Triton
CONSULTING

DB2 Shadow Tables: **Bringing Analytics to Data**

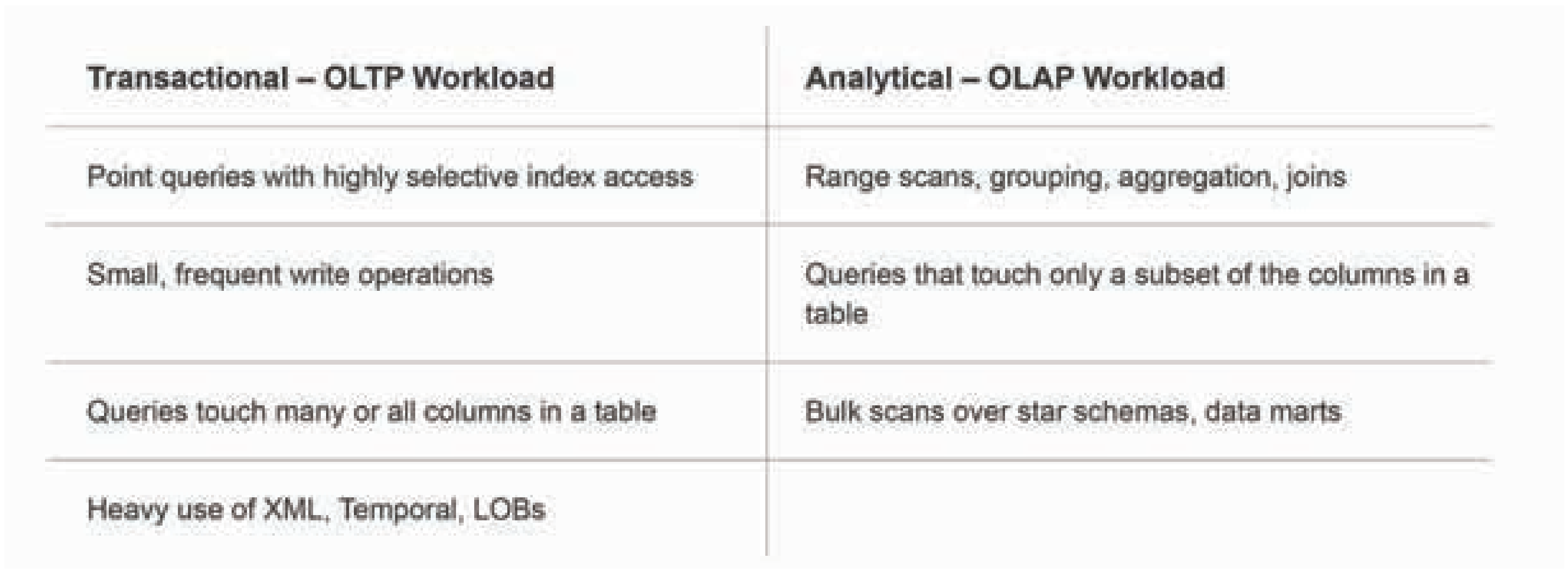
Somu Chakrabarty
Iqbal Goralwalla
Triton Consulting

Contents

Background

OLTP and OLAP – Poles Apart

A front-end related OLTP database is characterised by a high number of short transactions (each accessing/pointing to a very small number of records) that are relatively quick to run. The back-end OLAP database usually has a relatively smaller number of medium to highly complex and long-running transactions (each accessing a very high number of records, often as large as the entire table).



Transactional – OLTP Workload	Analytical – OLAP Workload
Point queries with highly selective index access	Range scans, grouping, aggregation, joins
Small, frequent write operations	Queries that touch only a subset of the columns in a table
Queries touch many or all columns in a table	Bulk scans over star schemas, data marts
Heavy use of XML, Temporal, LOBs	

Typically customers having a mixed OLTP and OLAP workload are forced to:

1. Either use a large number indexes, performance enhancing objects like Materialized Query Tables (MQTs), and a huge amount of DBA tuning effort in a single database to speed up the complex OLAP workload, or
2. Keep two separate database systems, by periodically copying data from the transactional database to a separate Data Warehouse or Data Mart for running analytic queries.

With the first option, if both OLTP and OLAP applications access the same table, locking on data by OLAP queries might slow down OLTP operations. Additional indexes created to serve OLAP applications might slow down OLTP operations. Our customers have experienced performance issues when transactional tables are also used for reporting or analytics. Reports run on these transactional tables do not run very well and often block transactional queries. As a result, customers are forced to schedule analytic queries in a timeframe when no transactional workload is running. However, finding that timeframe is often difficult or impossible with 24x7 online requirements for many of the databases.

With the second option, customers use some form of asynchronous replication or load utility or some kind of ETL operations to refresh the OLAP database from the OLTP database at regular intervals. The main drawbacks with this architecture are:

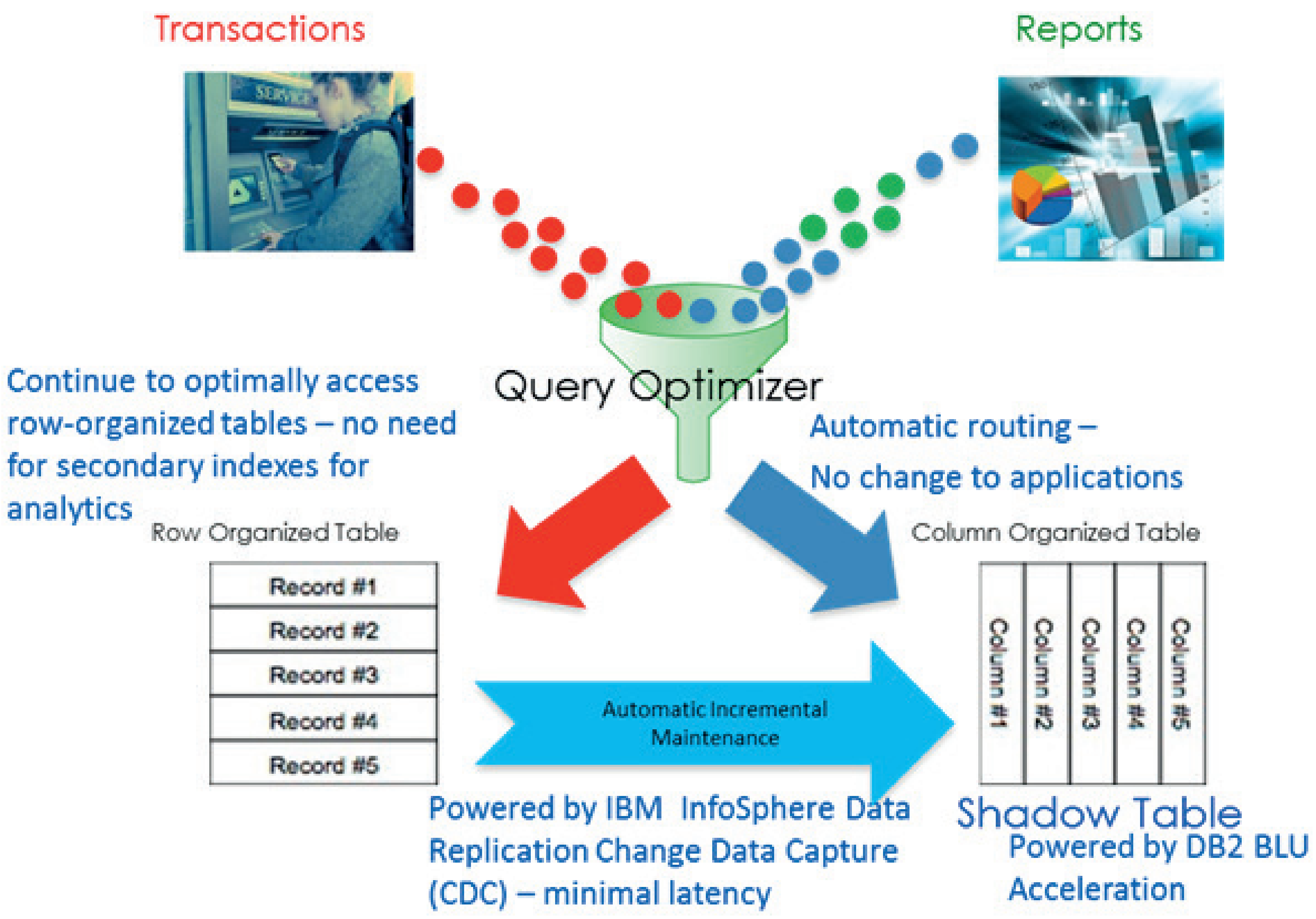
- There is increased business need to run real-time reporting and analytics directly on the live transactional data.
- Businesses do not want the hassle of building and administrating a separate database for reporting,

because of complex ETL setup, massive data movement. This architecture can be quite expensive.

Shadow Tables

To address the customer needs outlined above, IBM has introduced the concept of shadow tables in DB2 V10.5 Fixpack 4 (also known as the “Cancun Release”) to avoid separate OLTP and OLAP databases and to improve analytic query performance in a single OLTP database without using much tuning and indexing efforts. In essence, shadow tables bring analytics to data.

A shadow table is a column-organised copy of a row-organised table that includes all columns or a subset of columns from the source row table. As such, it leverages the power of BLU Acceleration which results in fast reporting. The DB2 optimiser automatically routes transactional queries to the row tables, whilst analytic queries are sent to the columnar shadow tables. This enables fast reporting directly on the transactional system without the performance impact to the transactional workload and eliminates the need to replicate data into a separate data warehouse or data mart. OLTAP (Online Transactional Analytic Processing), depicted in the diagram below, is now a reality.


Transactions
Reports
Query Optimizer
Continue to optimally access row-organized tables – no need for secondary indexes for analytics
Automatic routing – No change to applications
Row Organized Table
Record #1
Record #2
Record #3
Record #4
Record #5
Column Organized Table
Column #1
Column #2
Column #3
Column #4
Column #5
Automatic Incremental Maintenance
Powered by IBM InfoSphere Data Replication Change Data Capture (CDC) – minimal latency
Shadow Table
Powered by DB2 BLU Acceleration

Getting Started

Shadow tables are implemented as materialized query tables (MQTs) that are maintained by replication. Using shadow tables, we can get the performance benefits of BLU Acceleration for analytic queries in an OLTP environment. Analytical queries against row-organised tables are automatically routed to shadow tables if the replication latency falls within a user-defined limit.

High Level Steps for Implementation

1. Installation or upgrade to DB2 10.5. FP4
2. Configuring DB2 for shadow tables usage
3. Installation and Configuration of Infosphere CDC
4. Creating shadow tables and initial population of shadow tables

5. InfoSphere CDC subscription and table mappings configuration
6. Enabling query routing to shadow tables

In this white paper, we will expand on Steps 2, 4, and 6. Details on Step 1 (Installation or upgrade to DB2 10.5FP4) can be found in the DB2 10.5 Knowledge Center (http://ibm.co/1IBTKDW or http://ibm.co/1CqxEAu), whilst details on Steps 3 and 5 can be found in Appendix A and B of this document.

Configuring DB2 for Shadow Tables

Before using shadow tables we have to configure some DB2 server parameters. We go though some of the important ones below (a full list can be found at http://ibm.co/1u6IKvy Shadow tables are essentially BLU Column-organised tables. So, the database environment must conform to the requirements for column-organised tables.

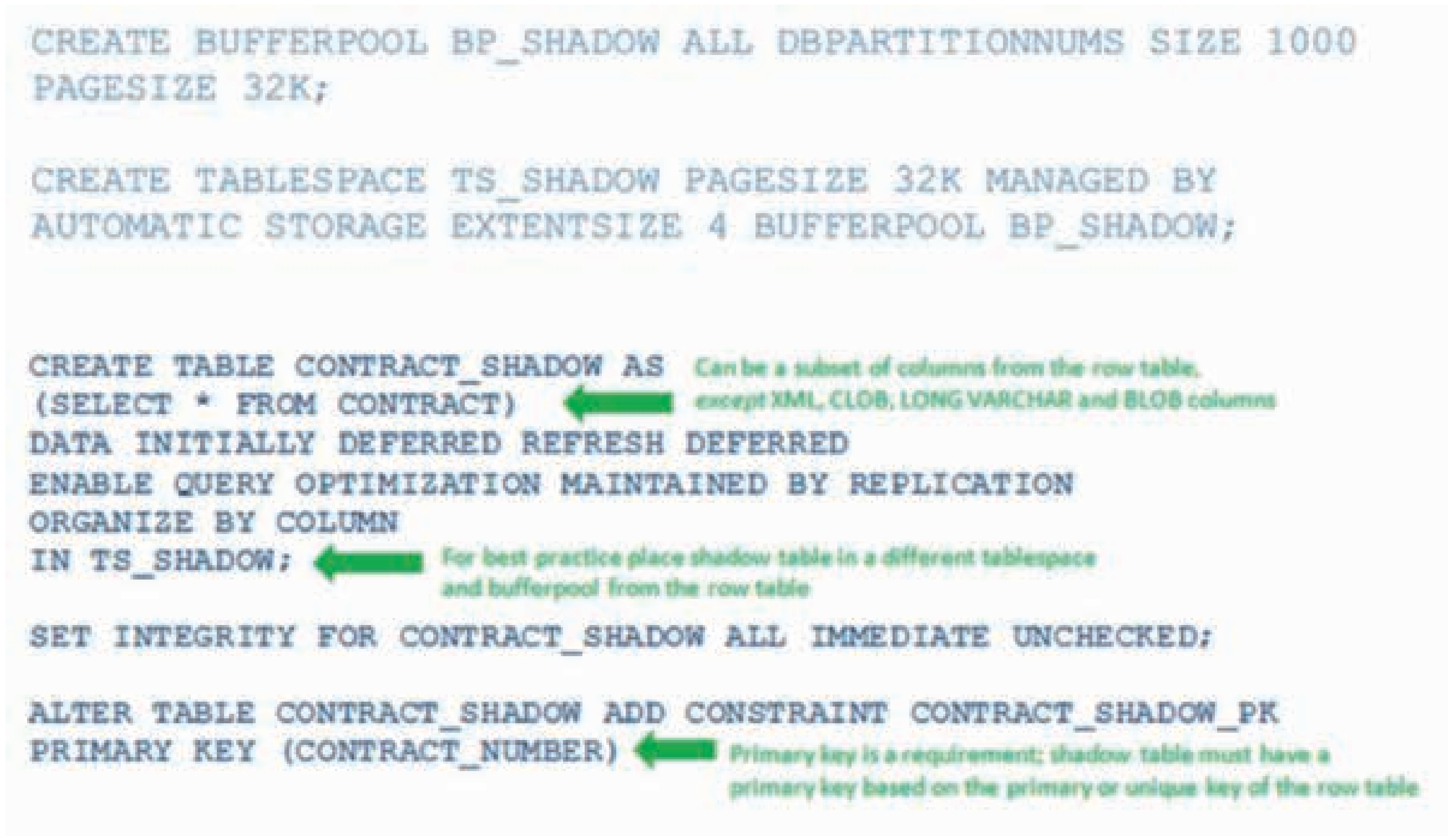
- Database code set UTF - 8
- DB2_WORKLOAD=ANALYTICS must *not* be set in the instance. This is because the shadow tables are generally used in an environment where the predominant workload is still OLTP.
- Database collating sequence IDENTITY IDENTITY_16BIT
- SORTHEAP (sort heap) and SHEAPTHRES_SHR (sort heap threshold for shared sorts) database configuration parameter - Not AUTOMATIC (set to a fixed high value suitable for column organised processing). This is because processing data for column-organised tables requires larger values for the SORTHEAP and SHEAPTHRES_SHR database configuration parameters than the values you might have for your OLTP environment.LOGARCHMETH1 (primary log archive method) database configuration parameter - set appropriately (required for CDC replication)

Creating Shadow Tables

In this white paper we will illustrate shadow tables using a simple dimensional schema which has a single row organised fact table (CONTRACT) and two row organised dimension/reference tables (CUSTOMER and CONTRACT_TYPE) related by simple integer foreign keys. CONTRACT_TYPE is just 3 rows defining the contract type: RemoteDBA, COD and COD_BILLABLE. CUSTOMER is around 200 rows that have details of customers, including address, post code, etc. CONTRACT contains 5,000,000 rows related to the two dimension tables.

Once the InfoSphere CDC components are set up and configured (see Appendix A, we can then create shadow tables.

Below we show how the CONTRACT_SHADOW table is created. The other two shadow tables, CONTRACT_TYPE_SHADOW and CUSTOMER_SHADOW are created similarly.



```
CREATE BUFFERPOOL BP_SHADOW ALL DBPARTITIONNUMS SIZE 1000
PAGESIZE 32K;

CREATE TABLESPACE TS_SHADOW PAGESIZE 32K MANAGED BY
AUTOMATIC STORAGE EXTENTSIZE 4 BUFFERPOOL BP_SHADOW;


CREATE TABLE CONTRACT_SHADOW AS
(SELECT * FROM CONTRACT)
DATA INITIALLY DEFERRED REFRESH DEFERRED
ENABLE QUERY OPTIMIZATION MAINTAINED BY REPLICATION
ORGANIZE BY COLUMN
IN TS_SHADOW;

SET INTEGRITY FOR CONTRACT_SHADOW ALL IMMEDIATE UNCHECKED;

ALTER TABLE CONTRACT_SHADOW ADD CONSTRAINT CONTRACT_SHADOW_PK
PRIMARY KEY (CONTRACT_NUMBER)
```

Can be a subset of columns from the row table, except XML, CLOB, LONG VARCHAR and BLOB columns

For best practice place shadow table in a different tablespace and bufferpool from the row table

Primary key is a requirement; shadow table must have a primary key based on the primary or unique key of the row table

Some points to note from the above shadow table creation statement:

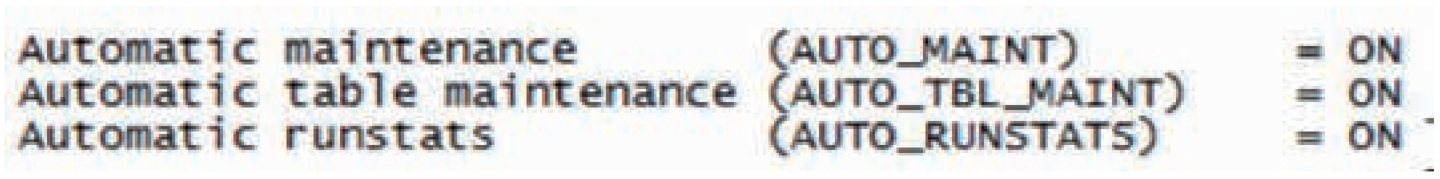
- A shadow table is created using a regular projection MQT definition.
- The MAINTAINED BY REPLICATION clause identifies the table as a shadow table.
- The shadow table can be defined to replicate all columns of the source table, or a subset of columns relevant to your query workload can be selected for replication.
- In order to accommodate different access patterns for OLTP and analytic workloads, it is best practice to have a separate bufferpool and tablespace for shadow tables.
- After it is created, a shadow table is placed in set integrity pending state. The SET INTEGRITY command needs to be issued to bring the shadow table out of the pending state.
- The shadow table needs to have a primary key corresponding to the row organised table's primary or unique key. This is required to maintain a 1:1 mapping between the source row table and its corresponding shadow table.
- It is important to note that each row organised table referenced in an analytic query must have a corresponding shadow table in order for the query to be routed to the shadow tables. As an example, if there is 3-table join in the analytic query, all the three row tables must have corresponding shadow tables.

