

ORACLE®

Heterogeneous Replication with Oracle Streams, including replication from Oracle to DB2 and DB2 to Oracle

Thomas Niewel
Principal Sales Consultant
Business Unit Database
Oracle Germany

Datatransfer with the Oracle Transparent Gateways

- Move of data / Replication from Oracle to DB2
 - Trigger
 - SQL*PLUS Copy Command
 - Streams

Datatransfer with the Oracle Transparent Gateways

- Move of data / Replication from Oracle to DB2
 - Triggers

```
CREATE OR REPLACE TRIGGER EMP_TRIGGER
AFTER UPDATE OF ENAME ON SCOTT.ORA_EMP
FOR EACH ROW
BEGIN
UPDATE SCOTT.DB2_EMP@tg4db2
SET ENAME = :NEW.ENAME
WHERE EMPNO = :NEW.EMPNO;
END;
```

Datatransfer with the Oracle Transparent Gateways

- Move of data / Replication from Oracle to DB2
- SQL*PLUS Copy Command

```
INSERT INTO table@DB2 SELECT * FROM oracle_table;
```

ORA-2025: All tables in the SQL statement must be at the remote database.

```
COPY FROM tniewel/tniewel@ORA9i -  
INSERT tniewel.employees@TO_DB2 -
```

```
USING SELECT EMPNO, -  
ENAME , JOB, MGR, HIREDATE, SAL ,  
COMM, DEPTNO, -  
'2002-10-20-12.00.00.000000' LAST_MODIFIED  
from emp
```

Datatransfer with the Oracle Transparent Gateways

- Move of data / Replication from DB2 to Oracle
 - Create Table
 - Insert
 - Copy
 - Snapshot
 - Streams

Datatransfer with the Oracle Transparent Gateways

Move of data / Replication from DB2 to Oracle

- CREATE TABLE EMP AS SELECT * FROM SCOTT.EMP@gateway;
- INSERT INTO EMP SELECT * FROM SCOTT.EMP@gateway;
- COPY FROM SCOTT/TIGER@gateway INSERT EMP USING SELECT * FROM SCOTT.EMP@gateway;

Datatransfer with the Oracle Transparent Gateways

- Move of data / Replication from DB2 to Oracle

- Snapshots

```
CREATE SNAPSHOT empdb2
PCTFREE 5 PCTUSED 60
TABLESPACE users
STORAGE (INITIAL 50K NEXT 50K)
REFRESH COMPLETE NEXT SYSDATE + 1
WITH ROWID
AS
SELECT * FROM SCOTT.EMP@gateway
WHERE deptno=20;
```

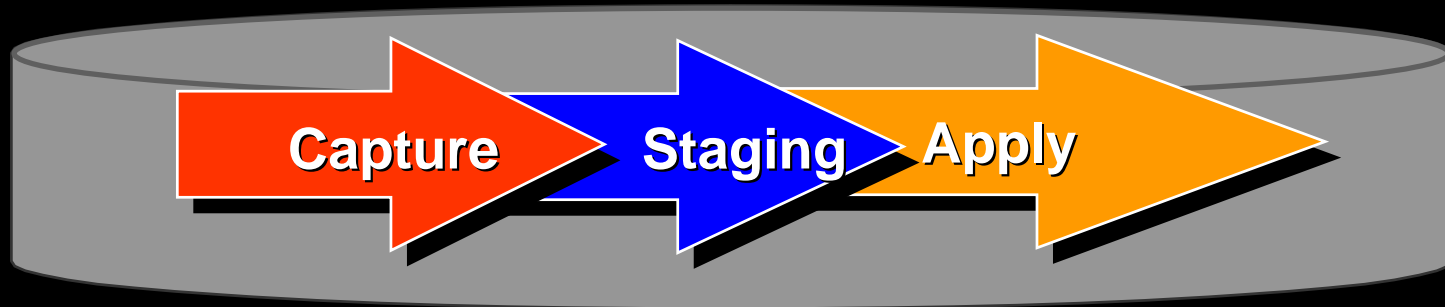
Oracle Streams

Heterogeneous Replication

Oracle Streams

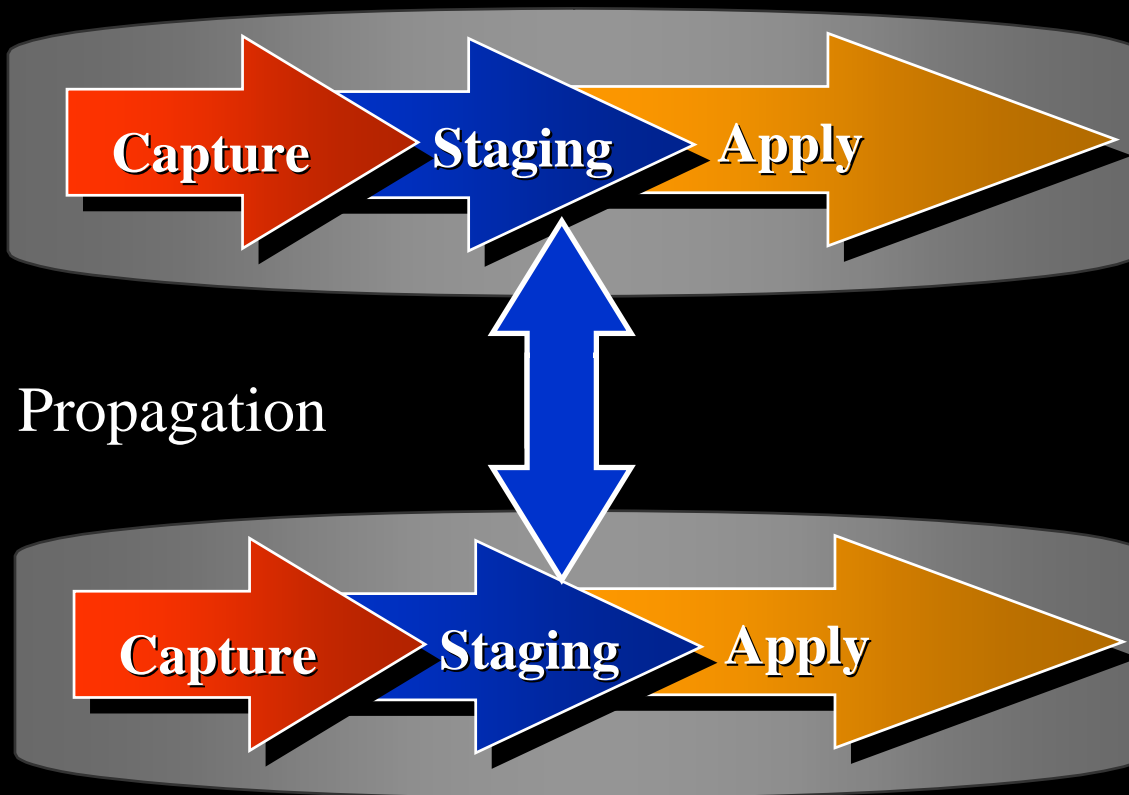
- **Simple solution for information sharing**
- **Provides**
 - **unique flexible replication**
 - **message queuing**
 - **data warehouse loading**
 - **database migration**
 - **application upgrade**
 - **event management and notification**

