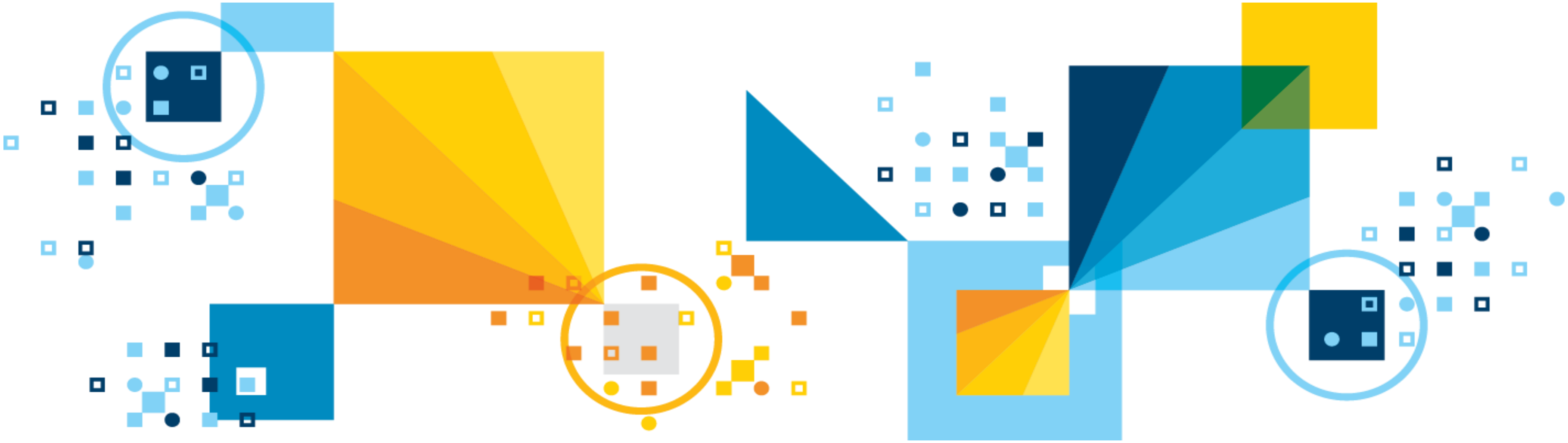
DB2 BLU Acceleration

Module ID	10108
Length	1.5 hours



For questions about this presentation contact askdata@ca.ibm.com

Disclaimer

THE INFORMATION CONTAINED IN THIS PRESENTATION IS PROVIDED FOR INFORMATIONAL PURPOSES ONLY. WHILE EFFORTS WERE MADE TO VERIFY THE COMPLETENESS AND ACCURACY OF THE INFORMATION CONTAINED IN THIS PRESENTATION, IT IS PROVIDED “AS IS” WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. IN ADDITION, THIS INFORMATION IS BASED ON IBM’S CURRENT PRODUCT PLANS AND STRATEGY, WHICH ARE SUBJECT TO CHANGE BY IBM WITHOUT NOTICE. IBM SHALL NOT BE RESPONSIBLE FOR ANY DAMAGES ARISING OUT OF THE USE OF, OR OTHERWISE RELATED TO, THIS PRESENTATION OR ANY OTHER DOCUMENTATION. NOTHING CONTAINED IN THIS PRESENTATION IS INTENDED TO, NOR SHALL HAVE THE EFFECT OF, CREATING ANY WARRANTIES OR REPRESENTATIONS FROM IBM (OR ITS SUPPLIERS OR LICENSORS), OR ALTERING THE TERMS AND CONDITIONS OF ANY AGREEMENT OR LICENSE GOVERNING THE USE OF IBM PRODUCTS AND/OR SOFTWARE.

Module Information

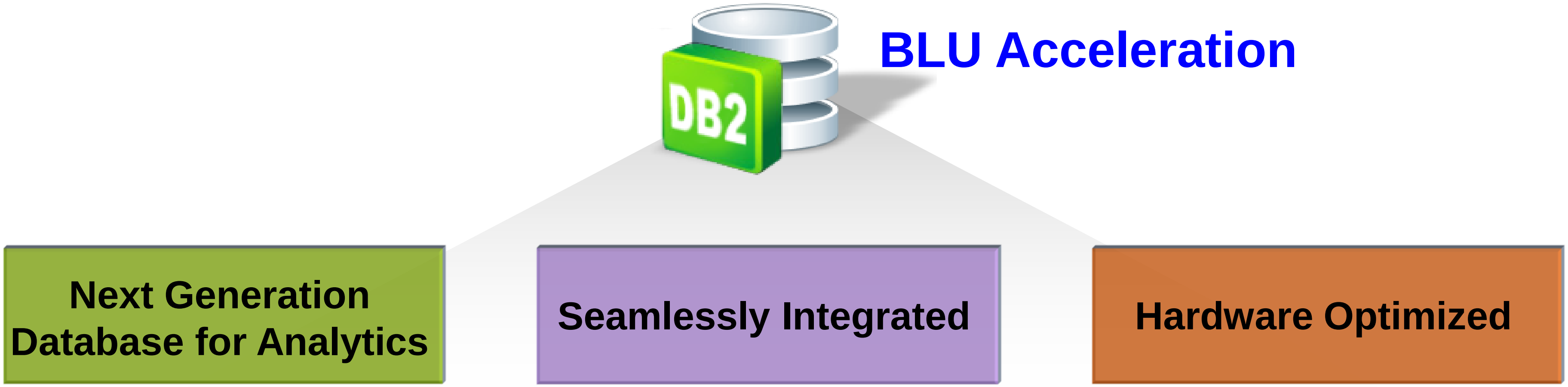
- You should have completed or acquired the necessary knowledge for the following modules in order to complete this module:
  - DB2 Fundamentals

- After completing this module, you should be able to:
  - Describe DB2 BLU Acceleration
  - Explain the concepts of:
    - Columnar storage
    - Shadow Table

Module Content

- **Value of BLU Acceleration**
- Getting started with BLU Acceleration
- LOAD & Compression
- Query Execution and Workload Management
- Shadow Table and HADR support
- Key Benefits of DB2 BLU

Value of DB2 with BLU Acceleration



Next Generation Database for Analytics

Seamlessly Integrated

Hardware Optimized

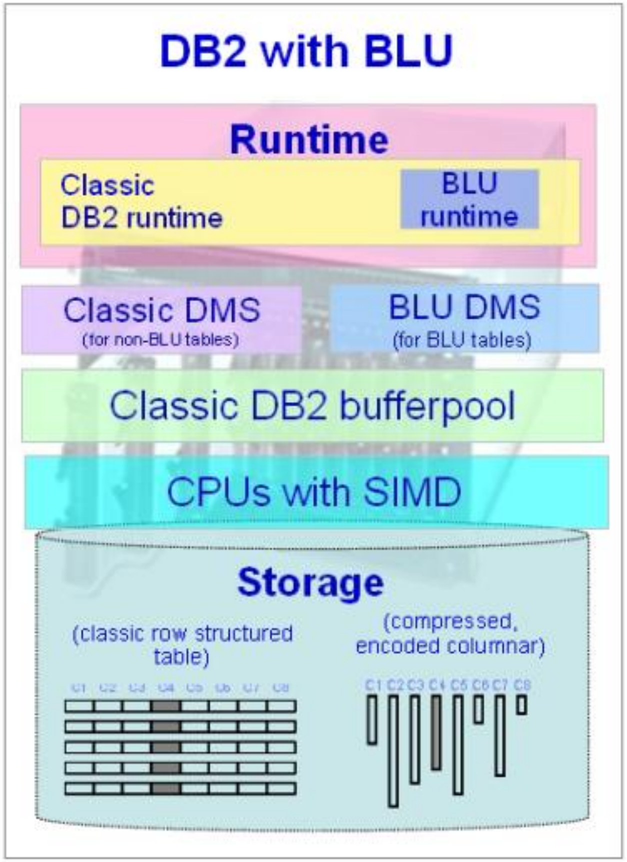
- **Extreme performance out-of-the-box**
- **Massive storage savings**
  - No indexes required
- **Lower cost of operational analytics**

- **Built seamlessly into DB2**
- **Consistent SQL, interfaces, administration**
- **Dramatic simplification**
  - Less to design
  - Less to tune
  - **Just Load and Go**
- **More Oracle ISV enablement for BLU**
  - Easier/Faster Migration of Oracle Data marts to DB2

- **In memory optimized**
  - Compressed in memory
- **Modern CPU Exploitation**
- **I/O optimized**
  - Only read columns of interest
- **Power 8 optimized**

What is DB2 with BLU Acceleration?

- **New innovative technology for analytic queries**
  - – Columnar storage
  - – New run-time engine with vector (aka SIMD) processing, deep multi-core optimizations and cache-aware memory management
  - – "Active compression" - unique encoding for further storage reduction beyond DB2 10 levels, and run-time processing without decompression
- **Value : Order-of-magnitude benefits in …**
  - – Performance
  - – Storage savings
  - – Simplicity !
- **"Revolution by Evolution"**
  - – Built directly into the DB2 kernel
  - – BLU tables can coexists with traditional row tables, same schema, tablespaces, buffer pools
  - – Query any combination of BLU or row data
  - – Memory-optimized (not "in-memory")



Seamless Integration into DB2

- **Built seamlessly into DB2 – integration and coexistence**
  - – Column-organized tables can coexist with existing, traditional, tables
    - Same schema, same storage, same memory
  - – Integrated tooling support
    - Optim Query Workload Tuner recommends BLU Acceleration deployments

- **Same SQL, language interfaces, administration**
  - – Column-organized tables or combinations of column-organized and row-organized tables can be accessed within the same SQL statement

- **Dramatic simplification – Just "Load and Go"**
  - – Faster deployment
    - Fewer database objects required to achieve same outcome
  - – Requires less ongoing management due to it's optimized query processing and fewer database objects required
  - – Simple migration
    - Conversion from traditional row table to BLU Acceleration is easy
    - DB2 Workload Manager identifies workloads to tune
    - Optim Query Workload Tuner recommends BLU Acceleration table transformations
    - Users only notice speed up; DBA's only notice less work!
  - – Management of single server solutions less expensive than clustered solutions

Super Fast, Super Easy – Create, Load, and Go!