You now have the empty shadow tables ready in the database. Before you can start replicating data, you need to add subscriptions between source row-organised tables and their corresponding shadow tables using CDC as described in Appendix B.

Initial Population of Shadow Tables

For the first refresh after the subscription mirroring is started CDC uses the technology of load utility (with a fixed set of options) under the hood to copy the content of each source row-organised table into the corresponding shadow table. To address the resource needs of the LOAD command, set the UTIL_HEAP_SZ database configuration parameter to an appropriately high starting value and AUTOMATIC.

You should also run runstats on the shadow table after the first refresh. After that, the best practice is keeping auto runstats set to on.



```
Automatic maintenance       (AUTO_MAINT)       = ON
Automatic table maintenance (AUTO_TBL_MAINT)   = ON
Automatic runstats          (AUTO_RUNSTATS)    = ON
```

Once the initial refresh is completed, the subscription will move into the Mirror Continuous state and CDC will continually replicate updates on the row-organised tables to the shadow tables. You can monitor statistics such as replication data activity and replication latency in the management console. Under normal circumstances, you will see the replication latency being a near constant line. If that is not the case (e.g. latency keeps on increasing), you may need to tune things such as increasing the I/O bandwidth or tuning certain CDC system parameters to keep the latency in check.

By default, CDC does the refresh of the shadow tables via a series of LOADs. To achieve maximum compression rate on the shadow tables, increase the value of the FASTLOAD_REFRESH_COMMIT_AFTER_MAX_OPERATIONS system parameter to the largest row count of the row-organised tables or the maximum value that CDC allows (2147483647). Doing so will ensure that the first LOAD into each shadow table sees as much data as possible to build a good compression dictionary.

Adhoc refreshing Shadow Tables outside CDC (without concurrent IUD)

It is possible to directly LOAD into a shadow table outside of CDC, allowing the possibility to tailor the LOAD options. When directly running LOAD on the shadow table, it is important to use either NONRECOVERABLE or COPY YES to avoid putting the tablespace in a backup pending state.

After the LOAD into the shadow table, you need to perform a “mark table capture point” on the row-organised table to inform CDC that the shadow table is in sync with the row-organised table at the current log position. Note CDC will not apply any IUDs that occur prior to this capture point to the shadow table. Hence, when using this option, it is up to the user to guarantee that there is no IUDs on the row-organized table during this processing. Once mark capture point is performed, CDC will start to replicate any IUDs after the capture point to the shadow table when the subscription is started.

Enabling Query Routing

The next step is to enable query routing to shadow tables. To enable the use of shadow tables, intra-partition parallelism must be enabled to allow runtime access to the column-organised shadow tables, and the optimiser must be directed to consider shadow tables for query optimisation when the replication latency is within an acceptable limit. This routing is for dynamic SQL queries only. No shadow table routing will take place for queries with RS and RR isolation levels.

Latency and Refresh Age

Replication latency is the amount of time that it takes for a transaction against a source row-organised table to be applied to a shadow table.

A shadow table defined with ENABLE QUERY OPTIMIZATION can be used to optimise the processing of queries based on a latency period if each of the following conditions is true:

- The CURRENT REFRESH AGE special register is set to duration other than zero or ANY.
- The CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION special register is set to contain only REPLICATION.
- The CURRENT QUERY OPTIMIZATION special register is set to 2 or a value greater than or equal to 5.

Not all applications can tolerate latency between the source row-organised table and the shadow table – this is a business decision. Different applications may have different latency limit. We must first identify which applications can tolerate latency and make use of shadow tables.

The refresh age is an input to the system by the DBA/administrator to specify to DB2 optimiser how much time lag between the source table and the shadow table is tolerated by business, which is actually the value of the CURRENT REFRESH AGE special register as a time stamp duration whose format is yyyymmddhhmmss, where y=year, 0-9999; m=month, 0-11; d=day, 0-30; h=hour, 0-23; m=minute, 0-59; and s=second, 0-59. The value can be truncated on the left, which means that you do not have to specify year, month, day, and so on if you want a refresh age of only a second. However, individual elements that are preceded by another element must include any leading zeros. For example, a refresh age of 10705 represents 1 hour, 7 minutes, and 5 seconds.

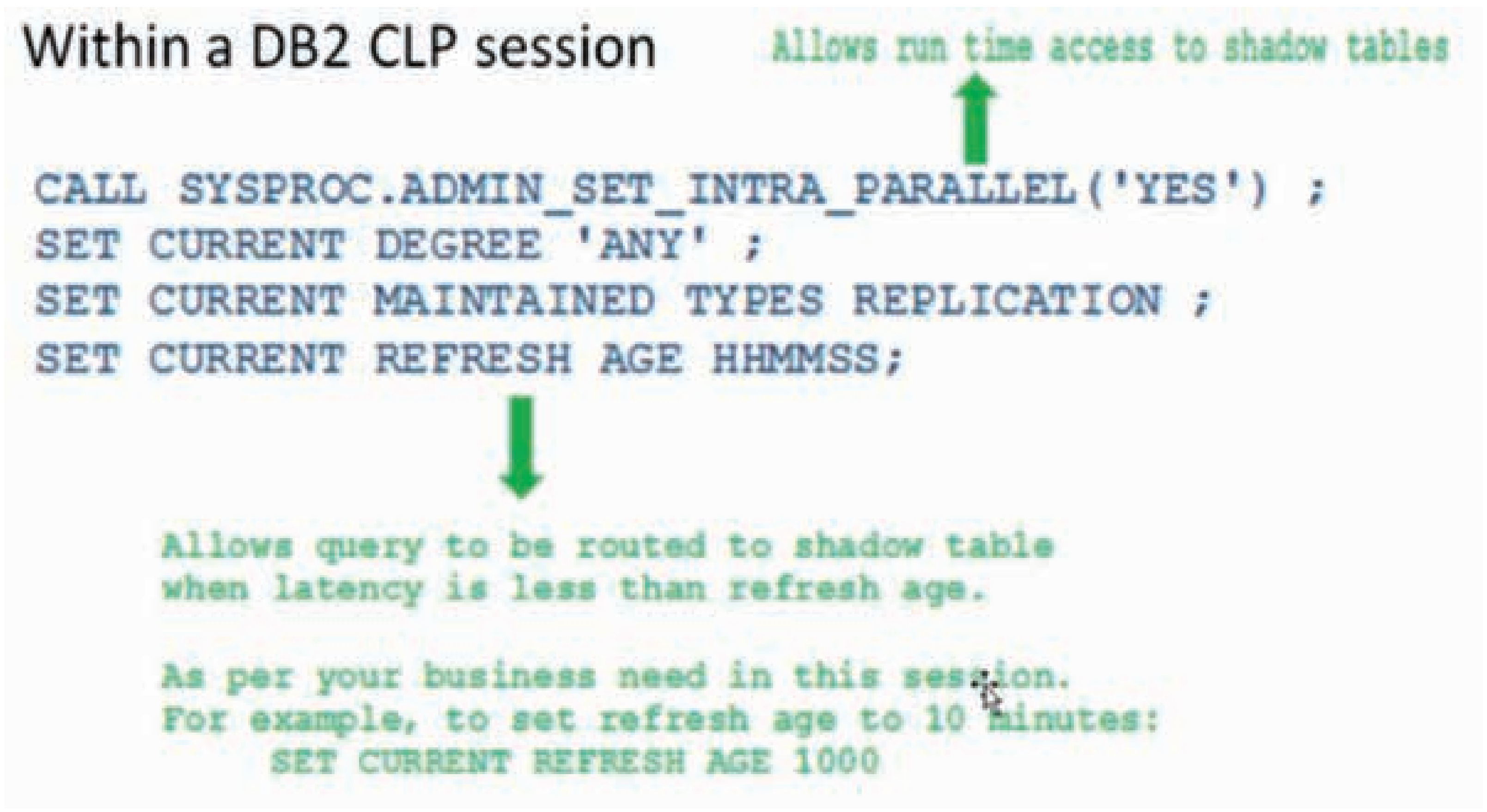
Replication latency information is shared automatically from Infosphere CDC to the DB2 server through a table called SYSTOOLS.REPL_MQT_LATENCY. This needs to be created manually, in every DB2 database in which shadow tables are used, as part of CDC setup process (see Appendix B for details).

Current Refresh Age:

- Default value 0 – never consider routing to shadow tables.
- Special value ANY – always consider shadow tables ignoring latency.
- Final decision of the optimiser whether to use shadow table or not is cost-based.

Two different ways to enable shadow table functionality and query routing to shadow tables are described below:

- Direct enablement within a DB2 command line processor (CLP) session or embed in the analytic/reporting application before running the analytic/reporting query from that same session or from that same application call:



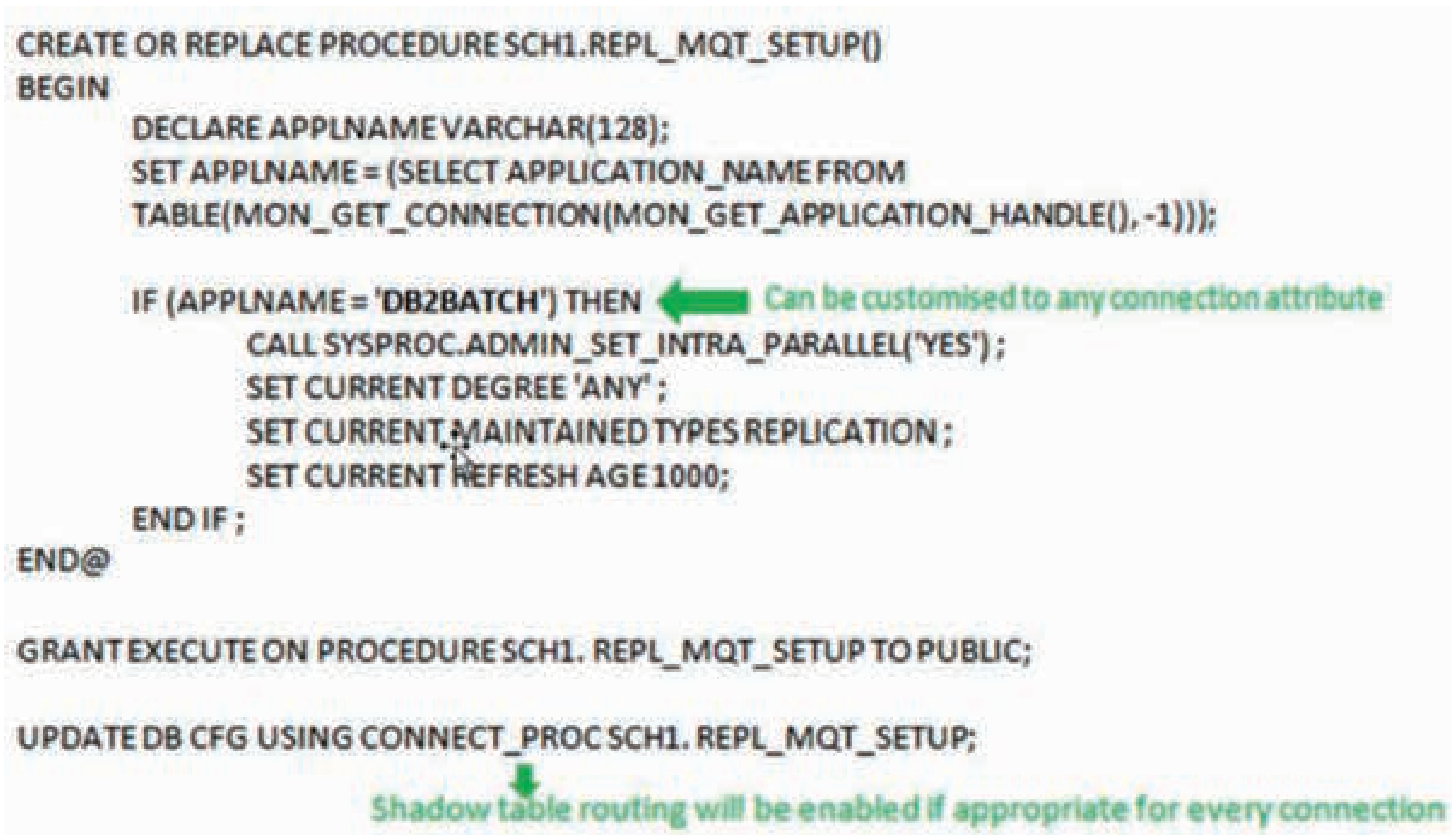
Within a DB2 CLP session

Allows run time access to shadow tables

```
CALL SYSPROC.ADMIN_SET_INTRA_PARALLEL('YES') ;
SET CURRENT DEGREE 'ANY' ;
SET CURRENT MAINTAINED TYPES REPLICATION ;
SET CURRENT REFRESH AGE HHMMSS;
```

Allows query to be routed to shadow table
when latency is less than refresh age.

As per your business need in this session.
For example, to set refresh age to 10 minutes:
SET CURRENT REFRESH AGE 1000

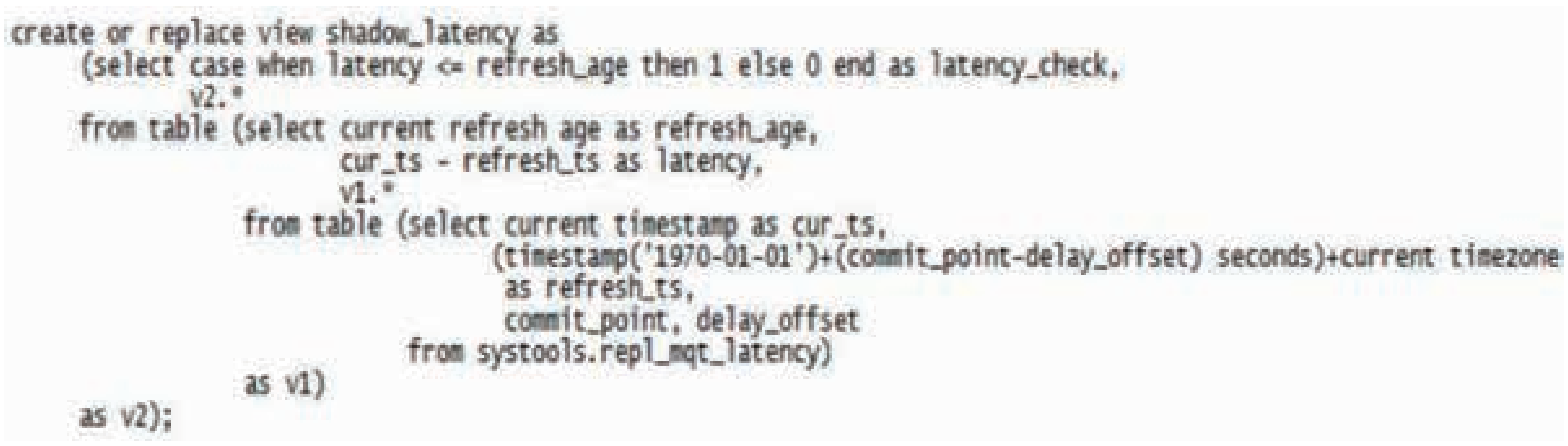


```
CREATE OR REPLACE PROCEDURE SCH1.REPL_MQT_SETUP()
BEGIN
        DECLARE APPLNAME VARCHAR(128);
        SET APPLNAME = (SELECT APPLICATION_NAME FROM
        TABLE(MON_GET_CONNECTION(MON_GET_APPLICATION_HANDLE(), -1)));

        IF (APPLNAME = 'DB2BATCH') THEN
                CALL SYSPROC.ADMIN_SET_INTRA_PARALLEL('YES') ;
                SET CURRENT DEGREE 'ANY' ;
                SET CURRENT MAINTAINED TYPES REPLICATION ;
                SET CURRENT REFRESH AGE 1000;
        END IF ;
END@

GRANT EXECUTE ON PROCEDURE SCH1. REPL_MQT_SETUP TO PUBLIC;

UPDATE DB CFG USING CONNECT_PROC SCH1. REPL_MQT_SETUP;
```

Can be customised to any connection attribute

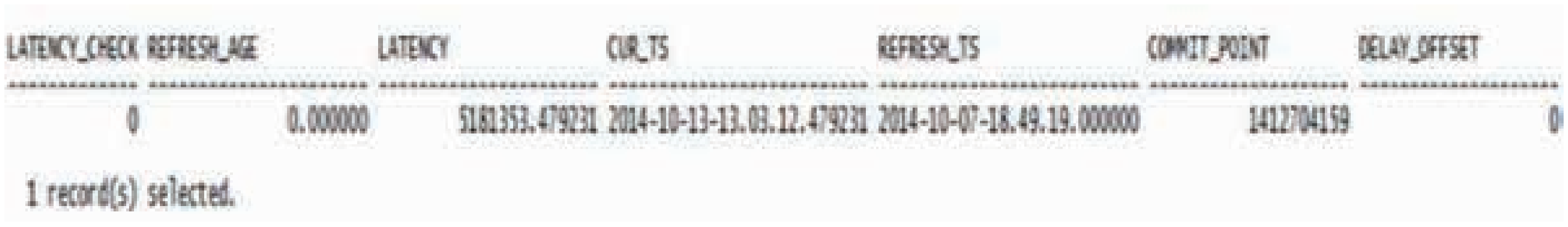
Shadow table routing will be enabled if appropriate for every connection

Also, create your own view to monitor latency and refresh age relation in the database.



```
create or replace view shadow_latency as
    (select case when latency <= refresh_age then 1 else 0 end as latency_check,
           v2.*
    from table (select current refresh age as refresh_age,
                       cur_ts - refresh_ts as latency,
                       v1.*
             from table (select current timestamp as cur_ts,
                                (timestamp('1970-01-01')+(commit_point-delay_offset) seconds)+current timezone
                                 as refresh_ts,
                                 commit_point, delay_offset
                    from systools.repl_mqt_latency)
             as v1)
    as v2);
```

Example output:



```
LATENCY_CHECK REFRESH_AGE            LATENCY                CUR_TS                     REFRESH_TS                 COMMIT_POINT         DELAY_OFFSET
------------- ---------------------- ---------------------- -------------------------- -------------------------- -------------------- --------------------
            0               0.000000         5181353.479231 2014-10-13-13.03.12.479231 2014-10-07-18.49.19.000000           1412704159                    0

  1 record(s) selected.
```

As per above view definition:

Latency_Check: 1 means current replication latency is shorter than the specified current refresh age value, 0 means replication latency is longer than the specified current refresh age.