Architecture Oracle Streams



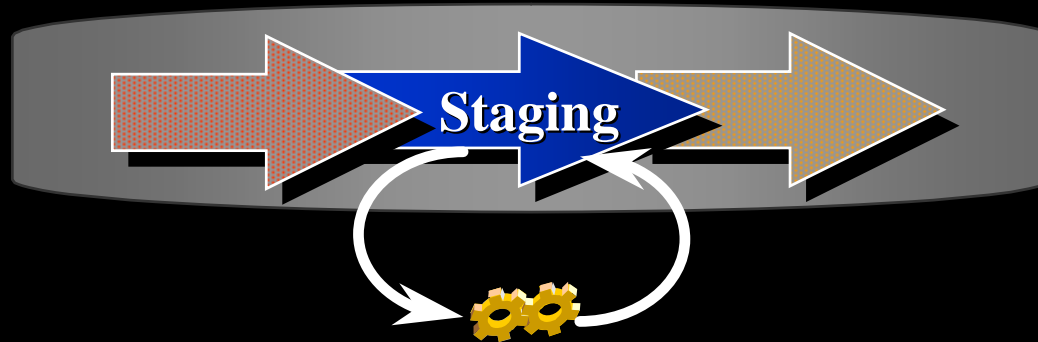
- **Basis Elements**
 - Capture
 - Staging
 - Apply

Multi-Database Streams



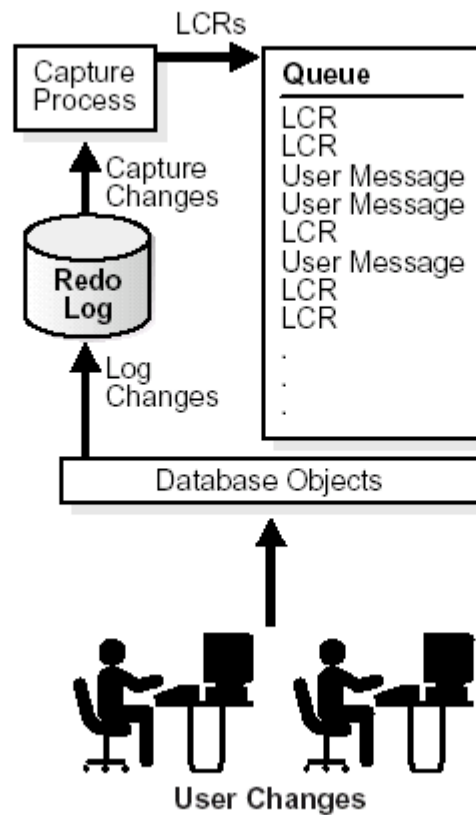
- A stream can contain elements from multiple databases
- Events flow between staging areas

