DB2 with BLU Acceleration: A rapid adoption guide

Naresh Chainani (naresh@us.ibm.com)
Senior Software Engineer
IBM

Weilin Lu (luw@us.ibm.com)
Advisory Software Engineer
IBM

Alexandria Burkleaux (burkleau@us.ibm.com)
Manager
IBM

Matthias Nicola (mnicola@us.ibm.com)
Senior Technical Staff Member
IBM Silicon Valley Lab

19 September 2013

You have probably heard how DB2® with BLU Acceleration can provide performance improvements ranging from 10x to 25x and beyond for analytical queries with minimal tuning. You are probably eager to understand how your business can leverage this cool technology for your warehouse or data mart. The goal of this article is to provide you with a quick and easy way to get started with BLU. We present a few scenarios to illustrate the key setup requirements to start leveraging BLU technology for your workload.

Introduction

In this article, we introduce DB2 with BLU Acceleration and demonstrate how you can start leveraging BLU technology for your applications and data. We walk you through several scenarios for adopting BLU:

- Upgrading from a previous DB2 release to DB2 10.5 to use BLU
- Start using BLU in an existing DB2 10.5 database
- Start using BLU with a new database on DB2 10.5
- Methods to create and populate column-organized tables
  - `CREATE TABLE` and `LOAD`
  - Convert row-organized table(s) to column-organized table(s)
  - Workload Table Organization Advisor

For each scenario, examples illustrate the ease with which BLU can be integrated into your business. We also show you the basics of configuring and evaluating the performance of your system. Finally, we explain some of the principles behind the technology to help you get the best compression and performance for your workload via good utilization of your hardware resources. This article is based on BLU capabilities as of DB2 10.5.0.1 (10.5 Fix Pack 1).

What is BLU Acceleration?

BLU Acceleration technology combines ease of use with unprecedented storage savings and performance acceleration for analytic workloads. This section gives an overview of the innovative in-memory, CPU, and I/O optimizations behind the BLU Acceleration technology.

Some of the key ideas that constitute BLU Acceleration technology include:

Ease of use: BLU Acceleration is designed to be very simple to use. From a DBA's perspective, you simply load and go! There are no secondary objects, such as indexes or MQTs, that need to be created to improve query performance. BLU Acceleration provides good out-of-the-box performance for analytic workloads.

Column-organized tables: Column-organized tables store each column on a separate set of pages on disk. Organizing data by column on disk reduces the amount of I/O needed for processing queries because only columns that are referenced in a query need to be accessed. Columnar organization favors analytic queries that access a large number of values from a subset of the columns and make heavy use of aggregations and joins. Only the column data needed to process the query is loaded into memory from disk. By materializing columns as late as possible, BLU takes advantage of filtering from predicates, leading to significant I/O savings. Finally, unlike some other vendors, BLU does not require all the active data to be loaded in memory before it can start executing the SQL.

Actionable compression: Column-organized tables are automatically compressed using an efficient compression algorithm. This industry leading compression results in significant storage savings over traditional database compression technologies. The DB2 query processing engine has the capability to evaluate predicates and perform complex operations like joins and aggregation on compressed data. This contributes to better query performance and allows for efficient use of memory because more compressed data can be stored in memory.

Data skipping: DB2 skips reading ranges of column values that are not needed to satisfy a query. Data skipping utilizes a small data structure called a synopsis table. The synopsis table is created and maintained automatically for column-organized tables and doesn't require any configuration or maintenance to use.

CPU optimized: BLU takes advantage of single instruction multiple data (SIMD) processing, if provided by the hardware. SIMD allows for the same instruction to be applied to multiple pieces of data in parallel at the hardware level. Additionally, BLU algorithms are designed to scale with the number of cores available in the system.

Scan friendly memory caching: Massive compression, data skipping, and late materialization means less data is needed in memory in order to process a query. When a query can't be processed entirely in memory and I/O is needed, BLU utilizes a memory caching algorithm that is optimized for scans to reduce the number of times specific pages are read from disk.