Database Design and Tuning

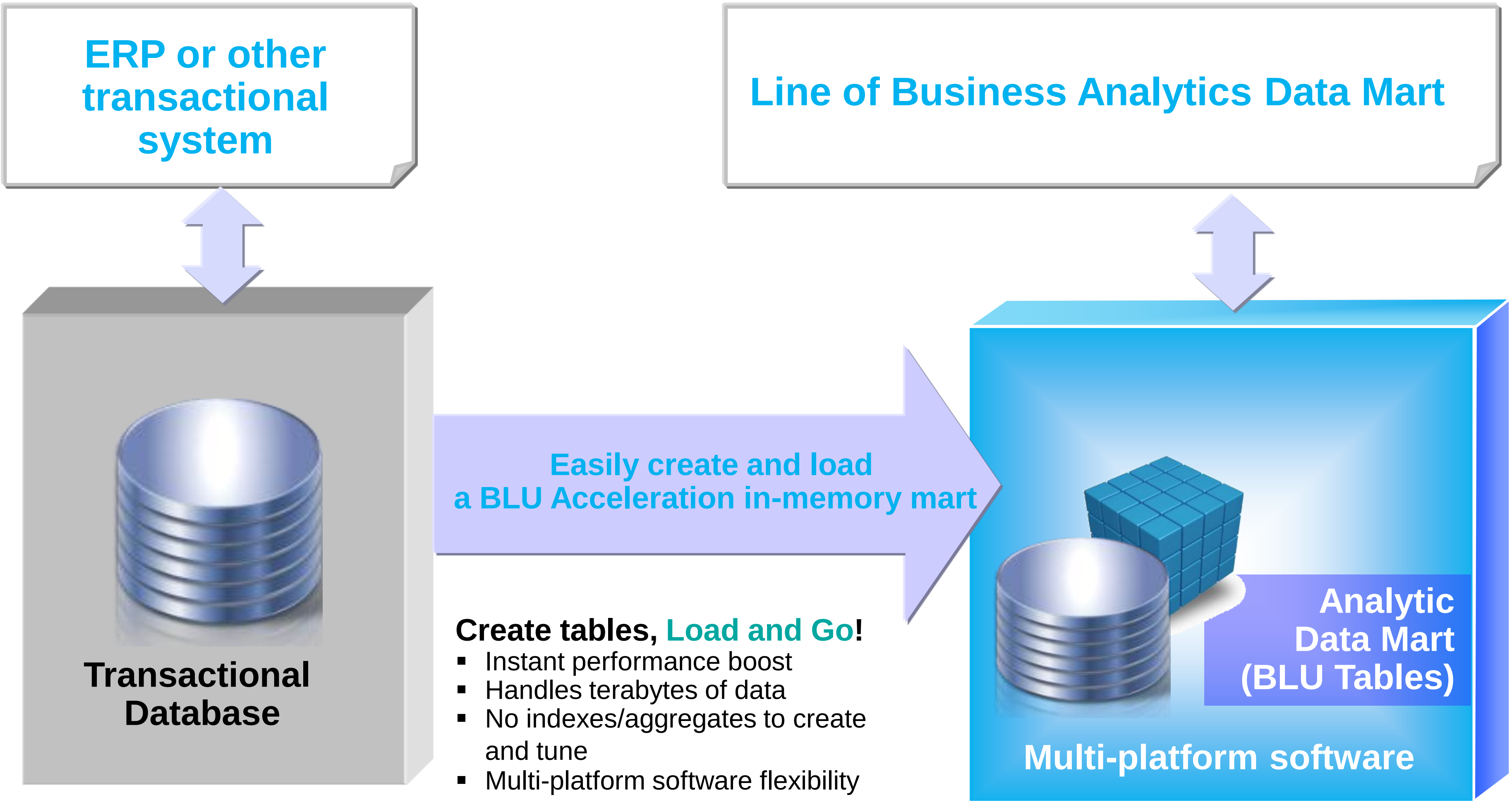
1. Decide on partition strategies
2. Select Compression Strategy
3. Create Table
4. Load data
5. Create Auxiliary Performance Structures
   - Materialized views
   - Create indexes
     - B+ indexes
     - Bitmap indexes
6. Tune memory
7. Tune I/O
8. Add Optimizer hints
9. Statistics collection

Repeat

DB2 with BLU Acceleration

1. Create Table
2. Load data

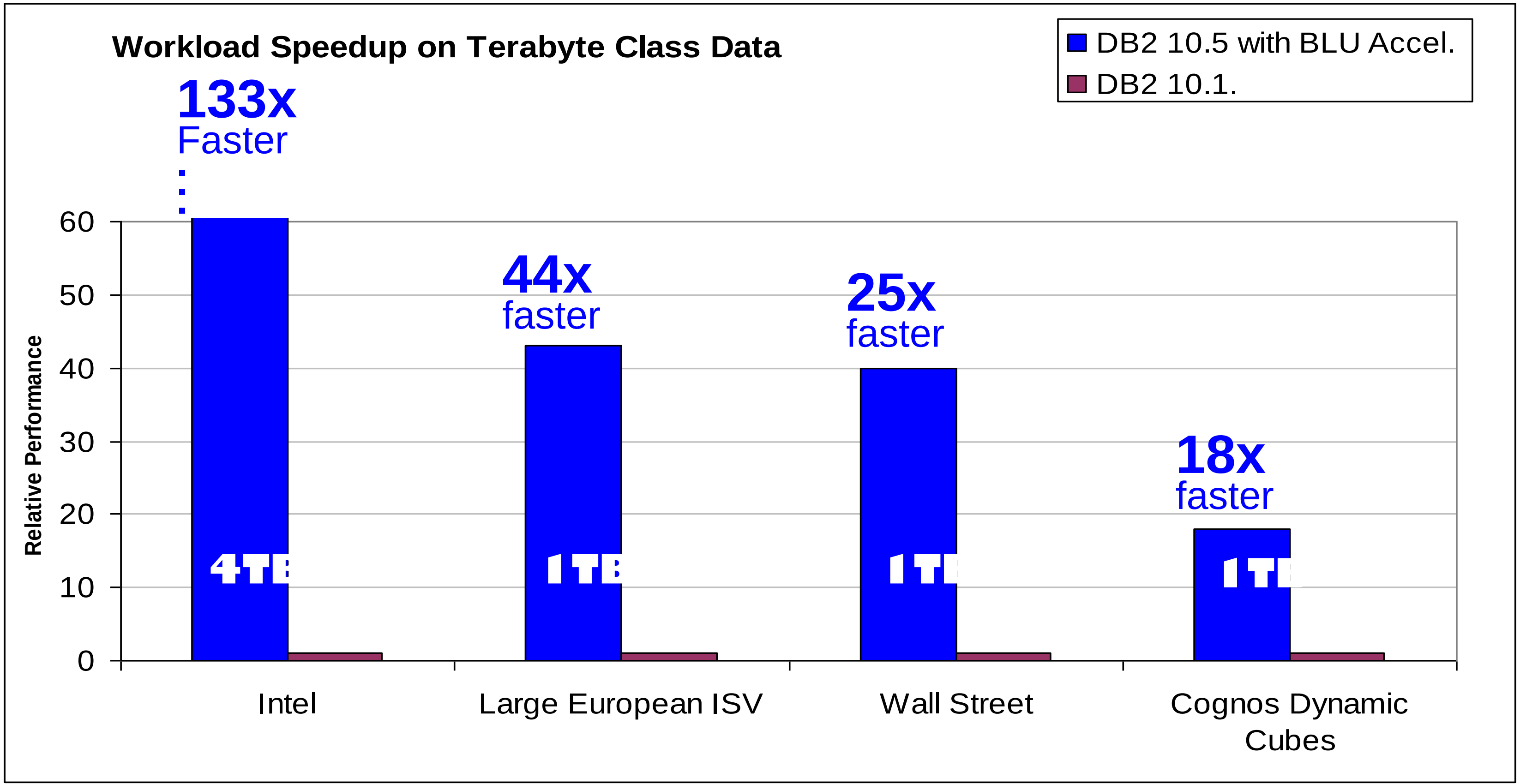
Analytics Data Mart
From Transactional Database


ERP or other transactional system
Line of Business Analytics Data Mart
Transactional Database
Easily create and load a BLU Acceleration in-memory mart
Analytic Data Mart (BLU Tables)
Multi-platform software

Create tables, Load and Go!

- Instant performance boost
- Handles terabytes of data
- No indexes/aggregates to create and tune
- Multi-platform software flexibility

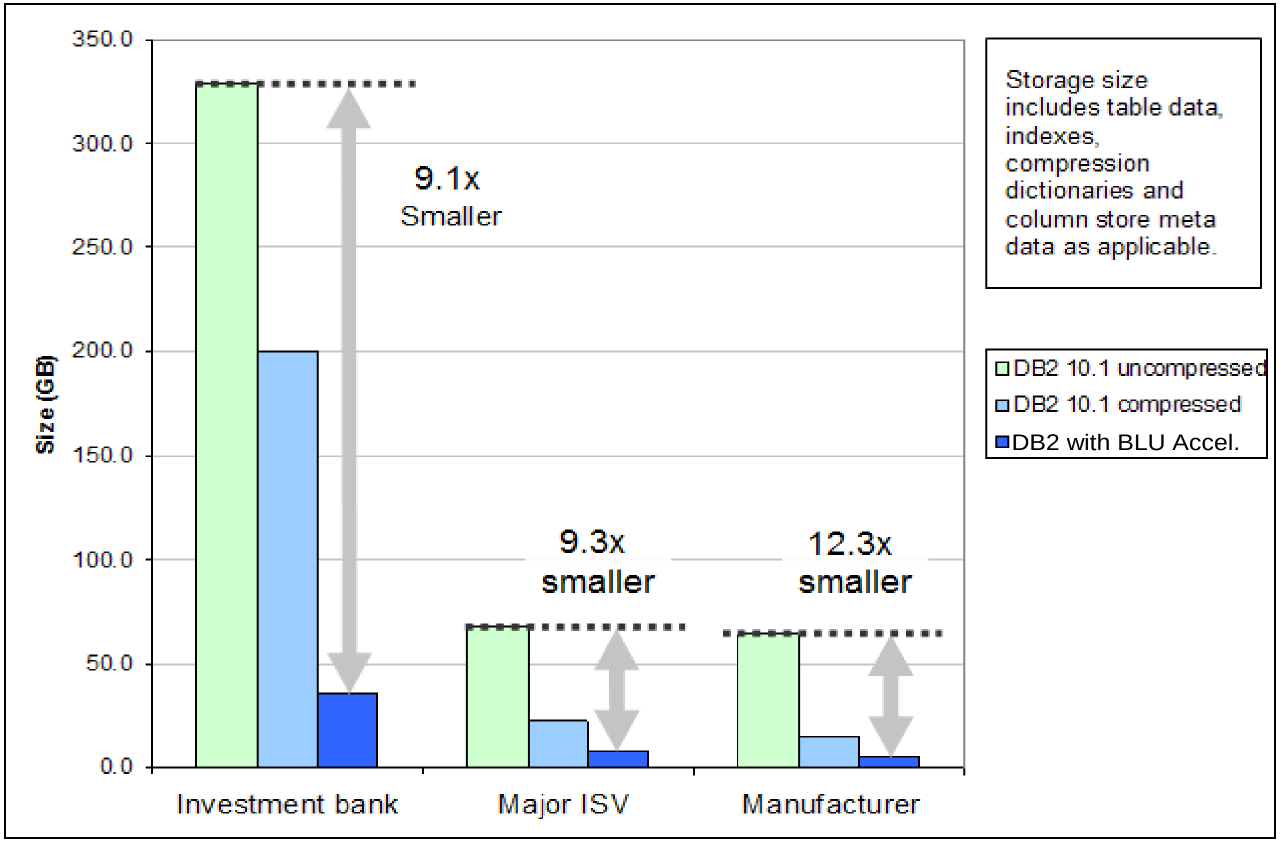
Terabyte Class Results, March-April 2013



> *“It was amazing to see the faster query times compared to the performance results with our row-organized tables.* ***The performance of four of our queries improved by over 100-fold! The best outcome was a query that finished 137x faster by using BLU Acceleration.”***
> - Kent Collins, Database Solutions Architect, BNSF Railway