Transformations



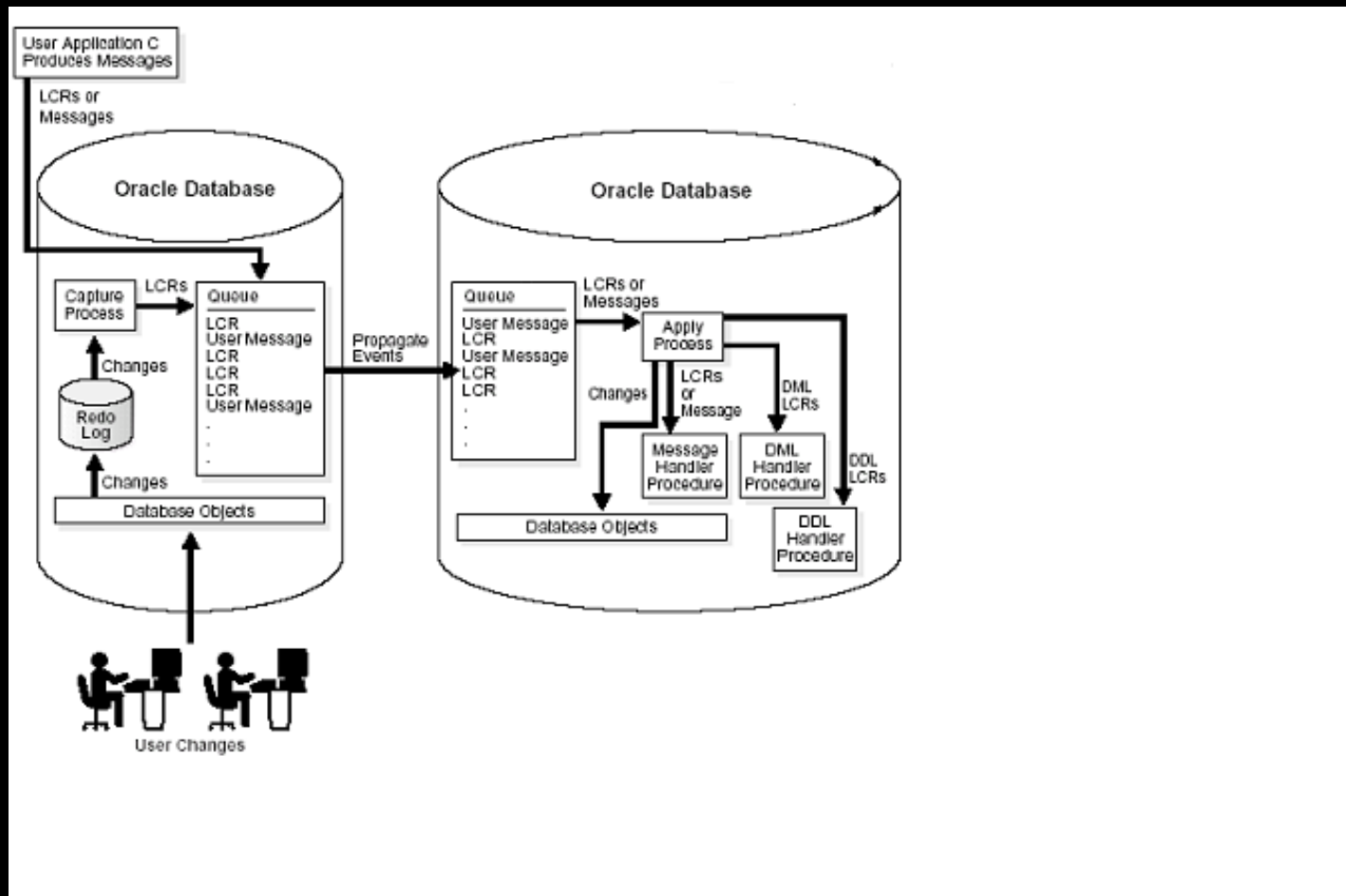
- **Transformations can be performed**
 - as events enter the staging area
 - as events leave the staging area
 - as events are propagated between staging areas
- **Transformation examples**
 - change format, data type, column name, table name

