LAB 06 – MDC

A. SET UP CARS TABLES EXAMPLE

To show MDCs in action, we need to create two otherwise identical tables in their own table spaces, but one is MDC and one is traditionally index clustered with indexes on all the other columns. The data we'll put into these tables is random, so each person in this lab will have a slightly different experience.

1. Make sure you position yourself in the C:/POT/06 MDC/ directory before continuing
2. To set up this lab, review and run these scripts:

MDC06001.CMD which executes ⇨ these 3 scripts:

MDC06002.DB2 & **MDC06003.DB2** & **MDC06004.BAT**

This should take about 5 minutes to complete.

Note: These scripts create the following:

Table: CARS	Regular table w/clustered index and regular indexes
Table: CARS_MDC	MDC table, with dimensions CARNAME, COLOR & YEAR
Table: CARS_TIMING_CHECK	Repository table used to collect timing statistics

Table space: CARS_DATA
Table space: CARS_MDC_DATA

Each table is in its own table space so as not to interfere with each other (the timing table is put in the userspace1 table space)

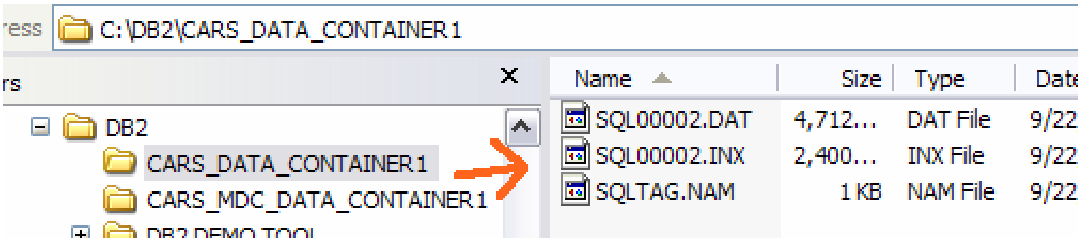
Procedure: CARS_INSERT_ALL	Used to populate both CARS and CARS_MDC with generated data
Procedure: CARS_FETCH_CHK	Used to perform fetches on both tables and time the results
Procedure: CARS_INSERT_CHK	Used to perform inserts on both tables and time the results
Procedure: CARS_UPDATE_CHK	Used to perform updates on both tables and time the results
Procedure: CARS_DELETE_CHECK	Used to perform deletes on both tables and time the results

B. EXPLORE MDC SPACE USAGE

First, let’s explore how the space allocation is used with our SMS table spaces. We could have used DMS table spaces, but SMS allows us to use the operating system to see allocation a little easier.