Significant Storage Savings

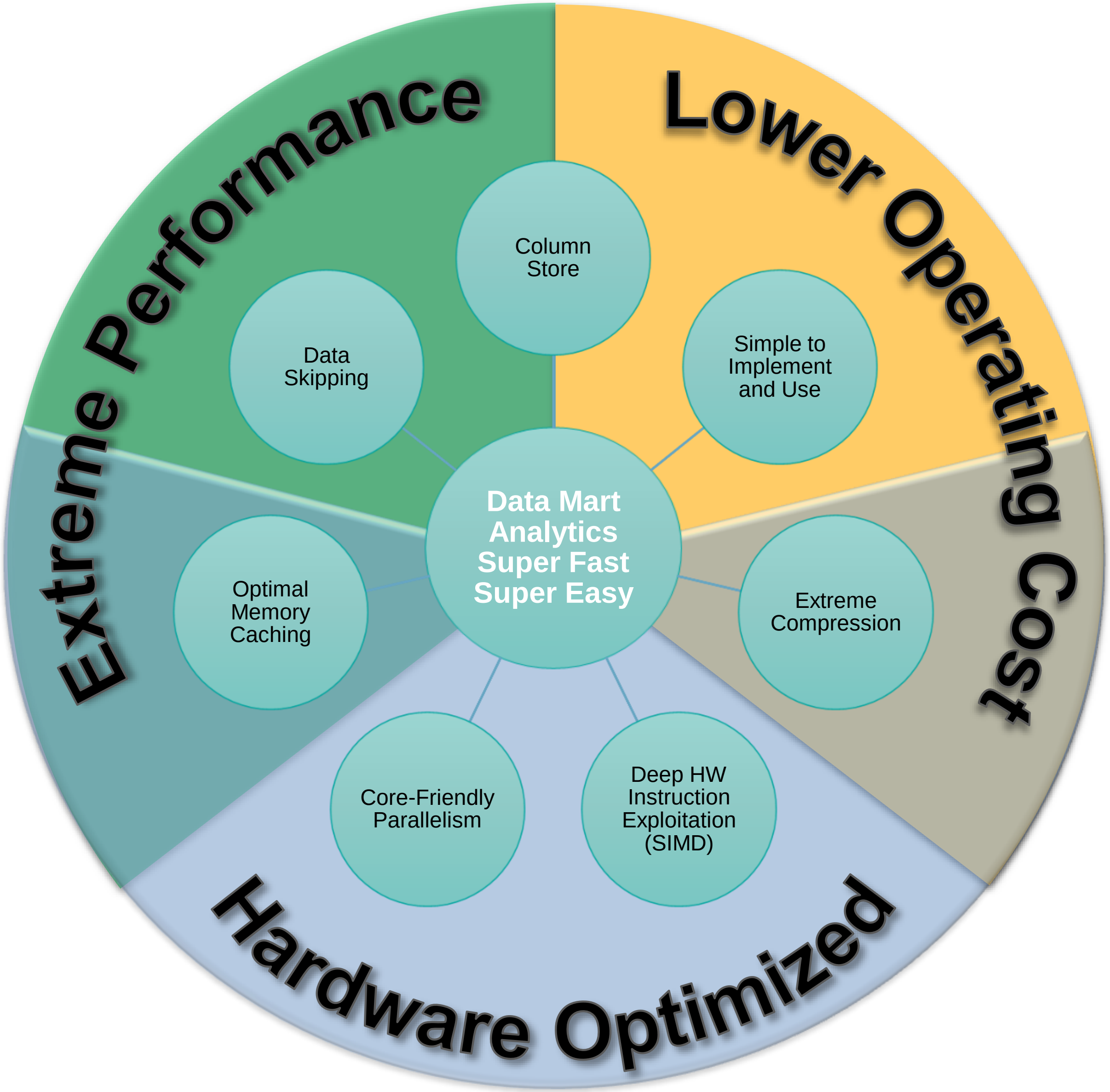
- ~2x-3x storage reduction vs DB2 10.1 adaptive compression (comparing all objects - tables, indexes, etc)
  - New advanced compression techniques
  - Fewer storage objects required



Storage size includes table data, indexes, compression dictionaries and column store meta data as applicable.

Lab tests – YMMV

The Seven Big Ideas of DB2 with BLU Acceleration


Extreme Performance
Lower Operating Cost
Hardware Optimized
Column Store
Data Skipping
Simple to Implement and Use
Data Mart Analytics Super Fast Super Easy
Optimal Memory Caching
Extreme Compression
Core-Friendly Parallelism
Deep HW Instruction Exploitation (SIMD)

7 Big Ideas: 1 Simple to Implement and Use

- **LOAD and then… run queries**
  - No indexes
  - No `REORG` (it's automated)
  - No `RUNSTATS` (it's automated)
  - No MDC or MQTs or Materialized Views
  - No partitioning
  - No statistical views
  - No optimizer hints

- **It is just DB2!**
  - Same SQL, language interfaces, administration
  - Reuse DB2 process model, storage, utilities

Coca-Cola Bottling Co. CONSOLIDATED

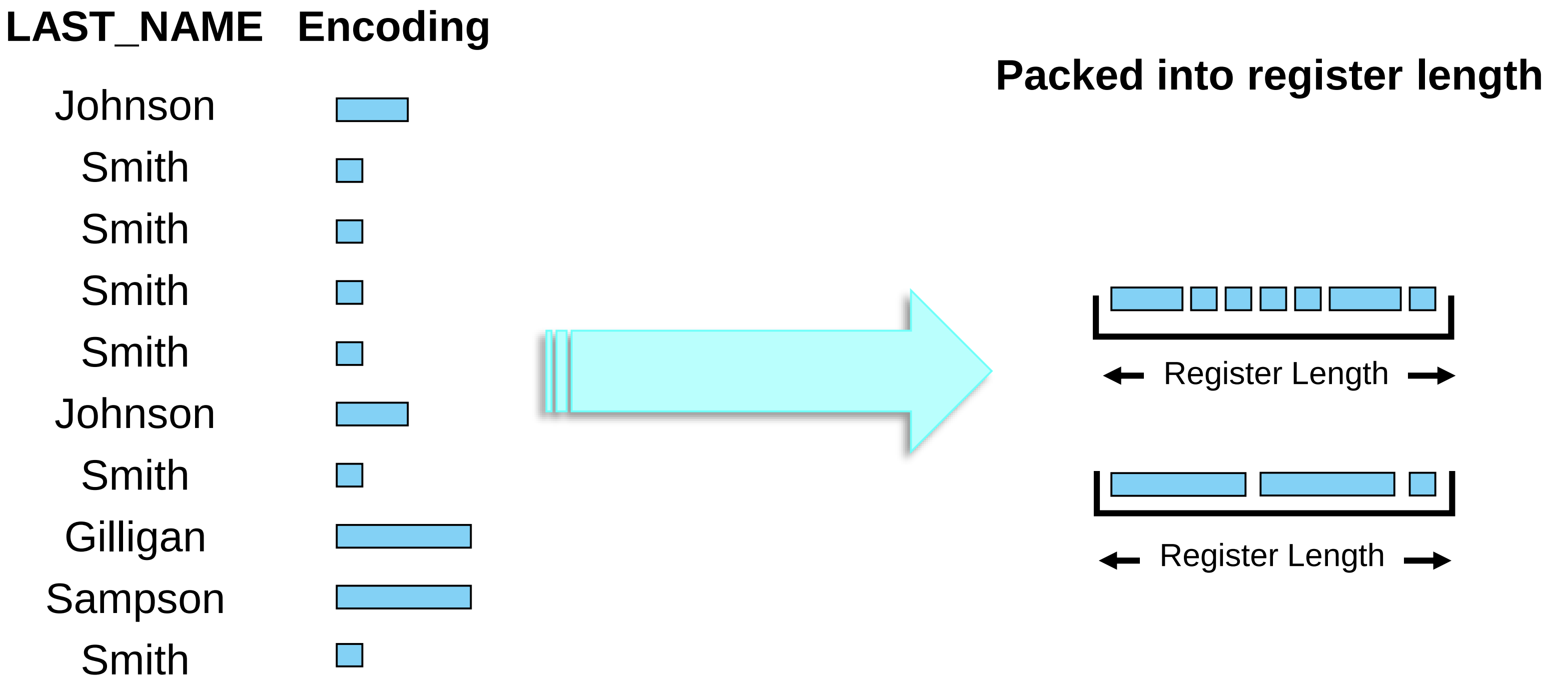
> "The BLU Acceleration technology has some obvious benefits: It makes our analytical **queries run 4-15x faster** and **decreases the size of our tables by a factor of 10x**. But it's when I think about **all the things I don't have to do with BLU**, it made me appreciate the technology even more: **no tuning, no partitioning, no indexes, no aggregates**."
>
> *-Andrew Juarez, Lead SAP Basis and DBA*

7 Big Ideas: 1 Simple to Implement and Use

- **One setting optimized the system for BLU Acceleration**
  - Set `DB2_WORKLOAD=ANALYTICS`
  - Informs DB2 that the database will be used for analytic workloads
- **Automatically configures DB2 for optimal analytics performance**
  - Makes column-organized tables the default table type
  - Enables automatic workload management
  - Enables automatic space reclaim
  - Sets up default page (32KB) and extent size (4) appropriate for analytics
  - Memory for caching, sorting and hashing, utilities are automatically initialized based on the server size and available RAM
- **Simple Table Creation**
  - If `DB2_WORKLOAD=ANALYTICS`, tables will be created column organized automatically
  - For mixed table types can define tables as `ORGANIZE BY COLUMN` or `ROW`
  - Compression is always on – No options
- **Easily convert tables from row-organized to column-organized**
  - `db2convert` utility

7 Big Ideas: 2 Compute Friendly Encoding and Compression

- Massive compression with approximate Huffman encoding
  - More frequent the value, the fewer bits it takes
- Register-friendly encoding dramatically improves efficiency
  - Encoded values packed into bits matching the register width of the CPU
  - Fewer I/Os, better memory utilization, fewer CPU cycles to process
- Encoded values do not need to be decompressed during evaluation
  - Predicates and joins work directly on encoded values



LAST_NAME Encoding

Johnson
Smith
Smith
Smith
Smith
Johnson
Smith
Gilligan
Sampson
Smith

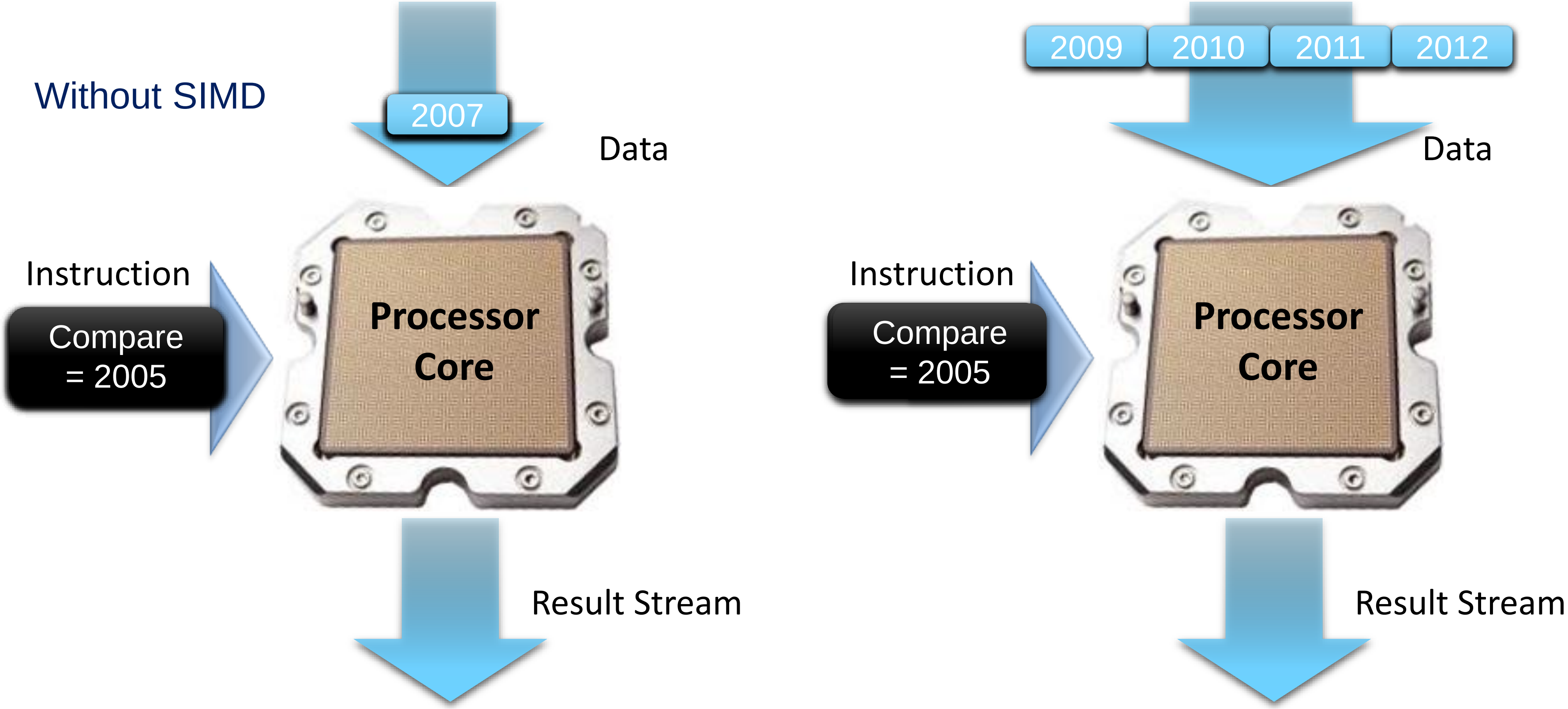
Packed into register length

Register Length

Register Length

7 Big Ideas: 3 Multiply the Power of the CPU

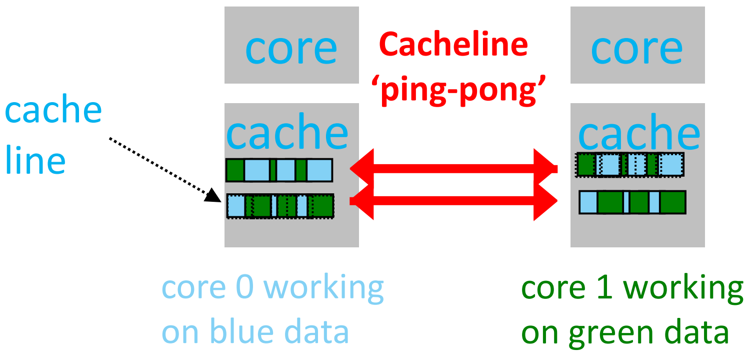
- **Performance increase with Single Instruction Multiple Data** (`SIMD`)
- **Using hardware instructions, DB2 with BLU Acceleration can apply a single instruction to many data elements simultaneously**
  - – Predicate evaluation, joins, grouping, arithmetic