The Capture Process

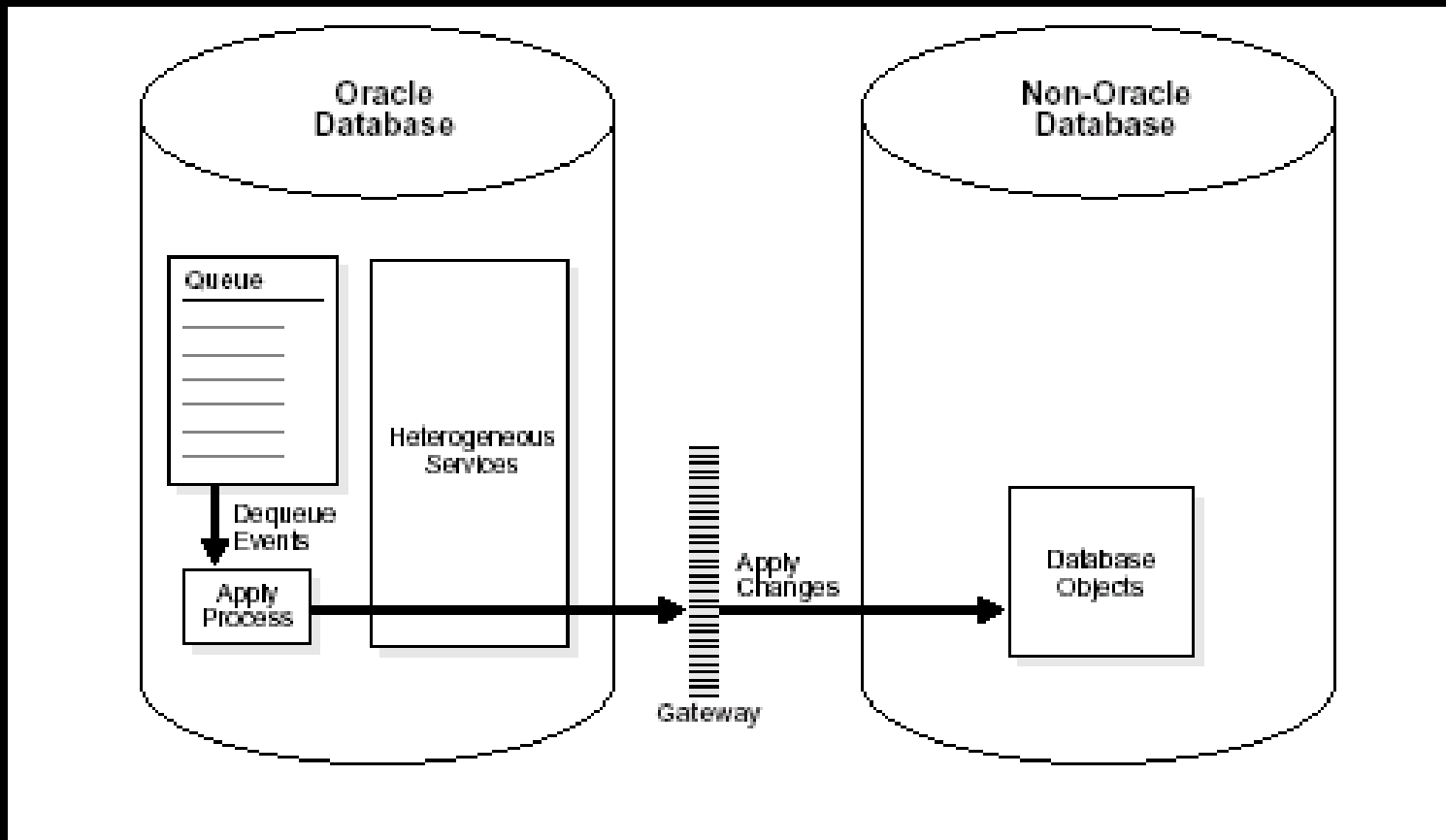


Architecture Oracle Streams

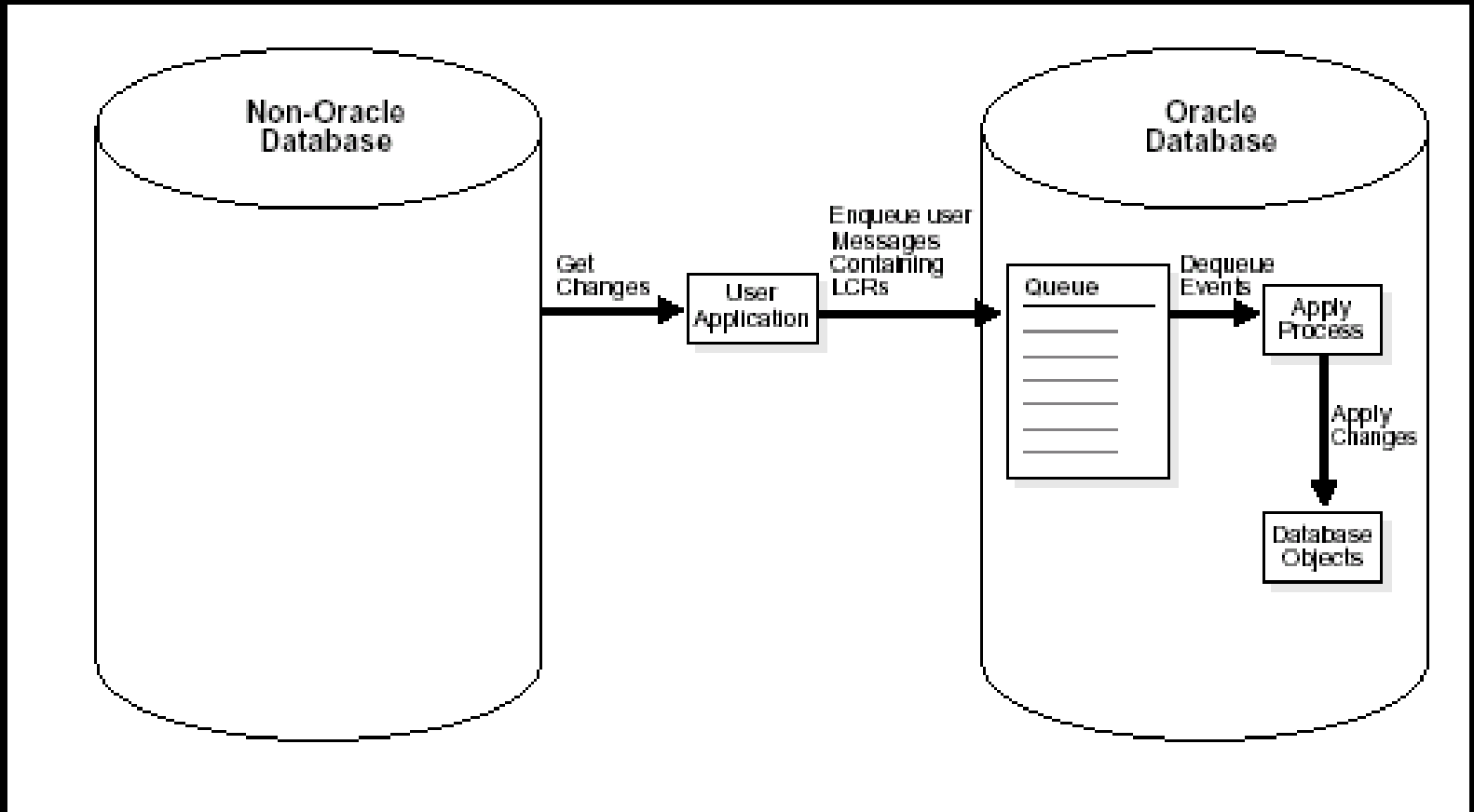
- Streams Flow Capture / Stage / Propagation / Apply (Non-Heterogeneous)



Oracle -> non Oracle Replikation



Non Oracle - > Oracle Replikation



Oracle Enterprise Manager

The screenshot displays the Oracle Enterprise Manager console interface. On the left, a tree view under 'Netzwerk' shows a hierarchy of database components. The 'Streams' folder is expanded, showing 'Administration', 'Erfassen', 'Weiterreichen', and 'Anwenden'. Under 'Anwenden', the 'TNIEWEL_APPLY1' queue is selected and highlighted in blue. The right pane shows the configuration for this queue. The 'Objekte' tab is active, displaying the following details:

Attribut	Wert
Name:	TNIEWEL_APPLY1
Queue:	TNIEWEL_QUEUE
Queue-Schema:	STRMADMIN
Status:	ENABLED

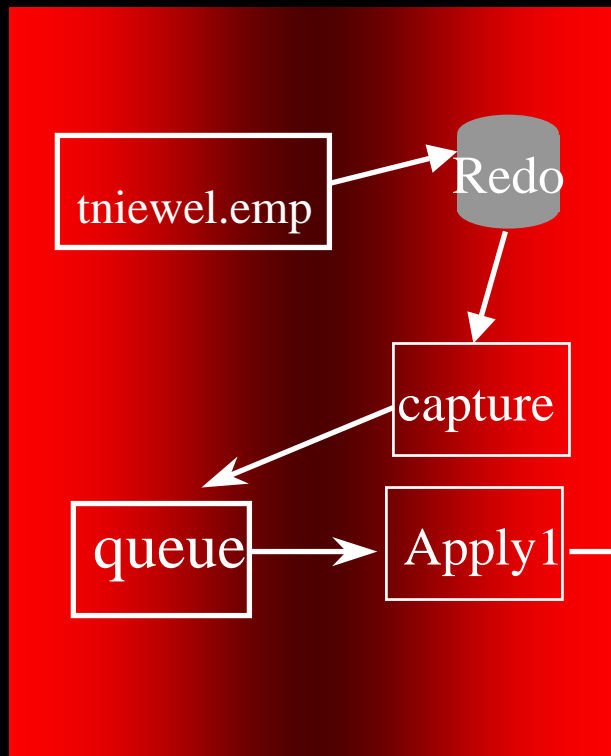
At the bottom right of the configuration pane, there is a 'Stopp' button. At the bottom of the console, there are buttons for 'Anwenden', 'Wiederherstellen', 'SQL zeigen', and 'Hilfe'.