1. Windows Explorer ⇨ C:\DB2
2. Review both table spaces:

CARS_DATA_CONTAINER1 & CARS_MDC_DATA_CONTAINER1

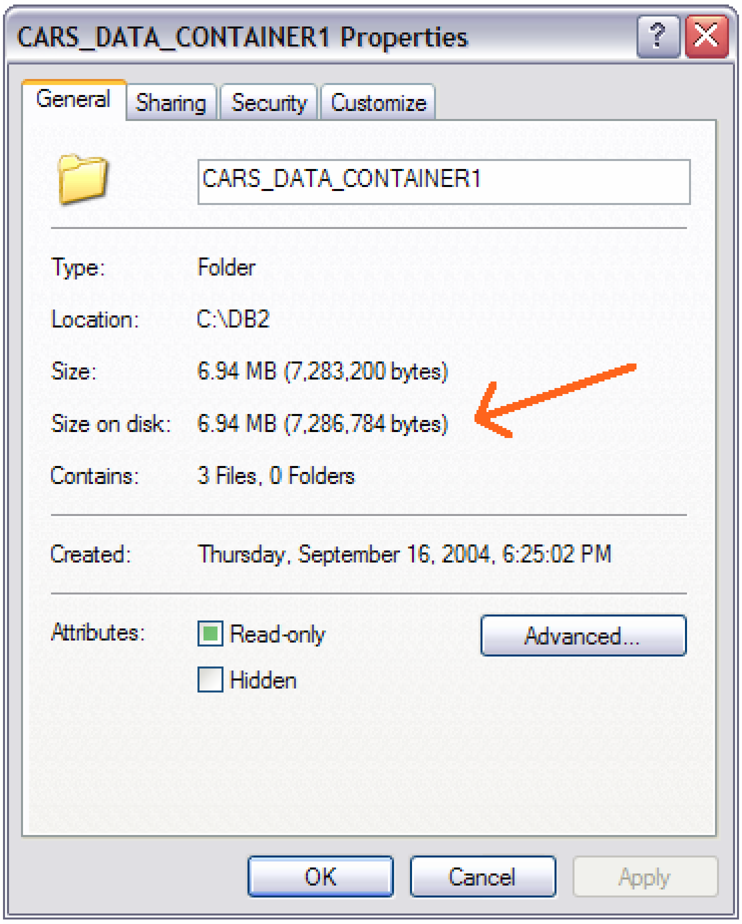


C:\DB2\CARS_DATA_CONTAINER1

Name	Size	Type	Dat
SQL00002.DAT	4,712...	DAT File	9/22
SQL00002.INX	2,400...	INX File	9/22
SQLTAG.NAM	1 KB	NAM File	9/22

3. Which has the bigger table usage?
4. Which has the bigger index usage?

5. Right click on subdirectory for each then ⇨ Properties ⇨ Size
    - You are using Windows to tell you total size of that subdirectory folder
    - Note total file size for CARS table: __________ CARS_MDC table : __________



CARS_DATA_CONTAINER1 Properties

CARS_DATA_CONTAINER1

Type: Folder

Location: C:\DB2

Size: 6.94 MB (7,283,200 bytes)

Size on disk: 6.94 MB (7,286,784 bytes)

Contains: 3 Files, 0 Folders

Created: Thursday, September 16, 2004, 6:25:02 PM

Attributes: ☑ Read-only ☐ Hidden

Section Answers:

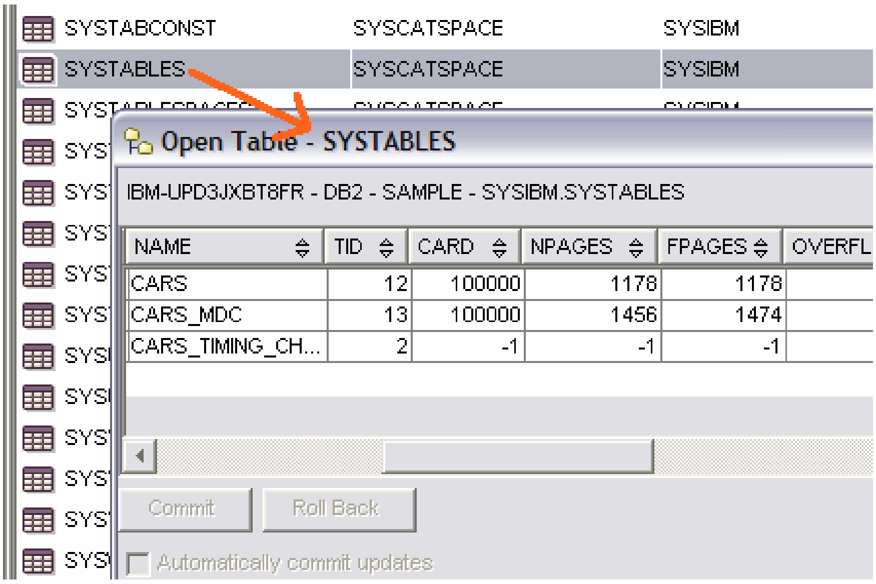
B3. MDCs usually use more data space, because they have to allocate a block (extent) at a time. However, this could vary depending upon non MDC usage of the PCTFREE and method of getting data into the table.