Latency: Latency value is calculated as per above view definition.

Commit_Point: A timestamp that represents the last time a commit was issued after applying changes to target shadow tables. It is the time at which the DELAY_OFFSET value is generated. It is a column of the SYSTOOLS.REPL_MQT_LATENCY table.

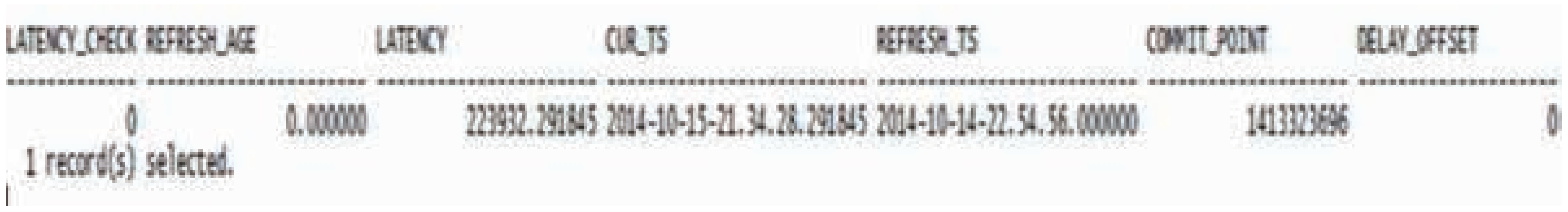
DELAY_OFFSET: the number of seconds between the time at which the source table data is read and the last time that applied changes to target shadow tables were committed. It is a column of the SYSTOOLS.REPL_MQT_LATENCY table

Routing Scenarios

Latency-based routing is a performance improvement technique that directs a query to a column-organised shadow table when the replication latency is within a user-defined threshold (current refresh age register in this case). Below we discuss 4 different scenarios and corresponding example outputs when the query is executed using the row-organised tables or executed using the shadow tables.

CURRENT REFRESH AGE=0 – never route to shadow tables ignoring the query structure and latency.

From shadow_latency_view:



```
LATENCY_CHECK REFRESH_AGE          LATENCY              CUR_TS                     REFRESH_TS                 COMMIT_POINT         DELAY_OFFSET
------------- -------------------- -------------------- -------------------------- -------------------------- -------------------- --------------------
            0             0.000000         223932.291845 2014-10-15-21.34.28.291845 2014-10-14-22.54.56.000000           1413323696                    0
  1 record(s) selected.
```

From EXPLAIN EXFMT output:

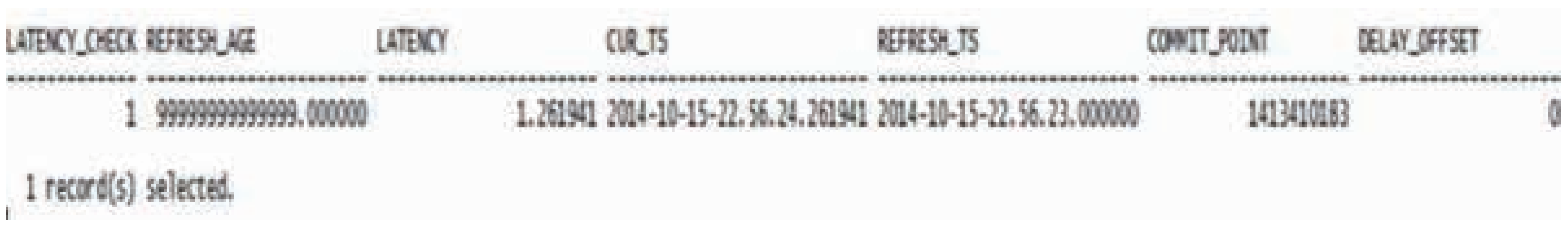
```
Extended Diagnostic Information:

--------------------------------

Diagnostic Identifier:  1

Diagnostic Details:     EXP0054W  The materialized query table “CQWROW  “.

                        “ITEM_SHADOW” was not considered for rewrite

                        matching because the CURRENT REFRESH AGE special

                        register is set to zero.

Diagnostic Identifier:  2

Diagnostic Details:     EXP0054W  The materialized query table “CQWROW  “.

                        “STORE_SALES_SHADOW” was not considered for rewrite

                        matching because the CURRENT REFRESH AGE special

                        register is set to zero.
```

CURRENT REFRESH AGE=ANY – always route to shadow tables ignoring the query structure and latency

You can use the InfoSphere CDC acceptable_latency_in_seconds_for_column_organized_tables parameter to increase the size of transactions against shadow tables by grouping. Larger transactions delay the commits. The acceptable_latency_in_seconds_for_column_organized_tables system parameter controls the interval between batched commits when InfoSphere CDC replication is active. InfoSphere CDC temporarily buffers transactions during replication in order to minimise the impact to the target database when applying to column-organised tables. This buffering can increase latency. If the latency reaches the point indicated by this system parameter InfoSphere CDC will stop buffering the data. Note that InfoSphere CDC will perform this buffering whenever there is at least one column-organised table being targeted in a subscription and that this buffering affects all the tables in the subscription.

```
acceptable_latency_in_seconds_for_column_organized_tables=5
```

```
db2 set current refresh age ANY
```



```
LATENCY_CHECK REFRESH_AGE           LATENCY               CUR_TS                     REFRESH_TS                 COMMIT_POINT         DELAY_OFFSET
------------- --------------------- --------------------- -------------------------- -------------------------- -------------------- --------------------
            1 99999999999999.000000              1.261941 2014-10-15-22.56.24.261941 2014-10-15-22.56.23.000000           1413410183                    0

  1 record(s) selected.
```

From EXPLAIN EXFMT output:

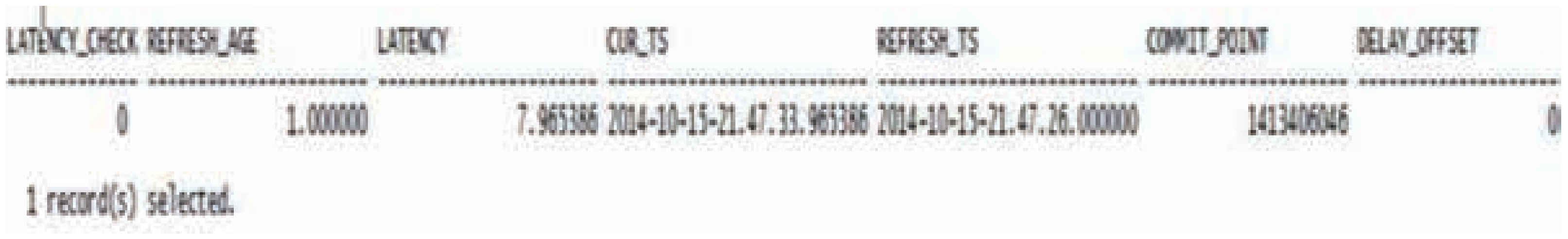
```
Diagnostic Identifier:  3
Diagnostic Details:     EXP0149W  The following MQT was used (from those
                        considered) in query matching: "CQWROW  ".
                        "ITEM_SHADOW".
Diagnostic Identifier:  4
Diagnostic Details:     EXP0149W  The following MQT was used (from those
                        considered) in query matching: "CQWROW  ".
                        "STORE_SALES_SHADOW".
```

CURRENT REFRESH AGE=hhmmss – route to shadow tables only if latency is shorter than specified refresh age

Set the refresh age to a high value of 30 minutes (3000).

From EXPLAIN EXFMT output

```
Diagnostic Identifier:  3
Diagnostic Details:     EXP0149W  The following MQT was used (from those
                        considered) in query matching: "CQWROW  ".
                        "ITEM_SHADOW".
Diagnostic Identifier:  4
Diagnostic Details:     EXP0149W  The following MQT was used (from those
                        considered) in query matching: "CQWROW  ".
                        "STORE_SALES_SHADOW".
```



```
LATENCY_CHECK REFRESH_AGE           LATENCY               CUR_TS                     REFRESH_TS                 COMMIT_POINT         DELAY_OFFSET
------------- --------------------- --------------------- -------------------------- -------------------------- -------------------- --------------------
            0              1.000000              7.965386 2014-10-15-21.47.33.965386 2014-10-15-21.47.26.000000           1413406046                    0

  1 record(s) selected.
```

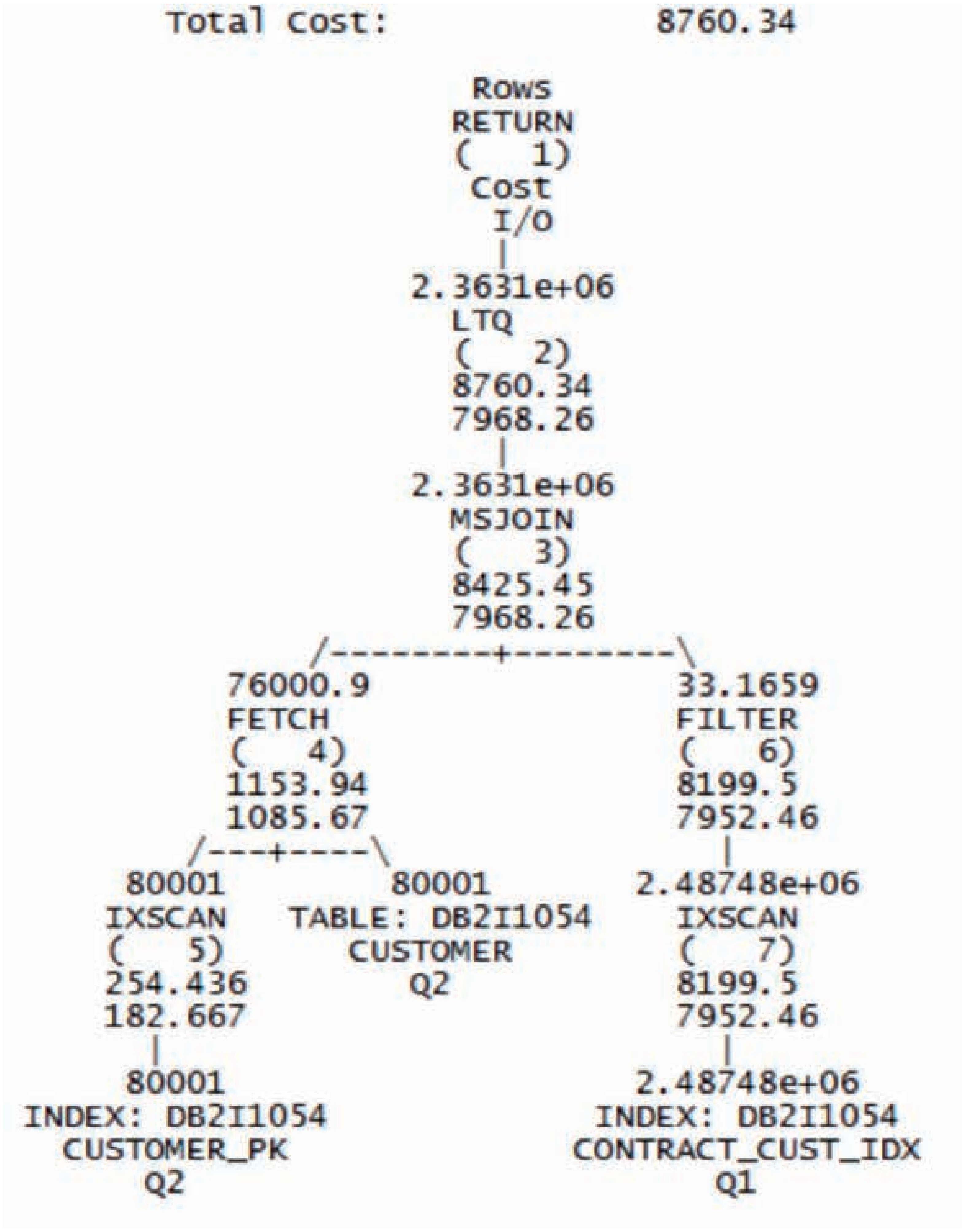
From EXPLAIN EXFMT output

```
Diagnostic Identifier:  1

Diagnostic Details:     EXP0087W  No materialized query table matching was

                        performed on the statement during query rewrite

                        because there is a replication-maintained

                        materialized query table defined on one of the

                        tables in the query and the current replication

                        latency is larger than the time value specified

                        in the CURRENT REFRESH AGE special register.

Diagnostic Identifier:  2

Diagnostic Details:     EXP0087W  No materialized query table matching was

                        performed on the statement during query rewrite

                        because there is a replication-maintained

                        materialized query table defined on one of the

                        tables in the query and the current replication

                        latency is larger than the time value specified

                        in the CURRENT REFRESH AGE special register.
```

Sort Memory Side Effect

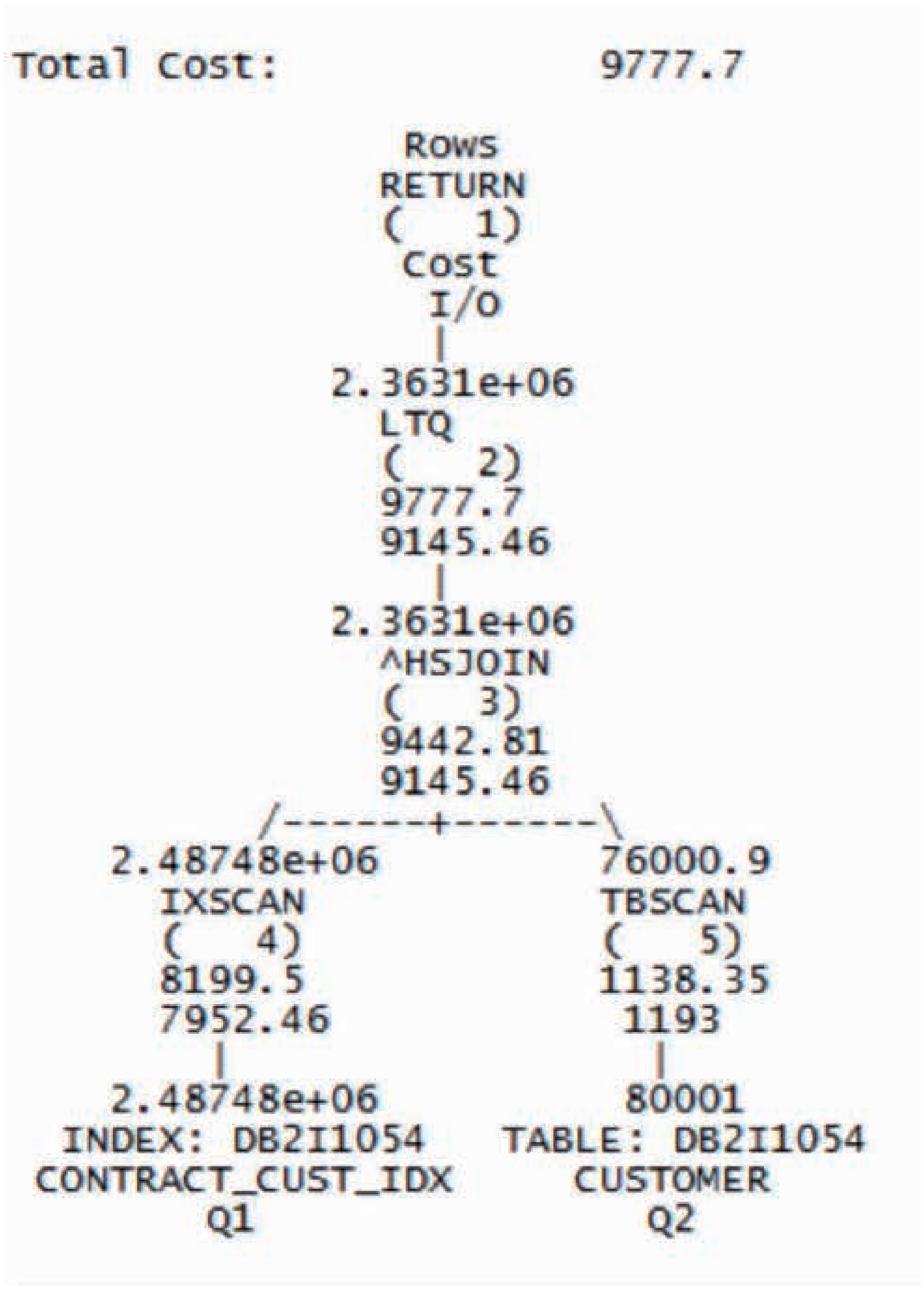
Column-organised data processing (and therefore shadow table data processing) requires very large allocations for SORTHEAP and SHEAPTHRES_SHR parameters. The values might be larger than those that are typically used in OLTP configurations. Although a larger SHEAPTHRES_SHR value is expected for any system that uses column-organised tables, a much larger SORTHEAP value could potentially cause access plan changes for some existing OLTP queries, depending on predicate and statistical distribution. For example, with a small SORTHEAP value related to OLTP transactions, the optimiser correctly chooses NLJOIN (nested loop join) or MSJOIN (merge sort join) because HSJOIN (hash sort join) is estimated to have significant sort spilling and those OLTP transactions are having good response times. But, with a very high SORTHEAP value, the optimizer may think there is no spilling and hence wrongly decide HSJOIN is cheaper, causing the OLTP queries to take almost twice as much time to execute. We noticed in some of our queries that access plans were using MSJOIN and index scan before the SORTHEAP was increased. After the SORTHEAP increase, the access plans got changed and the optimiser decided to use HSJOIN and a table scan. This meant that in order to maintain the SLAs of the OLTP queries we would have to place the columnar tables in a separate database which is not ideal. Here is an example of this scenario:

With a small OLTP related SORTHEAP value (1024, for example) our OLTP query was using the following access plan:



```
      Total Cost:             8760.34

                        Rows
                       RETURN
                       (   1)
                        Cost
                         I/O
                         |
                     2.3631e+06
                       LTQ
                       (   2)
                       8760.34
                       7968.26
                         |
                     2.3631e+06
                       MSJOIN
                       (   3)
                       8425.45
                       7968.26
                /--------+--------\
           76000.9                 33.1659
           FETCH                   FILTER
           (   4)                  (   6)
           1153.94                 8199.5
           1085.67                 7952.46
          /---+----\                 |
      80001        80001          2.48748e+06
     IXSCAN   TABLE: DB2I1054       IXSCAN
     (   5)       CUSTOMER          (   7)
     254.436         Q2             8199.5
     182.667                        7952.46
       |                              |
      80001                       2.48748e+06
  INDEX: DB2I1054              INDEX: DB2I1054
    CUSTOMER_PK               CONTRACT_CUST_IDX
       Q2                            Q1
```

Notice that MSJOIN and index scan are being used. We now wanted to use shadow (columnar) tables. However, trying to access the shadow tables resulted in a SQL0955C sort heap error. We then increased SHEAPTHRES_SHR to 611049 and SORTHEAP to 32768. The access plan now changed and the OLTP query took longer to complete:



```
Total Cost:                9777.7

                   Rows
                  RETURN
                  (   1)
                   Cost
                    I/O
                    |
                2.3631e+06
                  LTQ
                  (   2)
                  9777.7
                  9145.46
                    |
                2.3631e+06
                  ^HSJOIN
                  (   3)
                  9442.81
                  9145.46
            /------+------\
     2.48748e+06          76000.9
       IXSCAN              TBSCAN
       (   4)              (   5)
       8199.5              1138.35
       7952.46              1193
         |                   |
     2.48748e+06           80001
   INDEX: DB2I1054     TABLE: DB2I1054
  CONTRACT_CUST_IDX       CUSTOMER
         Q1                  Q2
```

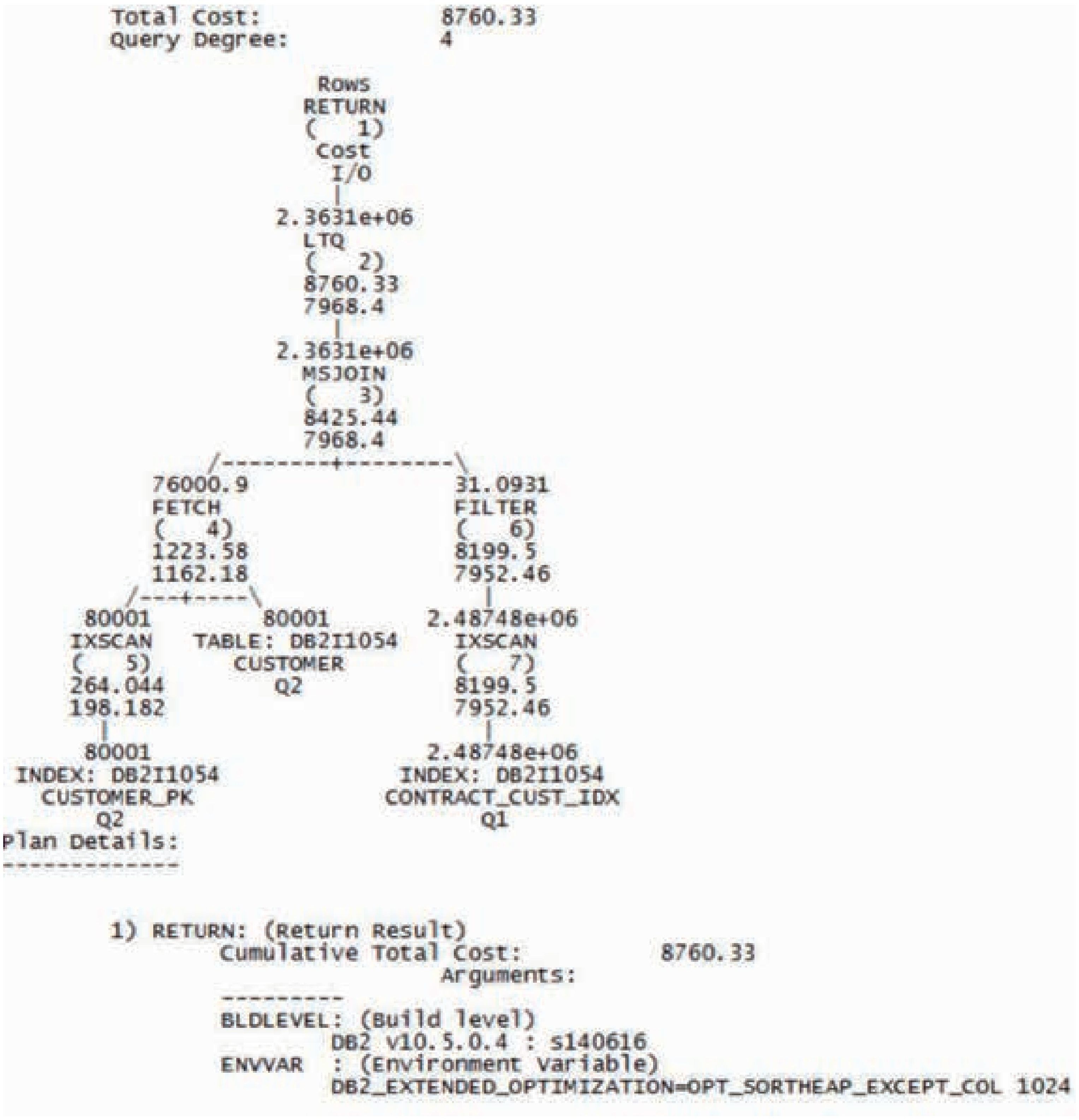
Notice that HSJOIN and table scan are now being used.

Using the DB2 registry variable DB2_EXTENDED_OPTIMIZATION, we can specify a lower SORTHEAP value (may be the original SORTHEAP value before it was increased to accommodate columnar tables) and force the optimizer to use that value for OLTP queries accessing row-based tables.

Note: DB2 will not use STMM in this case. OLTP sortheap value will be fixed.

db2set DB2_EXTENDED_OPTIMIZATION=”OPT_SORTHEAP_EXCEPT_COL 1024”

If the query is not using any column organised tables then the optimiser can decide to ignore the database level large SORTHEAP value and to use instead the value specified in this registry setting (1024 in this example). This is illustrated by the access plan below:

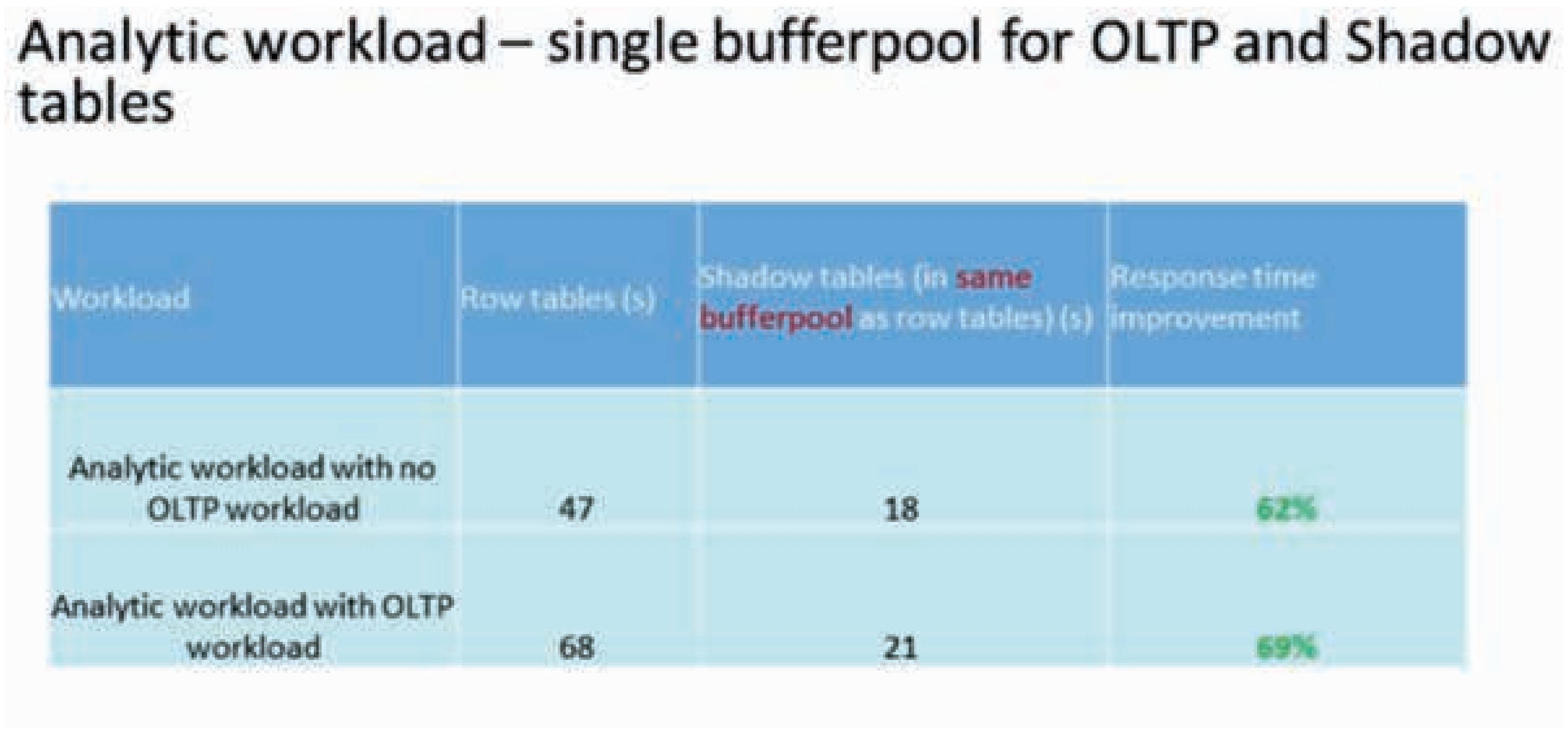


```
        Total Cost:                8760.33
        Query Degree:              4

                          Rows
                         RETURN
                         (   1)
                          Cost
                           I/O
                           |
                       2.3631e+06
                         LTQ
                         (   2)
                         8760.33
                         7968.4
                           |
                       2.3631e+06
                         MSJOIN
                         (   3)
                         8425.44
                         7968.4
                /--------+--------\
           76000.9                  31.0931
           FETCH                    FILTER
           (   4)                   (   6)
           1223.58                  8199.5
           1162.18                  7952.46
         /---+----\                    |
      80001        80001          2.48748e+06
      IXSCAN   TABLE: DB2I1054      IXSCAN
      (   5)      CUSTOMER          (   7)
      264.044        Q2             8199.5
      198.182                       7952.46
        |                              |
      80001                       2.48748e+06
   INDEX: DB2I1054             INDEX: DB2I1054
     CUSTOMER_PK               CONTRACT_CUST_IDX
        Q2                            Q1
Plan Details:
-------------

        1) RETURN: (Return Result)
                Cumulative Total Cost:          8760.33
                                Arguments:
                ---------
                BLDLEVEL: (Build level)
                        DB2 v10.5.0.4 : s140616
                ENVVAR  : (Environment Variable)
                        DB2_EXTENDED_OPTIMIZATION=OPT_SORTHEAP_EXCEPT_COL 1024
```

Shadow Tables – In Action

In this section, we present results we obtained by running typical analytic/reporting workloads (joins, aggregation, etc.) in our OLTP database which has the CONTRACT_TYPE, CUSTOMER, and CONTRACT row organised tables and their corresponding shadow tables.

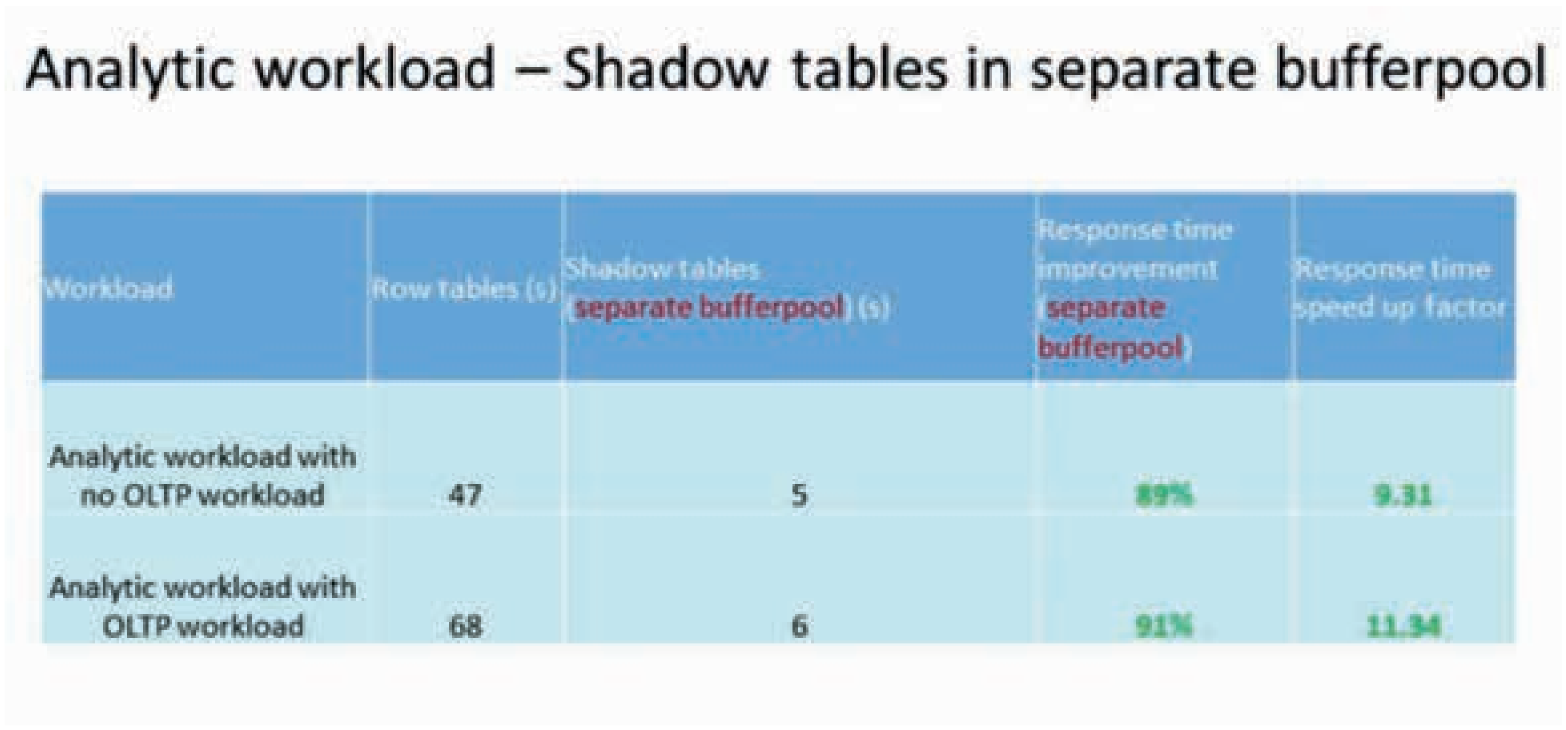
1. Performance of analytic workload using 3 tables above with no other OLTP workload
    a. On row tables (before CDC is active)
    b. On shadow tables (with row and shadow tables in the same bufferpool)
    c. On shadow tables (with row and shadow tables in separate bufferpools)
2. Performance of same analytic workload with OLTP workload
    a. Repeat 1a) – 1c)



Analytic workload – single bufferpool for OLTP and Shadow tables

Workload	Row tables (s)	Shadow tables (in same bufferpool as row tables) (s)	Response time improvement
Analytic workload with no OLTP workload	47	18	62%
Analytic workload with OLTP workload	68	21	69%

We can see here that using shadow tables has really reduced the response time for our analytic workload dramatically.

