IBM® DB2® for Linux®, UNIX®, and Windows®

Best practices
Optimizing analytic workloads using DB2 10.5 with BLU Acceleration

Jessica Rockwood
Senior Manager, DB2 LUW Performance Benchmarking, IBM Canada Lab

Roman B. Melnyk
Senior Information Developer, IBM Canada Lab

Michael Kwok
Senior Manager, DB2 LUW Warehouse Performance, IBM Canada Lab

Berni Schiefer
Distinguished Engineer, Information Management Performance and Benchmarks, IBM Canada Lab

Issued: September 2013

Executive summary

DB2 10.5 with BLU Acceleration is a combination of complementary innovations from IBM that simplifies and speeds up analytic workloads. It is easy to implement and is self-optimizing. BLU Acceleration eliminates the need for indexes, aggregates (for example, MQTs or materialized views), or time-consuming database tuning to achieve top performance and storage efficiency. No SQL or schema changes are required to take advantage of this breakthrough technology.

These innovations, which are introduced in DB2® for Linux®, UNIX ®, and Windows® Version 10.5 (DB2 10.5), are designed to help you quickly find answers to more complex business questions while keeping costs down.

DB2 10.5 with BLU Acceleration offers a rich set of features that help you to meet these goals, including column-organized storage, actionable compression, parallel vector processing, and data skipping. In combination, these features provide an in-memory, CPU-optimized, and I/O-optimized solution.

This paper gives you an overview of these technologies, recommendations on hardware and software selection, guidelines for identifying the optimal workloads for BLU Acceleration, and information about capacity planning, memory, and I/O. A section on system configuration shows you how IBM's focus on simplicity enables you to set up DB2 10.5 so that it automatically makes optimal configuration choices for analytic workloads. Other sections describe how to implement and use DB2 with BLU Acceleration. The focus of this best practices paper is on ease of use. You will learn how BLU Acceleration works and what it is doing "under the covers", which will give you a real appreciation of the simplicity that is built into BLU Acceleration and show you that it really does deliver "super analytics, super easy".

Introduction

BLU Acceleration is a new collection of technologies for analytic queries that are introduced in DB2 for Linux, UNIX, and Windows Version 10.5 (DB2 10.5[1]). At its heart, BLU Acceleration is about providing faster answers to more questions and analyzing more data at a lower cost. DB2 with BLU Acceleration is about providing order-of-magnitude benefits in performance, storage savings, and time to value.

These goals are accomplished by using multiple complementary technologies, including:

- The data is in a *column store*, meaning that I/O is performed only on those columns and values that satisfy a particular query.
- The column data is compressed with *actionable compression*, which preserves order so that the data can be used without decompression, resulting in huge storage and CPU savings and a significantly higher density of useful data held in memory.
- *Parallel vector processing*, with multi-core parallelism and single instruction, multiple data (SIMD) parallelism, provides improved performance and better utilization of available CPU resources.
- *Data skipping* avoids the unnecessary processing of irrelevant data, thereby further reducing the I/O that is required to complete a query.

These and other technologies combine to provide an in-memory, CPU-optimized, and I/O-optimized solution that is greater than the sum of its parts.

BLU Acceleration is fully integrated into DB2 10.5, so that much of how you leverage DB2 in your analytics environment today still applies when you adopt BLU Acceleration. The simplicity of BLU Acceleration changes how you implement and manage a BLU-accelerated environment. Gone are the days of having to define secondary indexes or aggregates, or having to make SQL or schema changes to achieve adequate performance.

This best practices paper focuses as much on “what you no longer need to do” as on “what you should do”. We direct your attention to a few key implementation items, and then you can let BLU Acceleration take over and provide you with optimal performance. It is analytics that is super fast and super easy – just load and go!

[1] This document is current with respect to DB2 10.5 FP1 (DB2 10.5.0.1) and later. The use of the most current fix pack is encouraged.

Before you start

Ensure that your system meets the necessary hardware and software requirements.

Hardware and software

BLU Acceleration is supported on POWER®/AIX® and x86/Linux platforms. Table 1 summarizes the supported operating systems, including both minimum and recommended versions, as well as recommended processors. Note that the minimum version requirements on these platforms are the DB2 10.5 requirements, which are applicable to both row- and column-organized tables.

Table 1. Hardware and operating system recommendations

Operating system	**Minimum version requirements**	**Recommended versions**	**Hardware recommendations**
AIX	AIX 6.1 TL7 SP6 or AIX 7.1 TL1 SP6	AIX 7.1 TL2 SP1 or higher	POWER7 or higher[2]
Linux x86 (64-bit only)	Red Hat Enterprise Linux (RHEL) 6, SuSE Linux Enterprise Server (SLES) 10 SP4 or SLES 11 SP2	RHEL 6.3 or higher SLES11 SP2 or higher	Intel Nehalem (or equivalent) or higher[3]

For optimal performance, we recommend that you maintain your system firmware at the latest levels, particularly if you are using virtualization.

BLU Acceleration is offered in DB2 Advanced Workgroup Server Edition, DB2 Advanced Enterprise Server Edition, and DB2 Developer Edition.

[2] Any POWER processor that supports DB2 10.5 is supported, but there are specific hardware optimizations on this processor.
[3] Any Intel processor that supports Linux x86 is supported, but there are specific hardware optimizations on this processor.