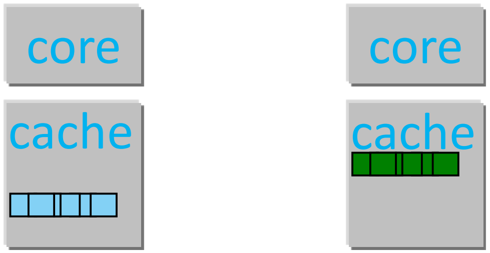


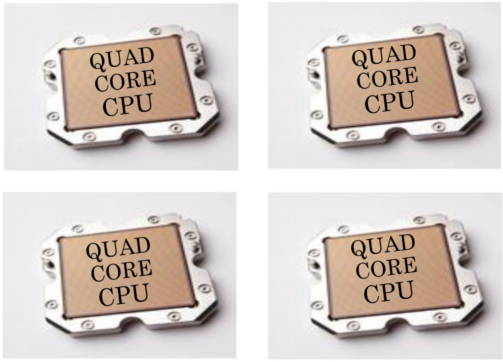
7 Big Ideas: 4 Core-Friendly Parallelism

- **BLU queries automatically parallelized across cores, and, achieve excellent multi-core scalability via …**
  - careful data placement and alignment
  - careful attention to physical attributes of the server
  - and other factors, designed to …

… maximize CPU cache hit rate & cacheline efficiency



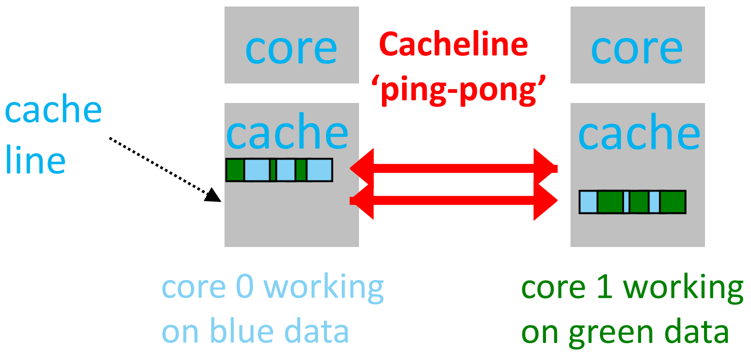


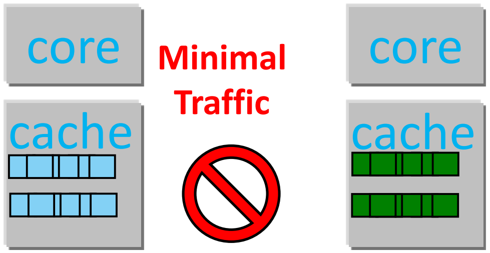


7 Big Ideas: 4 Core-Friendly Parallelism

- **BLU queries automatically parallelized across cores, and, achieve excellent multi-core scalability via …**
  - careful data placement and alignment
  - careful attention to physical attributes of the server
  - and other factors, designed to …

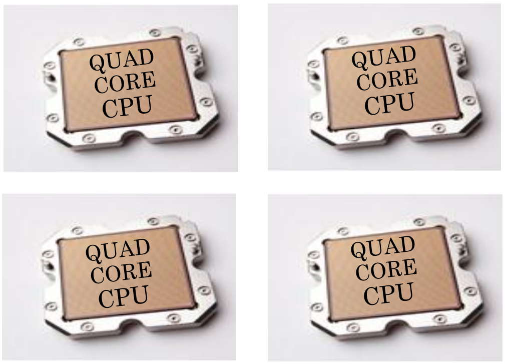
… maximize CPU cache hit rate & cacheline efficiency





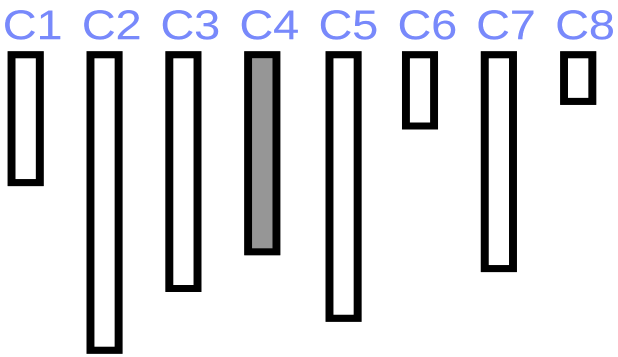
BNSF RAILWAY

> *"During our testing, we couldn't help but notice that DB2 10.5 with BLU Acceleration is excellent at utilizing our hardware resources. The core-friendly parallelism that IBM talks about was clearly evident and I didn't even have to partition the data across multiple servers."*
>
> \- Kent Collins, Database Solutions Architect, BNSF Railway



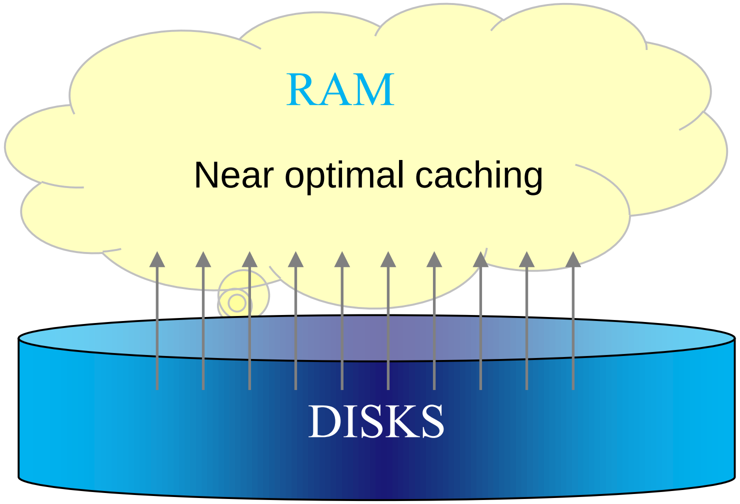
7 Big Ideas: 5 Column Store

- **Minimal I/O**
  - – Only perform I/O on the columns and values that match query
  - – As queries progresses through a pipeline the working set of pages is reduced
- **Work performed directly on columns**
  - – Predicates, joins, scans, etc. all work on individual columns
  - – Rows are not materialized until absolutely necessary to build result set
- Improved memory density
  - – Columnar data kept compressed in memory
- **Extreme** compression
  - – Packing more data values into very small amount of memory or disk
- Cache efficiency
  - – Data packed into cache friendly structures



7 Big Ideas: 6 Scan-Friendly Memory Caching

- New algorithms cache in RAM effectively
- High percent of interesting data fits in memory
  - We leave the interesting data in memory with the new algorithms
- Data can be larger than RAM
  - No need to ensure all data fits in memory
  - Optimization for in memory and I/O efficiency

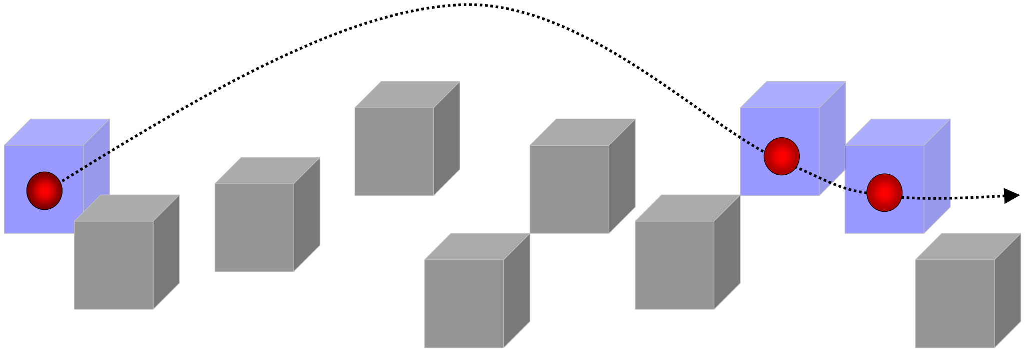


7 Big Ideas: 7 Data Skipping

- Automatic detection of large sections of data that do not qualify for a query and can be ignored

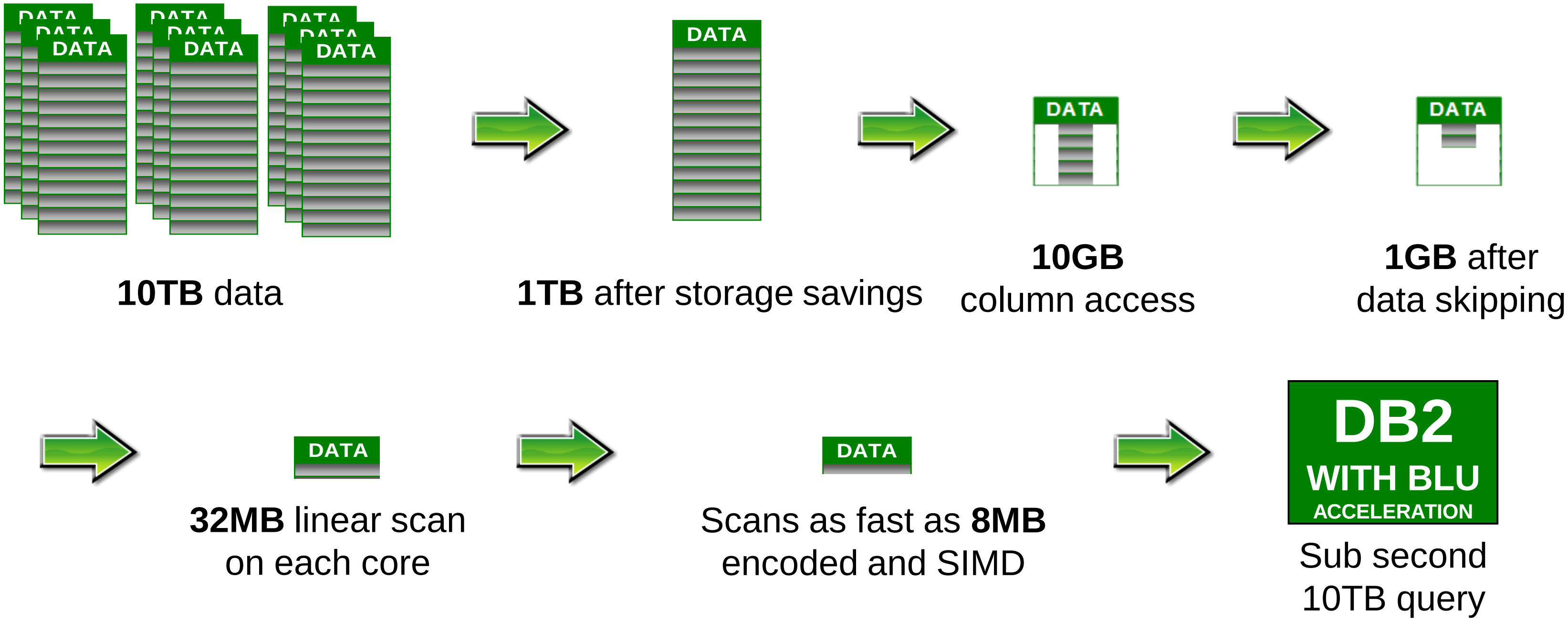
- Order of magnitude savings in all of I/O, RAM, and CPU