Examples

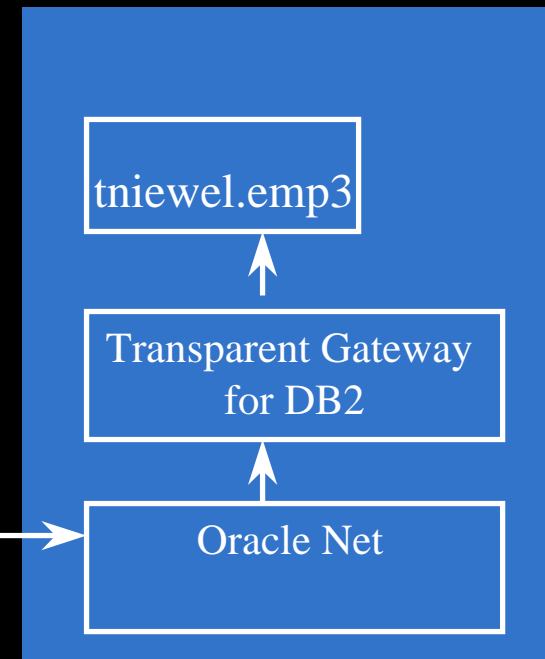
Examples

Replication Oracle - > DB2

Oracle - Unix



DB2 z/OS



DBLINK

Step1: Create a Queue

```
connect strmadmin/strmadmin
```

```
BEGIN
```

```
DBMS_STREAMS_ADM.SET_UP_QUEUE(
```

```
queue_table => 'tniewel_queue_table',
```

```
queue_name => 'tniewel_queue');
```

```
END;
```

Step 2 : Define the Capture Process

```
/* create a capture process for tniewel.emp */  
BEGIN  
DBMS_STREAMS_ADM.ADD_TABLE_RULES(  
table_name => 'tniewel.emp',  
streams_type => 'capture',  
streams_name => 'tniewel_capture',  
queue_name => 'tniewel_queue',  
include_dml => true,  
include_ddl => true,  
include_tagged_lcr => false);  
END;
```

Step 3: Define a Conversion Function

```
CREATE OR REPLACE FUNCTION strmadmin.convert_table_name
    (p_in_data in SYS.AnyData)
RETURN SYS.AnyData IS
    out_data SYS.LCR$_ROW_RECORD;
    tc pls_integer;
BEGIN
    - Typecast AnyData to LCR$_ROW_RECORD
    tc := p_in_data.GetObject(out_data);
    IF out_data.get_object_name() = 'EMP'
    THEN
        -- Transform in_data to out_data
        out_data.set_object_name('EMP3');
    END IF;
RETURN SYS.AnyData.ConvertObject(out_data);
END;
```

Step 4: Define Rules and a Rule Set

```
DECLARE action_ctx_dml SYS.RE$NV_LIST; action_ctx_ddl SYS.RE$NV_LIST;
ac_name VARCHAR2(30) := 'STREAMS$_TRANSFORM_FUNCTION';
BEGIN
action_ctx_dml := SYS.RE$NV_LIST(SYS.RE$NV_ARRAY());
action_ctx_dml.ADD_PAIR(ac_name,
SYS.ANYDATA.CONVERTVARCHAR2('strmadmin.convert_table_name'));

DBMS_RULE_ADM.CREATE_RULE_SET(
rule_set_name => 'strmadmin.apply_ruleset',
evaluation_context => 'SYS.STREAMS$_EVALUATION_CONTEXT');

DBMS_RULE_ADM.CREATE_RULE(
rule_name => 'strmadmin.apply_rule',
condition => ' :dml.get_object_owner() = ''TNIIEWEL'' AND ' ||
' :dml.get_object_name() = ''EMP''',
action_context => action_ctx_dml);

DBMS_RULE_ADM.ADD_RULE(
rule_name => 'strmadmin.apply_rule',
rule_set_name => 'strmadmin.apply_ruleset');
end;
```

Step 5: Define an Apply Process

```
BEGIN
DBMS_APPLY_ADM.CREATE_APPLY(
queue_name => 'strmadmin.tniewel_queue',
apply_name => 'tniewel_apply1',
rule_set_name => 'strmadmin.apply_ruleset',
message_handler => null,
apply_user => null,
apply_database_link => 'DEG1',
apply_captured => true);
END;
```

Set the Instantiation SCN (optional)

```
DECLARE
iscn NUMBER; -- Variable to hold instantiation SCN value
BEGIN
iscn := DBMS_FLASHBACK.GET_SYSTEM_CHANGE_NUMBER();
DBMS_APPLY_ADM.SET_TABLE_INSTANTIATION_SCN(
source_object_name => 'tniewel.emp',
source_database_name => 'ORC92',
instantiation_scn => iscn,
apply_database_link => 'DEG1.DE.ORACLE.COM');
END;
```

Step 6: Start the Capture and Apply Process

Begin

```
sys.DBMS_APPLY_ADM.START_APPLY(  
  apply_name => 'tniewel_apply1');  
END;
```