Identifying optimal workloads for BLU Acceleration

In general, column-organized tables significantly speed up analytic workloads or data mart types of workloads. A data mart or warehouse typically has queries that have grouping, aggregation, range scans, or joins; queries that access a subset of a table's columns; and database designs that often include a star or snowflake schema.

DB2 10.5 supports using both row-organized and column-organized tables in the same database, even in the same table spaces and buffer pools. The benefit of having column organization and row organization in the same database is that you can choose to optimize the table layout based on the workload. Row-organized tables have long been optimized for online transactional processing (OLTP) environments. Now with column-organized tables sitting right next to row-organized tables, you have the best of both worlds in the same database.

In fact, IBM provides Optim™ Query Workload Tuner[4] to identify and validate tables that would benefit from column organization based on the application workload.

Capacity planning

This section provides some guidelines for the number of cores and the amount of database memory that are recommended to achieve optimal performance for a BLU-accelerated workload.

Processor cores

DB2 with BLU Acceleration exploits multi-core parallelism and leverages SIMD parallelism to accelerate the performance of analytic queries. The effectiveness of these optimizations is directly impacted by the number of cores that are available to DB2 with BLU Acceleration.

A *minimum* of 8 processor cores is required[5] for BLU-accelerated workloads. Today a single processor chip typically has 8-12 cores. If the DFT_DEGREE database configuration parameter has the default setting of ANY, queries use a degree of intrapartition parallelism that is equal to the number of cores. When several queries are running concurrently, the degree is automatically scaled back, based on the number of active agent threads on the system, to maintain efficient execution.

Higher data volumes, higher degrees of query concurrency, and greater query complexity would benefit from a larger number of cores.

[4] Optim Query Workload Tuner is part of DB2 10.5 AESE and AWSE
[5] http://pic.dhe.ibm.com/infocenter/prodguid/v1r0/clarity-reports/report/html/softwareReqsForProductByComponent?deliverableId=254B4BA0C5F011E18183F12B0925FE36&duComponent=Server_8388E5E04AD611E2A6D822020925FE1B

Memory

DB2 with BLU Acceleration is designed to effectively leverage large memory configurations. To realize the greatest benefit from the in-memory analytics that is built into DB2 10.5, consider systems that have at least 64 GB of RAM for production use. As you scale up the system maintain a ratio of 8GB of RAM per core.

For information on how to allocate this memory within DB2, see "Best practices for configuration".

Input/Output

BLU Acceleration has more random I/O access than in row-organized data marts that typically perform large sequential scans. Consider storing key column-organized tables on I/O subsystems and devices that provide high random I/O access. Examples of storage that provides better random I/O access are solid state drives (SSD),flash-based storage or enterprise SAN storage systems BLU-accelerated tables that are frequently accessed benefit from improved I/O subsystem performance, particularly when the table exceeds the memory size that is available for use by BLU Acceleration.

There are both internal and external SSD or flash solutions. Internal flash storage might not have any write cache, and that can affect write operations involving, for example, temporary tables that are created and populated during query processing.

Common examples of storage that would provide good random I/O characteristics for use with DB2 with BLU Acceleration include FlashSystem (810/820), Storewize V7000 (with SSD), XIV, as well as enterprise SAN storage such as DS8K.

Best practices for configuration

The simplicity of managing a BLU-accelerated environment is one of the key benefits of this technology. There is even a new option for the DB2_WORKLOAD registry variable to automatically set configuration parameters that are most relevant to BLU Acceleration and the performance of analytic workloads.

Single setting for analytic workloads

If you have a database that is dedicated to analytic workloads, you can use a simple setting to configure DB2 and the database for BLU Acceleration.

By setting the DB2_WORKLOAD registry variable to ANALYTICS before you create the database, DB2 is aware that the system will be used for analytic workloads. The new database is automatically configured for optimal analytics performance. Remember to not disable AUTOCONFIGURE in the CREATE DATABASE command. For example:

```
db2set DB2_WORKLOAD=ANALYTICS
```

```
db2start
db2 create db mydb
```

Be sure to set this registry variable (when applicable) before creating your analytics database. DB2 will do the rest.

Table 2 describes in detail what happens to a newly created database when DB2_WORKLOAD=ANALYTICS.

Table 2. Database behavior that is influenced by DB2_WORKLOAD=ANALYTICS

Category	**Automatic best practice settings**

General database configuration parameters	DFT_TABLE_ORG = COLUMN
	
	PAGESIZE = 32 KB
	
	DFT_EXTENT_SZ = 4
	
	DFT_DEGREE = ANY

Database memory configuration parameters	CATALOGCACHE_SZ, SORTHEAP, and SHEAPTHRES_SHR are set to a value that is higher than the default and optimized for the hardware that you are using.
Intraquery parallelism	Enabled for any workload (including SYSDEFAULTUSERWORKLOAD) that specifies MAXIMUM DEGREE DEFAULT, even if the database manager configuration parameter INTRA_PARALLEL = OFF.
DB2 workload management	The default concurrency threshold on the SYSDEFAULTMANAGEDSUBCLASS service subclass is enabled to ensure maximum efficiency and utilization of the server.
Automatic space reclamation	Sets AUTO_MAINT = ON and AUTO_REORG = ON, and space reclamation is performed for column-organized tables by default.

The DB2_WORKLOAD registry variable is set at the instance level. It influences all new databases in that instance. If the instance is going to have multiple databases, not all of which are BLU-accelerated data marts, simply set the registry variable before you create the databases that you want optimized for analytic workloads, then unset the registry variable after these databases are created. Alternatively, follow the recommendations below for optimizing a pre-existing database, and apply only the database-level recommendations.

