Oracle Streams: Step by Step

- What is Oracle Streams?
- What is Advanced Queues (AQ)?
- What is Change Data Capture (CDC)?
- What is a Logical Change Record (LCR)?
- How Streams Works
- How do you configure Streams, AQ and CDC?
- How do you use Streams in your programs?
  - Adding records automatically
  - Adding record programmatically
  - Pulling records off the queue

Who am I

- Lewis R Cunningham
- Oracle ACE
- Certified PL/SQL Developer
- Author
- Blogger
- Database Geek
- Member of SOUG, IOUG, ODTUG, ACM

What is Oracle Streams?

- Streams enables data and event movement
- Movement may be within or between databases
- Can implement replication
- Data and events can be captured
  - explicitly: programatically
  - Implicitly: redo log
- Performant
  - Log mining reduces overhead
  - Downstreams mining reduces it even more

What is Oracle Streams?, cont'd

- Streams Features
  - Distinct capture/propagation/apply services
  - Message Queueing (via AQ)
  - Publish Subscribe
  - Rules based Capture/Transform/Apply
  - Database integration
  - Easy configuration
  - Flexible
  - Inline transformations

What is Oracle AQ?

- Oracle's messaging solution
- Queue based
  - Persistent queues protect data
  - Buffered, non-persistent queues offer maximum performance
- Oracle provides all of the maintenance code
- Secure – Queue Level Access Control
- Messages Types
  - Raw
  - XML
  - Object
  - Anydata

What is Oracle AQ?, cont'd

- Publish/Subscribe
  - Multi-consumer
  - Multi-message types
- RAC integration
- API support
  - OCI (C), OCCI (C++)
  - Java JMS
  - PL/SQL
  - OO40 (COM)
  - Internet Data Access (IDAP)
  - ODBC via Heterogeneous Gateways

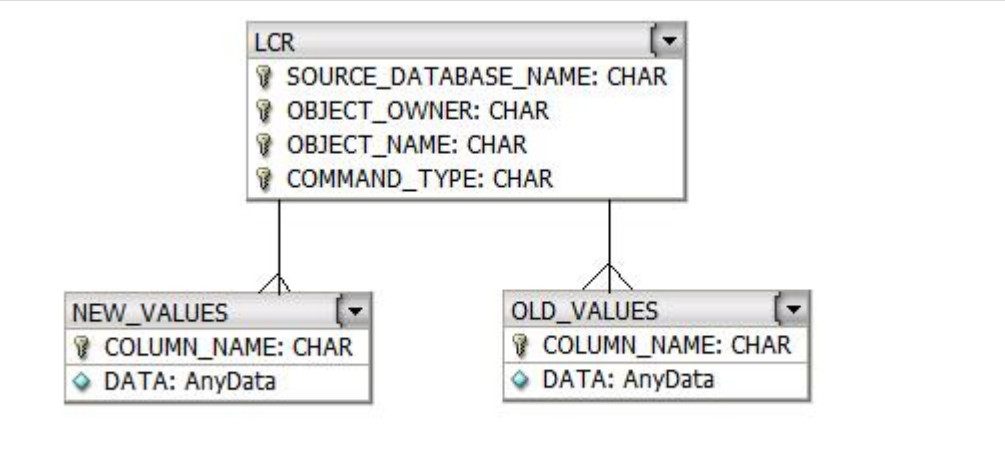
What is Change Data Capture?

- CDC efficiently identifies changed data and make it available for downstrean use
- CDC enables efficient warehouse loading
  - No flat files
  - Direct data movement
- Changes are queued
- Multiple subscribers are allowed
- Subscribers can receive changed data
- Data may be transformed
- Changes can include DDL and DML

