



# **Best Practices for Change Data Capture and replication from Oracle into Teradata using WisdomForce DatabaseSync**

**Note\*:** The latest copy of this document is available at

<http://www.wisdomforce.com/resources/docs/databasesync/DatabaseSyncBestPracticesforTeradata.pdf>

## **Table of Contents**

<i>Overview .....</i>	<i>2</i>
<i>Chapter 1: DatabaseSync Components.....</i>	<i>2</i>
<i>Chapter 2: DatabaseSync Apply Modes.....</i>	<i>4</i>
<i>Chapter 3: DatabaseSync Requirements and Limitations.....</i>	<i>5</i>
<i>Chapter 4: Considerations for Teradata destinations .....</i>	<i>6</i>

## Overview

**WisdomForce DatabaseSync™** ([www.wisdomforce.com](http://www.wisdomforce.com)) is Change Data Capture (CDC) database replication software. DatabaseSync captures transactional changes, directly from Oracle redo log files and replicates the changes to destination databases in a flexible manner.

**Teradata RDBMS** ([www.Teradata.com](http://www.Teradata.com)) is a massively parallel processing system running shared nothing architecture. Teradata provides integrated, optimized and extensible technology for a single application-neutral repository of your current and historical data, forming the framework of the business intelligence architecture.

This document describes important points for capturing most recent transactions from Oracle using DatabaseSync and applying into Teradata.

## Chapter 1: DatabaseSync Components.

WisdomForce DatabaseSync (or DBSync) is composed of four main components.

**DBSync Manager** is a graphical application used to generate the configuration file, used by the other components. The configuration file contains the definitions for the source database and tables, destination database and tables, the mappings between the tables and other configuration settings. A schema conversion option, allow to generate compatible table definitions for various destination databases, using the Oracle table definitions.

**DBSync InitialSync** is a command line utility used for first time synchronization of data for SQL Replay or Merge Apply methods. The synchronization needs to be performed before starting replication of the table changes. For each synchronized table InitialSync notes the Oracle SCN (Change number), this identifies the point in time from which to apply changes for this particular table on the destination.

InitialSync extracts the data using regular Oracle OCI calls. Since Oracle maintains read consistency for the select statement, the data in the source tables can be changed normally while InitialSync is running. The data changes performed on the tables during InitialSync will be applied correctly to the destination table.

InitialSync can synchronize multiple tables in parallel. The number of threads used is controlled by a parameter.

InitialSync uses direct load APIs for loading the data. For Teradata destinations, Teradata Parallel Transporter with the FastLoad or MultiLoad driver is used.

Another WisdomForce product – FastReader - extracts data from Oracle by reading the data directly from the Oracle data files for better performance.

FastReader is integrated with DBSync and can be used for faster first time Synchronization instead of DBSync InitialSync

**DBSync Extractor** parses any new data in the Oracle redo log files and extracts the data changes for the tables specified in the configuration file. Both the committed and the non-committed changes are extracted. The captured changes are written to an on disk queue, this queue is used by DBSync Apply. Log files produced on any platform can be parsed on any other platform, for example log files from HPUNIX maybe parsed on Windows or Linux. The performance of DBSync Extract is in the range of 10-70 MB of redo log files parsed per second, depending on the hardware and IO speed accessing the redo logs.

**DBSync Applier** reads only the committed changes present in the on disk queue and applies these changes to the destination. After application of changes, the data for processed transactions is purged from the on disk queue. The apply process is batch based, every execution combines all the changes produced committed transactions since the last application and applies the changes to the destination in one or several destination transactions. In case of apply failure, the entire apply batch is rolled back on the destination database and may be retried again after correction of any issues.

## **Chapter 2: DatabaseSync Apply Modes.**

There are three apply modes available in DBSync Applier:

**SQL Replay Apply** executes, on the destination table, the equivalent SQL statements to the changes performed on the source. For every row update, insert or delete on the source there will be an insert, update or delete on the destination. SQL calls are executed using bulk binding where possible. When parallel execution is used, changes are applied using multiple threads and multiple connections the destination database. Changes are spread among threads and the correct order of execution is preserved by using internal locking, based on row identifiers. SQL Replay should only be used for Teradata destinations with low volume of transactional changes, for best performance, use Merge Apply.

**Audit/Log Table Apply** is a table in the destination database, the table is populated by a new row for every update, insert or delete in the original table. Each row in the log table contains a before change and after change image for every column in the original table as well as additional columns containing the metadata of the operation. The optional metadata columns are operation type, time, SCN, transaction id, commit SCN and commit time. Each replicated table needs to have a separate log table in the destination. The DBSync Manager can be used to generate appropriate Create Table statements with compatible data types for the destination database. Initial synchronization is not applicable for log table destinations.

**Merge Apply** is a combination of the first two methods. For every replicated table, a regular table and a log table should be present on the destination. For every apply batch, the log tables are truncated and populated by the changes present in the current apply batch. The changes to table data are merged from the log table to the destination table, using several batch SQL statements executed on the destination. The new rows are inserted to the destination table, the updated rows are updated in the destination table and the deleted rows are deleted from the destination table. For Teradata destinations Merge Apply is the recommended apply method.

The SQL merge statements used by Merge Apply are optimized to the particular data changes in the apply batch. If for example, the batch only contains inserts for a particular table then a simple “Insert as select” statement will be performed. In case the batch contains updates and deletes for a particular table, two joins will be performed to the destination table, one join for updates and one join for deletes.