To optimize a pre-existing database by configuring the recommended settings, set the registry variable and then issue the AUTOCONFIGURE command. The following example shows you how to optimize the pre-existing database MYDB for analytics and BLU Acceleration:

```
db2set DB2_WORKLOAD=ANALYTICS
db2start
db2 connect to mydb
db2 autoconfigure apply db
```

After auto-configuration completes (either as part of the CREATE DATABASE command or an AUTOCONFIGURE command), review the sizes of the buffer pools, sort heap, sort heap threshold, and utility heap. You can fine tune the memory sizes based on your specific application needs and your expected data volumes. The following section provides more details about this type of review.

Database and database manager configuration

BLU Acceleration uses database shared sorts (the memory size for which is configured by the SHEAPTHRES_SHR database configuration parameter). In DB2 10.5 with BLU Acceleration, the self-tuning memory manager (STMM) does not adjust the values of the SHEAPTHRES_SHR and SORTHEAP database configuration parameters automatically. Instead, these parameters are set with static values by the AUTOCONFIGURE command when DB2_WORKLOAD=ANALYTICS.

At the instance level, the SHEAPTHRES configuration parameter is a soft target value that DB2 tries to maintain for all memory that is consumed by both private and shared memory-intensive operations, such as hash join and hash aggregation. This value is not just about sort memory, but about all “working memory”. Keep the instance-level SHEAPTHRES configuration parameter at its default value of 0, which leaves the tracking of sort memory consumption at the database level only (SHEAPTHRES_SHR).

However, at the database level, the value of the SHEAPTHRES_SHR parameter determines the limit for the allocation of sort memory. When a certain percentage of the value of SHEAPTHRES_SHR is attained, the DB2 database manager begins to throttle the memory that is allocated to sort memory consumers to prevent memory over-commitment, and those consumers might therefore receive only their minimum allocation requirement.

To ensure that aggregation-intensive operations, which are common in analytic workloads, have sufficient sort memory to perform optimally, take the following steps:

- Set the SHEAPTHRES_SHR parameter to 40-50% of the total memory that is available to the database, which is determined by the value of the DATABASE_MEMORY database configuration parameter.

- Set the SORTHEAP parameter to 5-20% of the value of the SHEAPTHRES_SHR parameter.

With increasing query concurrency and complexity, *decrease* the SORTHEAP to SHEAPTHRES_SHR ratio. In general, you can consider increasing the value of the SHEAPTHRS_SHR parameter, but beware of substantially increasing the value of the SORTHEAP parameter without understanding overall memory requirements across the entire workload.

Table 3 summarizes what the memory distribution within the database should be from a memory configuration standpoint. Your specific application and workload characteristics might suggest different values.

Table 3. Recommended memory distribution for a BLU-accelerated database as a function of workload concurrency

Low concurrency (< 20 concurrent workloads)	**High concurrency (> 20 concurrent workloads)**
40% buffer pool	25% buffer pool
40% SHEAPTHRES_SHR	50% SHEAPTHRES_SHR
SORTHEAP = SHEAPTHRES_SHR / 5	SORTHEAP = SHEAPTHRES_SHR / 20

Because of their complexity, analytic queries are more likely to benefit from an increased value of the STMTHEAP database configuration parameter, which specifies the size of the statement heap (a work space for the SQL compiler during compilation of an SQL statement). If your query returns SQL0437W with reason code 1, consider increasing the STMTHEAP value. Start by increasing the value of STMTHEAP by 50%; if the warning persists, continue increasing the value by the same percent increment, as needed.

DB2 10.5 sets the value of the UTIL_HEAP_SZ database configuration parameter to AUTOMATIC for new databases. Consider setting the UTIL_HEAP_SZ parameter to AUTOMATIC for existing databases. This setting enables the automatic resizing of the utility heap by DB2 load and other utilities.

Adopting BLU Acceleration

This section is all about implementing and using BLU Acceleration. It provides recommendations on how to build and load column-organized tables, details about column-organized tables and query execution, and guidelines for monitoring your database.

If you want to convert one or more row-organized tables in your database into column-organized tables, you can simply use the **`db2convert`** command to automate the process. This command calls the ADMIN_MOVE_TABLE stored procedure to perform the conversion, and shares its options. The **`db2convert`** command drops any dependent objects (such as secondary indexes or aggregates) as part of the conversion operation, as shown in the following example, because they are not required on column-organized tables:

```
$ db2convert -d mydb -z db2inst1 -t mytable

Conversion notes for exceptional table(s)
-----------------------------------------
 Table: DB2INST1.MYTABLE:
    -Secondary indexes will be dropped from the table.

Enter 1 to proceed with the conversion.
Enter 2 to quit.
```

Guidelines for creating column-organized tables

Creating a column-organized table is very easy. All you need is a table name, column names, and their data types. If you set the DFT_TABLE_ORG database configuration parameter to COLUMN, you do not need to specify the ORGANIZE BY COLUMN clause. Here is an example of the syntax to define a column-organized table:

```
CREATE TABLE mytable (
      c1 INTEGER NOT NULL,
      c2 INTEGER,
      …
      )
ORGANIZE BY COLUMN IN <table_space_name>
```

No additional specifications are required. The previous example includes the IN *table_space_name* clause. For improved manageability, it is recommended that you create fact tables in their own table space. For example, with a fact table in one table space and dimension tables in a different table space, you can have different storage groups and buffer pools for each table space (and its associated table type) as well as independent backup/restore fact tables. If there are enforced primary key constraints or unique constraints, the unique indexes that enforce those constraints should also be in their own table space, which you can specify by using the INDEX IN clause.