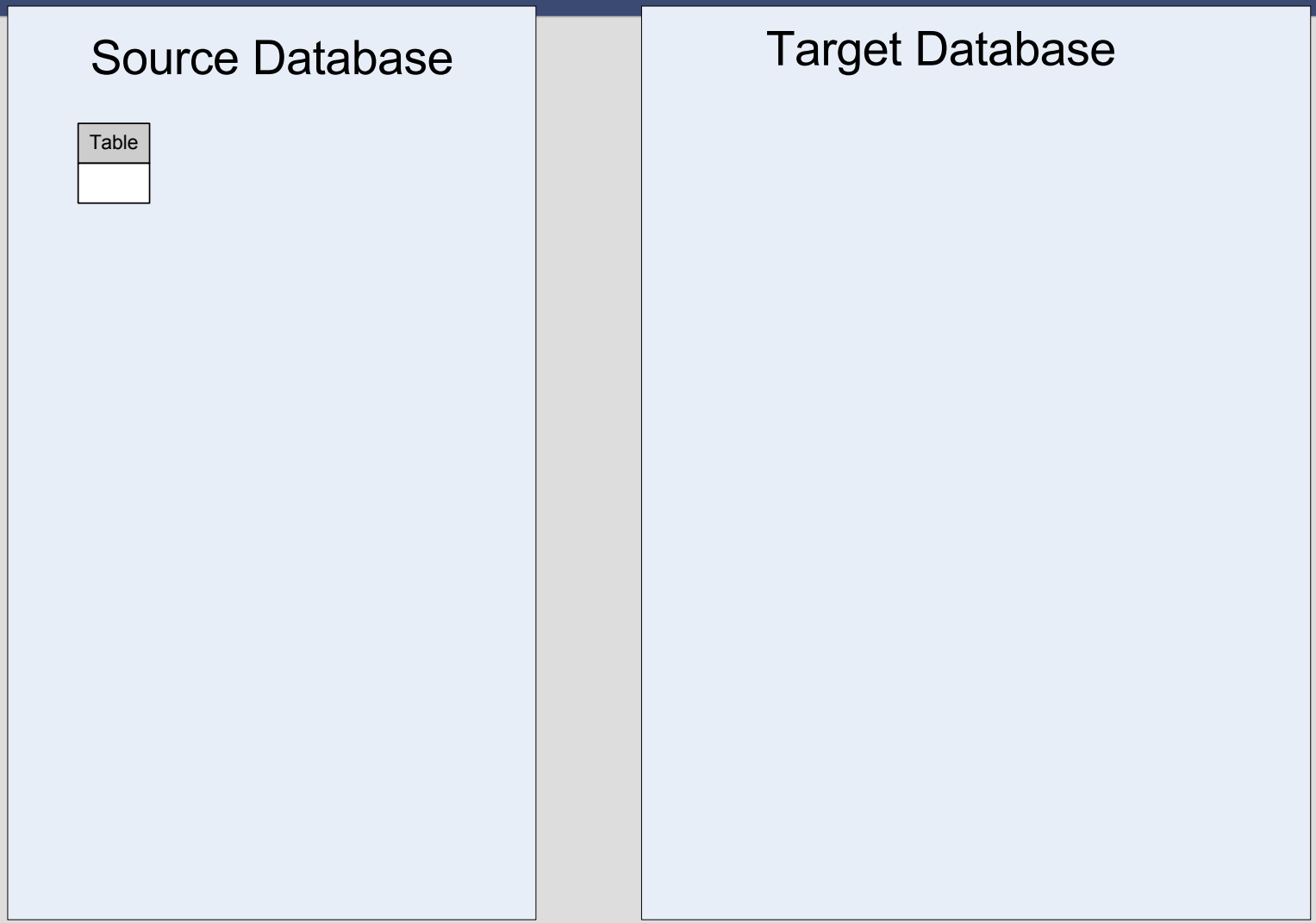
What is an LCR?

- LCR = Logical Change Record
- Committed records generate an LCR to the redo log
- An LCR shows the difference between an existing record and the new record
- For DDL, the record is a database object
- For DML, the record is table data

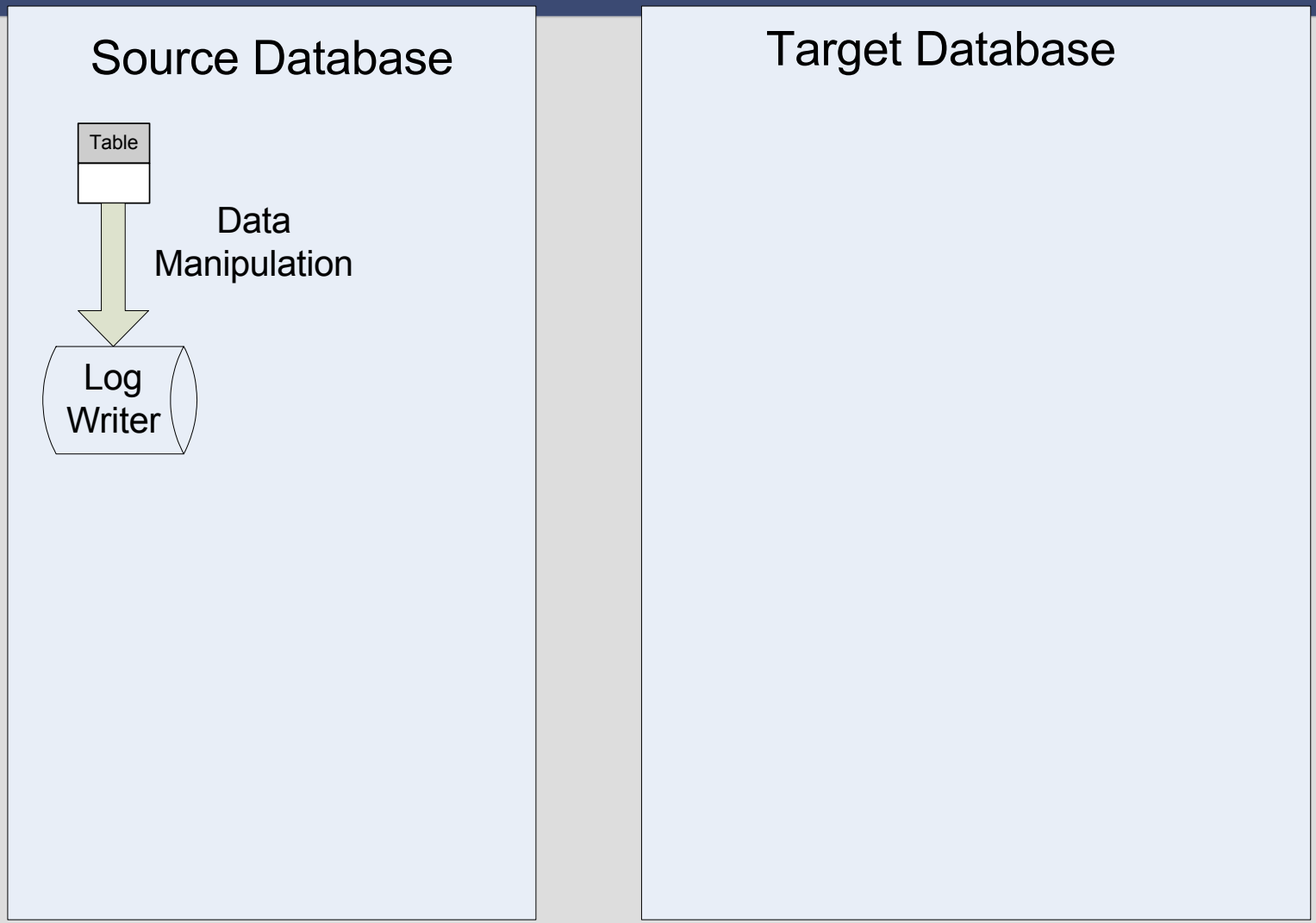
What is an LCR?