- No DBA action to define or use – truly invisible
  - Persistent storage of min. and max. values for sections of data values



7 Big Ideas: How DB2 with BLU Acceleration Helps
~Sub second 10TB query – An Optimistic Illustration

- The system – 32 cores, 10TB table with 100 columns, 10 years of data
- The query: `SELECT COUNT(*) from MYTABLE where YEAR = '2010'`
- The optimistic result: sub second 10TB query! Each CPU core examines the equivalent of just 8MB of data



10TB data

1TB after storage savings

10GB column access

1GB after data skipping

32MB linear scan on each core

Scans as fast as **8MB** encoded and SIMD

Sub second 10TB query

Module Content

- Value of BLU Acceleration
- **Getting started with BLU Acceleration**
- LOAD & Compression
- Query Execution and Workload Management
- Shadow Table and HADR support
- Key Benefits of DB2 BLU

Getting Started with BLU: Platforms and Hardware

- Supported platforms
  - Linux 64-bit on Intel/AMD/zSeries hardware
    - RHEL 5 or higher, SLES 10 SP4, SLES 11 SP2
  - AIX on Power hardware
    - AIX 6.1 TL7 SP6, AIX 7.1 TL1 SP6
  - Windows Operating Environment
    - Windows 2008 or higher, Windows 7 or higher
  - Inspur K-UX 2.1 64-bit

- For best results, use:
  - Intel Nehalem or better
  - Power 8

Capacity Recommendations

	Small	Medium	Large
raw data (CSV)	~1TB	~5TB	~10TB
Minimum:			
#cores	8	16	32
Memory	64GB	256GB	512GB
High-end perf:			
#cores	16	32	64
Memory	128 – 256GB	384 – 512GB	1 – 2TB

Assumption: all data is active and equally “hot”.

Will your workload benefit from BLU?

Probably:

- Analytical workloads, data marts, etc.
- Grouping, aggregation, range scans, joins
- Queries touch only a subset of the columns in a table
- Star Schema
- SAP Business Warehouse

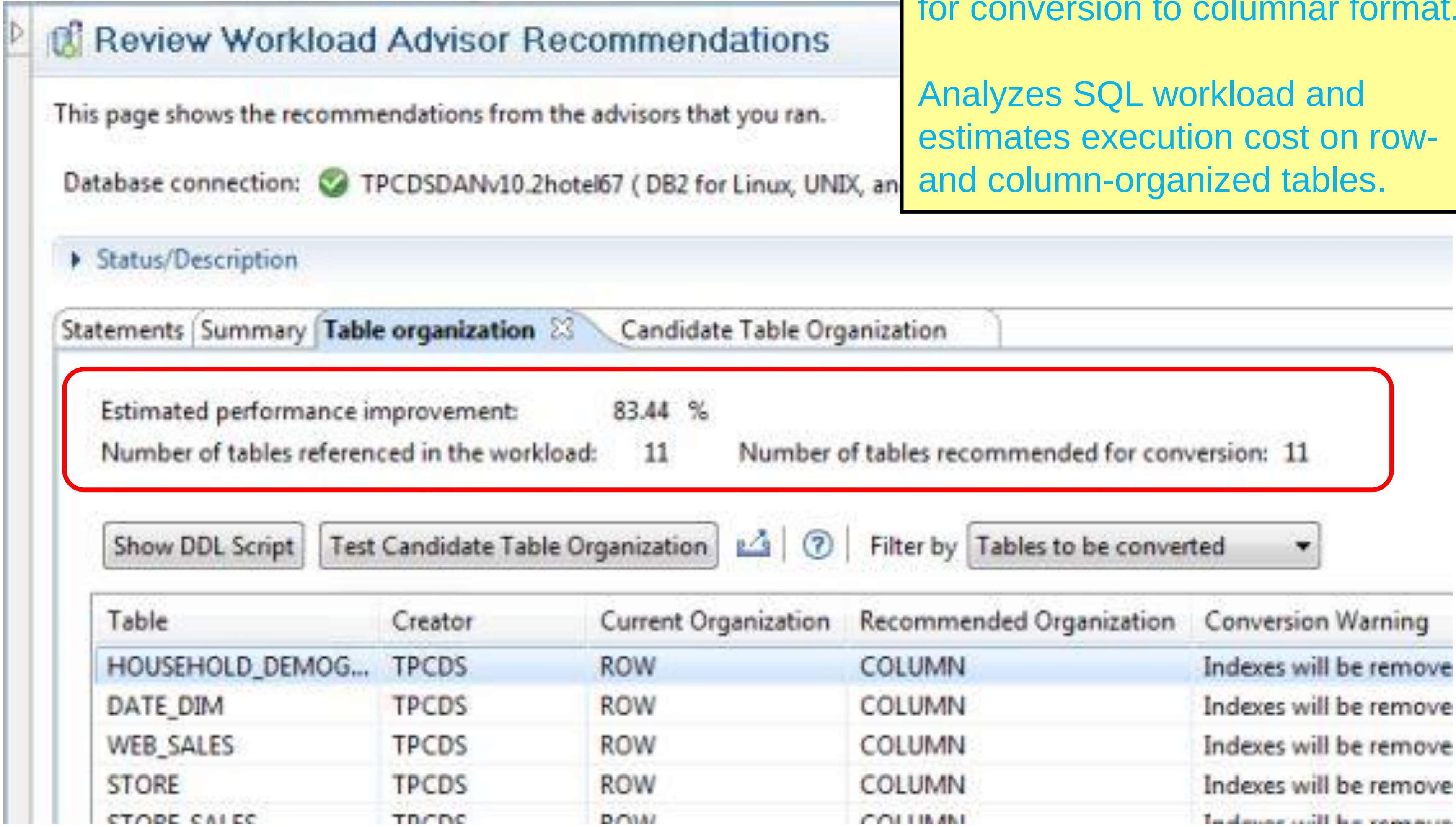
Probably not:

- OLTP
- Point access to 1 or few rows
- Insert, Update, Delete of few rows per transaction
- Queries touch many or all columns in a table
- Use of XML, Temporal, LOBs, etc. which are not supported in BLU yet

IBM Optim Query Tuner

Advisor identifies candidate tables for conversion to columnar format.

Analyzes SQL workload and estimates execution cost on row- and column-organized tables.



Review Workload Advisor Recommendations

This page shows the recommendations from the advisors that you ran.

Database connection: ✓ TPCDSDANv10.2hotel67 (DB2 for Linux, UNIX, an

Estimated performance improvement: 83.44 %

Number of tables referenced in the workload: 11 Number of tables recommended for conversion: 11

Filter by: Tables to be converted

Table	Creator	Current Organization	Recommended Organization	Conversion Warning
HOUSEHOLD_DEMOG...	TPCDS	ROW	COLUMN	Indexes will be remove
DATE_DIM	TPCDS	ROW	COLUMN	Indexes will be remove
WEB_SALES	TPCDS	ROW	COLUMN	Indexes will be remove
STORE	TPCDS	ROW	COLUMN	Indexes will be remove
STORE_SALES	TPCDS	ROW	COLUMN	Indexes will be remove

db2set DB2_WORKLOAD=ANALYTICS

- Set DB2_WORKLOAD=ANALYTICS **before** creating your database
- Don’t disable AUTOCONFIGURE

- For an existing database:
  - set DB2_WORKLOAD=ANALYTICS
  - then run AUTOCONFIGURE
    to get some (but not all) of the recommended settings
    e.g. dft_table_org = COLUMN, dft_extent_sz = 4, auto_reorg = ON

- Ideally, you won’t need to set anything else!
- Verify that sort heap, utility heap, and BPs are large

Creating a column-organized table

- Example:

```
CREATE TABLE sales_col (
   c1 INTEGER NOT NULL,
   c2 INTEGER,
    ...
   PRIMARY KEY (c1) ) ORGANIZE BY COLUMN;
```

Columnar tables are always compressed by default.

- If dft_table_org = COLUMN (e.g. DB2_WORKLOAD= ANALYTICS):
  - ORGANIZE BY COLUMN is the default and can be omitted
  - Use ORGANIZE BY ROW to create row-organized tables

- Do not specify compression, MDC, or partitioning for BLU tables.
- Do not create indexes
- **NEW!** Create column-organized user-maintained materialized query tables (MQTs).
- **NEW!** Use the ADD COLUMN clause of the ALTER TABLE statement to add any column type that is supported for column-organized tables
- **NEW!** DECFLOAT is now supported as a column type on column-organized table

Informational Uniqueness

- DB2 10.5 introduces informational uniqueness constraints
  - Primary keys and unique constraints can be enforced or not enforced
  - Enforced uniqueness remains the default
  - Informational (i.e., `NOT ENFORCED`) constraints do not enforce uniqueness
  - Valuable when data is coming from a trusted source
  - Be sure data is cleansed and truly unique!
- Benefits
  - Less storage required! No index is created to enforce the constraint.
  - No runtime overhead to maintain unique indexes during `LOAD, INSERT, UPDATE, or DELETE`
  - The uniqueness definition informs the query compiler of unique data, enabling opportunities for superior query execution plans

```
CREATE TABLE t1 (c1 INTEGER NOT NULL, c2 INTEGER,
                                 PRIMARY KEY (c1) NOT ENFORCED);

ALTER TABLE t1 ADD CONSTRAINT unique1 UNIQUE (c2) NOT ENFORCED;
```

Storage orientation

- Input data is in row format (CSV files, etc.):