Seamlessly integrated: BLU Acceleration is fully integrated into DB2. This means that regardless of table organization, your applications use the same SQL interfaces, backup and recovery scripts, and `LOAD` and `EXPORT` commands that you are familiar with.

BLU Acceleration exploits advanced hardware capabilities where available and at the same time it is flexible to function without them to leverage existing hardware you may have.

Ideal workload characteristics for BLU

The BLU Acceleration feature is intended for analytic or data mart workloads. Such workloads typically involve regular reporting queries as well as ad-hoc business intelligence queries that can't be tuned in advance. If your workload is primarily transaction processing, you may want to consider using row-organized tables. The following table identifies some of the workload characteristics that are optimal for column-organized tables and others that are well-suited for row-organized tables:

Table 1. Table organization based on workload attributes

Use column-organized tables	Use row-organized tables
Analytical workloads, data marts	Online transaction processing workloads
Queries that contain grouping, aggregation, range scans, or joins	Queries that access a single row or few rows
Queries that access a subset of table columns	Applications that insert or update less than 100 rows within a single transaction
Star schema based database design	Queries that access the majority or all of the columns within a table
SAP Business Warehouse application	Applications that use XML, temporal, or LOB data within tables

When you work with column-organized tables, you need to consider some environment and application restrictions before you attempt to move your application to BLU. For more information, refer to the following DB2 Information Center topics:

- Column organized tables
- CREATE TABLE statement
- INSERT, UPDATE, and DELETE (including MERGE) restrictions for column-organized tables

System requirements

In DB2 10.5, BLU Acceleration is supported on following operating systems:

- Linux 64-bit on Intel/AMD hardware
  - RHEL 6, SLES 10 SP4, SLES 11 SP2, or higher
- AIX on Power hardware

    - AIX 6.1 TL7 SP6, AIX 7.1 TL1 SP6, or higher
- Refer to the relevant section for your platform here for general DB2 installation prerequisites.

System sizing guidelines

The design of DB2 BLU Acceleration with respect to SIMD (Single Instruction Multiple Data), multi-core parallelism exploitation, and the volume of data being processed, favors a larger core count. BLU algorithms are designed to effectively leverage large memory configurations. A good general rule of thumb is:

- For production use, a minimum of 8 cores and 64GB of RAM are recommended.
- As your system scales, make sure BLU has at least 8 GB of main memory per core.
- Consider storing column-organized tables on I/O subsystems that exhibit good random I/O characteristics particularly when active table data exceeds memory size.
- With increasing data volumes and greater query complexity, your BLU-accelerated workload would benefit from a larger number of cores and more memory.

Scenarios to adopt BLU

In this section, we describe multiple scenarios to adopt BLU Acceleration on DB2 10.5 depending on your starting point. For instance, you may be upgrading from an older DB2 release or you may already be on DB2 10.5 and want to leverage BLU for some of your tables. Jump directly to the scenario that applies to your situation:

- Scenario 1: Upgrading from a previous DB2 release to DB2 10.5 to use BLU
- Scenario 2: Start using BLU in an existing DB2 10.5 database
- Scenario 3: Start using BLU with a new database on DB2 10.5

Scenario 1: Upgrading from a previous DB2 release to DB2 10.5 to use BLU

You can upgrade databases created in DB2 versions 9.7, 9.8, and 10.1 to DB2 version 10.5. A database that was created prior to DB2 9.7 must be upgraded to DB2 versions 9.7 or 10.1 before it can be upgraded to DB2 10.5.

Pre-upgrade checks

- The existing database must use `UNICODE` with `IDENTITY` or `IDENTITY_16BIT` collation to leverage BLU Acceleration. There is no option to change the collation of an existing database. In this case, you must recreate the database with the correct collation.
- Run the `db2ckupgrade` tool from your DB2 10.5 instance to verify if your database is ready to be upgraded to DB2 10.5:

```
db2ckupgrade edwdb -l upgrade.log
```

- If the database is ready for upgrade, you should see this message returned from db2ckupgrade:

DBT5508I The db2ckupgrade utility completed successfully. The database or databases can be upgraded.

- Take an offline full database backup.