How it works

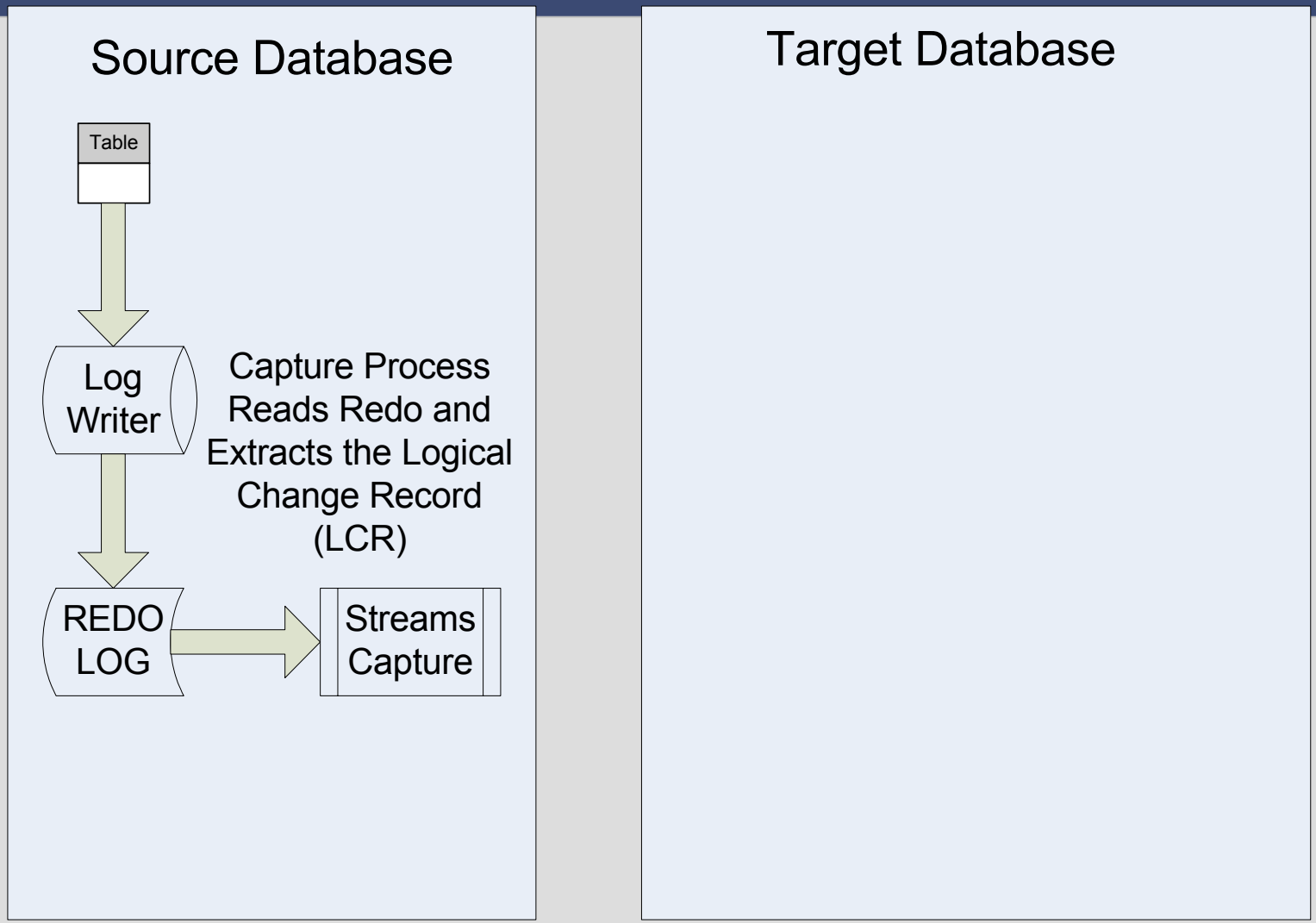


Data Changes

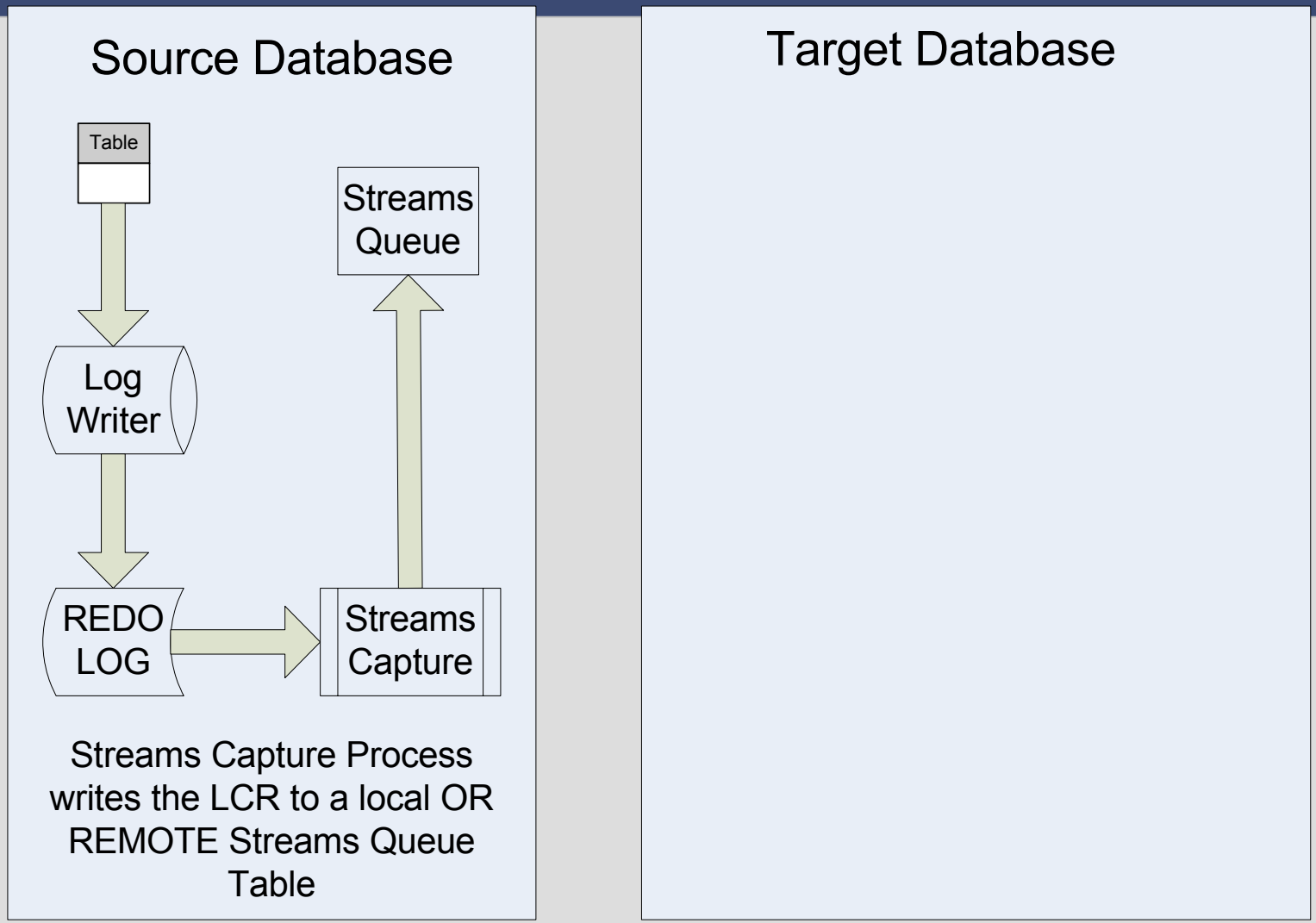


DB Writes to OS File

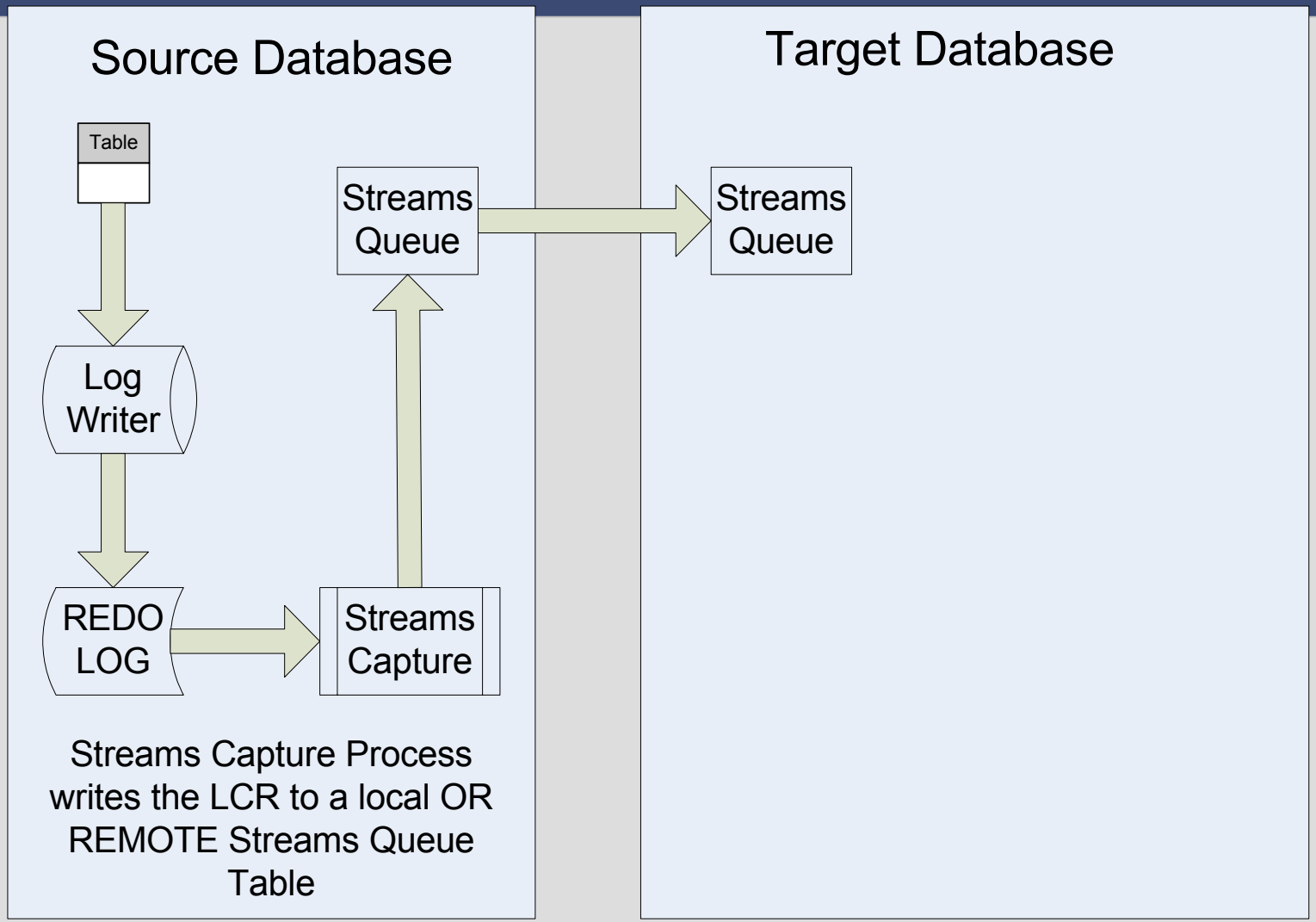
Streams Takes Over



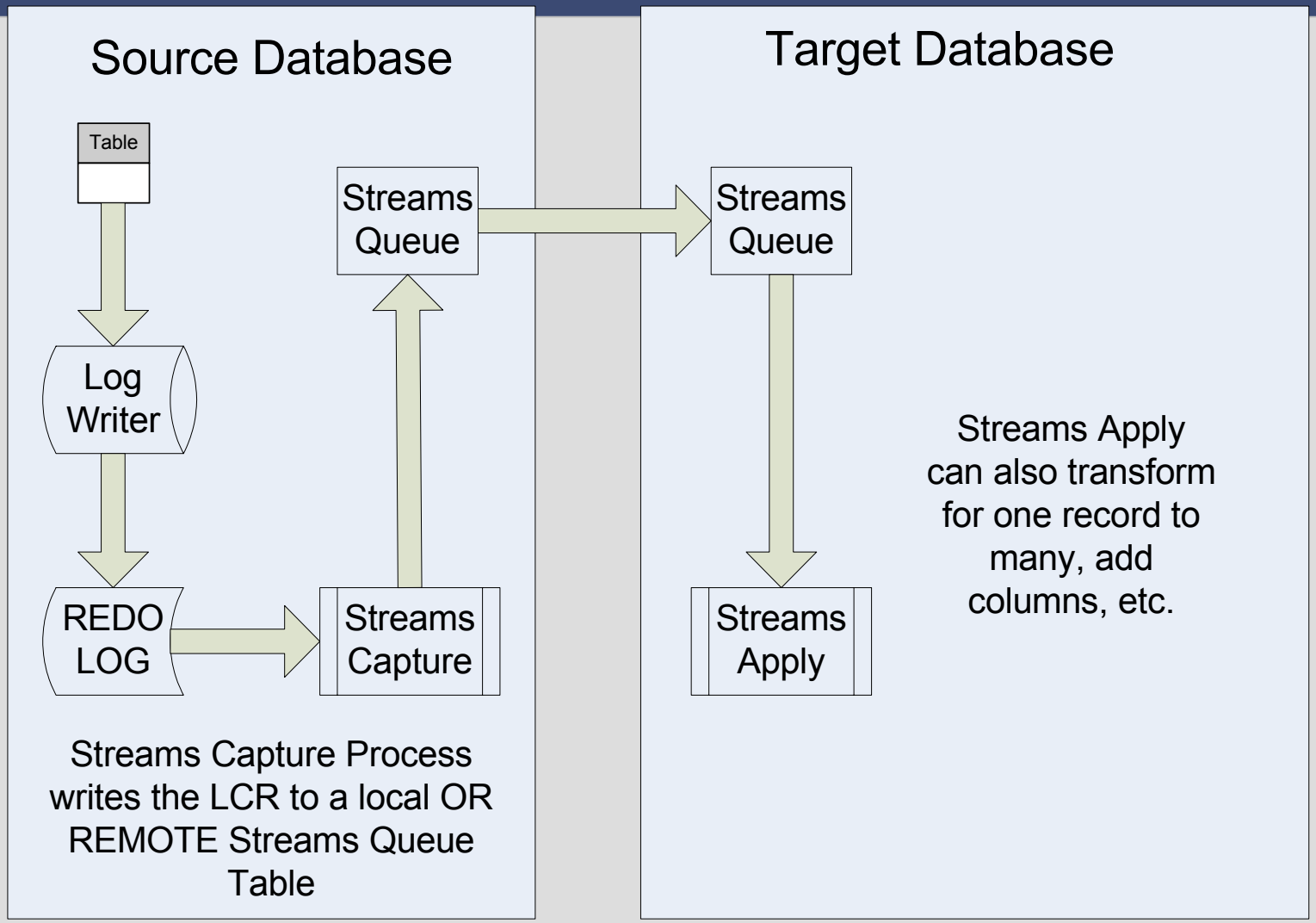
Streams Queues The LCR



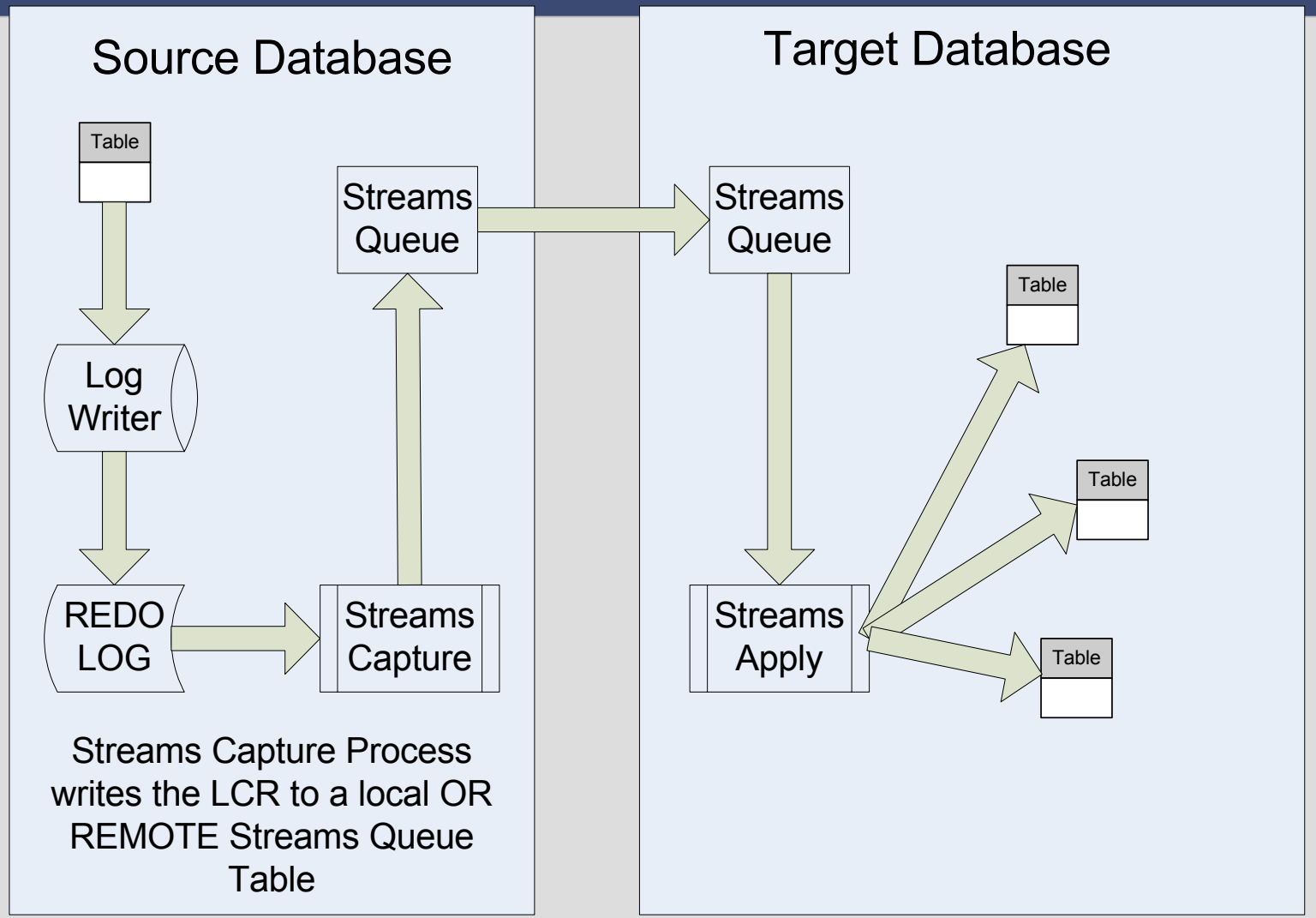
LCR Propagates



Changes Are Applied



Target Database



Configuring Streams

- The Scenario
  - Capture changes to the Employee table in 10g ORCL instance)
  - Send changes to Employee_audit in 9i (SECOND instance)
  - We will start with capture and apply in 10g
  - Once the capture and apply is working, we will modify the stream to send the data to 9i

Configuring Streams 10g

- ***Create Streams Admin***
- ***Setup CDC Queue***
- ***Create Capture Rule***
- ***Instantiate SCN***