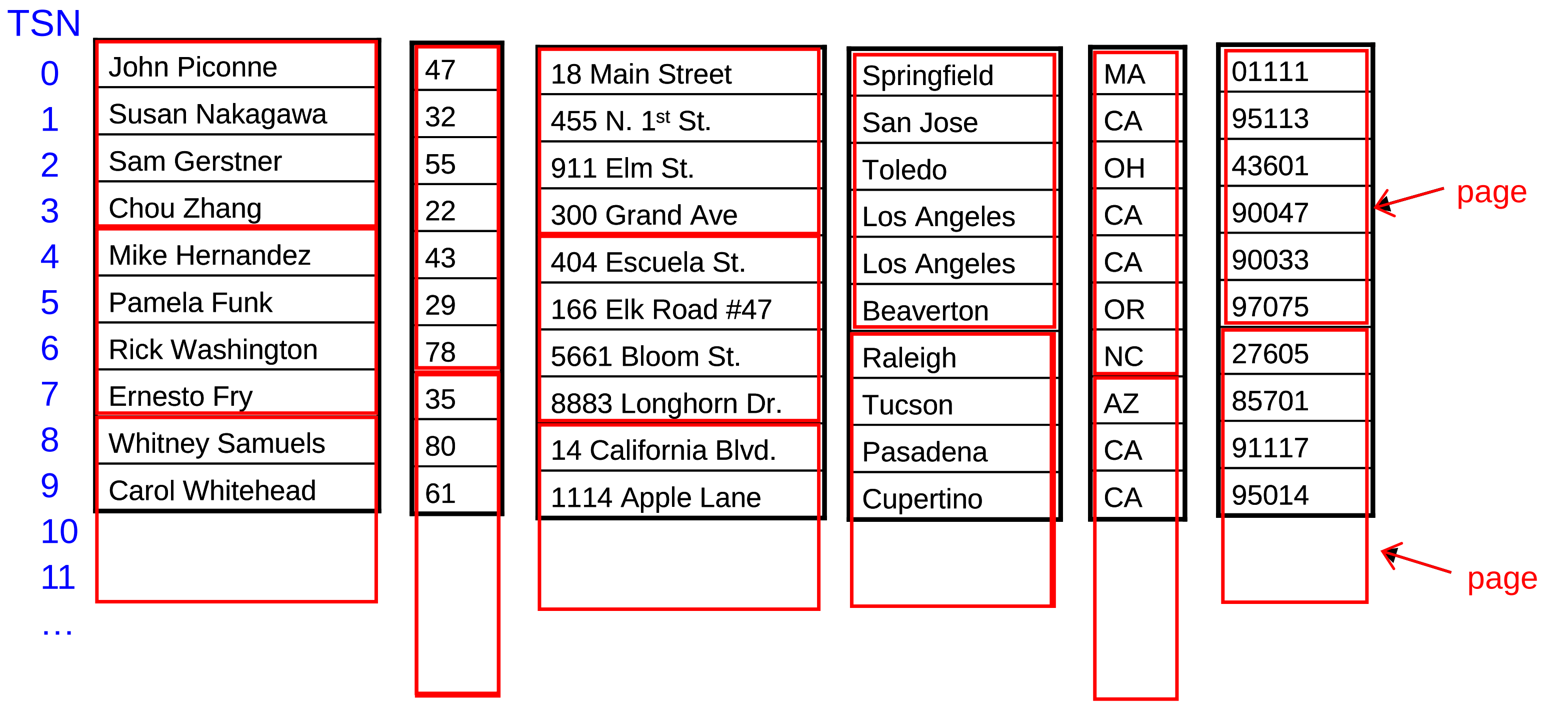
John Piconne	47	18 Main Street	Springfield	MA	01111
Susan Nakagawa	32	455 N. 1st St.	San Jose	CA	95113
Sam Gerstner	55	911 Elm St.	Toledo	OH	43601
Chou Zhang	22	300 Grand Ave	Los Angeles	CA	90047
Mike Hernandez	43	404 Escuela St.	Los Angeles	CA	90033
Pamela Funk	29	166 Elk Road #47	Beaverton	OR	97075
Rick Washington	78	5661 Bloom St.	Raleigh	NC	27605
Ernesto Fry	35	8883 Longhorn Dr.	Tucson	AZ	85701
Whitney Samuels	80	14 California Blvd.	Pasadena	CA	91117
Carol Whitehead	61	1114 Apple Lane	Cupertino	CA	95014

- Gets compressed and converted to columnar format upon LOAD/Insert

Columnar storage in DB2 (conceptual)

- Separate set of extents and pages for each column

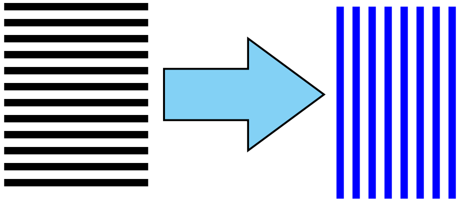
TSN = Tuple Sequence Number



TSN						
0	John Piconne	47	18 Main Street	Springfield	MA	01111
1	Susan Nakagawa	32	455 N. 1st St.	San Jose	CA	95113
2	Sam Gerstner	55	911 Elm St.	Toledo	OH	43601
3	Chou Zhang	22	300 Grand Ave	Los Angeles	CA	90047
4	Mike Hernandez	43	404 Escuela St.	Los Angeles	CA	90033
5	Pamela Funk	29	166 Elk Road #47	Beaverton	OR	97075
6	Rick Washington	78	5661 Bloom St.	Raleigh	NC	27605
7	Ernesto Fry	35	8883 Longhorn Dr.	Tucson	AZ	85701
8	Whitney Samuels	80	14 California Blvd.	Pasadena	CA	91117
9	Carol Whitehead	61	1114 Apple Lane	Cupertino	CA	95014
10						
11						
…						

- Typically, column-organized tables use less space than row-organized tables
- Column-organized tables with many columns and few rows can be larger than row-organized tables if many mostly empty extents
- TSNs (like logical Row ID) used to stitch column values that belong together in a row during query processing

Converting existing tables: db2convert



- Converts a row-organized table into a column-organized table
- Calls ADMIN_MOVE_TABLE
- Has the same options and restrictions as ADMIN_MOVE_TABLE
- **NEW!** Cancel pending conversion changes by running the **db2convert** command with the new **-cancel** parameter

```
db2convert
          -d <database-name>   (this is the only mandatory parameter)
          -stopBeforeSwap
          -continue                    (resumes a previously stopped conversion)
          -z <schema-name>
          -t <table-name>
          -ts <tablespace for new table>
          -opt <ADMIN_MOVE_TABLE options>  (e.g. COPY_USE_LOAD)
          …
```

What you see in the DB2 catalog: TABLEORG

- Which tables are column-organized?
  - – New column in syscat.tables: TABLEORG

```
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.
```

For column-organized tables, COMPRESSION is always blank because you cannot enable/disable compression.

What you see in the DB2 catalog: Synopsis Tables

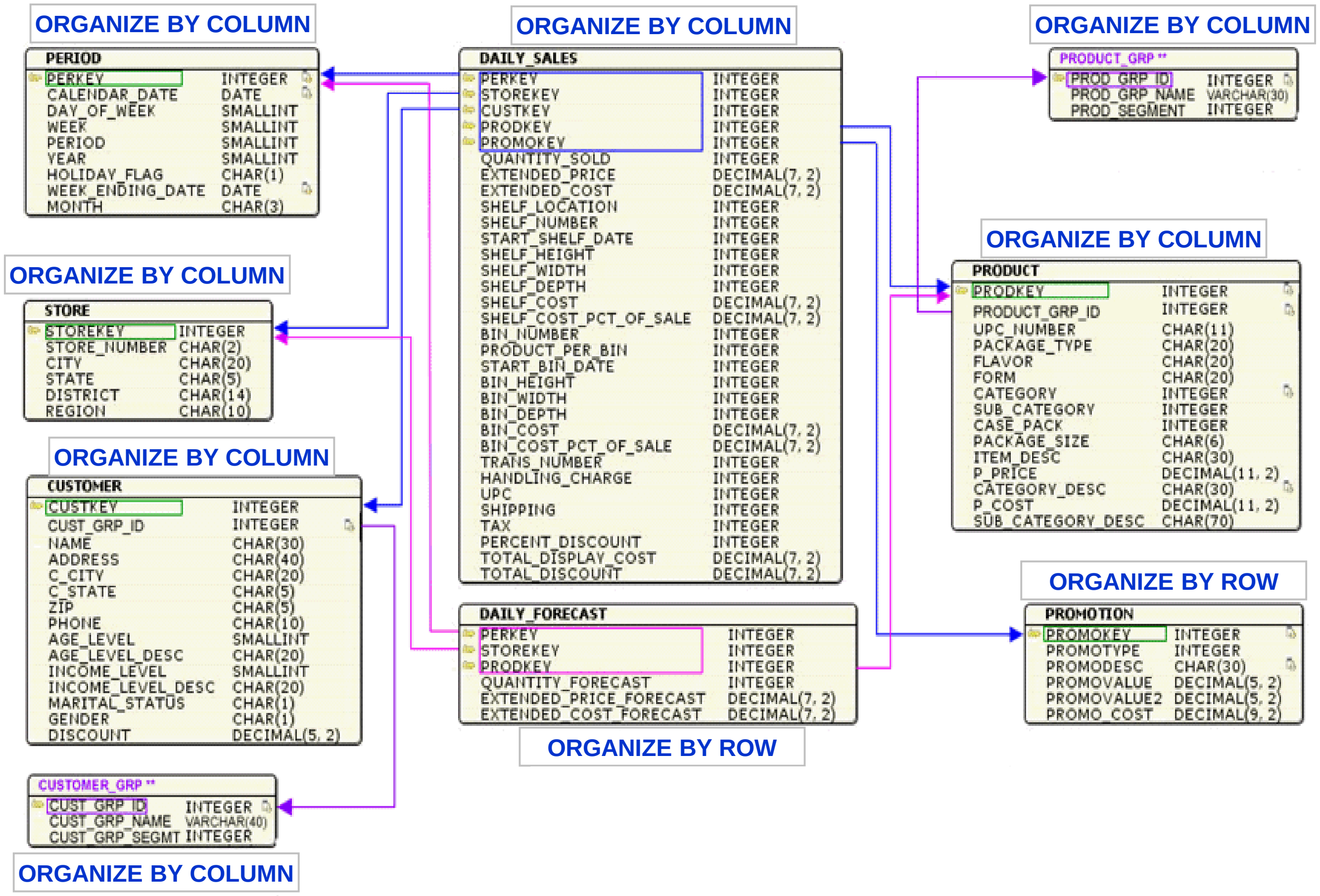
- For each columnar table there is a corresponding *synopsis table*, automatically created and maintained
  - Size of the synopsis table: ~0.1% of the user table
  - 1 row for every 1024 rows in the user table

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

TABSCHEMA  TABNAME                                  TABLEORG
---------- ---------------------------------------- --------
MNICOLA    SALES_COL                                C
SYSIBM     SYN130330165216275152_SALES_COL          C