Upgrade DB2 instance

- Run `db2iupgrade` to upgrade the existing DB2 instance to DB2 10.5:

```
db2iupgrade pdxdb2
```

Upgrade databases

There are 2 options that are commonly used to upgrade databases:

- Option 1: In-place upgrade using the `UPGRADE DATABASE` command

```
upgrade database edwdb rebindall
```

DB20000I The UPGRADE DATABASE command completed successfully.

- Option 2: Side-by-side upgrade using the `RESTORE` command

Restore your database from the backup image taken on an older DB2 release. At the end of the restore operation, the database will automatically be upgraded.

```
restore database edwdb from /backupfs taken at 20130626183207
```

SQL2555I The database was restored and then successfully upgraded to the current DB2 release where you issued the RESTORE DATABASE command.

You are now ready to tune the upgraded database for leveraging BLU technology in just a few steps.

Scenario 2: Start using BLU in an existing DB2 10.5 database

DB2 supports creating row-organized tables and column-organized tables in the same database. In fact, the same query *can* include both table types, although for optimal performance, we recommend joining column-organized tables with other column-organized tables.

If you already have an existing `UNICODE` database on DB2 10.5 with `IDENTITY` or `IDENTITY_16BIT` collation and a subset of tables are frequently accessed in analytic queries, you might want to create these tables as column-organized tables. In another use case, you might want to deploy a new analytic application that takes advantage of BLU Acceleration. In order to do so, tune the database for analytics by using the following simple steps:

Setup environment

1. db2set db2_workload=analytics: This registry variable indicates that the database is intended to be used for an analytic workload. When you subsequently run the `AUTOCONFIGURE` command, DB2 automatically configures the database for optimal analytic performance relative to the capacity of the hardware. This one-touch tuning saves you from having to manually tune the database.

2. Restart the instance using `db2stop force` followed by `db2start` and connect to your database.
3. autoconfigure apply db and dbm: This command configures the database for analytics when it is run after `DB2_WORKLOAD` is set to `ANALYTICS`. To only review the configuration changes that the command makes, issue `AUTOCONFIGURE APPLY NONE`.

Enable automatic storage

With the trend toward autonomics, only automatic storage table spaces are supported for column-organized tables. The next few commands demonstrate how to start using automatic storage. You can skip them if you are already familiar with automatic storage table spaces.

- If you are not already using automatic storage table spaces, you can get started by issuing the `CREATE STOGROUP` command:
```
create stogroup mystg on '/dbfs1' set as default
```

- To convert existing DMS table spaces to use automatic storage, issue the two `ALTER TABLESPACE` commands:
```
alter tablespace june_tbsp managed by automatic storage
alter tablespace june_tbsp rebalance
```

- To create new automatic storage table spaces, issue the `CREATE TABLESPACE` command. Because this table space will be used to store column-organized table data, use the recommended page size of 32K and the extent size of 4 pages.
```
create tablespace col_tbsp pagesize 32k extentsize 4
```

Congratulations. Now you are ready to create column-organized tables in your database. Refer to the methods defined in Methods to create and populate column-organized tables.

Scenario 3: Start using BLU with a new database on DB2 10.5

Creating a new database is the typical approach used during testing or a proof of concept, as it provides a fast path to experience BLU technology.

Setup environment

1. db2set db2_workload=analytics: This registry variable indicates that the database is intended to be used for an analytic workload.
2. Restart the instance using `db2stop force` followed by `db2start`.

Create a new Unicode database

```
create database edwdb using codeset utf-8 territory us collate using identity
```

Create automatic storage table spaces

```
create tablespace col_tbsp
```

Because the database was created after setting DB2_WORKLOAD=ANALYTICS, the default page size used for the table space is 32K and the extent size is 4 pages.

With these simple steps, you are ready to create column-organized tables in your database. Refer to the methods defined in Methods to create and populate column-organized tables.