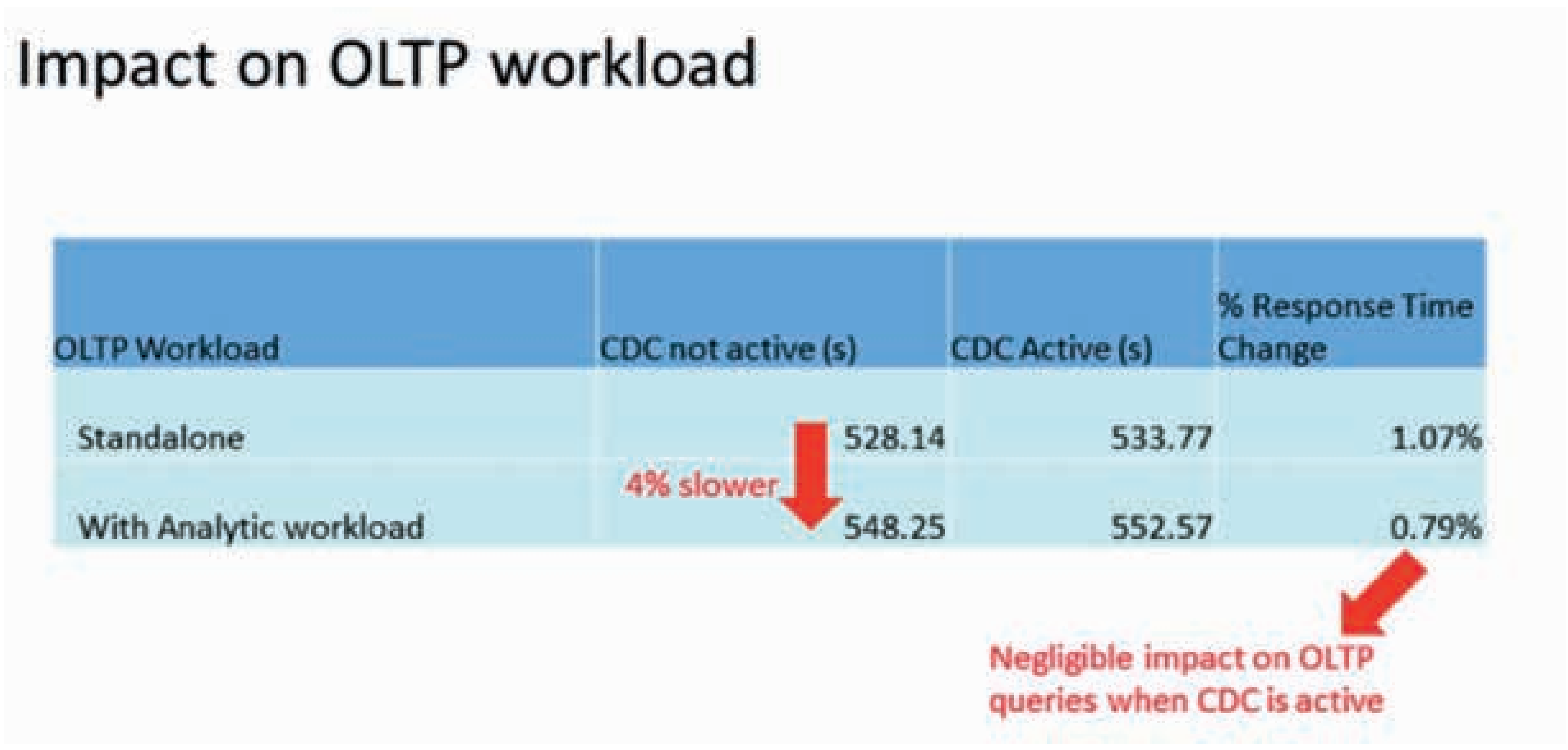


Analytic workload – Shadow tables in separate bufferpool

Workload	Row tables (s)	Shadow tables (separate bufferpool) (s)	Response time improvement (separate bufferpool)	Response time speed up factor
Analytic workload with no OLTP workload	47	5	89%	9.31
Analytic workload with OLTP workload	68	6	91%	11.34

Bufferpools are critical to database performance. Since OLTP and analytic workloads have different access patterns, it is best practice to keep separate tablespaces and bufferpools for OLTP and shadow tables to avoid any contention. Here we can see that using two separate bufferpools have further reduced the response time as we have expected.

3. Impact on OLTP workload – before and after CDC is active
4. Impact on OLTP workload when analytic workload also taking place – before and after CDC is active.

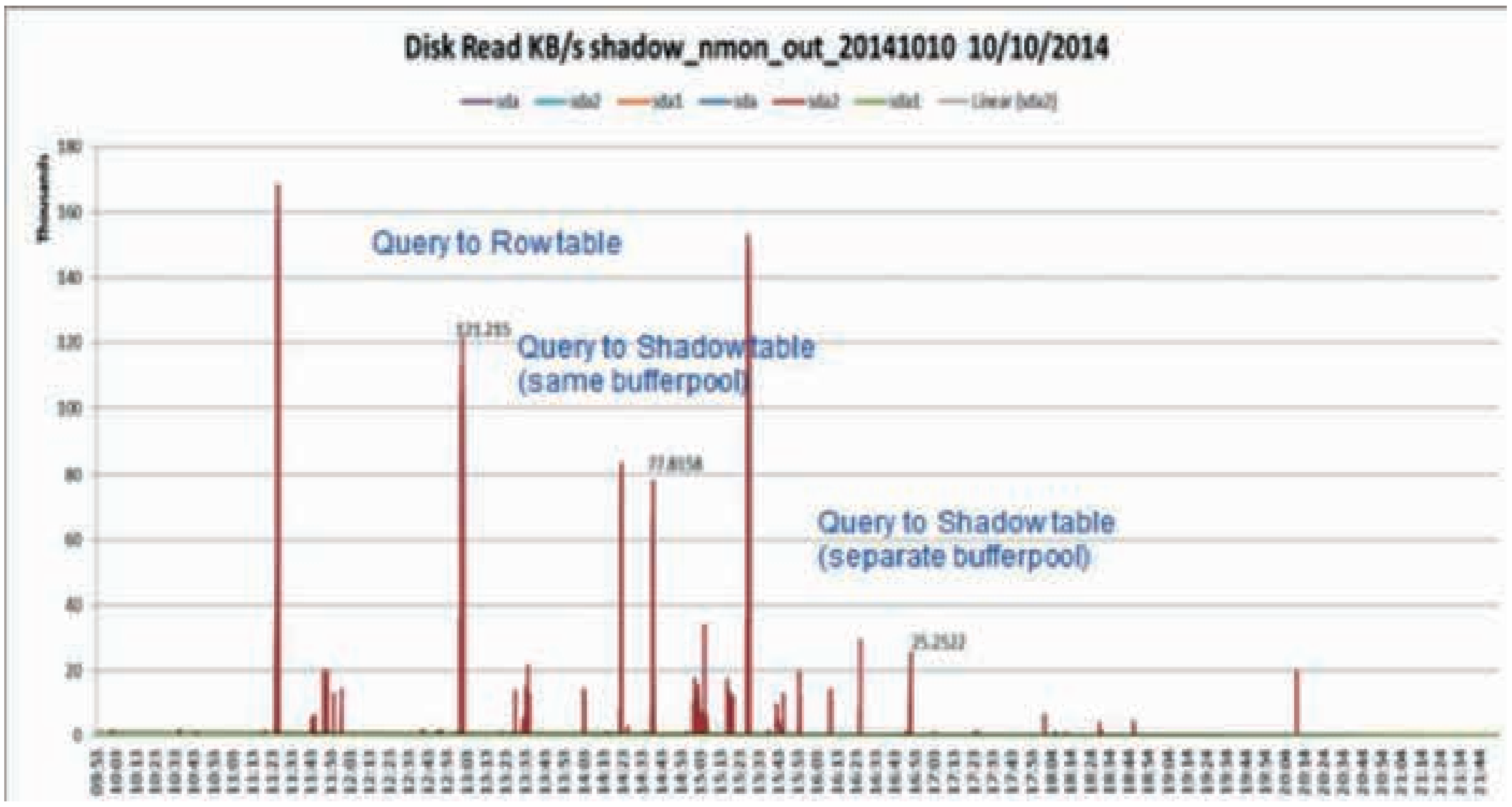


Impact on OLTP workload

OLTP Workload	CDC not active (s)	CDC Active (s)	% Response Time Change
Standalone	528.14	533.77	1.07%
With Analytic workload	548.25	552.57	0.79%

4% slower
Negligible impact on OLTP queries when CDC is active

System level overhead (CPU/Memory/Disk) to maintain CDC and Shadow table replication

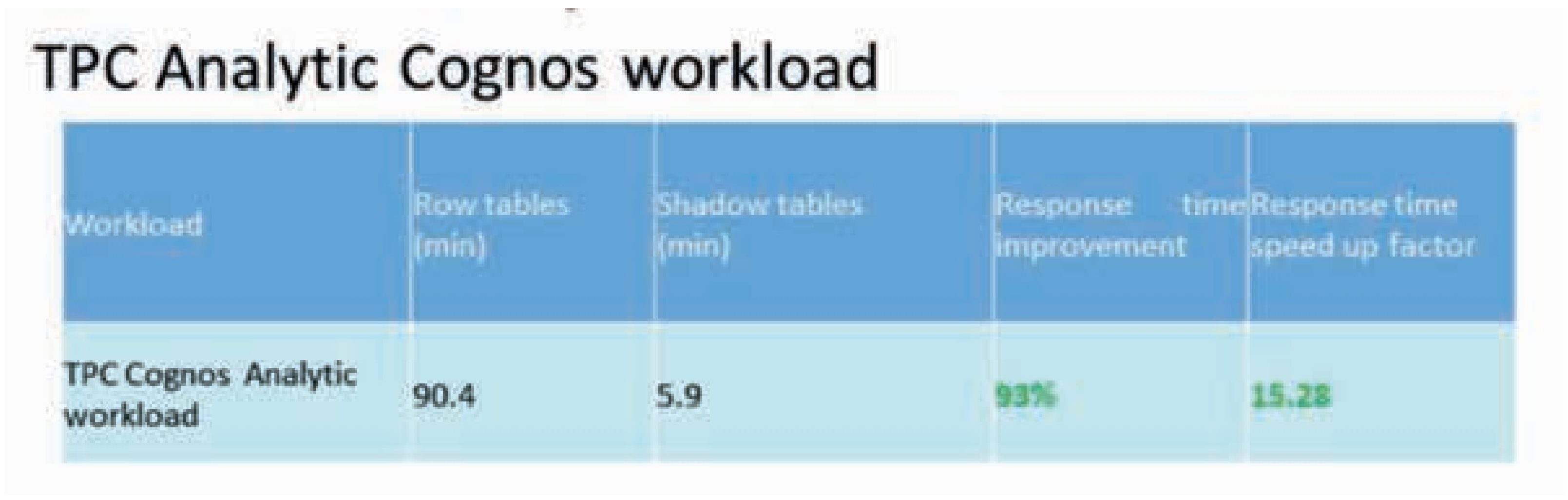


As you could see above disk read rates decreased as the access path got routed to shadow table and then decreased further when separate bufferpools were used as you would expect.

We then looked at a more complex TPC Analytic Cognos workload to see whether running reports on OLTP tables would have a faster response time with shadow tables in place. The workload comprised of the following 13 reports:

1. Sales by Category and Classification
2. Weekly Sales by Category by Location
3. Store Net Profit Margin
4. Net Profit Margin by Store by 4th Quarter
5. Gross Profit Across All Stores
6. Sales of Electronics Across All Stores
7. Sales of Electronics at Individual Stores
8. Weekly Electronic Sales
9. Gross Store Profit in 4Q of 2001 and 2002
10. Total Store Revenue
11. Total Store Revenue in 4Q of 2001 and 2002
12. Sales by Month in 3Q/4Q of 2002
13. Wholesale Cost of Electronics by Month in 2002

The results were amazing as seen in the table below:



TPC Analytic Cognos workload

Workload	Row tables (min)	Shadow tables (min)	Response time improvement	Response time speed up factor
TPC Cognos Analytic workload	90.4	5.9	93%	15.28

The whole workload had a 93% improvement in response time with a 15x speed up factor. Individual queries in the workload showed a speed up factor from 2x to 276x!

Shadow Tables Hints and Tips

In this section we summarise all the hints and tips as we found in our usage of shadow tables. Some of these are already covered in earlier sections, but included them here for completeness.

- DB2_WORKLOAD=ANALYTICS must *not* be set in the instance because the shadow tables are generally used in an environment where the predominant workload is still OLTP.
- Shadow table columns can be a subset of row table columns, excluding those columns like CLOB, BLOB etc not yet available in columnar technology as mentioned before. So, you should include only needed keys and interested measures.
- Need to project a primary or unique key. Create shadow table primary key corresponding to row-organised table’s projected primary / unique key. This is required for one-to-one mapping between row and corresponding shadow table for the CDC replication. Also, it enables new optimisation for point update/delete via underlying index.
- Each row-organised table referenced in a query must have a shadow table. This is a requirement for the optimiser. Otherwise, the query will not be routed to shadow tables.
- Shadow table routing is not available queries with RS, RR isolation level support.
- Available at optimization level 2 or >= 5 (default is 5).
- Shadow table routing is available for dynamic query only.
- auto_runstats and auto_reorg = on
- Dedicate tablespace and buffer pool for shadow tables separate from row-organised tables. This is a fundamental best practice because row tables and shadow tables have different access patterns and table organization and different workload. This is substantiated by the results in the previous Section

(Shadow Tables – In Action), where using a separate bufferpool for shadow tables gave an extra performance boost over using single shared bufferpool.

- Use recommended table space attributes for column-organised table – pagesize=32K and extent_size=4
- Minimal impact to OLTP transactions via asynchronous maintenance
  - Capture engine scrapes DB2 logs for deltas
  - Apply engine consolidates updates to shadow tables. Leverages new DB2 Cancun Release 10.5.0.4 index scan driven updates
- CDC Replication needs a single subscription per database containing all the shadow tables.
- The very first refresh after the subscription mirroring is started CDC uses the technology of load utility under the hood to copy the content of each source row-organised table into the corresponding shadow table. To address the resource needs of the LOAD command, set the UTIL_HEAP_SZ database configuration parameter to an appropriately high starting value and AUTOMATIC.
- CDC parameter acceptable_latency_in_seconds_for_column_organized_tables value should be less than the DB2 database refresh age special register value.
- Shadow tables are powered by BLU Acceleration and automatically benefit from inherent BLU extreme compression. We saw 75%-90+% compression savings for shadow tables in our results. Needless to say, we should create shadow tables only for relevant OLTP tables where reports need to be run. Additionally, as we have mentioned before, shadow tables can be created using only a subset of columns that are necessary for reporting from the source table. This would further reduce the storage footprint of shadow tables.

Shadow Tables and HADR

Columnar tables were not replicated to the HADR standby database before FP4. This prevented a true DR solution. With FP4 Cancun Release columnar tables can now be replicated to the standby database. This means shadow tables (which are column-organised) can be replicated to a standby server using HADR (though we still cannot read any column-organised tables in Read-Only-Standby as of FP4).

Extra consideration needs to be taken into account when deploying shadow tables in an HADR environment. From a topological point of view, a separate CDC instance is required on each of the Primary and Standby server. On the Primary server, the CDC instance is active to replicate the changes from the row-organised tables to the shadow tables.

To HADR, a shadow table is treated like any other column-organised table; hence HADR will replicate updates to shadow tables from the Primary to the Standby. Given the shadow tables are maintained by HADR on the Standby, the CDC instance should not be activated on the Standby.

In this setup, it is recommended to put the CDC Access Sever on a separate system so that it is not impacted on failover.

Shared log archiving between the Primary and Standby is required to ensure that CDC log capture has access to all of the log files on failover.

With CDC, there are two types of metadata that are used to synchronise the source and target tables:

- The operational metadata (which is information such as the instance signature and bookmarks) is stored in the database and is replicated by HADR to the standby.
- The configuration metadata (which is information such as subscriptions and mapped tables) is stored in the CDC home directory which is outside of the database, so it is not replicated by HADR. Therefore, after you have set up shadow tables with HADR, any configuration metadata changes on the Primary also need to be reflected on the Standby. CDC provides a dmbackup command to backup the configuration metadata on the Primary to be copied over to the Standby.

After a failover or role switch, and the HADR primary role is active on the Standby server, CDC needs to be manually started to maintain the shadow tables on the new Primary.

Summary and Conclusion

By bringing analytics to data, shadow tables address the needs for OLTAP customers by providing the following value propositions:

- Achieve significant performance for reports currently running on your OLTP system using Shadow tables.
- Improve OLTP performance by remove indexes previously used for reporting.
- Simplify infrastructure by moving reports running on another system to your OLTP system.

Appendix

Appendix A: Installation and Configuration of Infosphere CDC

Shadow tables are maintained by IBM InfoSphere Change Data Capture for DB2 (InfoSphere CDC), a component of the InfoSphere Data Replication product. InfoSphere CDC asynchronously replicates DML statements that are applied on the source table to the shadow table. By default, all applications access the source tables. Queries are automatically routed by the optimiser to the source table (row-organised) or the shadow table (column-organised copy of the source table) by using a latency-based algorithm that prevents applications from accessing the shadow table when the latency is beyond the user-defined limit. So, to implement shadow tables we will have to first install and configure Infosphere CDC. The install binaries are supplied as part of DB2 installs. Ideally you should install CDC in the same DB2 server where the database is existing. The high-level steps for installing and configuring CDC are (assuming DB2 10.5 Cancun FP4 instance is already installed):

Installing the IBM InfoSphere CDC Access Server

- First you will need to create a userid in the operating system for CDC access server, for example, db2iscdc.
- Then, unzip the access server install file and run the setup file.
- Use the default port number 10101 if it is available.
- Go to the install bin directory (for example - /home/db2iscdc/bin/AccessServer/bin), then start the access server by: `nohup ./dmaccessserver &`
- To shutdown the access server use the command: `./dmshutdownserver`
- Create an admin account inside access server, for example, `./dmcreateuser cdcadmin 'cdc access admin fullname' 'cdc access admin' pa$$w0rd sysadmin true false true`

Installing InfoSphere CDC instance for DB2 for LUW

- Make sure the database uses archived log mode and create a table schema DB2ISCDC (to hold CDC metadata tables).
- Create the SYSTOOLS.REPL_MQT_LATENCY table in the relevant DB2 database using the command:

```
db2 "CALL SYSPROC.SYSINSTALLOBJECTS('REPL_MQT', 'C', 'IBMDB2SAMPLEREL',CAST (NULL AS
VARCHAR(128)))"
```

- Login to db2iscdc and change directory to the install binary location and execute the setup file.
- Use the default port number 10901 for the 1st CDC instance (10902 for the 2nd instance and so on) if it is available.
- Enter the DB2 instance and the database name where the shadow tables will be created.
- Use ./dmconfigurets from the install bin folder to list status, delete, add and merge CDC instances.

