unlimited on system
quota unlimited on users
quota unlimited on sysaux ;

grant create session, create table, create sequence,
select_catalog_role, execute_catalog_role, create database
link, dba to cdc_stg_pub ;

grant execute on dbms_aqadm to cdc_stg_pub ;

grant execute on dbms_capture_adm to cdc_stg_pub ;

grant execute on dbms_apply_adm to cdc_stg_pub ;

grant execute on dbms_propagation_adm to cdc_stg_pub ;

grant execute on dbms_streams_adm to cdc_stg_pub ;
```

```
begin

  dbms_rule_adm.grant_system_privilege (
    privilege => DBMS_RULE_ADM.CREATE_RULE_SET_OBJ,
    grantee => 'cdc_stg_pub',
    grant_option => FALSE);

  dbms_rule_adm.grant_system_privilege (
    privilege => DBMS_RULE_ADM.CREATE_RULE_OBJ,
    grantee => 'cdc_stg_pub',
    grant_option => FALSE);

end ;
/

create user cdc_stg_user
identified by cdc_stg_user
default tablespace users
temporary tablespace temp ;

grant connect, resource to cdc_stg_user ;
```

Make sure you use a default tablespace for your database users in order to prevent database objects from being created in the SYSAUX tablespace.

Create the database links

In order to setup distributed CDC you have to create a database link from the source publisher CDC_SOURCE_PUB to the staging publisher CDC_STG_PUB and vice versa. Once you do not need to make any more changes to the CDC setup you can remove the database link from the staging publisher to the source publisher (the other database link has to remain in place in order for distributed CDC to work).

Logon using SQL Plus to CDC_SOURCE_PUB on the source database and create a database link to CDC_STG_PUB (use the password you used to create the CDC_STG_PUB user). Note that because of the global_names=true setting the name of the database link **must** be the same as the name of the staging database.

```
create database link cdccbstg
connect to cdc_stg_pub
identified by cdc_stg_pub
using 'cdccbstg' ;
```

Verify whether the database link works using the following select statement:

```
select * from dual@cdccbstg ;
```

Do not continue unless this query executes successfully and returns an X. Some debugging tips:

- When a database is created, its global database name may be suffixed with a domain that is different from your default domain. If that is the case then you can use a different name for the database link, and use this other name going forward. To find out a database's global name (for the staging database), use as a user connected to the staging database:
  ```
  select * from global_name ;
  ```
  Alternatively, if you can change the global database name for the staging database without effecting any other applications that may run against the database, use the following statement to change your database's global database name (connect as SYS to the staging database):
  ```
  alter database rename global_name to CDCCBSTG.US.ORACLE.COM
  ;
  ```
- If you have to recreate your database link, use the following statement to drop the database link and recreate it:
  ```
  drop database link cdccbstg ;
  ```
- If you get ORA-02085: database link CDCCBSTG connects to CDCCBSTG.<domain> then you may have to drop the database link and recreate it using the <domain> suffix. Depending on the default domain on your source database you may have to include the <domain> suffix in the select statement also.

Next logon using SQL Plus to CDC_STG_PUB and create a database link to CDC_SOURCE_PUB (use the password you used to create the CDC_SOURCE_PUB user). Note that because of the global_names=true setting the name of the database link **must** be the same as the name of the source database.

```
create database link cdccbs
connect to cdc_source_pub
identified by cdc_source_pub
using 'cdccbs' ;
```

Again, make sure that the database link is valid by running the following statement:

```
select * from dual@cdccbs ;
```

Do not continue unless this statement successfully returns an X. The same debugging tips that were mentioned earlier in this section also apply to the creation of the database link from the staging database to the source database.

SET UP AND ENABLE CDC

First you have to create the source tables and define the supplemental logging settings on the source database. Once that is completed, you setup the entire CDC configuration by using PL/SQL calls on the staging database. The commands you

execute on the staging database use the database link between the staging and the source databases in order to setup the underlying Streams configuration on the source database.

Note that once you started using CDC you should continue to use CDC calls to make modifications to the setup. Directly using calls to the Oracle Streams administration packages to modify the underlying Streams setup is not supported for a CDC setup, and can lead to inconsistencies in the underlying definitions supporting CDC.

The statements below include a lot of data dictionary select statements that you can use to make sure that the underlying setup is correct and help you understand the definitions that get created. Of course running these select statements is not required to successfully setup CDC.

Create the source tables

Connect to user CDC_SOURCE at the source database. Issue the following commands[6]:

```
create table emp as select * from scott.emp ;

create table dept as select * from scott.dept ;

-- ensure empno is always logged for updates against emp
alter table emp add supplemental log group log_group_emp(empno)
always ;