BEGIN

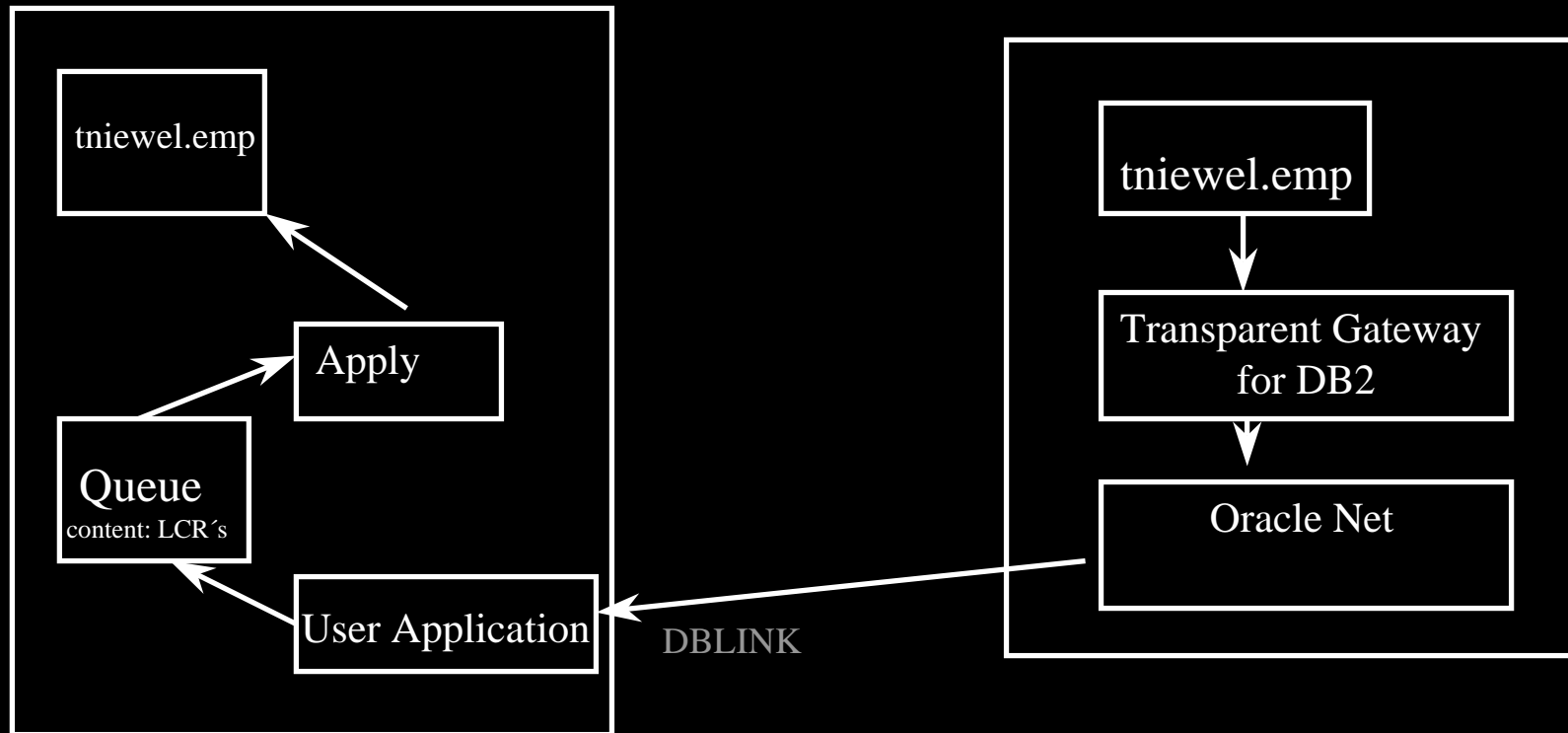
```
  DBMS_CAPTURE_ADM.START_CAPTURE(  
    capture_name => 'TNIIEWEL_CAPTURE');  
END;
```

Example 2

Replication DB2 -> Oracle

Oracle Unix/Windows

DB2 z/OS

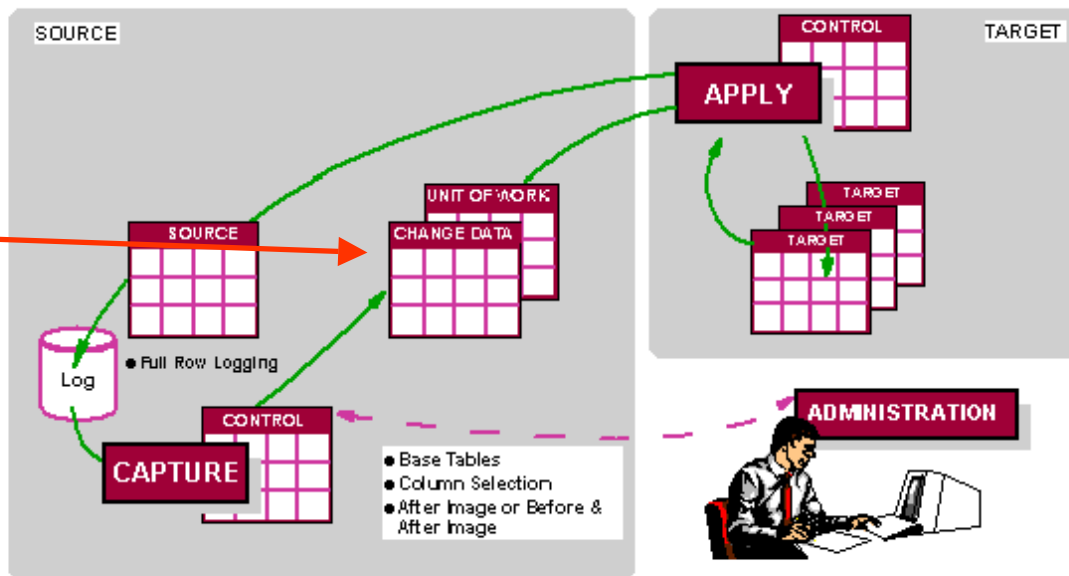


Replication from DB2 to Oracle

Architecture Datapropagator Relational

IBM Relational Replication

- Captures base table changes from log
- Capture runs locally to the source
- Maintains transaction consistency
- Automatically maintains staging tables