- Use the following command from the install bin folder to start the CDC instance ISCDC1: `nohup ./dmts64 -I ISCDC1 &`
- Use the following command from the install bin folder to stop the CDC instance ISCDC1: `./dmshutdown -I ISCDC1`
- 4 DB2ISCDC schema tables are now created automatically: TS_AUTH, TS_BOOKMARK, TS_CONFAUD, TS_DDLAUD.
- Using dmset command set the configuration parameters for the CDC instance:
  - `staging_store_disk_quota_gb=1`
  - `maintain_replication_mqt_latency_table=true`
  - `acceptable_latency_in_seconds_for_column_organized_tables=mmss`
  - `global_max_batch_size=1024`
  - `mirror_auto_restart_interval_minutes=2`

Installing the IBM InfoSphere CDC Management Console

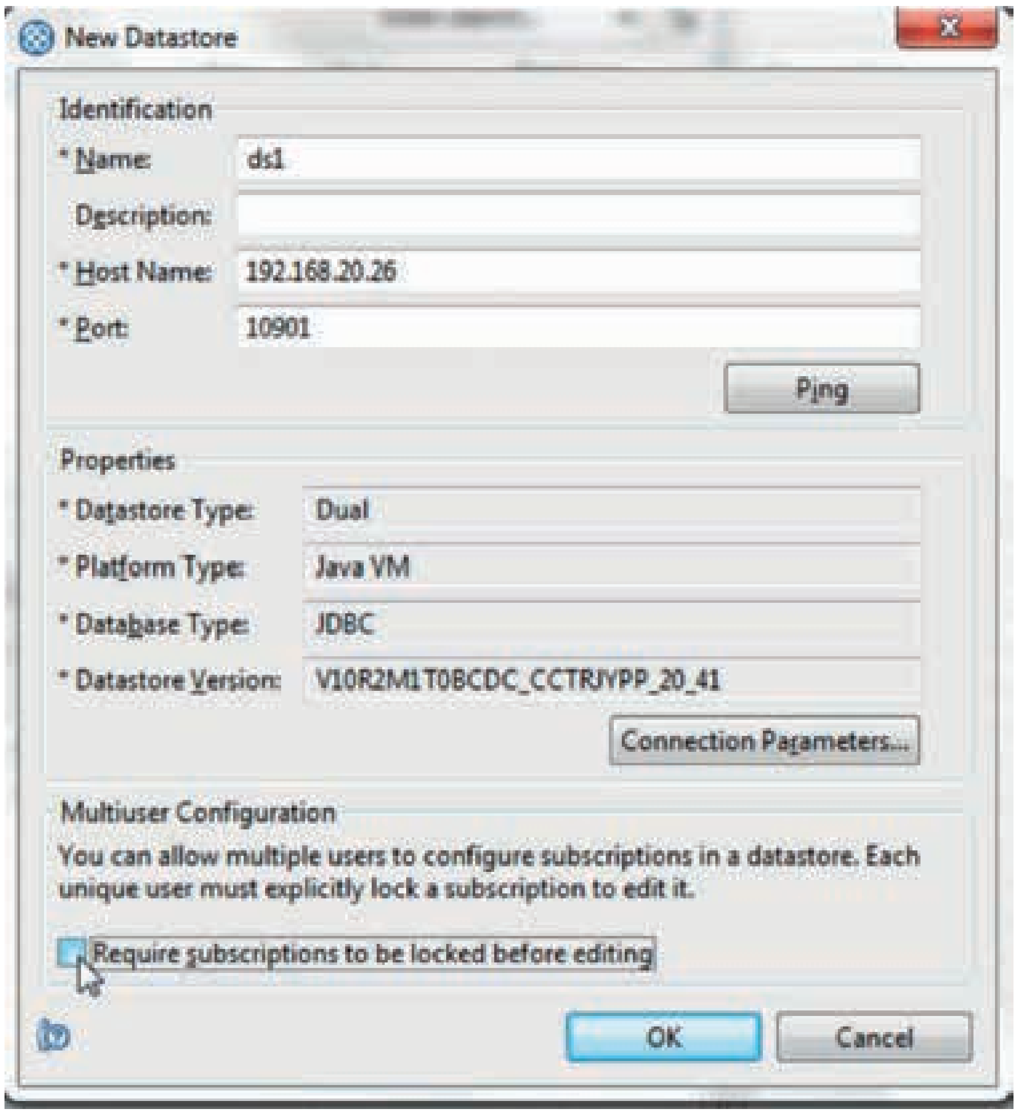
- This is a Windows based GUI client tool.
- The install files are supplied in the same place along with CDC Access server and CDC instance installs.
- Unzip the install zip file and then execute the setup.exe file.
- Login to the management console using the access server login cdcadmin.

Appendix B: InfoSphere CDC subscription and table mappings configuration

Configuring a data store

These can be done using CDC access server command line interface, but can be done through Management Console. Make sure that the corresponding CDC instance (example ISCDC1) is running.

Configure a single data store that identifies the replication engine to the access server.


New Datastore

Identification

* Name:	ds1
Description:	
* Host Name:	192.168.20.26
* Port:	10901

Properties

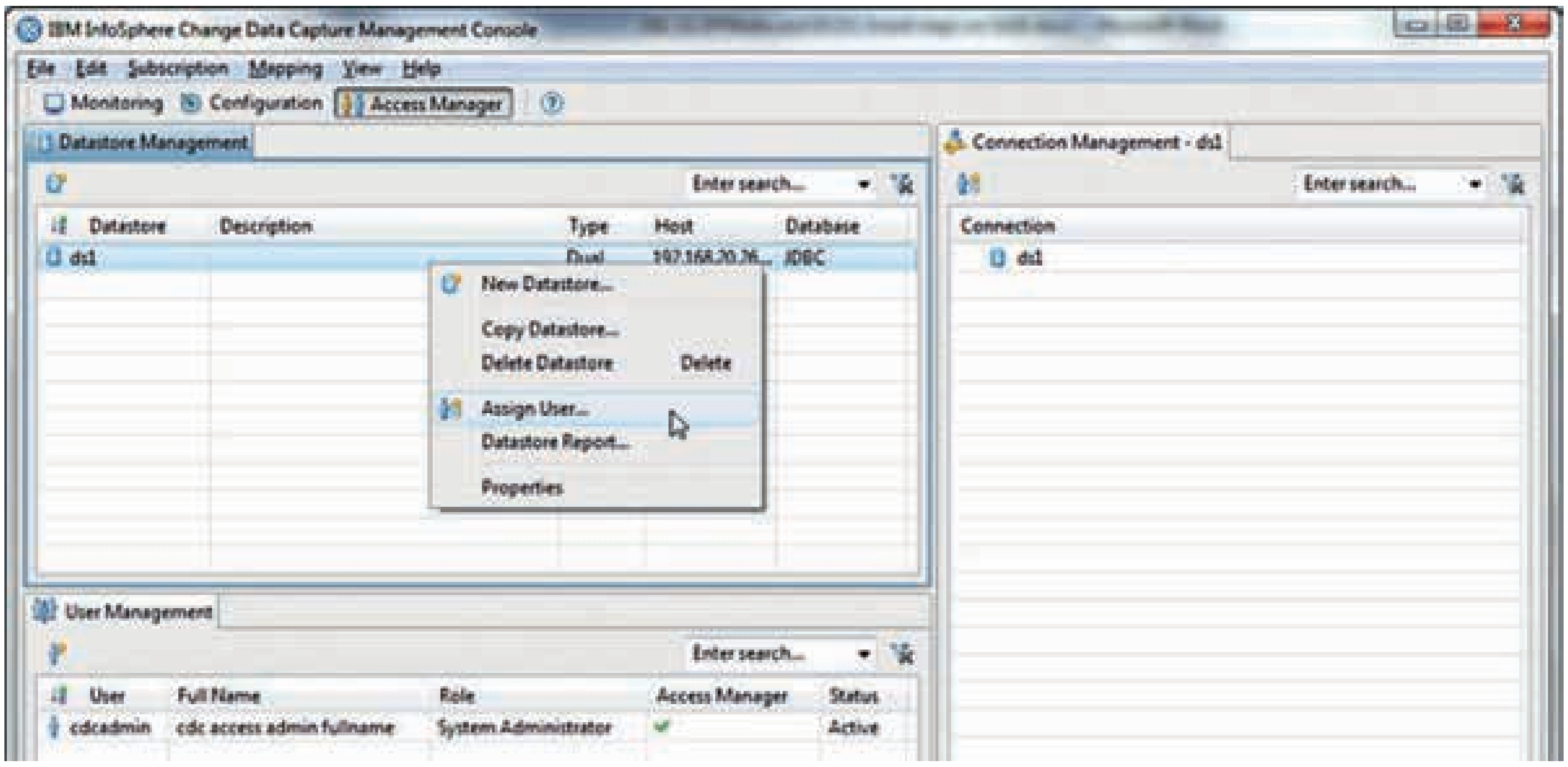
* Datastore Type:	Dual
* Platform Type:	Java VM
* Database Type:	JDBC
* Datastore Version:	V10R2M1T0BCDC_CCTRJYPP_20_41

Multiuser Configuration

You can allow multiple users to configure subscriptions in a datastore. Each unique user must explicitly lock a subscription to edit it.

☐ Require subscriptions to be locked before editing

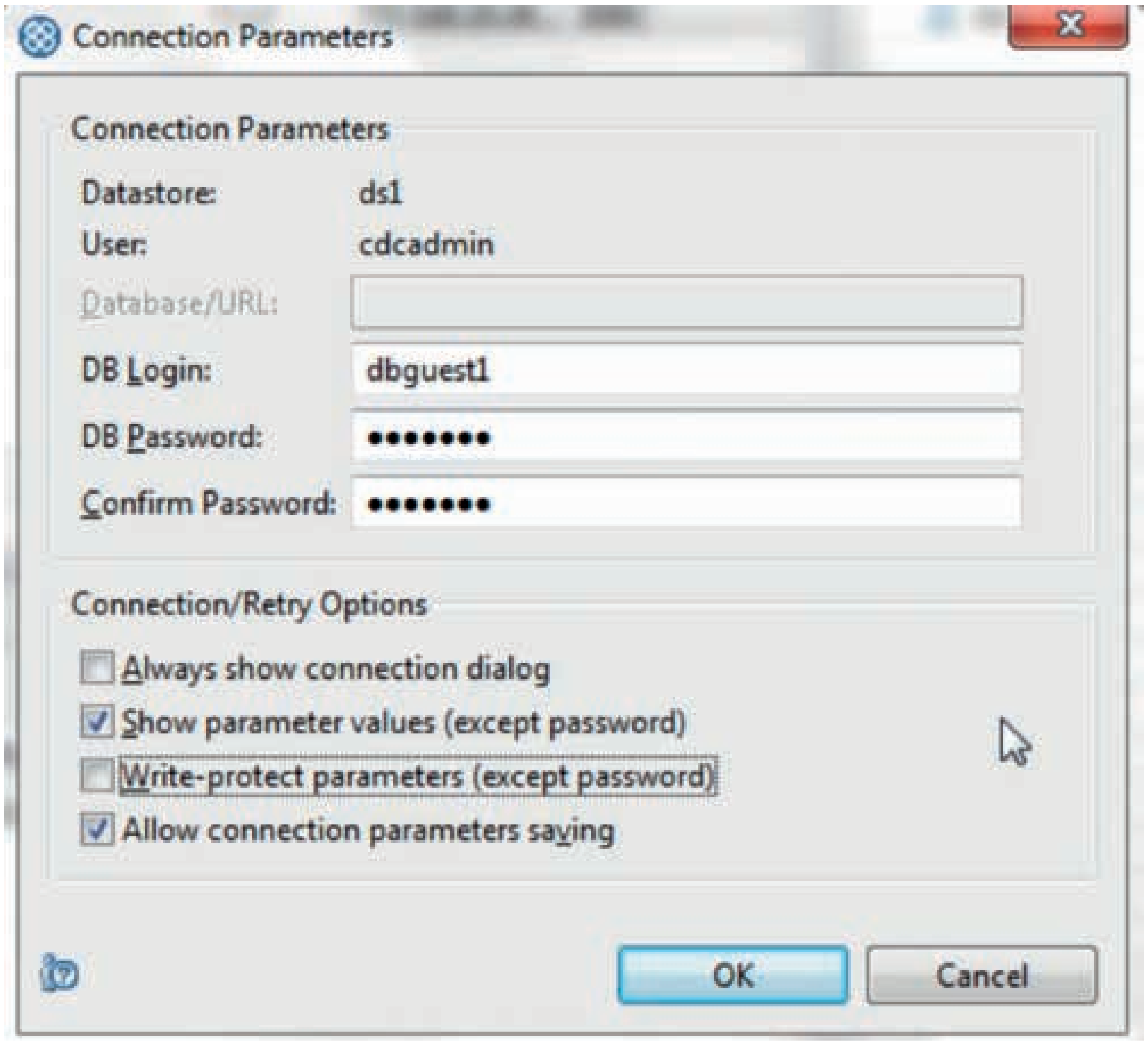
- Then configure a data store connection that maps the data store to an authorisation ID.


IBM InfoSphere Change Data Capture Management Console
File Edit Subscription Mapping View Help
Monitoring Configuration Access Manager
Datastore Management
Connection Management - ds1
Enter search...
New Datastore...
Copy Datastore...
Delete Datastore Delete
Assign User...
Datastore Report...
Properties
User Management

Datastore	Description	Type	Host	Database
ds1		Dual	192.168.20.76...	JDBC

Connection
ds1

User	Full Name	Role	Access Manager	Status
cdcadmin	cdc access admin fullname	System Administrator	✓	Active



Datastore:	ds1
User:	cdcadmin
Database/URL:	
DB Login:	dbguest1
DB Password:	•••••••
Confirm Password:	•••••••

- ☐ Always show connection dialog
- ☑ Show parameter values (except password)
- ☐ Write-protect parameters (except password)
- ☑ Allow connection parameters saying

```
db2iscdc@Monty:~/bin/AccessServer/bin> ./dmlistdatastores

Datastore       Hostname    Port OS     DB    Version

---------------------- --------------- ----- ---------- --------- --------------------
----

    ds1          192.168.20.26  10901 Java VM  JDBC
    V10R2M1T0BCDC_CCTRJYPP_20_41
```

Creating CDC Subscriptions

A CDC subscription is a group of table mappings and is a logical unit for replication. A subscription provides a single point of control for common operations such as start mirroring, stop mirroring, or start refresh for a set of tables that must be maintained at the same time.

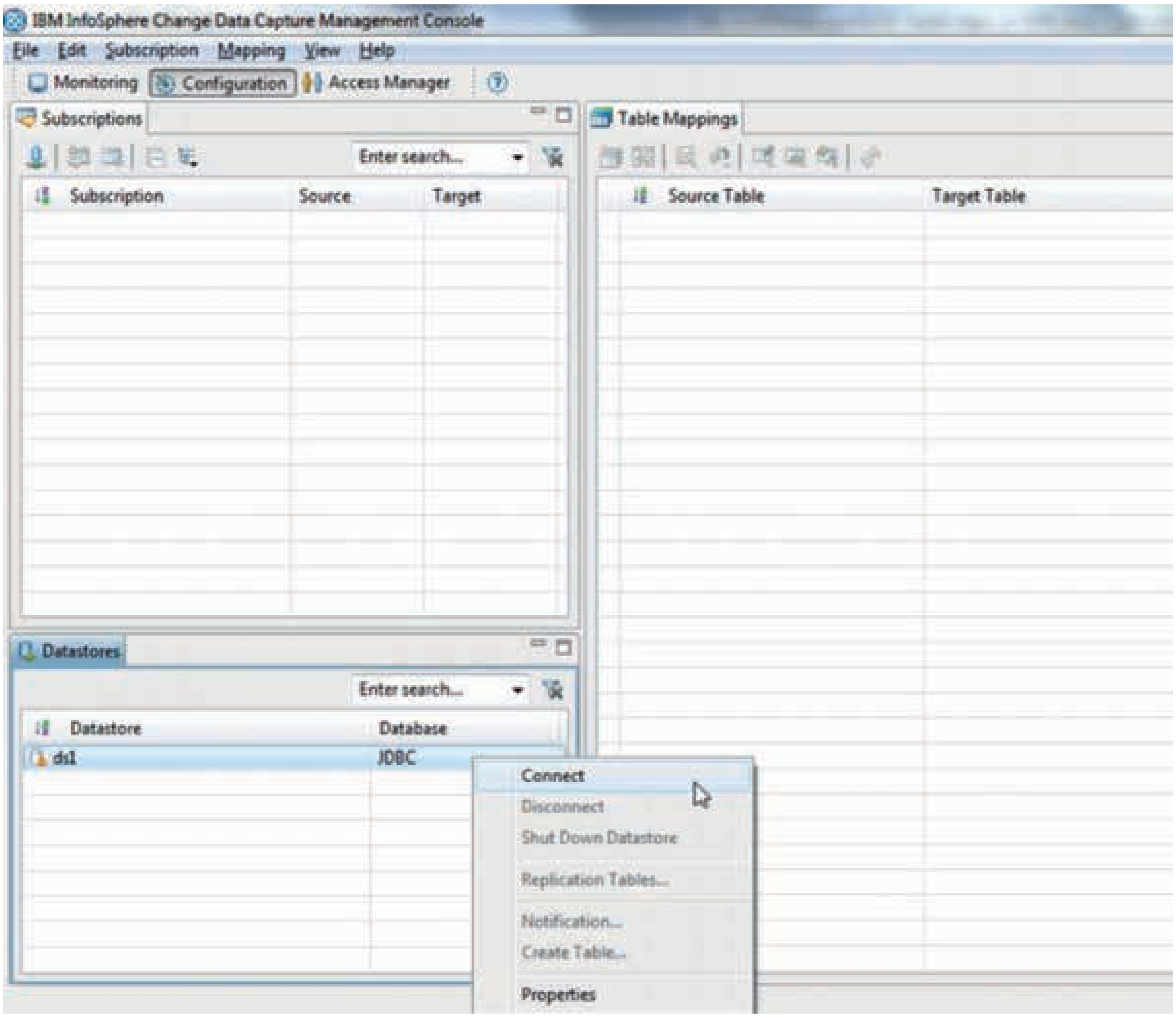
Mirroring is the process of continuous replication of changed data from the source system to the target system, whereas refresh is the process that synchronizes the target table with the current contents of the source table.