B4. Non MDC tables usually use more index space because they have to create a rid for each ***row.*** MDCs only create a bid for each new ***block***.

C. EXPLORE MDC CATALOG STATISTICS

So what does an MDC table look like from a catalog standpoint vs. a traditional clustered index table? To see this:

1. Control Center ⇨ Tables ⇨ SYSIBM.SYSTABLES ⇨ double click to select data
2. Filter for table names starting: **CARS%**



SYSTABCONST SYSCATSPACE SYSIBM
SYSTABLES SYSCATSPACE SYSIBM

Open Table - SYSTABLES

IBM-UPD3JXBT8FR - DB2 - SAMPLE - SYSIBM.SYSTABLES

NAME	TID	CARD	NPAGES	FPAGES	OVERFL
CARS	12	100000	1178	1178	
CARS_MDC	13	100000	1456	1474	
CARS_TIMING_CH...	2	-1	-1	-1	

3. Review these columns:

CARD	total cardinality of table (number of rows)
FPAGES	total number of pages
NPAGES	total number of pages on which rows exist
ACTIVE BLOCKS	blocks used for MDC table

- Notice system generated vs. user defined indexes for both tables

4. Control Center ⇨ Views ⇨ SYSCAT.INDEXES ⇨ double click to select data
5. Filter for TABNAME LIKE **CARS%**

- Notice system generated vs. user defined indexes for both tables

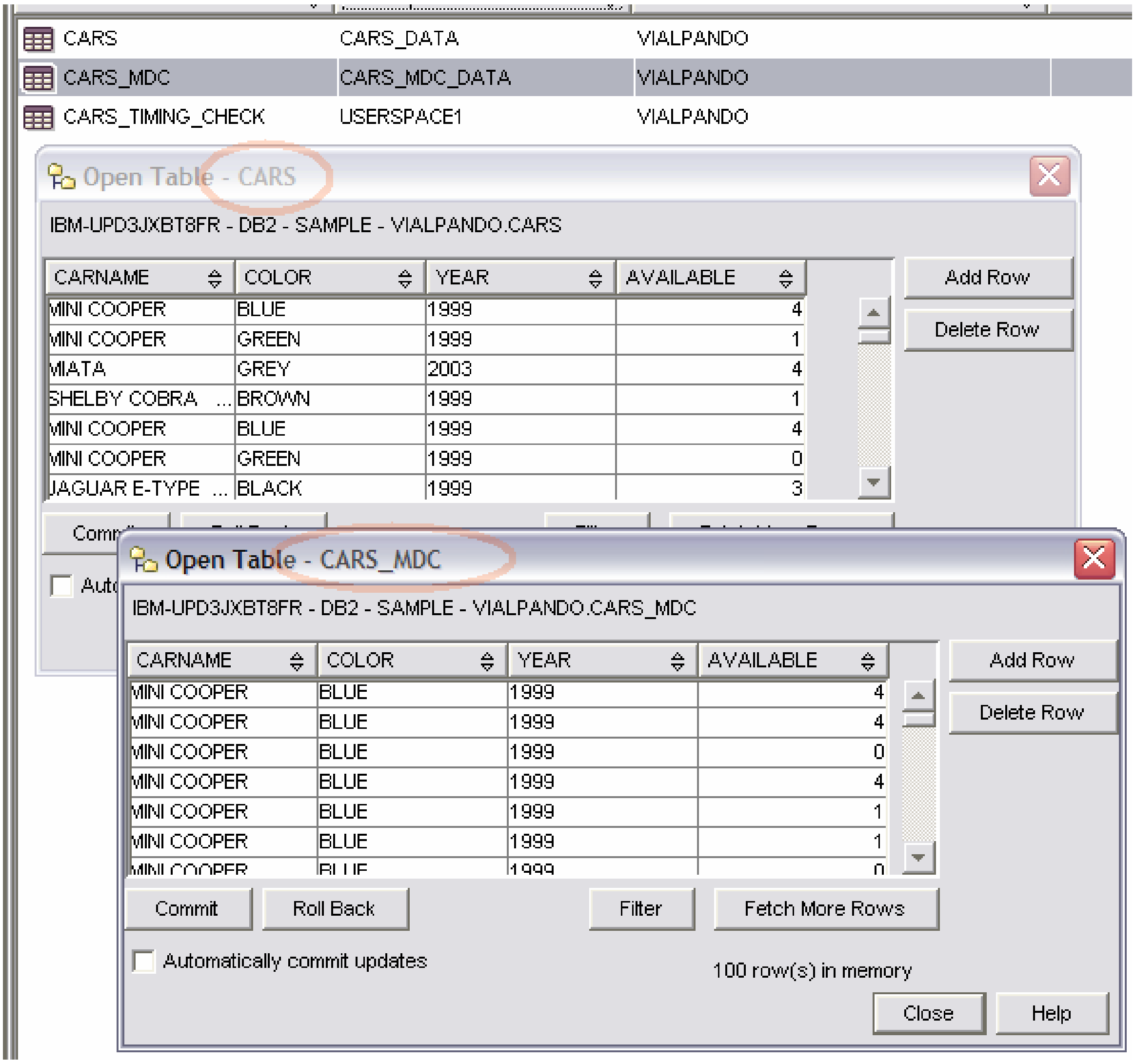
6. While in this view, review these columns:

INDEXTYPE	
NLEAF	leaf pages total
FIRSTKEYCARD	column1 cardinality
FIRST2KEYCARD	column1 & 2 cardinality together… and so on
CLUSTERRATIO	how well data is grouped
NUMRIDS	total number of row ids

D. EXPLORE MDC TABLE ORGANIZATION

Let’s look at how these two tables differ in the way the data is added to the tables themselves:

1. Control Center ⇨ Tables ⇨ create a filter on NAME LIKE **CARS%**
2. Double click on **CARS** & **CARS_MDC** to see what the data looks like



CARS	CARS_DATA	VIALPANDO
CARS_MDC	CARS_MDC_DATA	VIALPANDO
CARS_TIMING_CHECK	USERSPACE1	VIALPANDO

Open Table - CARS

IBM-UPD3JXBT8FR - DB2 - SAMPLE - VIALPANDO.CARS

CARNAME	COLOR	YEAR	AVAILABLE
MINI COOPER	BLUE	1999	4
MINI COOPER	GREEN	1999	1
MIATA	GREY	2003	4
SHELBY COBRA ...	BROWN	1999	1
MINI COOPER	BLUE	1999	4
MINI COOPER	GREEN	1999	0
JAGUAR E-TYPE ...	BLACK	1999	3

Open Table - CARS_MDC

IBM-UPD3JXBT8FR - DB2 - SAMPLE - VIALPANDO.CARS_MDC

CARNAME	COLOR	YEAR	AVAILABLE
MINI COOPER	BLUE	1999	4
MINI COOPER	BLUE	1999	4
MINI COOPER	BLUE	1999	0
MINI COOPER	BLUE	1999	4
MINI COOPER	BLUE	1999	1
MINI COOPER	BLUE	1999	1
MINI COOPER	BLUE	1999	0

☐ Automatically commit updates

100 row(s) in memory

3. Notice that first row of the **CARS** data is the first *grouping* of rows of **CARS_MDC** data

4. [Fetch More Rows] in **CARS_MDC** until you hit another new grouping;
    - How does this coincide with the CARS table?
5. Select counts on both tables for certain colors or certain car names… both tables are identical in content, but not organization. Prove this to yourself. Example:

    select carname, count(*) from cars group by carname;

    select carname, count(*) from cars_mdc group by carname;
6. Write down 8 or 9 different colors in the space below that you find in the data. We'll be using these colors for the next exercise.