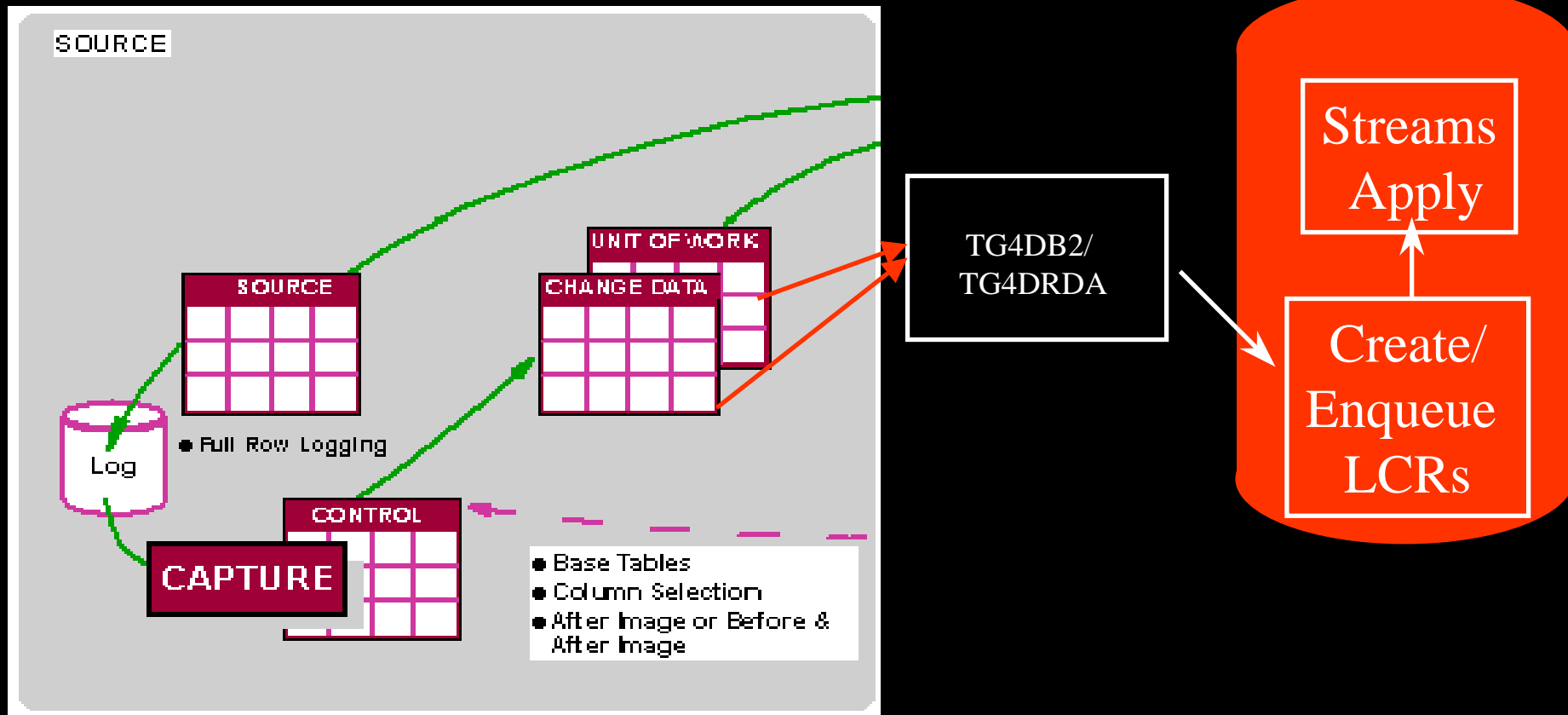


CD Table
UOW Table

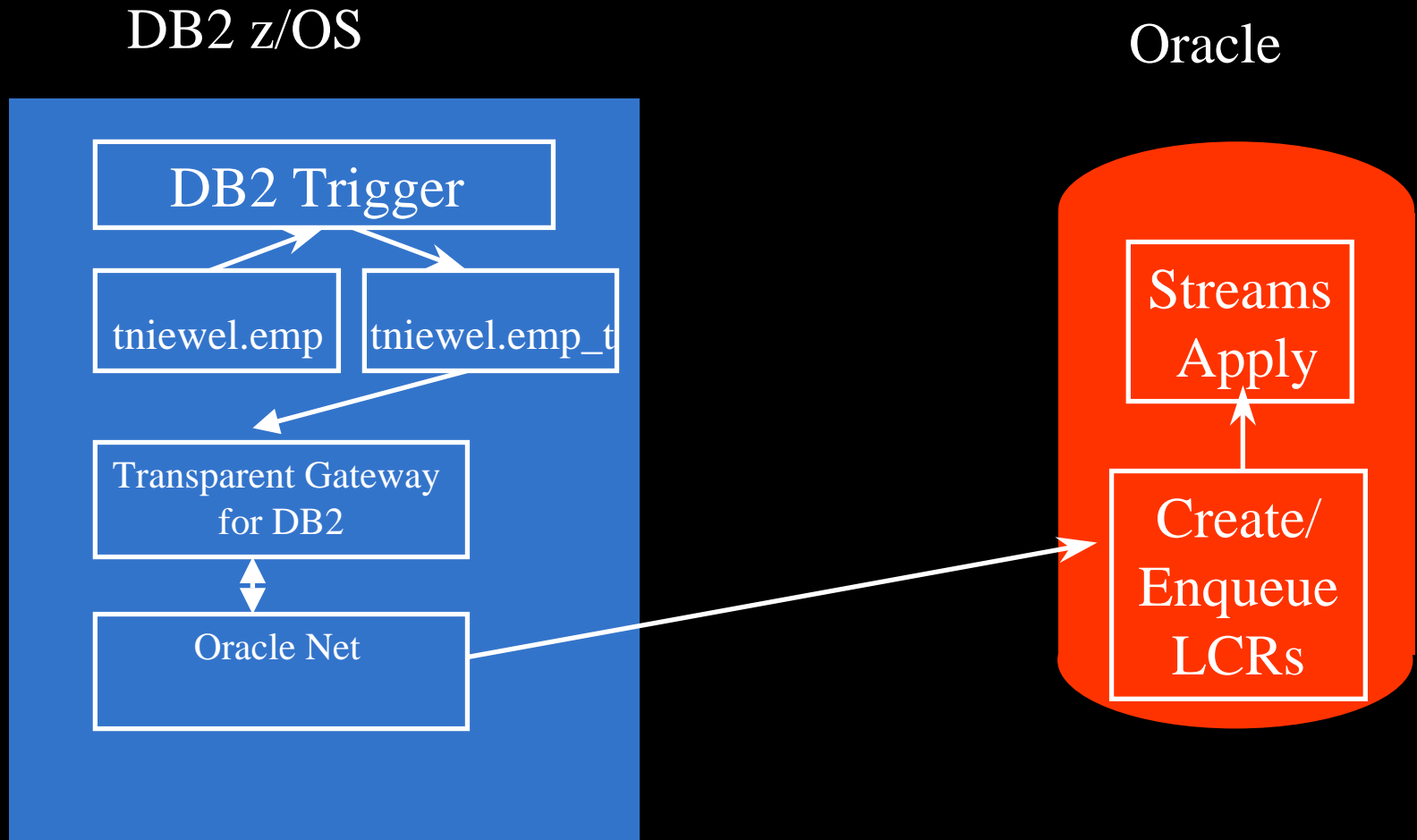
Content DPROP CD/UOW Tables

- **IBMSNAP_UOWID**
Foreign Key of the UOW Table
- **IBMSNAP_INTENTSEQ**
Global Sequence Number.
- **IBMSNAP_OPERATION**
DML Operation 'I', 'U' or 'D'
- **AFTER-IMAGE**
- **BEFORE-IMAGE**

Architecture IBM Dprop Relational with Oracle Streams



DB2 Trigger



Apply Processes

```
BEGIN  
  
DBMS_APPLY_ADM.CREATE_APPLY(  
queue_name => 'tniewel_queue1',  
apply_name => 'apply3',  
apply_captured => false);  
  
END;  
  
EXEC DBMS_APPLY_ADM.START_APPLY('APPLY3');
```

Procedure to create „Row LCR’s“ (1)

```
CREATE OR REPLACE PROCEDURE construct_row_lcr(  
source_dbname VARCHAR2,  
cmd_type VARCHAR2,  
obj_owner VARCHAR2,  
obj_name VARCHAR2,  
old_vals SYS.LCR$_ROW_LIST,  
new_vals SYS.LCR$_ROW_LIST) AS  
  
eopt DBMS_AQ.ENQUEUE_OPTIONS_T;  
mprop DBMS_AQ.MESSAGE_PROPERTIES_T;  
enq_msgid RAW(16);  
row_lcr SYS.LCR$_ROW_RECORD;
```

Procedure to create „Row LCR’s“ (2)

```
BEGIN
  -- add subscriber
  mprop.SENDER_ID := SYS.AQ$_AGENT('strmadmin', NULL, NULL);
  row_lcr := SYS.LCR$_ROW_RECORD.CONSTRUCT(
    source_database_name => source_dbname,
    command_type => cmd_type, object_owner => obj_owner,
    object_name => obj_name, old_values => old_vals,
    new_values => new_vals);
  -- Enqueue the created row LCR
  DBMS_AQ.ENQUEUE(
    queue_name => 'tniewel_queue1',
    enqueue_options => eopt,
    message_properties => mprop,
    payload => SYS.AnyData.ConvertObject(row_lcr),
    msgid => enq_msgid);