When possible, use NOT NULL as the default attribute for columns, except in cases where you actually need to store null values. Every nullable column in a column-organized table is actually a two-column group of columns: one for the nullability attribute and one for the actual column. These nullable columns cause extra work to be done on each row during query processing.

It is also useful to add primary key, informational foreign key, and any informational check constraints after your initial data has been loaded.

To determine whether a table is defined as row-organized or column-organized, you can query the SYSCAT.TABLES catalog view. A new column, TABLEORG, contains "C" for column-organized and "R" for row-organized. The following example shows you how to query the catalog view:

```
SELECT tabname, tableorg, compression
  FROM   syscat.tables
  WHERE  tabname like 'SALES%';


TABNAME                         TABLEORG COMPRESSION
------------------------------- -------- -----------
SALES_COL                       C
SALES_ROW                       R        N

  2 record(s) selected.
```

The COMPRESSION column for column-organized tables is blank because the data in such tables is always compressed.

You can define a primary key by using either the CREATE TABLE statement or an ALTER TABLE statement if the table exists, just as you would for row-organized tables. DB2 10.5 introduces an option for specifying not enforced (*informational*) primary key constraints or unique constraints. Because the NOT ENFORCED clause does not guarantee uniqueness, use this option only when you are certain that the data is unique. However, not enforced primary key constraints or unique constraints require significantly less storage and less time to create, because no internal B-tree structure is involved. Informational primary key, foreign key, and check constraints can be very beneficial to the query optimizer. If your data has been cleansed through the use of rigorous extract, transform, and load (ETL) procedures, we recommend that you take advantage of informational constraints, especially primary key constraints.

An *enforced* primary key constraint consumes space because a unique index is required to enforce the constraint. Like any index, this unique index increases the cost of insert, update, or delete operations. Verify that your application would benefit from and requires enforcement of primary key or unique constraints.

Synopsis tables and data skipping

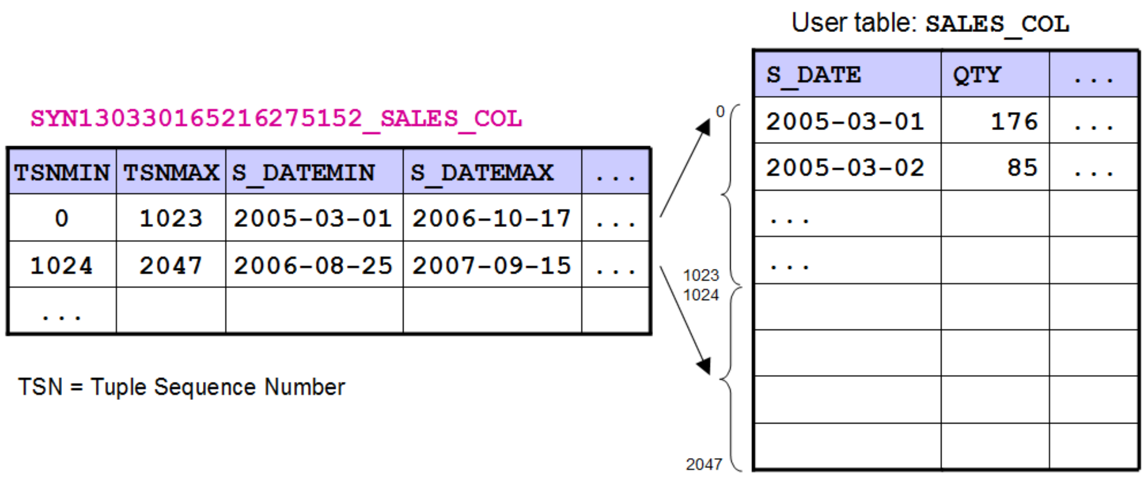
To improve scan performance, BLU Acceleration uses *data skipping*: skipping over data that is of no interest to a query. For example, suppose a query calculates the sum of last month's sales. There is no need to look at data that is older than the last month. With data skipping, BLU Acceleration can automatically skip over data that does not apply.

Internally, each column-organized user table has a companion metadata table, called a *synopsis table*. The synopsis table contains all of the user table's non-character columns (that is, datetime, Boolean, or numeric columns) and those character columns that are part of a primary key or foreign key definition. The synopsis table stores the minimum and maximum values for each column across a range of rows (identified by tuple sequence number[6], TSN) and uses those values for data skipping during query

[6] The TSN indicates which column values belong together as a logical row.

processing. A synopsis table is approximately 0.1% of the size of its user table. There is 1 row in the synopsis table for every 1024 rows in the user table.

Figure 1. A synopsis table and its relationship to a column-organized user table



TSNMIN	TSNMAX	S_DATEMIN	S_DATEMAX	...
0	1023	2005-03-01	2006-10-17	...
1024	2047	2006-08-25	2007-09-15	...
...				

S_DATE	QTY	...
2005-03-01	176	...
2005-03-02	85	...
...		
...		