The source and target tables with defined primary keys, and a configured data store must already exist. Important:

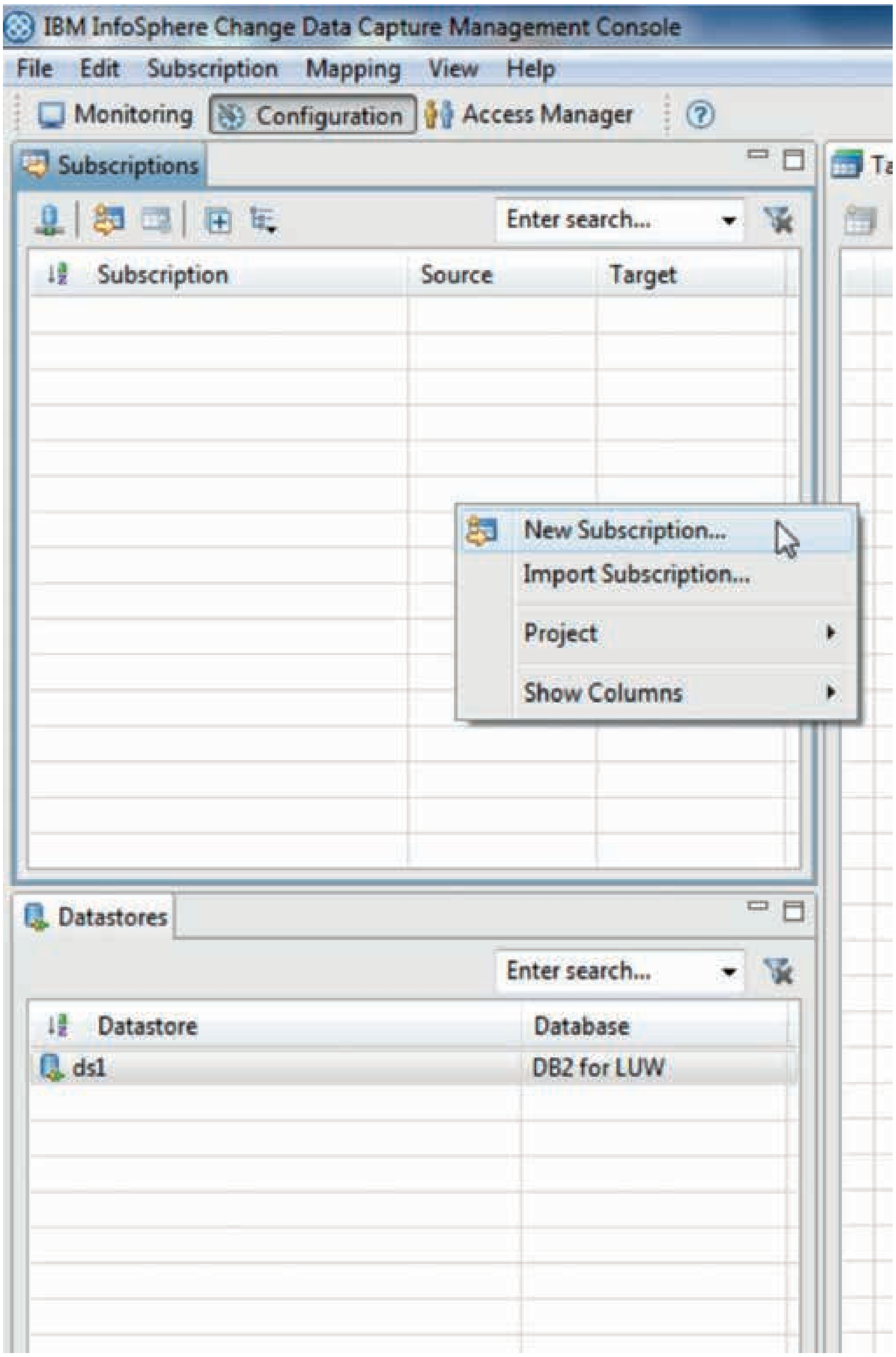
Be sure to create only one subscription for all shadow tables in the same database so that there is only one synchronisation point for all shadow tables in the database. Open CDC management console (make sure you start both the CDC Access Server and CDC replication instance first). First connect to the data store and right click at the datastore entry.

Maintenance of the shadow tables through CDC is an asynchronous process and hence has minimal impact to OLTP transactions.

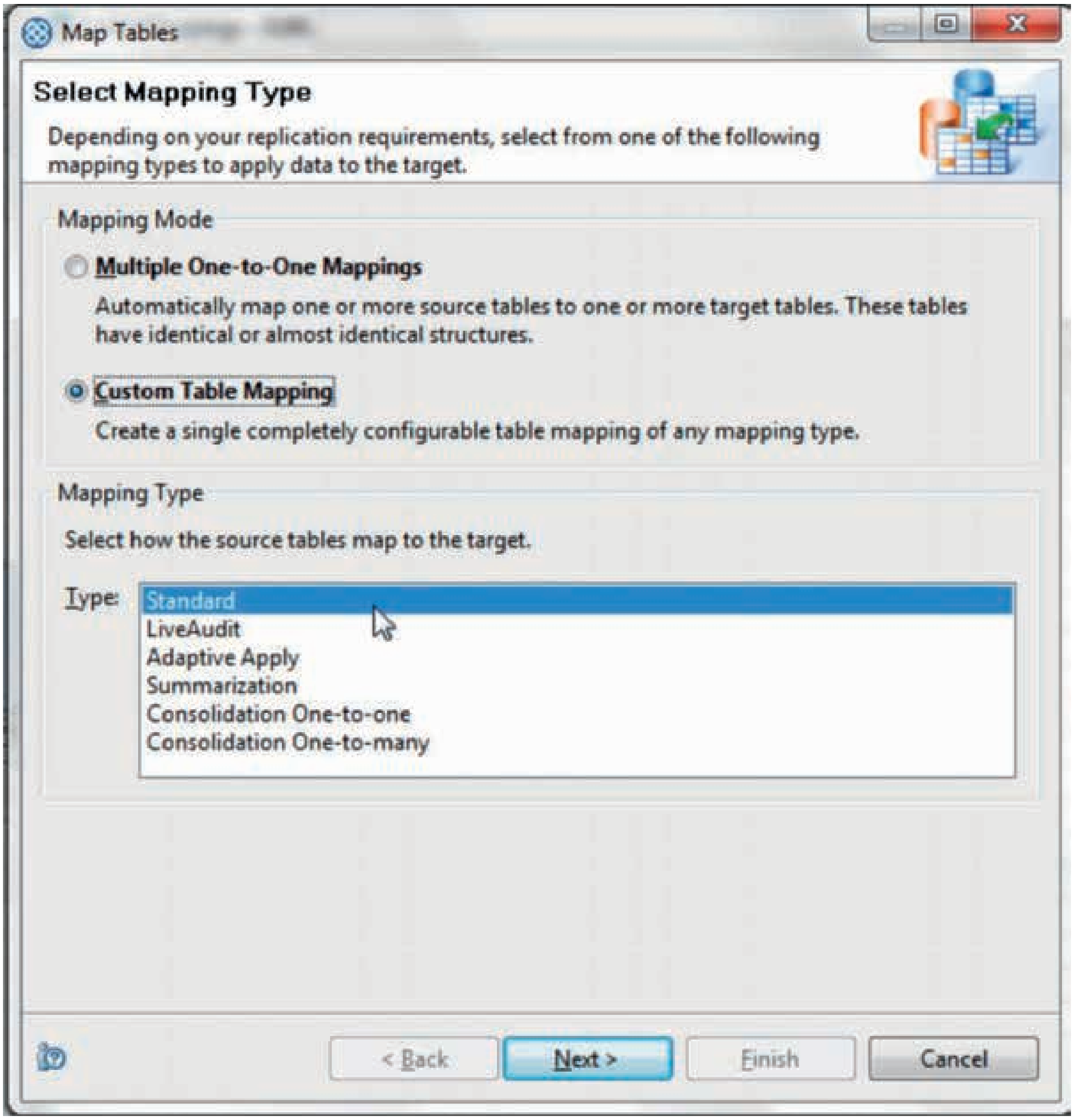
CDC replication involves a capture engine that scrapes the DB2 logs for update deltas and feed these deltas to the apply engine to maintain the shadow tables. The “apply” to the shadow tables is further optimised to leverage the new Cancun index scan driven update feature for column-organised tables.



Then, create a new subscription.



Datastore	Database
ds1	DB2 for LUW



Select Mapping Type

Depending on your replication requirements, select from one of the following mapping types to apply data to the target.

Mapping Mode

( ) **Multiple One-to-One Mappings**

Automatically map one or more source tables to one or more target tables. These tables have identical or almost identical structures.

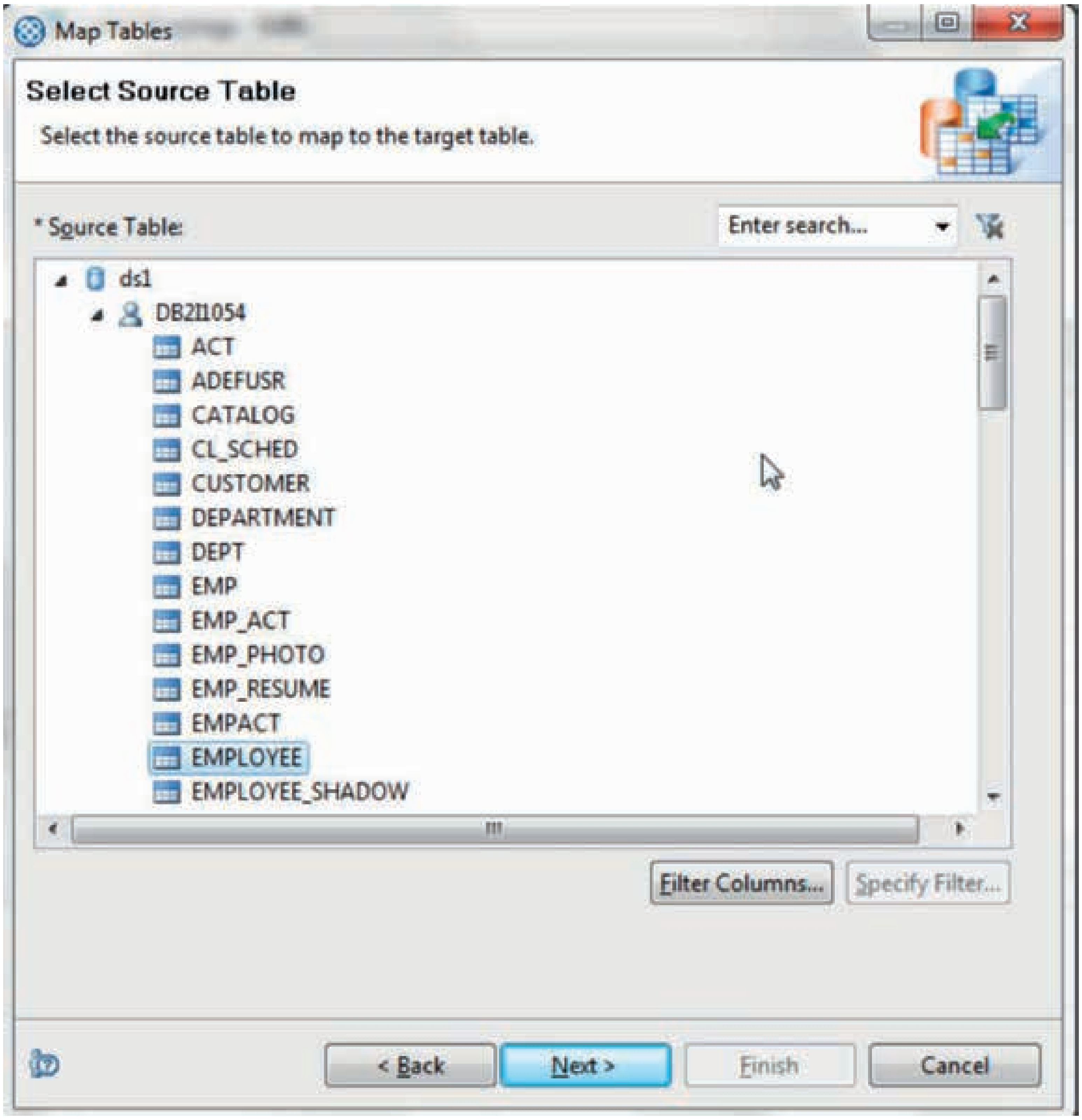
[✓ **Custom Table Mapping**]

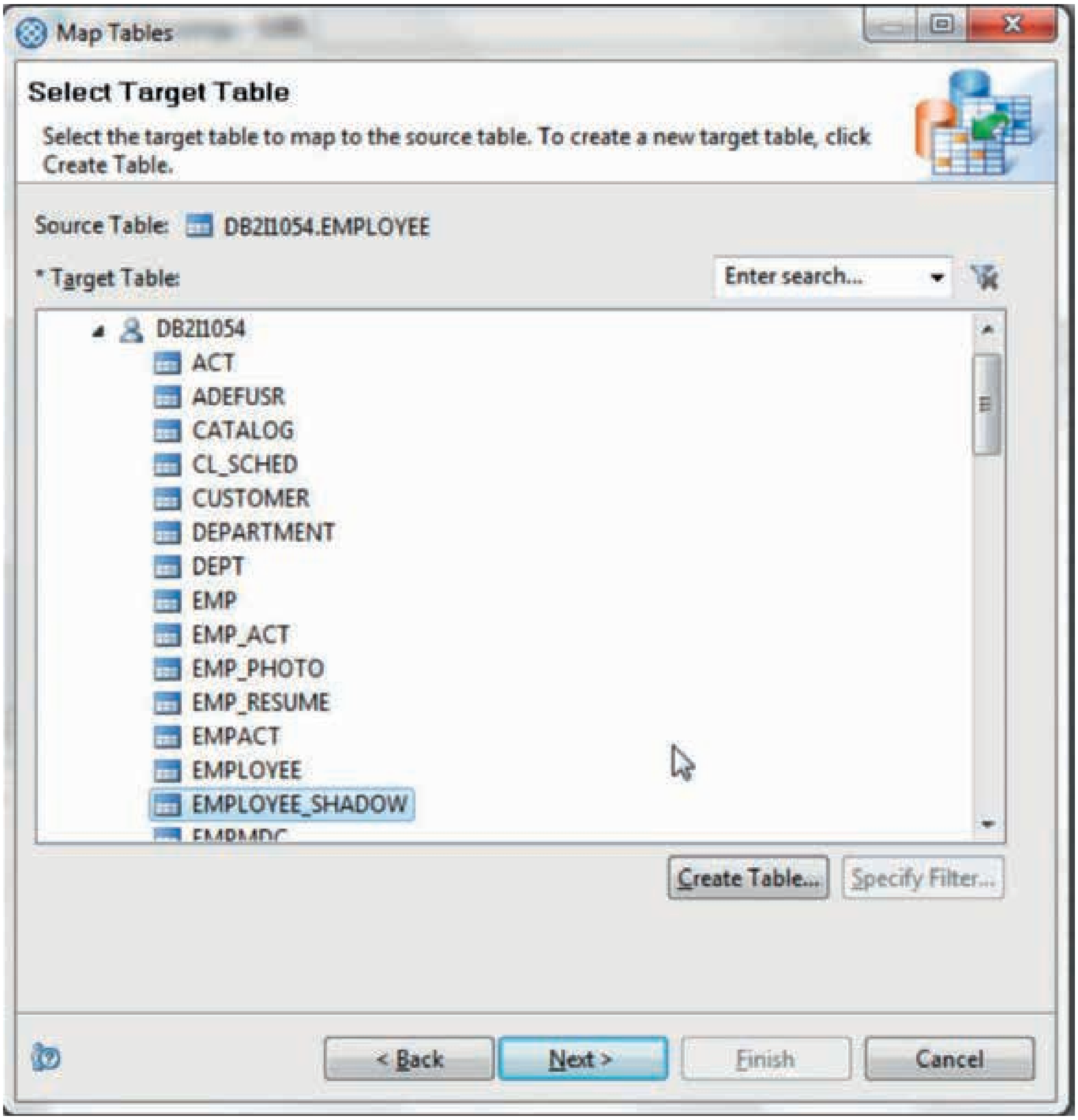
Create a single completely configurable table mapping of any mapping type.

Mapping Type

Select how the source tables map to the target.

Type: [✓ Standard]
LiveAudit
Adaptive Apply
Summarization
Consolidation One-to-one
Consolidation One-to-many


Map Tables
Select Source Table
Select the source table to map to the target table.
* Source Table:
Enter search...
ds1
DB2I1054
ACT
ADEFUSR
CATALOG
CL_SCHED
CUSTOMER
DEPARTMENT
DEPT
EMP
EMP_ACT
EMP_PHOTO
EMP_RESUME
EMPACT
EMPLOYEE
EMPLOYEE_SHADOW
Filter Columns...
Specify Filter...
< Back
Next >
Finish
Cancel



Map Tables

Select Target Table

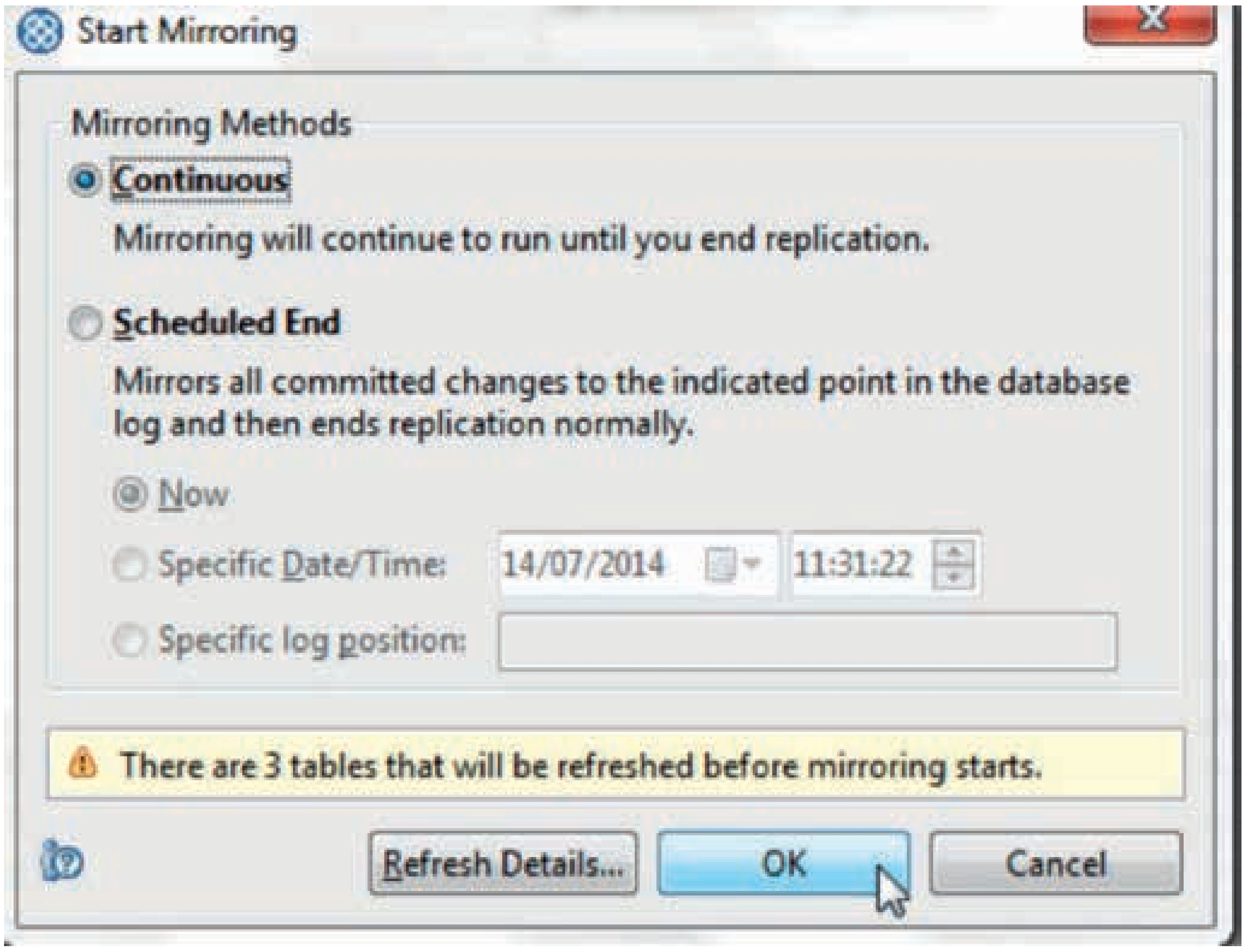
Select the target table to map to the source table. To create a new target table, click Create Table.