END construct_row_lcr;
```

Create Row LCR's (1)

Declare

```
empno_new number(10); ename_new varchar2(10);  
newunit1 SYS.LCR$_ROW_UNIT; newunit2 SYS.LCR$_ROW_UNIT;  
.....  
newvals SYS.LCR$_ROW_LIST;  
oldunit1 SYS.LCR$_ROW_UNIT; oldunit2 SYS.LCR$_ROW_UNIT;  
.....  
oldvals SYS.LCR$_ROW_LIST;
```

BEGIN

```
newunit1 := SYS.LCR$_ROW_UNIT(  
    'EMPNO', SYS.AnyData.ConvertNumber(empno_new), DBMS_LCR.NOT_A_LOB, NULL, NULL);  
newunit2 := SYS.LCR$_ROW_UNIT(  
    'ENAME', SYS.AnyData.ConvertVarchar2('Mueller'),DBMS_LCR.NOT_A_LOB,NULL, NULL);  
newvals := SYS.LCR$_ROW_LIST(newunit1,newunit2,newunit3,newunit4,newunit5,  
    newunit6,newunit7);
```

Create Row LCR's (2)

```
oldunit1 := SYS.LCR$_ROW_UNIT('EMPNO',
    SYS.AnyData.ConvertNumber(empno_old), DBMS_LCR.NOT_A_LOB, NULL, NULL);
/*-----*/
oldunit2 := SYS.LCR$_ROW_UNIT('ENAME',
    SYS.AnyData.ConvertVarchar2(ename_old), DBMS_LCR.NOT_A_LOB, NULL, NULL);
oldvals :=
SYS.LCR$_ROW_LIST(oldunit1,oldunit2,oldunit3,oldunit4,oldunit5,oldunit6,oldunit7);

/* Construct row lcr */

    construct_row_lcr(
        source_dbname => 'ORC92',
        cmd_type => 'UPDATE',
        obj_owner => 'TNIEWEL',
        obj_name => 'EMP',
        old_vals => oldvals,
        new_vals => newvals);
    commit;
.....
```

Summary

- Oracle Streams can be used as an efficient tool for data replication
- Heterogeneous systems can be used as a source of/target for replication
- The complete infrastructure of Oracle Streams can be used by Streams heterogeneous replication

Q&A

ORACLE®