You can determine the name of the synopsis table for a particular user table by querying the SYSCAT.TABLES catalog view, as shown in the following example:

```
SELECT tabschema, tabname, tableorg
  FROM syscat.tables
  WHERE tableorg = 'C';

TABSCHEMA    TABNAME                        TABLEORG
------------ ------------------------------ --------
DB2INST1     SALES_COL                          C
SYSIBM       SYN130330165216275152_SALES_COL    C
                                        2 record(s) selected.
```

It is important to understand how synopsis tables are used to optimize data skipping during the processing of specific queries. As mentioned, the synopsis table stores minimum and maximum values for all numeric, datetime, primary key, and foreign key columns. If you have a column that stores dates, it is best to use the DATE data type rather than character values so that the synopsis table enables effective data skipping.

Loading data

Column-organized tables are typically loaded from input data sources that are organized in row format. The data is compressed and converted into column-organized format during load or insert operations. A column-organized table is typically smaller than a comparable row-organized table because of better compression and less space required for auxiliary structures such as indexes or aggregates.

BLU Acceleration uses approximate Huffman encoding, prefix compression, and offset compression. Approximate Huffman encoding is a frequency-based compression that uses the fewest number of bits to represent the most common values. In other words, the most common values compress the most.

BLU Acceleration builds column compression dictionaries as part of the initial load operation on a column-organized table. Additional page-level dictionaries might be created to take advantage of local data clustering at the page level and to further compress the data. To optimize the compression dictionaries and ensure optimal compression of the data, ensure that the initial load (or dictionary creation) operation uses a representative sample of the data. However, new values that are not covered by column compression dictionaries can still be compressed by page-level dictionaries.

Although input data parsing, key options for the LOAD command, and general load semantics for column-organized tables and row-organized tables are the same, there are a few key differences:

- Loading data into column-organized tables has a new ANALYZE phase.
- Data is converted from row-organized input into fully formatted and compressed column-organized pages.
- The synopsis tables are maintained.

Table 4 outlines the activities that occur during several phases of a load operation into a column-organized table.

Table 4. Phases of the load process for column-organized tables

Load phase	**Activity**
ANALYZE	This phase occurs only if dictionaries are needed; histograms are built to track the frequency of data values in all columns, and column compression dictionaries are built.
LOAD	• Column and page compression dictionaries are used to compress the data. • Compressed values are written to data pages. • Synopsis tables are maintained.
BUILD	Any unique indexes for enforced primary key constraints are built. Although this phase always occurs for column-organized tables, in the case of row-organized tables, it depends on the existence of indexes.

A load operation into a column-organized table automatically collects statistics according to the profile that is defined for the table. If a profile does not exist, default automatic statistics (equivalent to the WITH DISTRIBUTION AND SAMPLED DETAILED INDEXES ALL clause) are collected by the RUNSTATS utility.

For optimal compression and load performance, there are two important considerations. First, the UTIL_HEAP_SZ database configuration parameter directly affects how many different values can be maintained during the compression dictionary build process. For example, if a histogram that uses memory from the utility heap can only contain 1000 different repeating column values, only those 1000 repeating values can be considered for compression, even though there could be thousands of different repeating values in a column, which could result in less efficient compression. It is crucial to review the UTIL_HEAP_SZ value. If the UTIL_HEAP_SZ parameter is set to AUTOMATIC, ensure that the database memory is large enough so that there is ample memory available for this parameter. Otherwise, update the UTIL_HEAP_SZ value to be as large as possible prior to a first load operation into a column-organized table (which builds the histograms and compression dictionary), and then reduce the value after the load operation completes. UTIL_HEAP_SZ can be updated dynamically.

Second, if possible, presort the data by columns that are frequently referenced by predicates that filter the fact table, or columns that are used to join highly filtered dimension tables. This type of presorting can improve compression ratios and improves the likelihood of data skipping (and better performance) by ensuring that a particular column value is clustered on fewer data pages.

To reduce the total time that is required to load large input data files into a column-organized table for the first time, you can reduce the duration of the ANALYZE phase by creating a column compression dictionary with only a portion of the input data file. This approach is outlined as follows:

1. Create the table. Do not create any primary keys at this time.

2. Obtain a subset of representative data from all of the data files that will be loaded into the database. If the data is already in a DB2 database (perhaps in a row-organized table), use a cursor with a sampling SELECT statement to obtain the sample.

3. Load the sample for the sole purpose of building the column compression dictionary, as in the following example:

```
LOAD FROM subset_data.csv OF DEL
  REPLACE RESETDICTIONARYONLY INTO mytable;
```

4. Load and compress the full set of source data by using this prebuilt column compression dictionary, as in the following example:

```
LOAD FROM all_data.csv OF DEL
```

```
INSERT INTO mytable;
```

Step 4 does not trigger the ANALYZE phase. Repeat the LOAD command until all of the data has been loaded.

5. Create enforced or informational primary key constraints and informational foreign key constraints.

6. Issue the RUNSTATS command to ensure a complete view of the table that was populated through this multistep process.

This approach reduces the total load time but can also reduce the compression efficiency if the sampled data is not a representative subset of the entire data set.

The new PCTENCODED column in the SYSCAT.COLUMNS catalog view represents the percentage of values that are encoded as a result of compression for a column in a column-organized table. If the overall compression ratio for your column-organized table is too low, check this statistic to see if values in specific columns were left uncompressed. If you see many columns with a very low value (or even 0) for PCTENCODED, the utility heap might have been too small when the column compression dictionaries were created. You might also see very low values for columns that were incrementally loaded with data that was outside of the scope of the column compression dictionaries.