Methods to create and populate column-organized tables

In this section, we describe how to create new column-organized tables or convert an existing row-organized table to a column-organized table. If the database was configured with DB2_WORKLOAD=ANALYTICS, newly created tables will be column-organized by default. Before creating column-organized tables, make sure that the characteristics of your workload align with the attributes discussed in the Ideal workload characteristics for BLU section. The Optim Query Workload Tuner (OQWT) can be used to identify tables that might be good candidates for being column-organized tables. Based on these recommendations, you can choose to create some or all tables as column-organized tables using one of the following methods:

Method 1: CREATE TABLE and LOAD

Create new tables with the same DDL as row-organized tables, and load the table data into the new table:

```
create table sales_col (tid bigint not null, saledate date,
                        saleprice decimal(12,2)) in col_tbsp organize by
                        column
```

The ORGANIZE BY COLUMN clause is optional if the database configuration parameter DFT_TABLE_ORG is set to COLUMN, which is the case when the database is configured for analytics. To create row-organized tables in such a database, specify the ORGANIZE BY ROW clause.

```
load from sales.del of del replace into sales_col
```

In addition to loading the table data, this initial load will automatically build the columnar compression dictionaries, synopsis table and collect statistics. For optimal compression, we recommend loading enough data (several gigabytes) that contains a good representative subset of data values. In cases with strongly unsorted input data, the effectiveness of data skipping can sometimes be improved by presorting the data being loaded on numeric or date/time column(s) that are frequently used in query predicates.

Method 2: Convert row-organized table(s) to column-organized table(s)

Use the db2convert command to convert one or all row-organized tables in a database into column-organized tables. The row-organized tables remain online during this conversion. The db2convert command invokes the ADMIN_MOVE_TABLE stored procedure. This conversion is one-way only, so be sure to back up the database or table spaces(s).

```
db2convert –d edwdb –z blu –t sales -ts col_tbsp
```

Convert the blu.sales table in the edwdb database from row-organized to column-organized format. Note that the -z option specifies the schema for the table and the -ts option species the table space in which the column-organized table should be created.

```
db2convert –d edwdb
```

Convert all tables in the edwdb database into column-organized tables. Upon successful conversion, `db2convert` returns:

SQL2446I The db2convert command completed successfully. All row-organized tables that satisfy the specified matching criteria have been converted to column-organized tables.

Refer here for more details on db2convert.

Method 3: Workload Table Organization Advisor

The OQWT can be used to identify tables that might be good candidates for being column-organized tables. OQWT bases its advice on table statistics and the query workload characteristics. It even indicates expected performance improvement for queries in the workload. After OQWT makes its recommendations for tables that would benefit from being column-organized, you can decide if you want to execute the script generated by the tool to implement the recommendations. Make sure to back up the database prior to making these changes. Refer to the tool documentation for more details.

At this time, you should be able to measure the compression savings and start running queries without any change to SQL. Indeed, there is no need for indexes or MQTs to be created. The only indexes for column-organized tables are the ones used for enforced primary keys or unique constraints. If the analytic data is already cleansed during ETL processing, create non-enforced constraints to allow DB2 to leverage this information.

Frequently asked questions

Before we proceed with measuring compression savings and performance improvement with column-organized tables, let's go over some questions that often come up at this stage.

- Q: What magic does `DB2_WORKLOAD=ANALYTICS` do under the covers?
  A: The idea behind this setting is to provide a simple way to configure the database for analytic workloads. With this setting, among other things, newly created tables are column-organized by default, intraquery parallelism is enabled, sort parameters like `SORTHEAP` and `SHEAPTHRES_SHR` are computed, the utility heap (`UTIL_HEAP_SZ`) is configured for loading data into column-organized tables, and the workload management concurrency threshold is enabled. If the database is created after this setting is in effect, the default database page size is set to 32K and the default extent size is set to 4 pages.

- Q: I have an existing database. Can I still benefit from the one-touch tuning provided by `DB2_WORKLOAD=ANALYTICS`?

A: Yes. After setting `DB2_WORKLOAD=ANALYTICS`, run the `AUTOCONFIGURE` command to get most of the settings configured. During table space creation, explicitly specify 32K page size and 4 extent size since the database level settings are likely different. To avoid having to specify the extent size on every table space creation, you could update `DFT_EXTENT_SZ` to 4 instead.