Source Table: DB2I1054.EMPLOYEE

* Target Table:

After the subscription is created start your subscription and begin replicating database changes to the shadow table. Be sure to create all table mappings before starting the subscription. You first need to add a subscription.

From the CDC Management Console: select Configuration > Subscriptions, then right click the subscription and select Start Mirroring.



Start Mirroring

Mirroring Methods

[✓ Continuous]

Mirroring will continue to run until you end replication.

Scheduled End

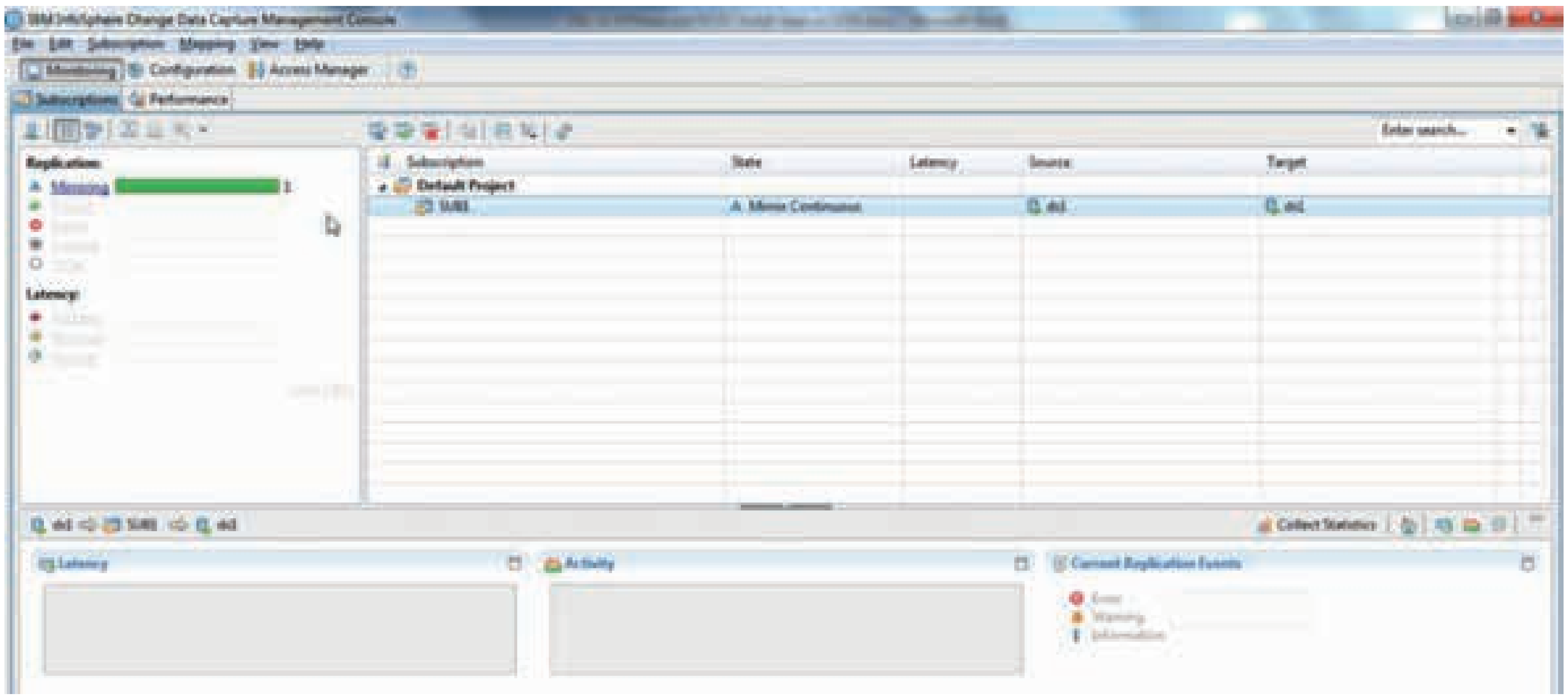
Mirrors all committed changes to the indicated point in the database log and then ends replication normally.

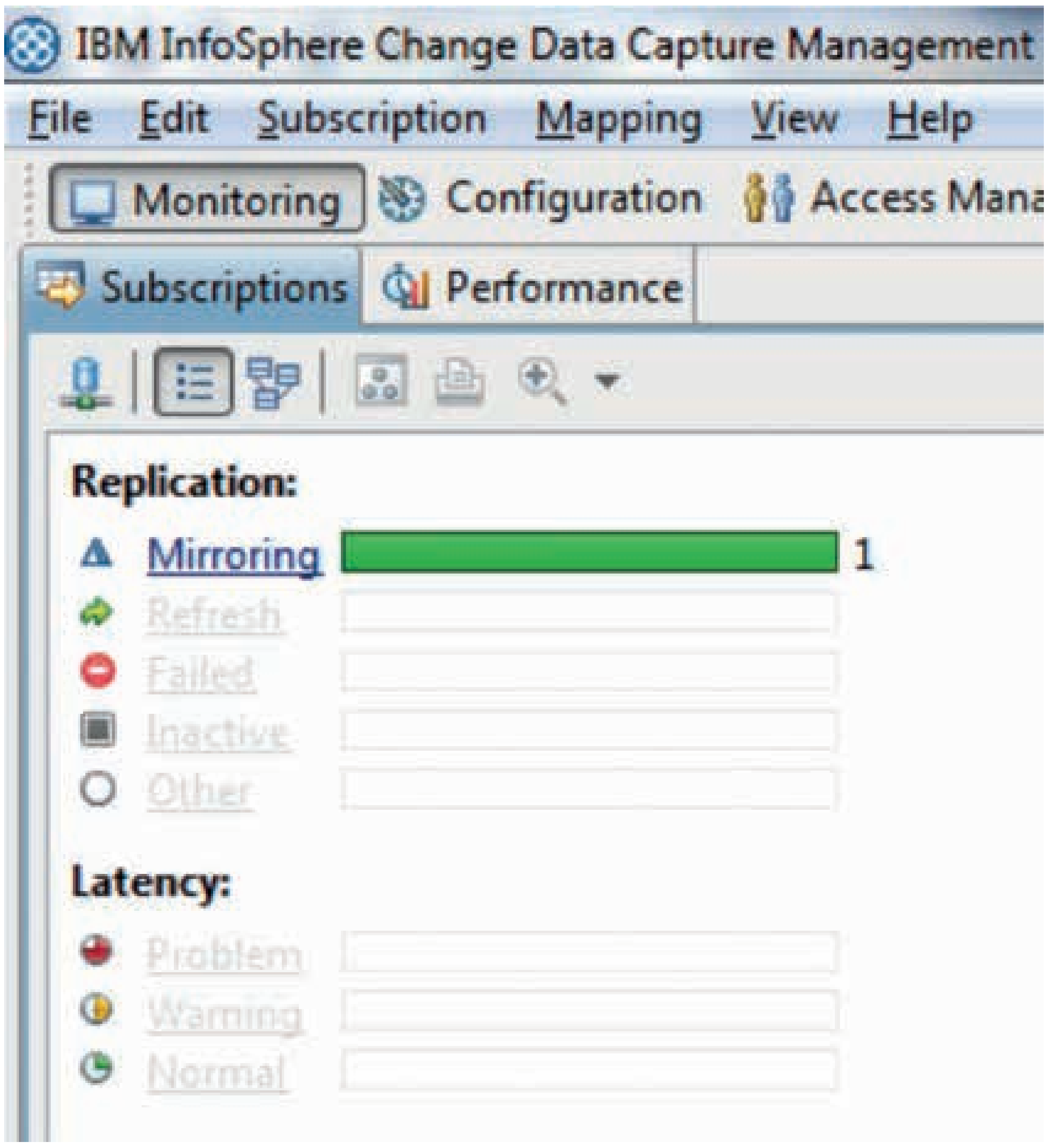
[✓ Now]

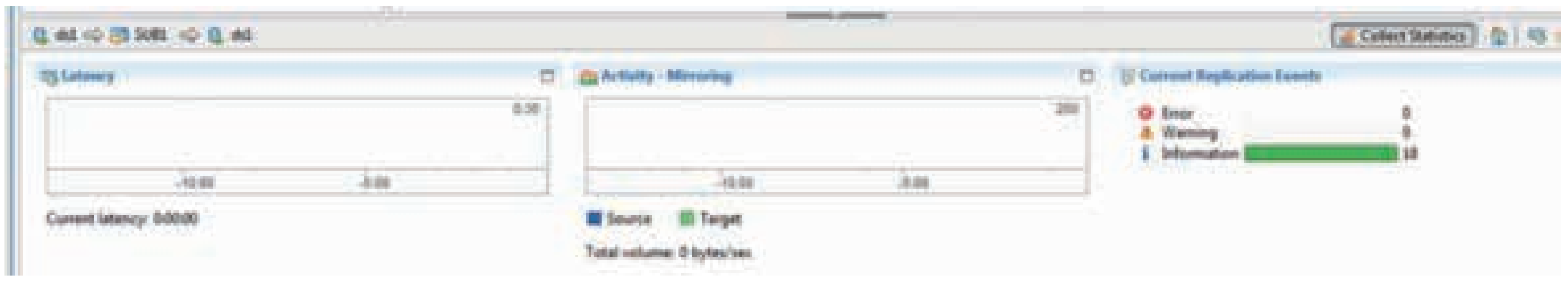
Specific Date/Time: 14/07/2014 11:31:22

Specific log position:

There are 3 tables that will be refreshed before mirroring starts.


IBM InfoSphere Change Data Capture Management Console
File Edit Subscription Mapping View Help
Monitoring Configuration Access Manager
Subscriptions Performance
Replication:
Mirroring 1
Latency:
Subscription State Latency Source Target
Default Project
SUB1 Mirror Continuous ds1 ds2
Latency
Activity
Current Replication Events
Error
Warning
Information
Collect Statistics

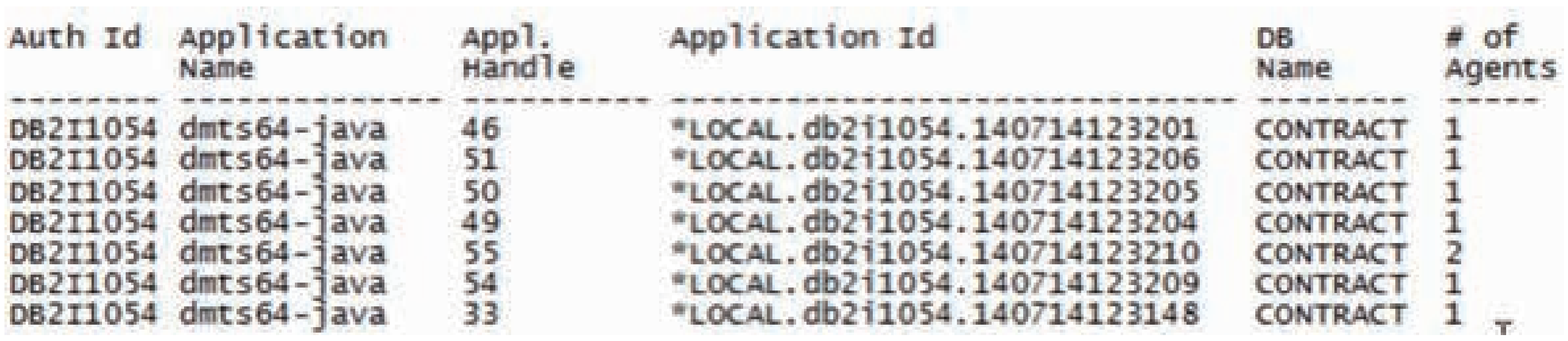

IBM InfoSphere Change Data Capture Management
File Edit Subscription Mapping View Help
Monitoring Configuration Access Mana
Subscriptions Performance
Replication:
Mirroring 1
Refresh
Failed
Inactive
Other
Latency:
Problem
Warning
Normal



The summary page shows a graphical view of latencies, activities, and event counts. Latency is the difference between the time at which a transaction is applied to the target and when that transaction occurred in the source. The activity charts show the amount of data (in bytes) or the number of apply operations that are being processed by CDC.

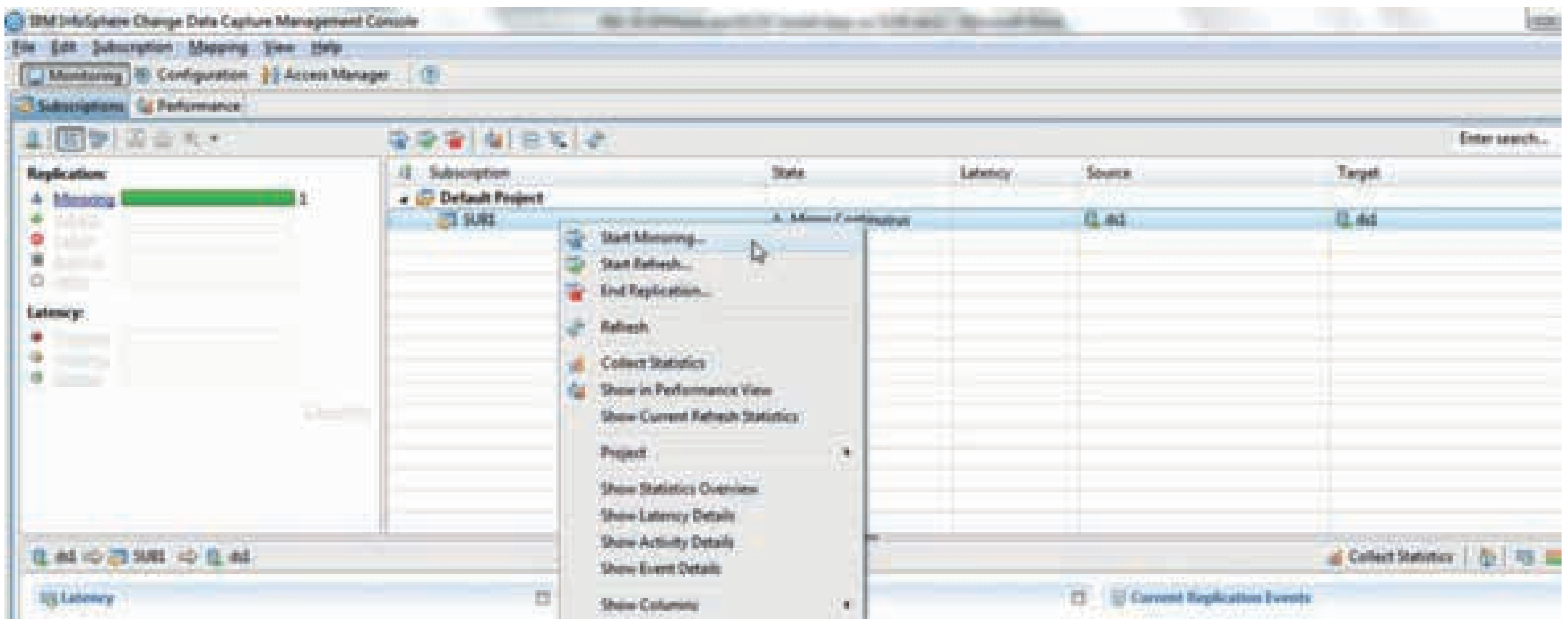
From the data store (DB2 instance):

```
db2 list applications
```



```
Auth Id  Application     Appl.      Application Id                  DB       # of
         Name            Handle                                     Name     Agents
-------- --------------- ---------- ------------------------------- -------- -----
DB2I1054 dmts64-java     46         *LOCAL.db2i1054.140714123201    CONTRACT 1
DB2I1054 dmts64-java     51         *LOCAL.db2i1054.140714123206    CONTRACT 1
DB2I1054 dmts64-java     50         *LOCAL.db2i1054.140714123205    CONTRACT 1
DB2I1054 dmts64-java     49         *LOCAL.db2i1054.140714123204    CONTRACT 1
DB2I1054 dmts64-java     55         *LOCAL.db2i1054.140714123210    CONTRACT 2
DB2I1054 dmts64-java     54         *LOCAL.db2i1054.140714123209    CONTRACT 1
DB2I1054 dmts64-java     33         *LOCAL.db2i1054.140714123148    CONTRACT 1
```

If you do db2stop and db2start on either source or target data store then the replication/mirroring breaks down. You will then need to start mirroring again by the CDC Admin console.


IBM InfoSphere Change Data Capture Management Console
File Edit Subscription Mapping View Help
Monitoring Configuration Access Manager
Subscriptions Performance
Replication:
Mirroring 1
Latency:
Subscription State Latency Source Target
Default Project
SUB1 Mirror Continuous dsl dsl
Start Mirroring...
Start Refresh...
End Replication...
Refresh
Collect Statistics
Show in Performance View
Show Current Refresh Statistics
Project
Show Statistics Overview
Show Latency Details
Show Activity Details
Show Event Details
Show Columns
Enter search...
dsl ⇨ SUB1 ⇨ dsl
Collect Statistics
Latency
Current Replication Events