  2 record(s) selected.
```

- **Synopsis Table is a meta-data that describes which ranges of values exist in which parts of the user table**
  - Enables DB2 to skip portions of table when scanning data during query
  - Benefits from data clustering, loading pre-sorted data

Mixing Row and Columnar Tables

- DB2 10.5 supports mixing row and columnar tables seamlessly
  - In the same tablespace and bufferpools
  - In the same query
- Best query performance for analytic queries usually occurs with all tables columnar
- Mixing row and columnar can be necessary
  - Point queries (highly selective access) favor row-organized tables with index access
  - Small, frequent, write operations favor row-organized tables



ORGANIZE BY COLUMN

PERIOD	
PERKEY	INTEGER
CALENDAR_DATE	DATE
DAY_OF_WEEK	SMALLINT
WEEK	SMALLINT
PERIOD	SMALLINT
YEAR	SMALLINT
HOLIDAY_FLAG	CHAR(1)
WEEK_ENDING_DATE	DATE
MONTH	CHAR(3)

ORGANIZE BY COLUMN

STORE	
STOREKEY	INTEGER
STORE_NUMBER	CHAR(2)
CITY	CHAR(20)
STATE	CHAR(5)
DISTRICT	CHAR(14)
REGION	CHAR(10)

ORGANIZE BY COLUMN

CUSTOMER	
CUSTKEY	INTEGER
CUST_GRP_ID	INTEGER
NAME	CHAR(30)
ADDRESS	CHAR(40)
C_CITY	CHAR(20)
C_STATE	CHAR(5)
ZIP	CHAR(5)
PHONE	CHAR(10)
AGE_LEVEL	SMALLINT
AGE_LEVEL_DESC	CHAR(20)
INCOME_LEVEL	SMALLINT
INCOME_LEVEL_DESC	CHAR(20)
MARITAL_STATUS	CHAR(1)
GENDER	CHAR(1)
DISCOUNT	DECIMAL(5, 2)

CUSTOMER_GRP **	
CUST_GRP_ID	INTEGER
CUST_GRP_NAME	VARCHAR(40)
CUST_GRP_SEGMT	INTEGER

ORGANIZE BY COLUMN

ORGANIZE BY COLUMN

DAILY_SALES	
PERKEY	INTEGER
STOREKEY	INTEGER
CUSTKEY	INTEGER
PRODKEY	INTEGER
PROMOKEY	INTEGER
QUANTITY_SOLD	INTEGER
EXTENDED_PRICE	DECIMAL(7, 2)
EXTENDED_COST	DECIMAL(7, 2)
SHELF_LOCATION	INTEGER
SHELF_NUMBER	INTEGER
START_SHELF_DATE	INTEGER
SHELF_HEIGHT	INTEGER
SHELF_WIDTH	INTEGER
SHELF_DEPTH	INTEGER
SHELF_COST	DECIMAL(7, 2)
SHELF_COST_PCT_OF_SALE	DECIMAL(7, 2)
BIN_NUMBER	INTEGER
PRODUCT_PER_BIN	INTEGER
START_BIN_DATE	INTEGER
BIN_HEIGHT	INTEGER
BIN_WIDTH	INTEGER
BIN_DEPTH	INTEGER
BIN_COST	DECIMAL(7, 2)
BIN_COST_PCT_OF_SALE	DECIMAL(7, 2)
TRANS_NUMBER	INTEGER
HANDLING_CHARGE	INTEGER
UPC	INTEGER
SHIPPING	INTEGER
TAX	INTEGER
PERCENT_DISCOUNT	INTEGER
TOTAL_DISPLAY_COST	DECIMAL(7, 2)
TOTAL_DISCOUNT	DECIMAL(7, 2)

DAILY_FORECAST	
PERKEY	INTEGER
STOREKEY	INTEGER
PRODKEY	INTEGER
QUANTITY_FORECAST	INTEGER
EXTENDED_PRICE_FORECAST	DECIMAL(7, 2)
EXTENDED_COST_FORECAST	DECIMAL(7, 2)

ORGANIZE BY ROW

ORGANIZE BY COLUMN

PRODUCT_GRP **	
PROD_GRP_ID	INTEGER
PROD_GRP_NAME	VARCHAR(30)
PROD_SEGMENT	INTEGER

ORGANIZE BY COLUMN

PRODUCT	
PRODKEY	INTEGER
PRODUCT_GRP_ID	INTEGER
UPC_NUMBER	CHAR(11)
PACKAGE_TYPE	CHAR(20)
FLAVOR	CHAR(20)
FORM	CHAR(20)
CATEGORY	INTEGER
SUB_CATEGORY	INTEGER
CASE_PACK	INTEGER
PACKAGE_SIZE	CHAR(6)
ITEM_DESC	CHAR(30)
P_PRICE	DECIMAL(11, 2)
CATEGORY_DESC	CHAR(30)
P_COST	DECIMAL(11, 2)
SUB_CATEGORY_DESC	CHAR(70)

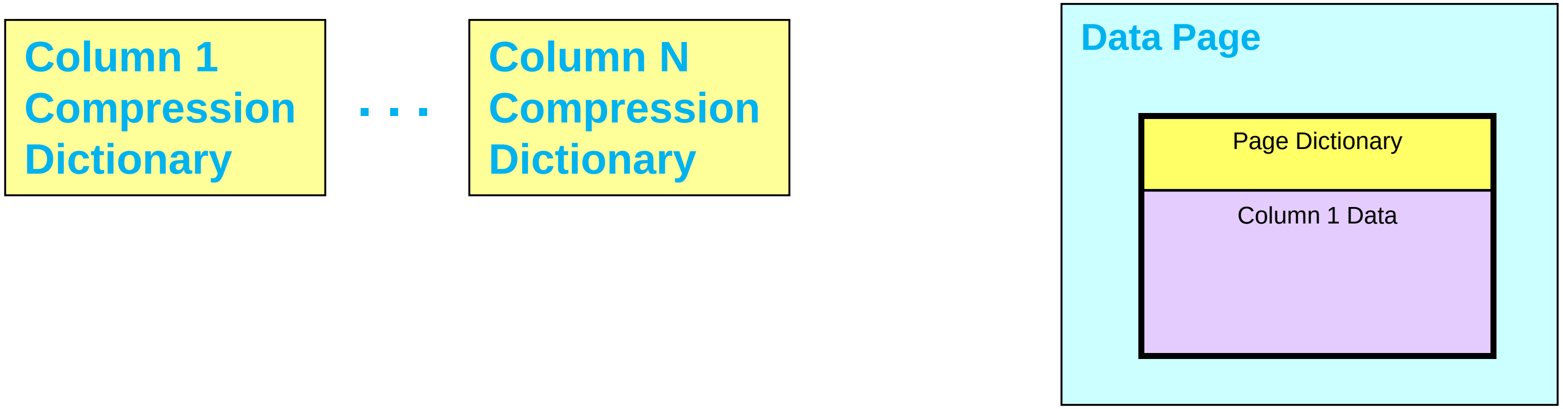
ORGANIZE BY ROW

PROMOTION	
PROMOKEY	INTEGER
PROMOTYPE	INTEGER
PROMODESC	CHAR(30)
PROMOVALUE	DECIMAL(5, 2)
PROMOVALUE2	DECIMAL(5, 2)
PROMO_COST	DECIMAL(9, 2)

Module Content

- Value of BLU Acceleration
- **Getting started with BLU Acceleration**
- **LOAD & Compression**
- Query Execution and Workload Management
- Shadow Table and HADR support
- Key Benefits of DB2 BLU

Compression Dictionaries for Column-Organized Tables



Column 1 Compression Dictionary

. . .

Column N Compression Dictionary

Data Page

Page Dictionary

Column 1 Data

- Column-level dictionaries: Always one per column
  - Dictionary populated during load replace, load insert into an empty table, or Automatic Dictionary Creation during Insert
- Page-level dictionaries: May also be created
  - Exploit local data clustering at page level to further compress data
  - Space savings must outweigh cost of storing page-level dictionaries

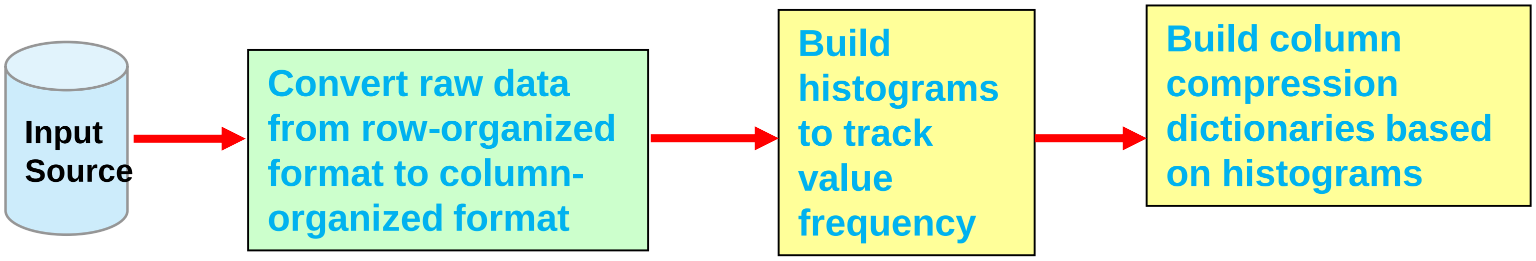
Load Example

```
LOAD FROM /db1/svtdbm1/data.del OF DEL INSERT INTO colTable1;

SQL3109N The utility is beginning to load data from file "/db1/svtdbm1/data.del".
SQL3500W The utility is beginning the "ANALYZE" phase at time "04/15/2013 14:56:02.272825".
SQL3519W Begin Load Consistency Point. Input record count = "0".
SQL3520W Load Consistency Point was successful.
SQL3515W The utility has finished the "ANALYZE" phase at time "04/15/2013 14:56:03.327893".

SQL3500W The utility is beginning the "LOAD" phase at time "04/15/2013 14:56:03.332048".
SQL3110N The utility has completed processing. "300000“ rows were read from the input file.
SQL3519W Begin Load Consistency Point. Input record count = "300000".
SQL3520W Load Consistency Point was successful.
SQL3515W The utility has finished the "LOAD" phase at time "04/15/2013 14:56:04.639261".

SQL3500W  The utility is beginning the "BUILD" phase at time "04/15/2013 14:57:06.848727".
SQL3213I  The indexing mode is "REBUILD".
SQL3515W  The utility has finished the "BUILD" phase at time "04/15/2013 14:59:07.487172".