-- ensure deptno is always logged for updates against dept
alter table dept add supplemental log group
log_group_dept(deptno) always ;
```

Set up CDC: create the change tables

Connect to the CDC publisher on the staging database, CDC_STG_PUB and issue the following command to create the hotlog change source on the staging database. This statement creates an associated, still disabled, Streams capture process, capture queue and queue table on the source database.

6 If you use Oracle Database 10*g* Release 2 as a source database then you do not need to run the statements to add the supplemental logs. By default Oracle Database 10*g* Release 2 will enable supplemental logging on a table's primary key or a unique key. If there is no key, then the supplemental logging will log **all** column values. To avoid getting all columns logged if you use Oracle Database 10*g* Release 2 as a source, issue the following two statements once you created the tables:

```
alter table emp add constraint emp_pk primary key (empno) ;
alter table dept add constraint dept_pk primary key (deptno)
;
```

```
begin
  dbms_cdc_publish.create_hotlog_change_source(
    change_source_name => 'emp_dept_src',
    description        => 'EMP and DEPT source',
    source_database    => 'cdccbs') ; -- database link name
end;
/
```

The `source_database` parameter has to have the database link name to the source database.

Note that even though you may have created the database link without a domain suffix, a domain suffix may have been added automatically. If that is the case, then the execution of the previous statement will fail with:

```
ERROR at line 1:
ORA-31485: invalid database link
ORA-06512: at "SYS.DBMS_SYS_ERROR", line 79
ORA-06512: at "SYS.DBMS_CDC_IPUBLISH", line 599
ORA-06512: at "SYS.DBMS_CDC_PUBLISH", line 114
ORA-06512: at line 2
```

In that case, use:

```
select db_link from user_db_links ;
```

to identify the full name of the database link, replace cdccbs in the previous statement with the full name (including the domain; the database link name is case insensitive) and re-issue the command. For example, the command that works for you may be:

```
begin
  dbms_cdc_publish.create_hotlog_change_source(
    change_source_name => 'emp_dept_src',
    description        => 'EMP and DEPT source',
    source_database    => 'cdccbs.us.oracle.com') ;
end;
/
```

Verify the change source definition that was created on the staging database:

```
select source_name, source_description, source_type
, source_database, capture_name, capture_queue_name
, capture_queue_table_name
from change_sources
where publisher = 'CDC_SOURCE_PUB'
and source_name = 'EMP_DEPT_SRC' ;
```

This query should return one record with the change source you just created.

Verify the change source definition on the source database by using the following cross database link data dictionary select on the staging database:

```
select cap.capture_name, q.name, qt.queue_table, cap.status
from dba_capture@cdccbs cap
, dba_queues@cdccbs q
, dba_queue_tables@cdccbs qt
where cap.queue_owner = 'CDC_SOURCE_PUB'
and q.owner = 'CDC_SOURCE_PUB'
and qt.owner = 'CDC_SOURCE_PUB'
and q.name = cap.queue_name
and qt.queue_table = q.queue_table
and cap.capture_name like '%EMP_DEPT%' ;
```

The result of the query shows one disabled capture process on the source database.

Next, as CDC_STG_PUB user, create a distributed hotlog change set on the staging database. This will create an associated, still disabled, streams apply process, an apply queue and apply queue table, as well as a Streams propagation definition from the capture stream at the source database.

```
begin
  dbms_cdc_publish.create_change_set(
    change_set_name    => 'emp_dept_set',
    description        => 'EMP and DEPT change set',
    change_source_name => 'emp_dept_src') ;
end ;
/
```

Verify the change set definition that was created on the staging database:

```
select set_name, set_description, change_source_name
, apply_name, queue_name, queue_table_name
from change_sets
where publisher = 'CDC_STG_PUB'
and set_name = 'EMP_DEPT_SET' ;
```

This query should return one record with the change set you just created.

Check the underlying Streams definition that was created as a result of the call to the CDC procedure:

```
select app.apply_name, q.name, app.status, qt.queue_table
from dba_apply app
, dba_queues q
, dba_queue_tables qt
where app.apply_user = 'CDC_STG_PUB'
and q.owner = 'CDC_STG_PUB'
and qt.owner = 'CDC_STG_PUB'
and q.name = app.queue_name
and qt.queue_table = q.queue_table
and app.apply_name like '%EMP_DEPT%' ;
```

The result of this query shows a disabled Streams apply process with its details.

Check the Streams propagation definition on the staging database:

```
select p.propagation_source_name, p.propagation_name
, p.staging_database, p.destination_queue, ps.change_set_name
from change_propagations p
, change_propagation_sets ps
where p.destination_queue_publisher = 'CDC_STG_PUB'
and ps.change_set_publisher = 'CDC_STG_PUB'
and ps.propagation_source_name = p.propagation_source_name
and ps.propagation_name = p.propagation_name
and ps.staging_database = p.staging_database
and p.propagation_source_name = 'EMP_DEPT_SRC' ;
```

The result shows one record with the propagation definition on the staging database that was created when you issued the CDC command[7].

Verify the Streams propagation definition on the source database by running the following cross database link statement on the staging database:

```
select propagation_name, source_queue_owner, source_queue_name
, destination_queue_owner, destination_queue_name
, destination_dblink
from dba_propagation@cdccbs
where destination_queue_owner = 'CDC_STG_PUB'
and propagation_name like '%EMP_DEPT%' ;
```

You should see one record with the details of the propagation on the source database. The result shows the propagation from a database queue owned by CDC_SOURCE_PUB to another queue owned by user CDC_STG_PUB on the staging database.

7 If you use Oracle Database 10*g* Release 2 as a source database, then this query will not return any rows. The Streams on the source database directly talk to the Streams on the staging database, without a propagation process on the staging database.

Still as CDC_STG_PUB user, create the change tables to capture the changes from the source tables. This creates the Streams apply rules on the staging database as well as the Streams capture rules on the source database.