### **Chapter 3: DatabaseSync Requirements and Limitations**

The following considerations have to be taking into account when deploying and using WisdomForce DatabaseSync:

- The source Oracle database must be in ARCHIVE LOG mode
- The source Oracle tables should be defined as LOGGING
- The source table must have a logical primary key to replicate updates and deletes; i.e. a set of columns that uniquely identifies a single row. An actual unique index or primary key definition is not required.
- The logical primary key columns need to be defined with supplemental logging in Oracle, this can be done using DBSync Manager.
- DBSync Extract must have read access to the Oracle archive log files.

## **Chapter 4: Considerations for Teradata destinations**

When replicating changes to Teradata, the Merge Apply mode must be used for best performance. Execution of single SQL statements for each row change is less efficient than merging the changes in bulk using Merge Apply.

Give consideration to the apply batch size used when applying changes to Teradata. Merge Apply into Teradata works best with larger apply batches, since every merge containing updates or deletes requires a hash join or nested loops between the log table and the destination table.

For best join performance, the logical primary key should be defined as the primary key in the destination table definition.

## Log Table example

Assuming the following table structure on Oracle:

```
CREATE TABLE PEOPLE
(
  ID NUMBER (10),
  NAME VARCHAR2 (50),
  ADDRESS VARCHAR2 (250),
  PRIMARY KEY (ID)
)
```

Assuming a compatible table structure on Teradata:

```
CREATE TABLE PEOPLE
(
  ID BIGINT (10),
  NAME VARCHAR (50),
  ADDRESS VARCHAR (250)
)
PRIMARY INDEX (ID)
```

There should be a log table on Teradata:

```
CREATE TABLE PEOPLE_LOG
(
  OP_XID DECIMAL(22),
  OP_CODE CHAR(1) NOT NULL,
  OP_TIME TIMESTAMP(0),
  OP_CMT_SCN BIGINT,
  OP_CMT_TIME TIMESTAMP(0),
  OP_NUM_IN_TX BIGINT NOT NULL,
  ID_NEW BIGINT (10),
  ID_OLD BIGINT (10),
  NAME_NEW VARCHAR (50),
  NAME_OLD VARCHAR (50),
  ADDRESS_NEW VARCHAR (250),
  ADDRESS_OLD VARCHAR (250)
)
PRIMARY INDEX (ID_OLD)
```

The following changes were performed on the source table:

```
insert into PEOPLE(ID,NAME,ADDRESS) values(2,'Kyle Brown','875 Paulson St.,
Bothell 98012, WA');
commit;
update PEOPLE set ADDRESS='123 Preston St., Bothell 98013, WA' where id=2;
commit;
delete from PEOPLE where id=2;
commit;
insert into PEOPLE(ID,NAME,ADDRESS) values(3,'Rose Brown','875 Paulson St.,
Bothell 98012, WA');
commit;
update PEOPLE set ADDRESS='4653 Preston St., Bothell 98013, WA' where id=3;
commit;
```

The log table would contain the following data after the apply execution.

OP_CODE	OP_TIME	OP_CMT_SCN	OP_CMT_TIME	OP_XID	OP_NUM_IN_TX	ID_NEW	ID_OLD	NAME_NEW	NAME_OLD	ADDRESS_NEW	ADDRESS_OLD
I	2/16/09 4:35 AM	201	2/16/09 4:35 AM	101	1	2	2	Kyle Brown		875 Paulson St., Bothell 98012, WA	
U	2/16/09 4:35 AM	202	2/16/09 4:35 AM	102	1	2	2			123 Preston St., Bothell 98013, WA	875 Paulson St., Bothell 98012, WA
D	2/16/09 4:35 AM	203	2/16/09 4:35 AM	103	1		2		Kyle Brown		123 Preston St., Bothell 98013, WA
I	2/16/09 4:36 AM	204	2/16/09 4:36 AM	104	1	3	3	Rose Brown		875 Paulson St., Bothell 98012, WA	
U	2/16/09 4:36 AM	205	2/16/09 4:36 AM	105	1	3	3			4653 Preston St., Bothell 98013, WA	875 Paulson St., Bothell 98012, WA

- OP\_CODE the operation type (I for insert, U for update, D for delete)
- OP\_TIME the time when the change was made
- OP\_CMT\_SCN the SCN of the commit
- OP\_CMT\_TIME the time when the commit occurred
- OP\_XID the source transaction ID
- OP\_NUM\_IN\_TX the number of the operation in the transaction
- ID\_NEW the after image of the ID column
- ID\_OLD the before image of the ID column
- NAME\_NEW the after image of the NAME column
- NAME\_OLD the before image of the NAME column
- ADDRESS\_NEW the after image of the ADDRESS column
- ADDRESS\_OLD the before image of the ADDRESS column

The primary key values are inserted into both the before and after image columns for inserts and updates (even if the primary key column was not modified). On update, only

the primary key (which should be defined with supplemental logging in Oracle) and the changed columns are present in the log tables. A column wasn't updated incase both the before and after image are both NULL.

When Merge Apply is performed, the merge query, using SQL queries to filter out the row with ID=2 (since it was deleted), merges the updates for inserted rows and only performs an insert for the row (3, 'Rose Brown', '4653 Preston St., Bothell 98013, WA' )