- ***Create Apply Rule***
- ***Add Supplemental Logging***
- ***Create DML Procedure***
- ***Create DML Handlers***
- ***Set Parameters***
- ***Start Apply***
- ***Start Capture***

Configuring Streams 9i

- ***Create AQ Table***
- ***Create AQ Queue***
- ***Start AQ Queue***
- ***Create Propagate on 10g***
- ***Create Dequeue Procedure***
- ***Dequeue Transactions***

Configuring Streams

Step	Config	Source	Target	Examples

1	One Time Setup	Create Streams Admin	Create Streams Admin	CREATE USER Streams;
				GRANT DBA TO Streams;
				BEGIN
				DBMS_STREAMS_AUTH.GRANT_ADMIN_PRIVILEGE(
				grantee => 'streams',
				grant_privileges => true);
				END;

2	CDC	Setup CDC Queue		BEGIN
				DBMS_STREAMS_ADM.SET_UP_QUEUE(
				queue_table => 'streams.streams_ent_queue_table',
				queue_name => 'streams.streams_ent_queue');
				END;

3	CDC	Create Capture Rule		BEGIN DBMS_STREAMS_ADM.ADD_TABLE_RULES(table_name => 'ces_ods.ent2', streams_type => 'capture', streams_name => 'capture_ent2', queue_name => 'streams.streams_ent_queue', include_dml => true, include_ddl =>

Configuring Streams

4	CDC	Instantiate SCN		DECLARE l_scn number; BEGIN l_scn := DBMS_FLASHBACK.GET_SYSTEM_CHANGE_NUMBER (); DBMS_APPLY_ADM.SET_TABLE_INSTANTIATION_SCN (source_object_name => 'ces_ods.ent2', source_database_name => 'lbces.world', instantiation_scn => l_scn

5	CDC	Create Apply Rule		DECLARE
				entity_rule_name_dml VARCHAR2(30);
				entity_rule_name_ddl VARCHAR2(30);
				BEGIN
				DBMS_STREAMS_ADM.ADD_TABLE_RULES(
				table_name => 'ces_ods.ent2',
				streams_type => 'apply',