```
begin
  dbms_cdc_publish.create_change_table(
    owner => 'cdc_stg_pub',
    change_table_name => 'emp_ct',
    change_set_name => 'emp_dept_set',
    source_schema => 'cdc_source',
    source_table => 'emp',
    column_type_list => 'empno number(4), ename varchar2(10),
job varchar2(9), mgr number(4), sal number(7,2), comm
number(7,2), deptno number(2)',
    capture_values => 'both',
    rs_id => 'y',
    row_id => 'n',
    user_id => 'n',
    timestamp => 'y',
    object_id => 'n',
    source_colmap => 'n',
    target_colmap => 'y',
    options_string => null) ;
end ;
/

grant select on emp_ct to cdc_stg_user ;
```

```
begin
  dbms_cdc_publish.create_change_table(
    owner => 'cdc_stg_pub',
    change_table_name => 'dept_ct',
    change_set_name => 'emp_dept_set',
    source_schema => 'cdc_source',
    source_table => 'dept',
    column_type_list => 'deptno number(2), dname varchar2(14),
loc varchar2(13)',
    capture_values => 'both',
    rs_id => 'y',
    row_id => 'n',
    user_id => 'n',
    timestamp => 'y',
    object_id => 'n',
    source_colmap => 'n',
    target_colmap => 'y',
    options_string => null) ;
end ;
/

grant select on dept_ct to cdc_stg_user ;
```

Check the change tables definitions on the staging database:

```
select change_table_name, change_set_name
, source_schema_name, source_table_name
from change_tables
where change_table_schema = 'CDC_STG_PUB'
and change_set_name = 'EMP_DEPT_SET'
order by change_table_name ;
```

This query should show you the two change tables you just created and which source tables' changes are going to be captured.

Verify the apply rules definition on the staging database:

```
select streams_name, streams_type, table_owner, table_name
, rule_type, source_database
from dba_streams_table_rules
where rule_owner = 'CDC_STG_PUB'
and table_owner = 'CDC_SOURCE'
order by table_name, rule_type, streams_type ;
```

The result of this query shows four Streams rules: DML and DDL apply rules for the two source tables.

Verify the capture and propagation rules definitions on the source database, by running the following cross database link statement on the staging database:

```
select streams_name, streams_type, table_owner
, table_name, rule_type, source_database
from dba_streams_table_rules@cdccbs
where rule_owner = 'CDC_SOURCE_PUB'
and table_owner = 'CDC_SOURCE'
order by table_name, rule_type, streams_type ;
```

For both source tables you should see a CAPTURE and PROPAGATION rule, for both DML and DDL (total of eight rules).

The setup of the underlying infrastructure is now complete. The capture, propagation and apply processes are still inactive.

Activate CDC

Activate the change set on the staging database by running the following command as CDC_STG_PUB on the staging database:

```
begin
  dbms_cdc_publish.alter_change_set(
    change_set_name => 'emp_dept_set',
    enable_capture  => 'Y') ;
end ;
/
```

Verify that the underlying Streams apply process was enabled:

```
select apply_name, status
from dba_apply
where apply_user = 'CDC_STG_PUB'
and apply_name like '%EMP_DEPT%' ;
```

The value for STATUS must show ENABLED. If it does not, then check the alert.log file and/or any trace files that were generated in the background dump directory for the database.

Activate the change source on the source database by issuing the following command as CDC_STG_PUB on the staging database:

```
begin
  dbms_cdc_publish.alter_hotlog_change_source(
    change_source_name => 'emp_dept_src',
    enable_source   => 'Y') ;
end ;
/
```

Verify the Streams capture definition on the source by running this cross database link statement from the staging database:

```
select capture_name, status
from dba_capture@cdccbs
where queue_owner = 'CDC_SOURCE_PUB'
and capture_name like '%EMP_DEPT%' ;
```

The outcome for STATUS must show ENABLED. If the capture process is not activated you should check the alert.log file for the source database.

CDC is now active. Changes applied to the source tables using regular DML[8] will be captured and end up in the change tables at the staging database.

Create a subscription

You will now create a subscription for the changes as the CDC_STG_USER user. Connect to the staging database as CDC_STG_USER, and issue:

```
begin
  dbms_cdc_subscribe.create_subscription(
    change_set_name   => 'emp_dept_set',
    description       => 'EMP and DEPT change subscription',
    subscription_name => 'emp_dept_sub1') ;
end ;
/

begin
  dbms_cdc_subscribe.subscribe(
    subscription_name => 'emp_dept_sub1',
    source_schema     => 'cdc_source',
    source_table      => 'emp',
    column_list       => 'empno, ename, job, mgr, sal, comm,
deptno',
    subscriber_view   => 'emp_chg_view') ;
end ;
/

begin
  dbms_cdc_subscribe.subscribe(
    subscription_name => 'emp_dept_sub1',
    source_schema     => 'cdc_source',
    source_table      => 'dept',
    column_list       => 'deptno, dname, loc',
    subscriber_view   => 'dept_chg_view') ;
end ;
/
```

Activate the subscription:

8 At this point direct path DML and implicit DML as part of partition maintenance operations are not supported. Note that these are not common for typical OLTP applications.

```
begin
  dbms_cdc_subscribe.activate_subscription(
    subscription_name => 'emp_dept_sub1') ;
end ;
/
```

Verify the CDC subscription definition on the staging database (still as CDC_STG_USER):

```
select s.subscription_name, s.set_name, s.description
, st.source_schema_name, st.source_table_name, st.view_name
, sc.column_name
from user_subscriptions s
, user_subscribed_tables st
, user_subscribed_columns sc
where s.subscription_name = 'EMP_DEPT_SUB1'
and st.handle = s.handle
and sc.handle = s.handle
and st.source_schema_name = sc.source_schema_name
and st.source_table_name = sc.source_table_name
order by st.source_schema_name, st.source_table_name
, sc.column_name ;
```

The result of this query shows ten records for the ten columns you subscribed to, across two subscriber views.

CDC IN ACTION

In order to see CDC in action, you will introduce changes to the source tables, verify the data in the change tables and extend and purge the subscription window.

Every time you extend the subscription window you will see more change data, assuming more changes have arrived in the change tables. The extend window functionality moves the end date/time of your subscription window forward implicitly by using the System Commit Number (SCN). The `PURGE_WINDOW` command on the other hand moves the start data/time forward (again, through the use of the SCN). The `EXTEND_WINDOW` and `PURGE_WINDOW` commands are independent, and can be called independently.

At any point in time you can purge the change tables. CDC keeps track of the subscriptions and knows whether or not change data is or will still be visible through a subscriber view. Data for which no subscribers exist anymore is removed from the change tables when you purge the change tables.

Perform DML against the source tables

Connect as CDC_SOURCE to the source database and introduce the following changes:

```
-- 10% raise for all salesmen:
update emp
set sal = 1.1 * sal
where job = 'SALESMAN' ;

commit ;

-- Hire John Doe in New York as analyst, reporting to KING
insert into emp values
(8000, 'DOE', 'ANALYST', 7839, trunc(sysdate), 4000, null, 10)
/

-- Close the operations department
delete from dept where dname = 'OPERATIONS' ;

commit ;

-- Give all clerks in Dallas a 5% commission
update emp
set comm = 0.05 * sal
where job = 'CLERK'
and deptno =
(select deptno from dept where loc = 'DALLAS') ;

commit ;
```

Read the captured changes from the change tables

Connect to CDC_STG_PUB on the staging database to verify the data in the change tables:

```
select operation$ operation
, to_char(timestamp$,'dd-mon-yyyy hh24:mi:ss') this_time
, empno, ename, sal, comm
from emp_ct
order by timestamp$ ;
```

The outcome should show something like[9] [10] [11]:

9 How quickly you will see the changes appear in the change table depends on the speed of the database servers you use, the network speed and latency, as well as the amount of memory you configured the database to use. In general, once up and running, all changes can be captured, propagated and applied within a few seconds after the transaction commits (depending on the size of the transaction; very large transactions modifying thousands or even millions or records would obviously take longer to capture entirely).

10 With Oracle Database 10*g* Release 2 as source database, unless you created the primary key on the EMP table as previously indicated, you would see all column values.

11 The `timestamp$` column is available because during the creation of the change tables the `timestamp` parameter was set to 'y'. If you don't set this parameter to 'y' the `timestamp$` column will not be available in the change table.

```
OP THIS_TIME                EMPNO ENAME             SAL       COMM
-- -------------------- ---------- ---------- ---------- ----------
UO 18-jan-2006 14:40:14       7499                  1600
UN 18-jan-2006 14:40:14       7499                  1760
UO 18-jan-2006 14:40:14       7521                  1250
UN 18-jan-2006 14:40:14       7521                  1375
UO 18-jan-2006 14:40:14       7654                  1250
UN 18-jan-2006 14:40:14       7654                  1375
UO 18-jan-2006 14:40:14       7844                  1500
UN 18-jan-2006 14:40:14       7844                  1650
I  18-jan-2006 14:40:32       8000 DOE              4000
UO 18-jan-2006 14:41:11       7369
UN 18-jan-2006 14:41:11       7369                              40
UO 18-jan-2006 14:41:11       7876
UN 18-jan-2006 14:41:11       7876                              55
```

Note that:

- Every update shows up twice, once as UO (Update Old) and once as UN (Update New). You can influence this behavior through the `capture_values` parameter when you create the change tables (allowed values OLD, NEW, BOTH).
- The `timestamp$` column reflects the date/time when the DML operation took place on the source, and is hence relative to the source database time.
- Thanks to the supplemental logging on EMPNO the value of EMPNO is always captured, even if you don't change it.