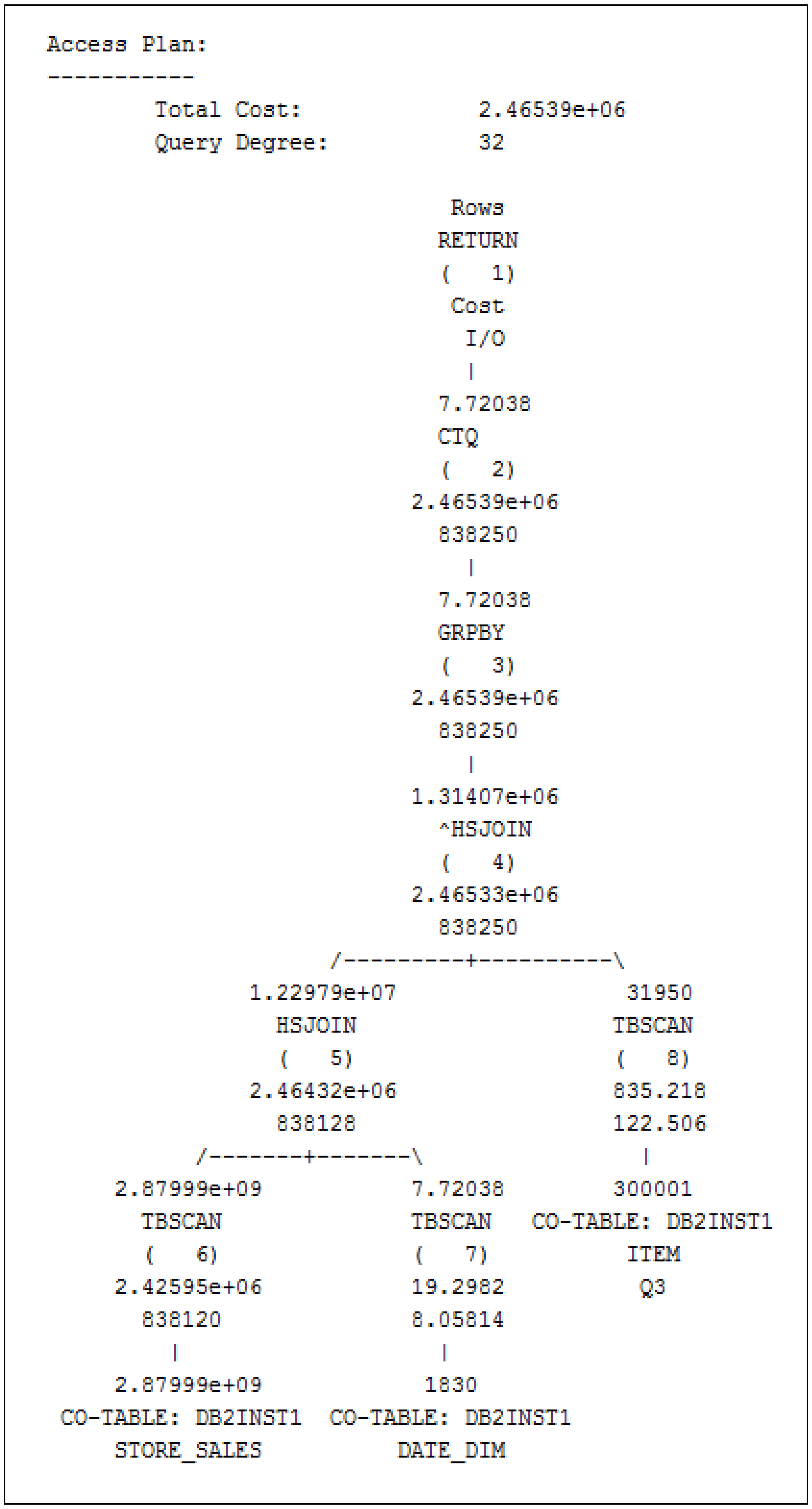
Query execution plans and the new CTQ operator

Access methods and join enumeration are greatly simplified with BLU Acceleration, because BLU Acceleration generally considers table scans, hash joins, and hash aggregation. Join order is still dictated by cost, so cardinality estimation is still important. Collecting column group statistics, for example, is helpful for both column-organized and row-organized tables.

Defining informational primary key constraints and unique constraints whenever possible is an excellent strategy for providing the optimizer with additional information that would not be available otherwise. This information is used to obtain good query execution plans for BLU-accelerated tables.

The new CTQ operator for query execution plans is used to indicate the transition between column-organized data processing (BLU Acceleration) and row-organized data processing. All operators that appear below the CTQ operator in an execution plan are optimized for column-organized tables. In good execution plans for column-organized tables, the majority of operators are below the CTQ operator, and only a few rows flow through the CTQ operator (Figure 3).