				streams_name => 'apply_ent2',

6	CDC	Add Any Sumplemental Logging		BEGIN
				DBMS_CAPTURE_ADM.INCLUDE_EXTRA_ATTRIBUTE(
				capture_name => 'capture_ent2',
				attribute_name => 'username',
				include => true);
				END;
				/

Configuring Streams

- ***Create DML Handler Proc***
  - ```
    CREATE OR REPLACE PROCEDURE emp_dml_handler(in_any
    IN ANYDATA) IS
      lcr             SYS.LCR$_ROW_RECORD;
      rc              PLS_INTEGER;
      command         VARCHAR2(30);
      old_values      SYS.LCR$_ROW_LIST;
    BEGIN
      -- Access the LCR
      rc := in_any.GETOBJECT(lcr);
      -- Get the object command type
      command := lcr.GET_COMMAND_TYPE();
    ```
  - ```
      -- I am inserting the XML equivalent of the LCR
    into the monitoring table.
      insert into streams_monitor (txt_msg)
         values (command ||
          DBMS_STREAMS.CONVERT_LCR_TO_XML(in_any) );
    ```

Configuring Streams

- ***Create DML Handler (cont'd)***
  - ```
    -- Set the command_type in the row LCR to INSERT
    lcr.SET_COMMAND_TYPE('INSERT');
    -- Set the object_name in the row LCR to EMP_DEL
    lcr.SET_OBJECT_NAME('EMPLOYEE_AUDIT');
    ```
  - ```
    -- Set the new values to the old values for update and delete
    IF command IN ('DELETE', 'UPDATE') THEN
        -- Get the old values in the row LCR
        old_values := lcr.GET_VALUES('old');
        -- Set the old values in the row LCR to the new values in the row LCR
        lcr.SET_VALUES('new', old_values);
        -- Set the old values in the row LCR to NULL
        lcr.SET_VALUES('old', NULL);
     END IF;
    ```