- Q: I understand `DB2_WORKLOAD=ANALYTICS` is recommended for BLU, but what if my SAP BW environment has `DB2_WORKLOAD` set to `SAP`?
  A: In order to avoid any impact to your SAP BW application environment, continue to set `DB2_WORKLOAD` to `SAP` and manually configure the database for analytics as described here.
- Q: How do I verify which tables are row-organized and which are column-organized?
  A: Query the `TABLEORG` column of the `syscat.tables` catalog view to verify if a table is row-organized (`TABLEORG=R`) or column-organized (`TABLEORG=C`):
  ```
  select tableorg from syscat.tables where tabname='SALES'
  ```
- Q: What is BLU an acronym for?
  A: BLU is not an acronym. It was an internal project name that became popular. One satisfied user has suggested Big Data, Lightning Fast, Ultra Easy.
- Q: How do I incrementally refresh the table data for my column-organized tables?
  A: Depending on the table availability requirements, you can add new data to an existing column-organized table either via the `INGEST` utility or `LOAD` with `INSERT` option. The `INGEST` utility allows for greater concurrency with ongoing SQL activity on the target table, which can be important to meet certain requirements such as continuous and real-time data refresh in a warehouse. If you use `IMPORT, INGEST` or `INSERT`, we recommend inserting greater than 1000 rows per transaction in order to amortize logging and other overheads.
- Q: Is data that is loaded after the initial `LOAD` compressed?
  A: Yes, data that is incrementally added to the table via `LOAD, INSERT, IMPORT`, or `INGEST` operations is compressed using the column-level dictionaries that the initial `LOAD` operation created. Additionally, BLU also uses page-level compression to compress new values that might not be represented in the column-level compression dictionary. This two-level approach of column-level and page-level compression is a proven technique to effectively address "data drift" and avoid deteriorating compression over time.

Measuring storage savings

Now that you have loaded some data into column-organized tables, you are probably eager to evaluate the storage savings with BLU compression. The following example query can be used to get the total on-disk footprint for the column-organized tables versus the same data loaded into row-organized tables:

```
select
    substr ( a.tabname,1,20 ) as tabname,
    rowcompmode, pctpagessaved, data_object_p_size,
    index_object_p_size, col_object_p_size,
    (data_object_p_size+index_object_p_size+col_object_p_size) as total_size_in_kb
from syscat.tables a, sysibmadm.admintabinfo b
where
    a.tabname = b.tabname and
    a.tabname like 'SALES%' and -- replace SALES with your table name
    a.tabschema = b.tabschema
with ur
```

In our setup, SALES_ROW is an uncompressed row-organized table while SALES_COL is the corresponding column-organized table. Figure 1 shows the output of the previous query:

Figure 1. Output of the previous query

```
TABNAME             ROWCOMPMODE PCTPAGESSAVED DATA_OBJECT_P_SIZE   INDEX_OBJECT_P_SIZE  COL_OBJECT_P_SIZE    TOTAL_SIZE_IN_KB
------------------- ----------- ------------- -------------------- -------------------- -------------------- --------------------
SALES_COL                                  84                  512                  768                84736                86016
SALES_ROW                                   0               435200                    0                    0               435200
```

In our example, the row-organized table consumes 435200KB of disk space while the same data loaded into a column-organized table consumes 86016KB, leading to a storage saving of 84%. Note that the `PCTPAGESSAVED` statistic is relative to uncompressed row-organized data. For a row-organized table, `DATA_OBJECT_P_SIZE` and `INDEX_OBJECT_P_SIZE` represent the physical size of the data object and index object respectively while `COL_OBJECT_P_SIZE` is not applicable and always 0. For a column-organized table, `DATA_OBJECT_P_SIZE` represents the physical size for some of the metadata for the table, `INDEX_OBJECT_SIZE` represents the physical size of the index object, while the `COL_OBJECT_P_SIZE` represents the physical size of the columnar data. In your tests, the row-organized table could be compressed using static compression (`ROWCOMPMODE=S`) or adaptive compression (`ROWCOMPMODE=A`). Column-organized tables typically yield storage savings of 10x over uncompressed row-organized tables and 2-3x over static or adaptive compressed row-organized tables.