```
select operation$ operation
, to_char(timestamp$,'dd-mon-yyyy hh24:mi:ss') this_time
, deptno, dname, loc
from dept_ct
order by timestamp$ ;
```

The query result should show:

```
OP THIS_TIME                 DEPTNO DNAME          LOC
-- -------------------- ---------- -------------- -----------
D  18-jan-2006 14:40:49         40 OPERATIONS     BOSTON
```

Extend the subscription window

As a subscriber to change data, you will now extend your subscription window in order to see the change data through your subscription views. Connect to the staging database as CDC_STG_USER and execute the following statement:

```
begin
  dbms_cdc_subscribe.extend_window(
    subscription_name => 'emp_dept_sub1') ;
end ;
/
```

Verify the data visible through the change views:

```
select operation$ operation
, to_char(timestamp$,'dd-mon-yyyy hh24:mi:ss') this_time
, empno, ename, sal, comm
from emp_chg_view
order by timestamp$ ;
```

The query should show a result similar to the following[12]:

```
OP THIS_TIME                EMPNO ENAME             SAL       COMM
-- -------------------- ---------- ---------- ---------- ----------
UO 18-jan-2006 14:40:14       7499                 1600
UN 18-jan-2006 14:40:14       7499                 1760
UO 18-jan-2006 14:40:14       7521                 1250
UN 18-jan-2006 14:40:14       7521                 1375
UO 18-jan-2006 14:40:14       7654                 1250
UN 18-jan-2006 14:40:14       7654                 1375
UO 18-jan-2006 14:40:14       7844                 1500
UN 18-jan-2006 14:40:14       7844                 1650
I  18-jan-2006 14:40:32       8000 DOE             4000
```

Note that you do not see the latest transaction that was already captured in the EMP_CT change table. The reason for that is because CDC does not know whether the transaction is in the middle of being captured (or propagated), or has completed capturing and propagating. Only when a newer transaction for the change set arrives (a transaction with a higher System Commit Number, SCN) CDC will know that the previous transaction has completely been captured and propagated[13].

```
select operation$ operation
, to_char(timestamp$,'dd-mon-yyyy hh24:mi:ss') this_time
, deptno, dname, loc
from dept_chg_view
order by timestamp$ ;
```

This query should show:

12 With Oracle Database 10*g* Release 1 or Release 2 as source database you would see all change records.

13 This reasoning does not apply with Oracle Database 10*g* Release 1 or Release 2 as a source database. In that case there is a "ping" transaction if the system is idle for a few seconds.

```
OP THIS_TIME                 DEPTNO DNAME          LOC
-- -------------------- ---------- -------------- -----------
D  18-jan-2006 14:40:49          40 OPERATIONS     BOSTON
```

Because the change in the DEPT table was not introduced in the last transaction against the change set all change data against DEPT shows up in the subscriber view.

Purge the subscription window

In a production situation individual subscribers will consume the changes they are interested in. Once done they will purge the subscription window to indicate that the change data is not needed anymore.

For example in a near real-time data warehouse, every 5 minutes you could extend the subscription window, run a SQL statement to load the data from the subscriber view into target data warehouse tables and purge the subscription window.

Logon to the staging database as CDC_STG_USER, and run:

```
begin
  dbms_cdc_subscribe.purge_window(
    subscription_name => 'emp_dept_sub1') ;
end ;
/
```

Verify that the change views don't contain any data anymore:

```
select * from emp_chg_view ;
```

```
select * from dept_chg_view ;
```

Both queries should return no records.

Purge the change tables

There are 3 different levels of granularity at which you can purge the change tables:

1) Per table: use `DBMS_CDC_PUBLISH.PURGE_CHANGE_TABLE`.
2) Per change set: use `DBMS_CDC_PUBLISH.PURGE_CHANGE_SET`.
3) For the entire staging database: use `DBMS_CDC_PUBLISH.PURGE`[14].

You will purge the change set. Connect to the staging database as CDC_STG_PUB and issue the following statement:

```
begin
  dbms_cdc_publish.purge_change_set(
    change_set_name => 'emp_dept_set') ;
```

14 As part of the setup CDC will automatically schedule a call to `DBMS_CDC_PUBLISH.PURGE` and execute it once every 24 hours. This job is visible through the `DBA_JOBS` data dictionary view and can be modified using calls to `DBMS_JOB` or through the Enterprise Manager console.