    _______ _______ _______ _______ _______ _______ _______ _______ _______

Section Answers:

D4. As each row is added to the non MDC table, the traditionally unordered row sets of a non MDC relational table just puts each row in as it gets them. But for MDCs, a block is allocated for each dimension "cell", so blocks are created for each row until it sees another row that fits into a previously created block. When an MDC sees that it already has a block available for rows that match in dimension, they put the data in that block. When you view the data, you start to see data being "organized" in this way.... it looks "sorted" but is not, it's "organized".

E. MDC PERFORMANCE TIMING TEST

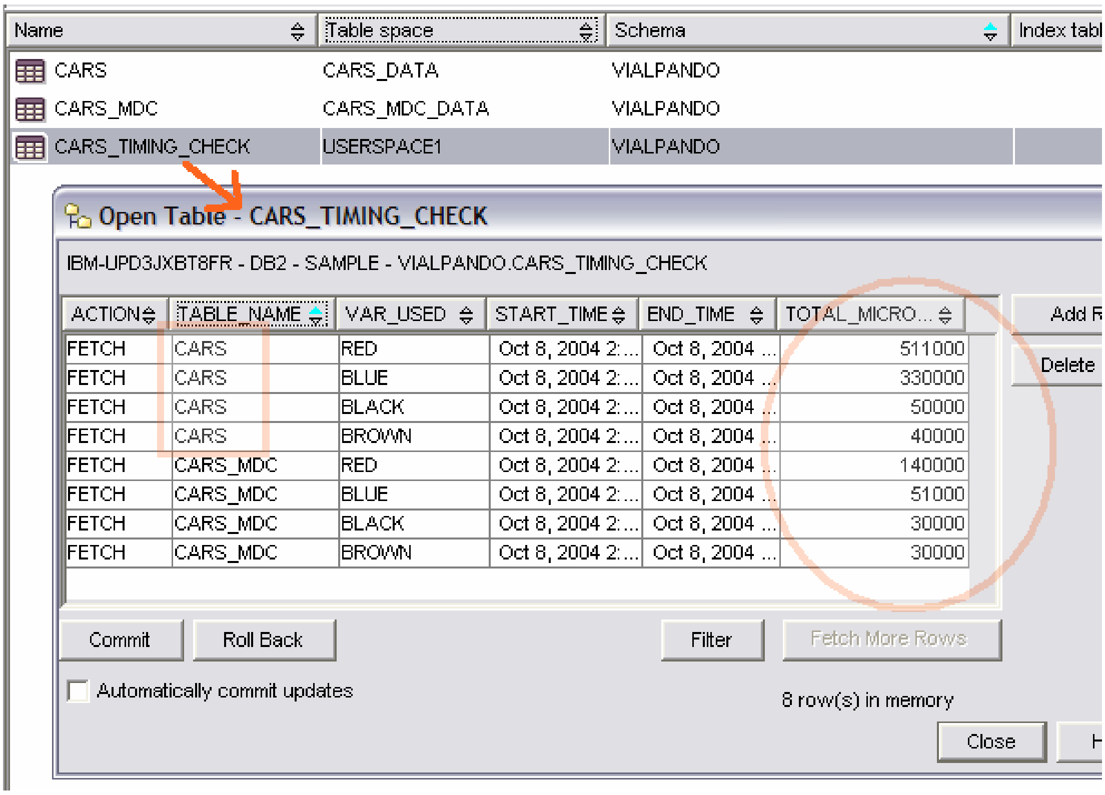
There are four stored procedures we will be using to test the timing of fetch, insert, update and delete actions against both tables. You can review the code for these stored procedures in the file **MDC06003.DB2** you executed earlier to satisfy yourself that they perform the exact same actions against both tables in the exact same way. The procedures further place the output of this test in the CARS_TIMING_CHECK table so you can review total time spent (in microseconds) on each action.

Note: Before you do this exercise, make sure you are no longer browsing any of the cars tables! Close any Control Center windows you may have open on the data or you may lock out transactions from this next set of exercises.

Disclaimer: The purpose of this test is not to try to prove anything regarding specific performance with MDCs, but is rather to get you to think about how performance might work with MDCs in your environment. We realize this test is neither complete enough nor large enough to be considered comprehensive in any way.