In many data centers, the storage subsystem cost dominates due to large data volumes and functionality like mirroring, replication, etc. So if you are already excited about the significant storage savings you are seeing with adaptive compression, look forward to doubling these savings with column-organized tables. Another direct impact of reducing the on-disk footprint is faster backup, restore, and smaller backup image(s).

Troubleshooting tips for compression

In some cases, your test results may not demonstrate the storage savings previously mentioned. There could be several factors impacting the storage savings, and here are few tips to help you figure out what might be going on:

- The compression dictionaries are built when data is initially loaded. If the initial data that was loaded was very small or not representative of the overall table data, the compression dictionaries will be sub-optimal. Issue `LOAD WITH REPLACE` to repopulate the table and rebuild the compression dictionaries.
- The table was populated using `INSERT, INGEST`, or `IMPORT` commands. Repopulate the tables with the `LOAD` utility for achieving the best compression.
- The utility heap (`UTIL_HEAP_SZ`) is too small. Check if the `PCTENCODED` statistic in `SYSCAT.COLUMNS` reports a low number (see example query below) even though you feel the column is a good candidate for compression. This per-column statistic reports the percentage of column values that are encoded or compressed. For optimal compression, `LOAD` needs enough working memory to analyze the data being loaded. In most cases, the utility heap should be configured automatically when the database is configured for analytics. If for some

reason you tuned it manually, increase it to be at least 1000000 pages by running `'update db cfg using util_heap_sz 1000000 automatic')`. Setting `UTIL_HEAP_SZ` to `AUTOMATIC` allows utilities to consider database memory overflow as part of available utility heap in addition to the underlying configured value. If memory is constrained in your test environment, you may temporarily increase the configured utility heap value for the duration of the initial `LOAD` operation.

```
select substr(colname,1,20), substr(typename,1,20), length, pctencoded
from syscat.columns
where tabname='SALES_COL' and tabschema='BLU'
```

Query performance

By this time, you have probably computed the dollar savings in storage costs that your business can achieve by leveraging DB2 BLU with its innovative compression techniques. The DB2 BLU query execution engine takes this further by working on compressed data. To the extent possible, predicate evaluation and complex operations like joins and grouping work on compressed data. Uncompressing as late as possible allows for efficient query execution and effective memory utilization yielding query performance gains ranging from 10x to 25x and more with a wide variety of analytic workloads. DB2 BLU is optimized for running analytic queries. These are queries that run in the order of several seconds to minutes using row-organized tables and crunch a lot of data, join with several tables in a star schema and perform aggregation.

You can use db2batch to compare query performance of column-organized tables against row-organized tables or competitive offerings from other database vendors. Here is a sample invocation of the db2batch tool. In DB2 10.5, queries over column-organized tables must use an isolation level of Uncommitted Read (UR) or Cursor Stability (CS) with currently committed semantics.

```
db2batch -d edwdb -iso ur -f query.clp
```

Troubleshooting tips for query performance

Here are few things to consider if the performance is not what you anticipated:

- If queries commonly join a column-organized fact table to row-organized dimension table(s), joins will not happen over compressed data, thereby hurting query performance. The best practice for optimal query performance is to join only column-organized tables in a query. If this is not possible, at least create the most filtering dimension tables as column organized. You can use OQWT to help determine which tables should be created as column-organized tables for a given workload.
- If the query is a simple query that is able to leverage an index on row-organized tables and return in milliseconds, it might not be an ideal candidate for BLU, which is designed for analytic workloads that typically scan large amounts of data and perform joins and aggregation.
- If many queries access all or most of the columns, consider creating the underlying tables as row-organized tables, where a data page contains all columns of a row.