```
end ;
/
```

Verify that the data previously visible through the change views has disappeared:

```
select operation$ operation
, to_char(timestamp$,'dd-mon-yyyy hh24:mi:ss') this_time
, empno, ename, sal, comm
from emp_ct
order by timestamp$ ;
```

The result of your query should look similar to the following[15]:

```
OP THIS_TIME                EMPNO ENAME             SAL       COMM
-- -------------------- ---------- ---------- ---------- ----------
UO 18-jan-2006 14:41:11       7369
UN 18-jan-2006 14:41:11       7369                              40
UO 18-jan-2006 14:41:11       7876
UN 18-jan-2006 14:41:11       7876                              55
```

Run:

```
select operation$ operation
, to_char(timestamp$,'dd-mon-yyyy hh24:mi:ss') this_time
, deptno, dname, loc
from dept_ct
order by timestamp$ ;
```

This query should not retrieve any rows.

Introduce more DML changes

Connect as CDC_SOURCE and introduce the following changes:

```
-- Open a marketing department in San Francisco
insert into dept
values (50, 'MARKETING', 'SAN FRANCISCO') ;

commit ;

-- Move John Doe under manager Clark
update emp set mgr = 7782 where empno = 8000 ;

commit ;

-- Lay off the clerks in Dallas
delete from emp where job = 'CLERK' and deptno =
(select deptno from dept where loc = 'DALLAS') ;

commit ;
```

15 If you use Oracle Database 10*g* Release 1 or Release 2 as a source database the change table will be empty. The reason is because you would have already consumed the changes through the subscriber views.

Extend the subscription window again

Connect as CDC_STG_USER to the staging database and execute the following statement:

```
begin
  dbms_cdc_subscribe.extend_window(
    subscription_name => 'emp_dept_sub1') ;
end ;
/
```

Verify the data visible through the change views:

```
select operation$ operation
, to_char(timestamp$,'dd-mon-yyyy hh24:mi:ss') this_time
, empno, ename, mgr, comm
from emp_chg_view
order by timestamp$ ;
```

The result should show something similar to[16]:

```
OP THIS_TIME                 EMPNO ENAME             MGR       COMM
-- -------------------- ---------- ---------- ---------- ----------
UO 18-jan-2006 15:40:48       7369
UN 18-jan-2006 15:40:48       7369                              40
UO 18-jan-2006 15:40:48       7876
UN 18-jan-2006 15:40:48       7876                              55
UO 18-jan-2006 16:49:26       8000                 7839
UN 18-jan-2006 16:49:26       8000                 7782
```

Note again that the employees who were laid off are not yet visible through the change view because the DML was performed in the last transaction against the tables in the change set.

```
select operation$ operation
, to_char(timestamp$,'dd-mon-yyyy hh24:mi:ss') this_time
, deptno, dname, loc
from dept_chg_view
order by timestamp$ ;
```

This query should show:

```
OP THIS_TIME                 DEPTNO DNAME          LOC
-- -------------------- ---------- -------------- -------------
I  18-jan-2006 16:49:26         50 MARKETING      SAN FRANCISCO
```

From here you would again consume the change data, purge the subscription window, and start all over again.

16 Again, the result would be different if you use Oracle Database 10*g* Release 1 or Release 2 as a source. You may actually see the last change as well, but not the ones that were consumed and purged before.

FINAL NOTES

CDC is built on top of the Oracle Streams infrastructure. In general, restrictions in the Streams infrastructure, such as limited support for certain data types, for example binary data types, apply to CDC as well. In order to tune the CDC processes, you tune the Oracle Streams configuration.

In Oracle Database 10*g* Release 2 CDC does not support direct path data load operations against the data source nor does it support implicit DML through partition maintenance operations. In a traditional OLTP environment these types of operations are rare.

CONCLUSION

Oracle's Change Data Capture is an extremely powerful feature that can capture changes against database tables without making any changes to the applications that work with the tables. Asynchronous CDC even minimizes the performance impact on the transactions against the tables. CDC is a key enabling feature for near real-time data warehousing, but can be applied to any environment that requires capturing incremental changes out of an Oracle database.

With Oracle Database 10*g* Release 2 you can set up an asynchronous distributed change data capture environment using Oracle Database 9*i* Release 2 or higher as a source. Many applications today still run on Oracle Database 9*i* Release 2 or on Oracle Database 10*g* Release 1. Oracle Database 10*g* Release 2 enables you to easily implement incremental change data capture on top of these applications today.

APPENDIX A CLEANUP THE ENVIRONMENT

To cleanup the environment you go through the reverse steps you went through to setup the environment. This appendix shows you how to get back to your original environment. The data dictionary select statements are there to verify whether the cleanup is correct, and to provide a better understanding of the underlying technology. These are not required to be run if you just want to remove the CDC setup.

Drop the subscription

Drop the subscription by connecting to CDC_STG_USER and issuing the following statement:

```
begin
  dbms_cdc_subscribe.drop_subscription(
    subscription_name => 'emp_dept_sub1') ;
end ;
/
```

Verify that the subscription was dropped:

```
select s.subscription_name, s.set_name, s.description
, st.source_schema_name, st.source_table_name, st.view_name
, sc.column_name
from user_subscriptions s
, user_subscribed_tables st
, user_subscribed_columns sc
where s.subscription_name = 'EMP_DEPT_SUB1'
and st.handle = s.handle
and sc.handle = s.handle
and st.source_schema_name = sc.source_schema_name
and st.source_table_name = sc.source_table_name
order by st.source_schema_name, st.source_table_name
, sc.column_name ;
```

This query should not return any rows. The change views have also been dropped.

Disable CDC

Switch of CDC by disabling the change set and disabling the change source. Connect to the staging database as CDC_STG_PUB and run:

```
begin
  dbms_cdc_publish.alter_change_set(
    change_set_name => 'emp_dept_set',
    enable_capture  => 'N') ;
end ;
/
```

Verify the Streams apply process definition to make sure it was switched of as a result of running the command:

```
select apply_name, status
from dba_apply
where apply_user = 'CDC_STG_PUB'
and apply_name like '%EMP_DEPT%' ;
```

This query should return status DISABLED for the apply process.

Disable the change source by issuing the following statement:

```
begin
  dbms_cdc_publish.alter_hotlog_change_source(
    change_source_name => 'emp_dept_src',
    enable_source   => 'N') ;
end ;
/
```

Verify the Streams capture process definition on the source database through the following cross database link query:

```
select capture_name, status
from dba_capture@cdccbs
where queue_owner = 'CDC_SOURCE_PUB'
and capture_name like '%EMP_DEPT%' ;
```

The capture process should show status DISABLED.

Drop the CDC database objects

First you will drop the change tables on the staging database. Connect to CDC_STG_PUB on the staging database and issue the following commands:

```
begin
  dbms_cdc_publish.drop_change_table(
    owner => 'cdc_stg_pub',
    change_table_name => 'dept_ct',
    force_flag => 'y') ;
end ;
/
```

```
begin
  dbms_cdc_publish.drop_change_table(
    owner => 'cdc_stg_pub',
    change_table_name => 'emp_ct',
    force_flag => 'y') ;
end ;
/
```

Verify that both change tables have been dropped correctly:

```
select change_table_name, change_set_name, source_schema_name
, source_table_name
from change_tables
where change_table_schema = 'CDC_STG_PUB'
and change_set_name = 'EMP_DEPT_SET'
order by change_table_name ;
```

This query should not return any rows.

Verify that the apply process on the staging database has been dropped:

```
select streams_name, streams_type, table_owner, table_name
, rule_type, source_database
from dba_streams_table_rules
where rule_owner = 'CDC_STG_PUB'
and table_owner = 'CDC_SOURCE'
order by table_name, rule_type, streams_type ;
```

This query should not return any rows.

Verify that the capture rules on the source database have been dropped by running this cross database link query from the staging database:

```
select streams_name, streams_type, table_owner, table_name
, rule_type, source_database
from dba_streams_table_rules@cdccbs
where rule_owner = 'CDC_SOURCE_PUB'
and table_owner = 'CDC_SOURCE'
order by table_name, rule_type, streams_type ;
```

This query should not return any rows.

Drop the CDC change set on the staging database, connected as CDC_STG_PUB:

```
begin
  dbms_cdc_publish.drop_change_set(
    change_set_name => 'emp_dept_set') ;
end ;
/
```

Verify that the change set was successfully removed:

```
select set_name, set_description, change_source_name
, apply_name, queue_name, queue_table_name
from change_sets
where publisher = 'CDC_STG_PUB'
and set_name = 'EMP_DEPT_SET' ;
```

This query should not return any rows.

Verify that all Streams apply definitions on the staging database were removed as a result of dropping the change set:

```
select app.apply_name, q.name, app.status, qt.queue_table
from dba_apply app, dba_queues q, dba_queue_tables qt
where app.apply_user = 'CDC_STG_PUB'
and q.owner = 'CDC_STG_PUB'
and qt.owner = 'CDC_STG_PUB'
and q.name = app.queue_name
and qt.queue_table = q.queue_table
and app.apply_name like '%EMP_DEPT%' ;
```

This query should not return any rows.

Make sure that all related Streams propagation definitions were removed:

```
select p.propagation_source_name, p.propagation_name
, p.staging_database, p.destination_queue, ps.change_set_name
from change_propagations p
, change_propagation_sets ps
where p.destination_queue_publisher = 'CDC_STG_PUB'
and ps.change_set_publisher = 'CDC_STG_PUB'
and ps.propagation_source_name = p.propagation_source_name
and ps.propagation_name = p.propagation_name
and ps.staging_database = p.staging_database
and p.propagation_source_name = 'EMP_DEPT_SRC' ;
```

This query should return no rows.

Verify that the propagation definition on the source database was removed also:

```
select propagation_name, source_accept v_source_desc prompt
'Provide a change source description for change source
&v_source_name: 'queue_owner, source_queue_name
, destination_queue_owner, destination_queue_name
, destination_dblink
from dba_propagation@cdccbs
where destination_queue_owner = 'CDC_STG_PUB'
and propagation_name like '%EMP_DEPT%' ;
```

This query should return no rows.

Finally, drop the hotlog change source on the source database, by executing the following statement on the staging database, connected as CDC_STG_PUB:

```
begin
  dbms_cdc_publish.drop_change_source(
    change_source_name => 'emp_dept_src') ;
end ;
/
```

Verify that the change source definition on the staging database has been removed:

```
select source_name, source_description, source_type
, source_database, capture_name, capture_queue_name
, capture_queue_table_name
from change_sources
where publisher = 'CDC_SOURCE_PUB'
and source_name = 'EMP_DEPT_SRC' ;
```

The query should not return any rows.

Also verify that the related Streams capture process and the queues on the source database have been removed through this cross database link query:

```
select cap.capture_name, q.name, qt.queue_table, cap.status
from dba_capture@cdccbs cap
, dba_queues@cdccbs q
, dba_queue_tables@cdccbs qt
where cap.queue_owner = 'CDC_SOURCE_PUB'
and q.owner = 'CDC_SOURCE_PUB'
and qt.owner = 'CDC_SOURCE_PUB'
and q.name = cap.queue_name
and qt.queue_table = q.queue_table
and cap.capture_name like '%EMP_DEPT%' ;
```

This query should also return no rows.

Drop the source tables

Connect to CDC_SOURCE on the source database, and drop the source tables:

```
drop table dept ;

drop table emp ;
```

Drop the database links

Connect to CDC_SOURCE_PUB and drop the database link to the staging publisher:

```
drop database link cdccbstg ;
```

Connect to CDC_STG_PUB and drop the database link to the source publisher:

```
drop database link cdccbs ;
```

Drop the users

Connect to SYSTEM or SYS on the source database and drop the CDC_SOURCE and CDC_SOURCE_PUB users:

```
drop user cdc_source ;

drop user cdc_source_pub cascade ;
```
[17]

Connect to SYSTEM or SYS on the staging database and drop the CDC_STG_USER and CDC_STG_PUB users:

```
drop user cdc_stg_user ;

drop user cdc_stg_pub cascade ;
```
[18]

17 The cleanup process leaves some Streams rules around, that are dropped as well if you use the drop user cascade call. Alternatively you could use the Streams APIs to drop the rules followed by a drop user without the cascade option.

18 The cleanup process may leave some Streams rules around, that are dropped as well if you use the drop user cascade call. Alternatively you could use the Streams APIs to drop the rules followed by a drop user without the cascade option.

APPENDIX B KNOWN ISSUES AND WORKAROUNDS

This appendix lists known issues that you may run into, depending on your environment and setup.

Oracle Streams propagation from 9.2.0.6 or 9.2.0.7

Oracle Streams propagation may not work if you use Oracle Database 9*i* version 9.2.0.6 or 9.2.0.7 as a source database (this problem is platform dependent). The issue is documented in bug 4285404. The problem surfaces in the alert.log of the source database as `ORA-25292 Cannot add buffer on the specified queue`. Backports are available for several platforms or can be requested through Oracle Support at http://metalink.oracle.com.

High System Commit Numbers (SCNs)

In an existing environment with many transactions on the source system you may run into bug 4649767 because of large system commit numbers. A fix for this problem will be available in Oracle Database 10.2.0.2. You can request a backport for the problem on 10.2.0.1 through Oracle Support at http://metalink.oracle.com.

Queue tables in SYSTEM and SYSAUX tablespace

Asynchronous distributed hotlog relies on underlying Oracle Streams functionality. Queue tables are created as a memory overflow for the Oracle Streams processes. With 10.2.0.1 the queue tables always end up in the SYSTEM tablespace on Oracle Database 9*i* and in the SYSAUX tablespace on an Oracle Database 10*g*. Bug 5024710 describes this problem and will be fixed in Oracle Database 10.2.0.3. You can request a backport for the problem on 10.2.0.1 or 10.2.0.2 through Oracle Support at http://metalink.oracle.com.

Incorrect username$ value

Environments that use different users within a single session to apply changes to a table may see incorrect values for the username$ column in the CDC change table (and subscription view). This problem is bug 5437445 which is fixed in Oracle Database 10.2.0.4 . You can request a backport for the problem on 10.2.0.1, 10.2.0.2 or 10.2.0.3 through Oracle Support at http://metalink.oracle.com.

Update column value to null

An update of a column value from any value (not null) to a null value is not correctly reflected in CDC. The old value still shows the original value. This problem is bug 5533982 which is fixed in Oracle Database 10.2.0.4. You can request a backport for the problem on 10.2.0.1, 10.2.0.2 or 10.2.0.3 through Oracle Support at http://metalink.oracle.com.

Asynchronous Change Data Capture Cookbook
April 2007
Author: Mark Van de Wiel
Contributing Authors:

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com