Fetch timing test

1. From a DB2 Command Window ⇨ **db2 call cars_fetch_chk('xxxx')** substituting xxxx with the color of your choice. This stored procedure will fetch all records from each table that has the color you indicated
   - Do this with three or four different colors to get more than one result for a fetch from both tables
   - You might want to add a fetch on a made up color you know that DOES NOT exist
   - Review the results in the CARS_TIMING_CHECK table and review the TOTAL_MICROSCNDS column for time spent on each fetch activity; which seems to run faster fetches, MCD or non?



Name	Table space	Schema	Index tabl
CARS	CARS_DATA	VIALPANDO	
CARS_MDC	CARS_MDC_DATA	VIALPANDO	
CARS_TIMING_CHECK	USERSPACE1	VIALPANDO	

Open Table - CARS_TIMING_CHECK

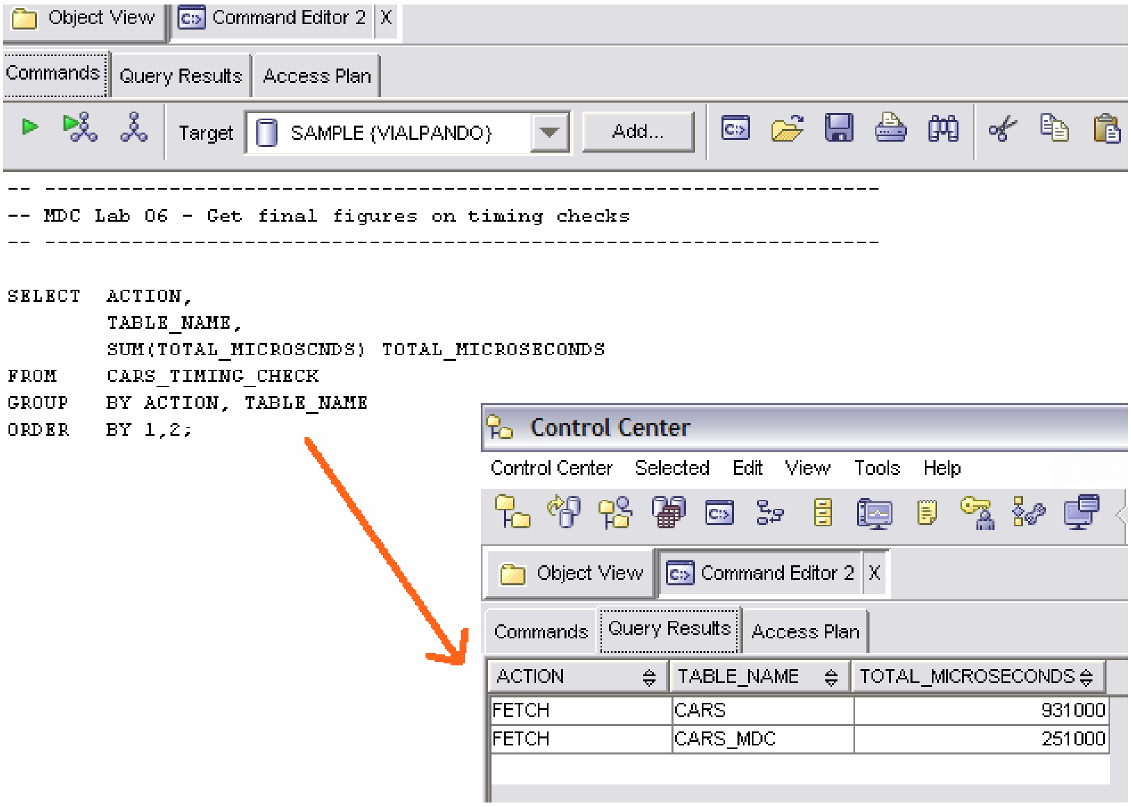
IBM-UPD3JXBT8FR - DB2 - SAMPLE - VIALPANDO.CARS_TIMING_CHECK