Figure 3. An example of a good query execution plan for column-organized tables



```
Access Plan:
------------
        Total Cost:             2.46539e+06
        Query Degree:           32

                              Rows
                             RETURN
                             (   1)
                              Cost
                               I/O
                               |
                             7.72038
                             CTQ
                             (   2)
                           2.46539e+06
                             838250
                               |
                             7.72038
                             GRPBY
                             (   3)
                           2.46539e+06
                             838250
                               |
                           1.31407e+06
                             ^HSJOIN
                             (   4)
                           2.46533e+06
                             838250
                     /---------+----------\
               1.22979e+07                 31950
                 HSJOIN                   TBSCAN
                 (   5)                   (   8)
               2.46432e+06                835.218
                 838128                   122.506
           /-------+-------\                |
     2.87999e+09           7.72038        300001
       TBSCAN              TBSCAN   CO-TABLE: DB2INST1
       (   6)              (   7)          ITEM
     2.42595e+06           19.2982          Q3
       838120              8.05814
         |                   |
     2.87999e+09            1830
 CO-TABLE: DB2INST1  CO-TABLE: DB2INST1
     STORE_SALES          DATE_DIM
```

There is comprehensive support for explain in DB2 10.5 with BLU Acceleration, not just with the engine tools (**`db2expln`** and **`db2exfmt`**), but also with a new version of the Optim Performance Manager and the Optim Query Workload Tuner.

Database maintenance

Automatic statistics collection is enabled by default for a database that contains either row-organized or column-organized tables. Because automatic statistics collection is also enabled by default during a load operation into a column-organized table, an explicit RUNSTATS operation might not be necessary, particularly in the case of single-step load operations.

Space is automatically reclaimed for column-organized tables when DB2_WORKLOAD=ANALYTICS or if the AUTO_REORG database configuration parameter is set to ON. If neither of these conditions is true, you can use a REORG TABLE…RECLAIM EXTENTS command to reclaim free space. To determine whether there is sufficient space to reclaim, you can query the RECLAIMABLE_SPACE column in the ADMINTABINFO administrative view. The following sample SQL returns the amount of reclaimable space for the column-organized table named MYTABLE:

```
SELECT SUBSTR(a.tabname,1,20) AS tabname, RECLAIMABLE_SPACE
  FROM SYSIBMADM.ADMINTABINFO
  WHERE tabname LIKE 'mytable%'
  ORDER BY tabname WITH UR
```

All other maintenance activities that involve column-organized tables, such as backup and restore, are identical to those that involve row-organized tables.

Concurrency control (workload management)

DB2 10.5 automatically controls the resource consumption of running queries. Although a large number of queries can be submitted concurrently, the number of queries that are allowed to execute at one time is limited to optimize the amount of memory and CPU resource that are available on the server for each running query.

DB2 10.5 introduces a new service subclass named SYSDEFAULTMANAGEDSUBCLASS. Any READ DML (a work type for work classes) queries whose estimated cost is greater than the default timeron threshold of 150000 run in the SYSDEFAULTMANAGEDSUBCLASS instead of the SYSDEFAULTSUBCLASS.

When DB2_WORKLOAD=ANALYTICS, the default concurrency threshold on the SYSDEFAULTMANAGEDSUBCLASS service subclass is enabled by default. The default concurrency threshold allows a fixed number of queries into the system at a time, specifically those queries whose estimated cost is greater than the default timeron threshold. The fixed number is based on a heuristic calculation that factors in system hardware attributes such as the number of CPU sockets, CPU cores, and threads per core. The limit is calculated based on the server configuration at the time of the CREATE DATABASE command. If changes are made to the server configuration, the concurrency threshold can be recalculated by running the AUTOCONFIGURE command against the database, as outlined in the following steps:

1. Issue the AUTOCONFIGURE command:

```
AUTOCONFIGURE APPLY NONE
```

2. Review the output, in particular the "Current and Recommended Values for System WLM Objects" section.

```
   Current and Recommended Values for System WLM Objects

Description                                   Current Value  Recommended Value
-----------------------------------------------------------------------------
Work Action SYSMAPMANAGEDQUERIES Enabled    = Y              Y
Work Action Set SYSDEFAULTUSERWAS Enabled   = Y              Y
Work Class SYSMANAGEDQUERIES Timeroncost    = 1.50000E+05    1.50000E+05
Threshold SYSDEFAULTCONCURRENT Enabled      = Y              Y
Threshold SYSDEFAULTCONCURRENT Maxvalue     = 16             18
```

   Use the recommended value for "Threshold SYSDEFAULTCONCURRENT Maxvalue" in place of *maxvalue* in the following SQL statement.

3. Issue the following ALTER THRESHOLD statement:

```
ALTER THRESHOLD SYSDEFAULTCONCURRENT
  WHEN CONCURRENTDBCOORDACTIVITIES > maxvalue
  AND QUEUEDACTIVITIES UNBOUNDED
  STOP EXECUTION
```

If necessary, you can define additional DB2 workload management controls to meet the service-level agreement (SLA) requirements of the workload.

Monitoring

DB2 10.5 introduces new monitor elements for column-organized tables. When you monitor database performance in a BLU-accelerated environment, you are seeking the answers to several key questions. The following sections address each of these questions and make recommendations about which monitor elements are relevant.

Is the sort heap adequate for good performance?

Review the monitor elements in Table 5 to assess how the sort heap is being used.

Table 5. Monitor elements that are relevant to column-organized tables, grouped by the areas that they monitor

Area	**Elements**
Sort heap	SORT_HEAP_ALLOCATED, SORT_SHRHEAP_ALLOCATED, SORT_SHRHEAP_TOP
Sort	ACTIVE_SORTS, SORT_OVERFLOWS

Join	ACTIVE_HASH_JOINS, TOTAL_HASH_JOINS, HASH_JOIN_OVERFLOWS, HASH_JOIN_SMALL_OVERFLOWS, POST_SHRTHRESHOLD_HASH_JOINS, POST_THRESHOLD_HASH_JOINS
Group by	ACTIVE_HASH_GRPBYS, TOTAL_HASH_GRPBYS, HASH_GRPBY_OVERFLOWS, POST_THRESHOLD_HASH_GRPBYS

"Overflows" indicate that a query could not obtain all of the sort memory that it needed, which can be the result of a SORTHEAP setting that is too small, or a result of attaining the SHEAPTHRES_SHR threshold because the SORTHEAP setting is too large for the level of concurrency in the workload. Review the value that is returned by the SORT_SHRHEAP_TOP monitor element and the SHEAPTHRES_SHR database configuration parameter setting to determine which case is applicable.