Configuring Streams

- ***Create DML Handler (cont'd)***
- ```
  -- Add a SYSDATE for upd_date
  lcr.ADD_COLUMN('new', 'UPD_DATE',
ANYDATA.ConvertDate(SYSDATE));
  -- Add a user column
  lcr.ADD_COLUMN('new', 'user_name',

lcr.GET_EXTRA_ATTRIBUTE('USERNAME'));
  -- Add an action column
  lcr.ADD_COLUMN('new', 'ACTION',
ANYDATA.ConvertVarChar2(command));
  ```

- ```
  -- Make the changes
  lcr.EXECUTE(true);
 commit;
END;
/
  ```

Configuring Streams

8	CDC	Create DML handlers		BEGIN DBMS_APPLY_ADM.SET_DML_HANDLER(object_name => 'ces_ods.ent2', object_type => 'TABLE', operation_name => 'INSERT', error_handler => false, user_procedure => 'streams.ces_dml_handler', apply_d

9	CDC	Set Any Additional Parameters		BEGIN
				DBMS_APPLY_ADM.SET_PARAMETER(
				apply_name => 'apply_ent2',
				parameter => 'disable_on_error',
				value => 'n');
				END;

10	CDC	Start Apply		BEGIN
				DBMS_APPLY_ADM.START_APPLY(
				apply_name => 'apply_ent2');
				END;

11	CDC	Start Capture		BEGIN
				DBMS_CAPTURE_ADM.START_CAPTURE(
				capture_name => 'capture_ent2');
				END;

Configuring Streams

12	AQ	Ceate AQ Q Table	Create AQ Q Table	BEGIN
				DBMS_AQADM.CREATE_QUEUE_TABLE(
				queue_table => 'lrc_src_ent3_mc_t',
				queue_payload_type => 'sys.xmltype',
				multiple_consumers => TRUE,
				compatible => '8.1');
				END;

13	AQ	Create AQ Q Q	Create AQ Q Q	BEGIN
				DBMS_AQADM.CREATE_QUEUE(
				queue_name => 'streams.lrc_src_ent3_mc_q',
				queue_table => 'streams.lrc_src_ent3_mc_t');
				END;

14	AQ	Start AQ Q	Start AQ Q	BEGIN
				DBMS_AQADM.START_QUEUE (
				queue_name => 'streams.lrc_src_ent3_mc_q');
				END;

15	AQ	Create Prop Schedule		BEGIN DBMS_STREAMS_ADM.ADD_TABLE_PROPAGATION_RULES(table_name => 'ces_ods.ent2', streams_name => 'ces_to_gems2', source_queue_name => 'streams.lrc_src_ent3_mc_q', destination_queue_name => 'streams.lrc_src_ent3_mc_q@lbindp.world',

Configuring Streams

- ***Create a DEQUEUE proc in 9i***

```
CREATE OR REPLACE PROCEDURE emp_dq (consumer IN
VARCHAR2) AS
  msg             ANYDATA;
  row_lcr         SYS.LCR$_ROW_RECORD;
  num_var         pls_integer;
  more_messages   BOOLEAN := true;
  navigation      VARCHAR2(30);
BEGIN
  navigation := 'FIRST MESSAGE';
  WHILE (more_messages) LOOP
    DBMS_OUTPUT.PUT_LINE('Looping thru messages');
    BEGIN
      DBMS_STREAMS_MESSAGING.DEQUEUE(
        queue_name   => 'strmadmin.streams_queue',
        streams_name => consumer,
        payload      => msg,
        navigation   => navigation,
        wait         => DBMS_STREAMS_MESSAGING.NO_WAIT);
```

Configuring Streams

- ***Create a DEQUEUE proc in 9i***
  - ```
    IF msg.GETTYPENAME() = 'SYS.LCR$_ROW_RECORD' THEN
      num_var := msg.GetObject(row_lcr);
      DBMS_OUTPUT.PUT_LINE(row_lcr.GET_COMMAND_TYPE ||
          ' row LCR dequeued');
    END IF;
          navigation := 'NEXT MESSAGE';
        COMMIT;
    EXCEPTION
    WHEN SYS.DBMS_STREAMS_MESSAGING.ENDOFCURTRANS THEN
              navigation := 'NEXT TRANSACTION';
    WHEN DBMS_STREAMS_MESSAGING.NOMOREMSGS THEN
              more_messages := false;
              DBMS_OUTPUT.PUT_LINE('No more messages.');
    WHEN OTHERS THEN
              RAISE;
    END;
    END LOOP;
    END;
    /
    ```
    ```

Summary

- Streams only looks complex
- It's conceptually simple and is implemented via a few procedures
- Questions?
- Thank you for coming