ACTION	TABLE_NAME	VAR_USED	START_TIME	END_TIME	TOTAL_MICRO...
FETCH	CARS	RED	Oct 8, 2004 2:...	Oct 8, 2004 ...	511000
FETCH	CARS	BLUE	Oct 8, 2004 2:...	Oct 8, 2004 ...	330000
FETCH	CARS	BLACK	Oct 8, 2004 2:...	Oct 8, 2004 ...	50000
FETCH	CARS	BROWN	Oct 8, 2004 2:...	Oct 8, 2004 ...	40000
FETCH	CARS_MDC	RED	Oct 8, 2004 2:...	Oct 8, 2004 ...	140000
FETCH	CARS_MDC	BLUE	Oct 8, 2004 2:...	Oct 8, 2004 ...	51000
FETCH	CARS_MDC	BLACK	Oct 8, 2004 2:...	Oct 8, 2004 ...	30000
FETCH	CARS_MDC	BROWN	Oct 8, 2004 2:...	Oct 8, 2004 ...	30000

☐ Automatically commit updates

8 row(s) in memory

- You can use script MDC06005.SQL to get totals from the CARS_TIMING_CHECK table;
    - Run it from the Command Editor (GUI tool) to see totals



```
-- ------------------------------------------------------------------
-- MDC Lab 06 - Get final figures on timing checks
-- ------------------------------------------------------------------

SELECT  ACTION,
        TABLE_NAME,
        SUM(TOTAL_MICROSCNDS) TOTAL_MICROSECONDS
FROM    CARS_TIMING_CHECK
GROUP   BY ACTION, TABLE_NAME
ORDER   BY 1,2;
```

ACTION	TABLE_NAME	TOTAL_MICROSECONDS
FETCH	CARS	931000
FETCH	CARS_MDC	251000

Generally speaking, MCD tables can be very efficient for fetching data. Though both tables are highly indexed, MCD indexes are smaller and thus more efficient. Further, you have automatic prefetching advantages for MCD tables. Too, the non MCD clustered table may degrade over time. Run 7 or 8 colors to see the numbers begin to really add up and imagine tables will millions or rows and warehouses with thousands of queries and you can start to get the idea of how this can be very beneficial.

Insert timing test

2. From a DB2 Command Window ⇨ **db2 call cars_insert_chk(‘xxxx’)**
    - Repeat as above; run the totaling script again to get a new running total
    - Try an insert on a made up color you know does not exist

Inserting into MDC tables is not its strong suit. It is generally better to LOAD into an MDC table for better efficiency. Remember, each new cell requires a brand new block allocation.

Update timing test

3. From a DB2 Command Window ⇨ **db2 call cars_update_chk(‘xxxx’)**

   - Repeat as above
   - Note: this procedure updates the color you chose by adding an X to the front of the data; review what your data looks like in the CARS table before proceeding to the next test

   Updating into MDC tables can be a mixed bag. Depending upon what is being updated, MDC tables can be faster or slower. In this exercise, we actually updated on one of the dimension columns which generally makes the MDC slower because it has to relocate the data to a new or different block. How did your test do? In any case, if you update on an MDC column that is NOT in the dimension, then it can be quite fast, especially if it has to do entire blocks at a time.

Delete timing test

4. From a DB2 Command Window ⇨ **db2 call cars_delete_chk(‘xxxx’)**

   - Repeat as above; make sure you take into account some colors have changed
   - Try a made up color that does not exist

   Deletes from an MDC table can also be quite fast. If you delete on a dimension column, MDCs know already all the blocks in that qualify quite quickly and can delete a block at a time. Also, it has less bid indexes to deal with. Your test should have shown an advantage here over the non MDC table.

5. Run script **MDC06005.SQL** to get final totals from the **CARS_TIMING_CHECK** table

   - You can click on the table name column to group these together
   - How did your MDC table do?