- Check if the query is spilling to temp space by looking at monitoring elements like `HASH_JOIN_OVERFLOWS` and `HASH_GRPBY_OVERFLOWS`. Also look at `POST_SHRTHRESHOLD_HASH_JOINS` and `POST_THRESHOLD_HASH_GRPBYS` to get the number of joins or group-by requests that were throttled back. Increase the `SORTHEAP` and `SHEAPTHRES_SHR` database configuration parameters to minimize spilling or throttling. A reasonable starting point is to set `SHEAPTHRES_SHR` to the size of the buffer pool and `SORTHEAP` to 1/20th of `SHEAPTHRES_SHR`. If the environment has a high degree of concurrency with many users running many analytic queries, the shared sort heap threshold may need to be increased manually. An alternative is to use WLM to limit the number of concurrent heavy queries allowed; the default concurrency is determined automatically based on hardware capabilities. Note that the `AUTOMATIC` setting for sort parameters is not supported when using column-organized tables. If you ran the `AUTOCONFIGURE` command or created the database after setting `DB2_WORKLOAD` to `ANALYTICS`, DB2 will configure the sort parameters to a reasonable size based on available memory.
- Make sure you considered the sizing recommendations described in the System requirements section of this paper.

- Last but not the least, is your baseline for this performance comparison a multi-node system like DB2 DPF? For instance, if a query takes 10 minutes on a DB2 DPF system and the same time on a single server BLU, make sure to normalize the comparison to take into account the number of cores and the total memory because that factors into the total cost of ownership. This tip is applicable any time you are comparing performance across servers that have different hardware capabilities.

Summary

BLU Acceleration in DB2 10.5 has a unique blend of three important attributes: simple, small, and fast. In this article, you learned how simple it is to set up BLU for your data mart or data warehouse by using column-organized tables. You realized the dollar savings from BLU's smaller on-disk storage footprint and the ability to explore new areas of the business by running complex, analytic queries significantly faster. Finally, you learned about the workload characteristics that the BLU technology is designed for.

Resources

- Refer to the DB2 10.5 Information Center.
- Read the Data Magazine article on BLU Acceleration Super Analytics, Super Easy.
- Watch this video on BLU Acceleration: Deep Dive on BLU Acceleration in DB2 10.5, Super Analytics, Super Easy.
- Refer to additional best practices for BLU Acceleration: Optimizing analytic workloads using DB2 10.5 with BLU Acceleration
- Get involved in the My developerWorks community. Connect with other developerWorks users while exploring the developer-driven blogs, forums, groups, and wikis.

About the authors

Naresh Chainani

Naresh Chainani has been a senior software engineer in the IBM Portland lab since 2001. His expertise spans several areas of DB2, with a particular emphasis on BLU Acceleration, multitemperature data management, table partitioning, and decimal floating-point. Recently, he led the development of several enhancements to the table partitioning feature in DB2. Prior to that, he led the development of the IEEE-compliant DECFLOAT data type. He frequently interacts with customers either to present or provide consulting assistance in the above areas as well as to seek valuable stakeholder feedback.

Weilin Lu

Weilin Lu has been working on DB2 UDB products for 13 years, with all of her time dedicated with the Quality Assurance team. Her recent project is with the BLU Acceleration on both system verification testing and functional verification testing. She holds a master's degree of computing and science from McMaster University, Canada, She also holds a master's degree of computer science and electric engineering from Zhejiang University, China. She was an associated professor at Zhejiang University before she moved to North America.

Alexandria Burkleaux

Alexandria (Alex) Burkleaux is a manager in the IBM Portland Lab, working on DB2 for Linux, UNIX, and Windows. She has worked on DB2 since 2001 as a development engineer, test engineer, and manager on the Table Partitioning feature, SQL Compatibility, HADR and, most recently, BLU Acceleration for DB2. She came to IBM as part of the Informix acquisition. At Informix, she worked as a developer on Informix XPS High Availability, Backup and Restore, and ISV Certification for the X/Open Backup Services API. Alex is an active customer advocate who frequently helps support customers during DB2 Early Release Programs.

Matthias Nicola

Matthias Nicola is a senior technical staff member at IBM's Silicon Valley Lab in San Jose, CA. He focuses on DB2 performance and benchmarking, XML, temporal data management, in-database analytics, BLU Acceleration, and other emerging technologies. He also works closely with customers and business partners to help

them design, optimize, and implement DB2 solutions. Previously, Matthias worked on data warehouse performance at Informix Software.