"Post threshold sorts" indicate sorts that are throttled below the SORTHEAP setting. "Small overflows" indicate an overflow of 10% or less.

Overflows or post threshold sorts might indicate the need to increase the SORTHEAP setting to reduce spills and improve performance, or the need to reduce the SORTHEAP setting relative to the SHEAPTHRES_SHR setting because of higher concurrency in the workload. In general, it is typically better to increase SHEAPTHRES_SHR rather than SORTHEAP, in order to improve overall workload performance.

Is the table suitably organized?

By dividing the NUM_COLUMNS_REFERENCED monitor element by the SECTION_EXEC_WITH_COL_REFERENCES monitor element, you can determine the average number of columns that a particular query accesses. If this average is much less than the number of columns in the table, the workload favors column organization.

How well are queries performing?

New monitor elements were added to measure specific aspects of column-organized data processing. These measurements include elapsed time (reported by the TOTAL_COL_TIME element), processing time excluding lock wait times, I/O, and other items (reported by the TOTAL_COL_PROC_TIME element), and the number of times the CTQ operator is accessed (reported by the TOTAL_COL_EXECUTIONS element).

You can use these elements in the context of other "time-spent" monitor elements to determine how much time was spent, per thread, performing column-organized data processing. For example, if you want to know what portion of a query was executed in column-organized form and what portion was executed in row-organized form, you can compare TOTAL_COL_TIME to TOTAL_SECTION_TIME. A large ratio suggests an optimal execution plan for BLU Acceleration.

For more information about the time-spent monitor elements, see http://pic.dhe.ibm.com/infocenter/db2luw/v10r5/index.jsp?topic=%2Fcom.ibm.db2.luw.admin.mon.doc%2Fdoc%2Fc0056890.html.

How is the buffer pool performing?

New versions of existing monitor elements (for example, COL_HIT_RATIO_PERCENT and COL_PHYSICAL_READS) were added to enable you to monitor buffer pool usage by column-organized tables.

How are the prefetchers performing?

New versions of existing monitor elements (for example, POOL_QUEUED_ASYNC_COL_PAGES and SKIPPED_PREFETCH_COL_P_READS) were added to enable you to monitor prefetching for column-organized tables.

Conclusion

The best practices that are presented in this paper are intended to get you up and running with BLU Acceleration, a new technology for analytic queries that is introduced in DB2 for Linux, UNIX, and Windows Version 10.5.

The paper began with an overview of the cornerstones of this technology, including column-organized tables, actionable compression, parallel vector processing, and data skipping.

This was followed by recommendations pertaining to hardware and software selection, guidelines for identifying the optimal workloads for BLU Acceleration, and information about capacity planning, memory, and I/O.

A section on system configuration included information on how to have DB2 automatically set up an optimal default configuration for analytic workloads with one easy step, plus information on how to derive the greatest benefits from the memory that is available on your system.

The remaining sections told you how to implement and best use DB2 with BLU Acceleration. We included the most important information about column-organized tables and told you how to ensure that synopsis tables are performing optimally. These sections also gave you best practices for loading data, and information that will help you to keep your system running at top performance levels.

By leveraging these best practices, you can use DB2 10.5 with BLU Acceleration to gain benefits for your enterprise by performing *analytic processing that is super fast and super easy – just load and go!*

Further reading

- Information Management best practices: http://www.ibm.com/developerworks/data/bestpractices/
- DB2 for Linux, UNIX, and Windows best practices: http://www.ibm.com/developerworks/data/bestpractices/db2luw/
- DB2 with BLU Acceleration: http://www-01.ibm.com/software/data/db2/linux-unix-windows/db2-blu-acceleration/
- "*Super Analytics, Super Easy - Introducing IBM DB2 10.5 with BLU Acceleration*": http://ibmdatamag.com/2013/04/super-analytics-super-easy/
- DB2 10.5 Information Center: http://pic.dhe.ibm.com/infocenter/db2luw/v10r5/index.jsp

Contributors

Significant contributions were made to the content of this paper by the following persons:

Chris Eaton
WW IM Technical Sales and Competitive Specialist, IBM Canada Lab

David Kalmuk
Architect, DB2 LUW Workload Management, Monitoring, Process Model, IBM Canada Lab

Calisto Zuzarte
Senior Technical Staff Member, IBM Canada Lab

Acknowledgements

We thank the following people whose valuable input helped to improve the quality of this paper:

Gopi Attaluri

Serge Boivin

Eric Koeck

Nela Krawez

Christina Lee

Sam Lightstone

Mathias Nicola

Huaxin Zhang

Notices

Trademarks

Contacting IBM

To provide feedback about this paper, write to db2docs@ca.ibm.com

To contact IBM in your country or region, check the IBM Directory of Worldwide Contacts at http://www.ibm.com/planetwide

To learn more about IBM Information Management products, go to http://www.ibm.com/software/data/