Number of rows read = 300000
Number of rows skipped = 0
Number of rows loaded = 300000
Number of rows rejected = 0
Number of rows deleted = 0
Number of rows committed = 300000
```

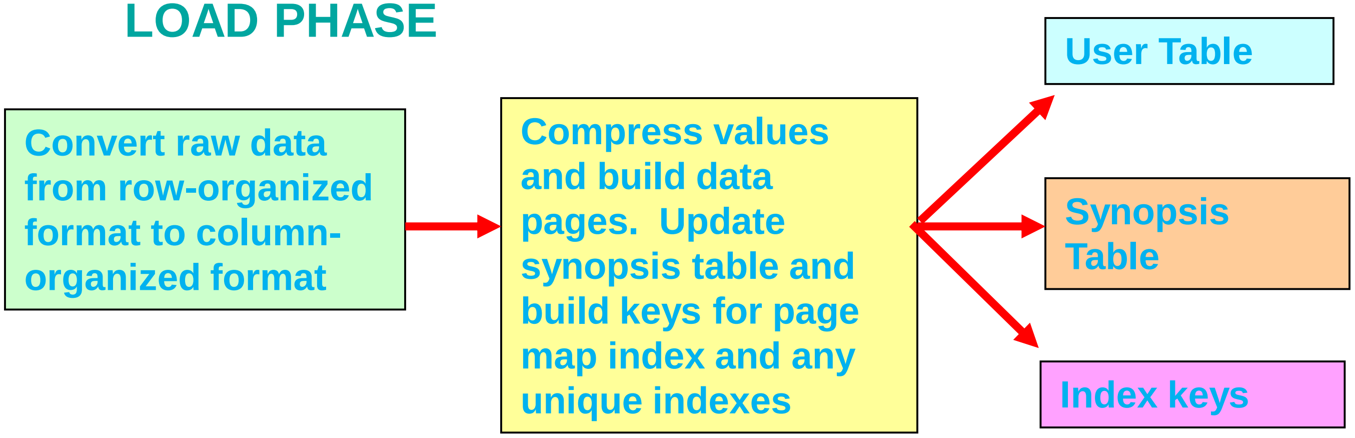
Load for Column-Organized Tables

ANALYZE PHASE only if dictionaries need to be built

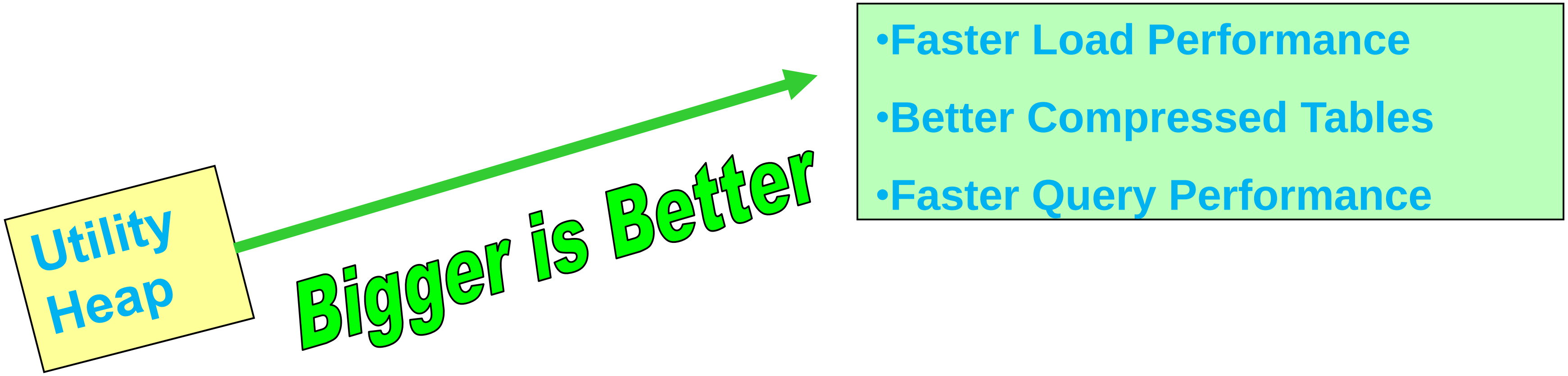
- sampling algorithms to make more efficient use of less memory


Input Source
Convert raw data from row-organized format to column-organized format
Build histograms to track value frequency
Build column compression dictionaries based on histograms

LOAD PHASE


Convert raw data from row-organized format to column-organized format
Compress values and build data pages. Update synopsis table and build keys for page map index and any unique indexes
User Table
Synopsis Table
Index keys

Memory Considerations for Load



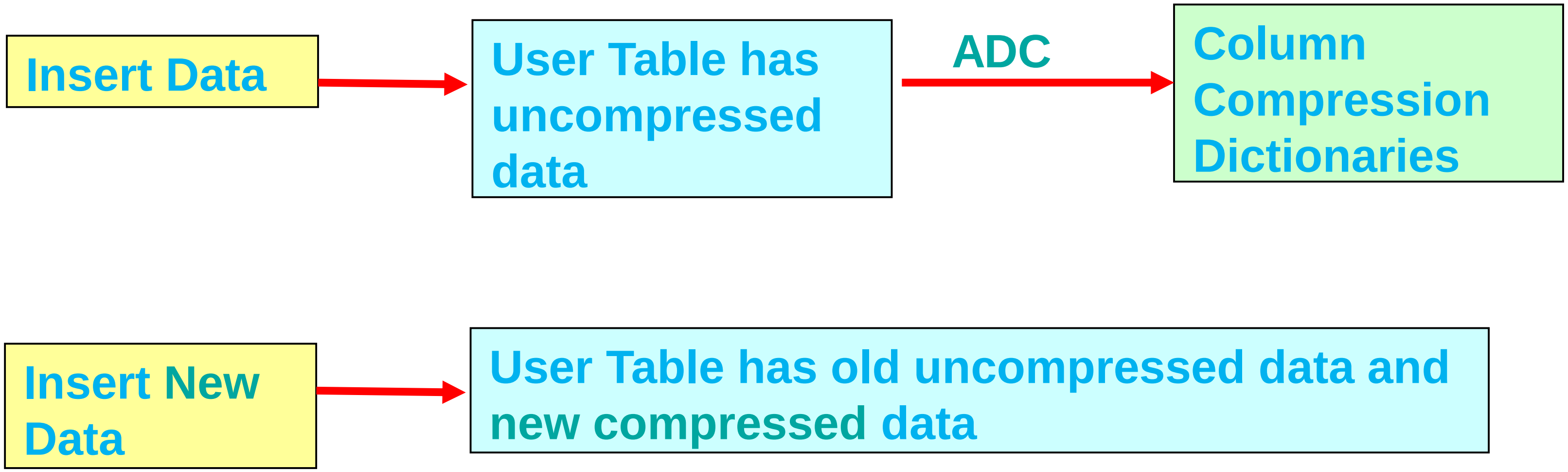
Utility Heap

Bigger is Better

- •Faster Load Performance
- •Better Compressed Tables
- •Faster Query Performance

- **Load allocates memory from utility heap**
- **util_heap_sz** recommendations:
  - At least **1,000,000** pages
  - **4,000,000** pages if database server has >= 128 GB of memory
  - If concurrent utilities need to be run, util_heap_sz should be increased to accommodate higher memory requirements
  - Consider reducing util_heap_sz after load completes to have more SORTHEAP memory for query usage

Inserting into Column-Organized Tables



Insert Data

User Table has uncompressed data

ADC

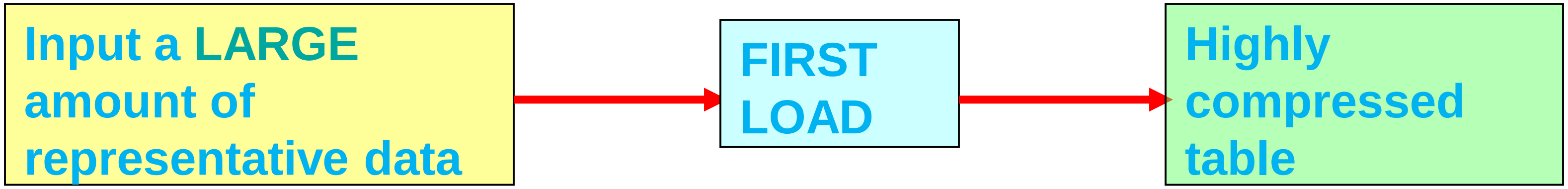
Column Compression Dictionaries

Insert New Data

User Table has old uncompressed data and new compressed data

- Initial data inserted before Automatic Dictionary Creation is uncompressed
- When threshold number of values inserted, ADC builds column compression dictionaries
    - Need enough input values to build effective dictionaries
- New values inserted after dictionaries are built are compressed
- Support both page-level and column-level dictionaries when data is inserted or ingested

Recommendations to get Good Compression



☑ **Load instead of Insert for initial dictionary creation**

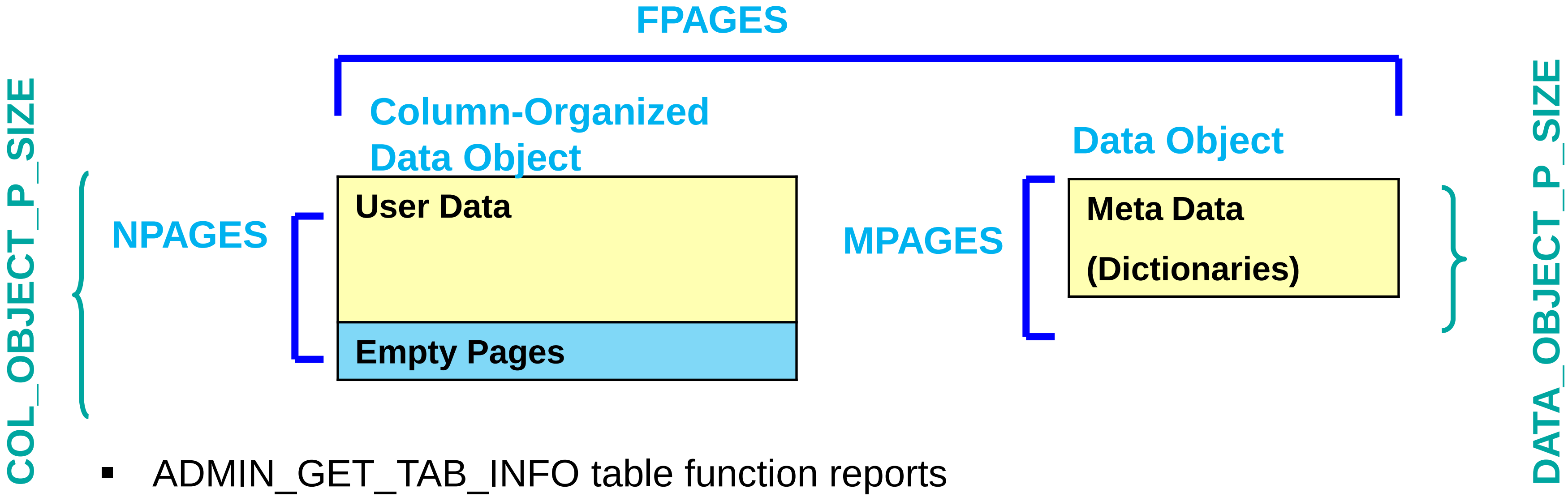
- Load utility can analyze more initial data than ADC during Insert and build better column compression dictionaries
- Values inserted before ADC won't be compressed at the column level

☑ **Use sufficiently large amount of representative data in 1st Load that builds dictionaries**

☑ **Set util_heap_sz >= 1,000,000 pages**

☒ Don't load a small initial subset of data for 1st Load

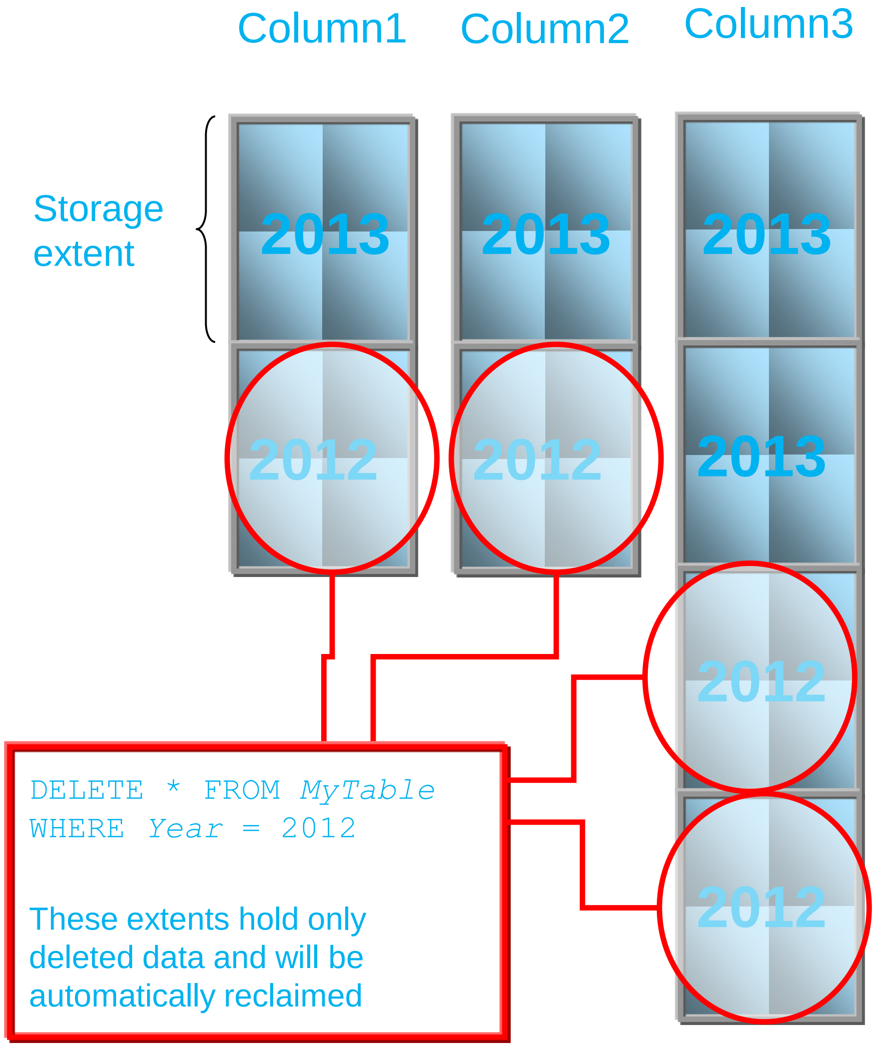
Calculating Column-Organized Storage Sizes



- ADMIN_GET_TAB_INFO table function reports
  - **COL_OBJECT_P_SIZE:** Physical size of column-organized data object containing user data
  - **DATA_OBJECT_P_SIZE:** Physical size of data object containing meta data
  - **INDEX_OBJECT_P_SIZE:** Physical size of the Page Map/Unique Indexes defined on the table
- TOTAL_SIZE = COL_OBJECT_P_SIZE + DATA_OBJECT_P_SIZE + INDEX_OBJECT_P_SIZE

Automatic Space Reclaim

- Automatic space reclamation
  - Frees extents with no active values
  - The storage can be subsequently reused by any table in the table space
- No need for costly DBA space management and `REORG` utility
- Enabled out-of-the box for column-organized tables when `DB2_WORKLOAD=ANALYTICS`
- Space is freed online while work continues
- Regular space management can result in increased performance of `RUNSTATS` and some queries



```
DELETE * FROM MyTable
WHERE Year = 2012
```

These extents hold only deleted data and will be automatically reclaimed

Module Content

- Value of BLU Acceleration
- **Getting started with BLU Acceleration**
- **LOAD & Compression**
- **Query Execution and Workload Management**
- Shadow Table and HADR support
- Key Benefits of DB2 BLU

Query Execution in DB2 with BLU Acceleration

- BLU Acceleration is more than columnar storage!
- BLU Acceleration = columnar data storage
  + columnar query runtime
  + *many other optimizations*

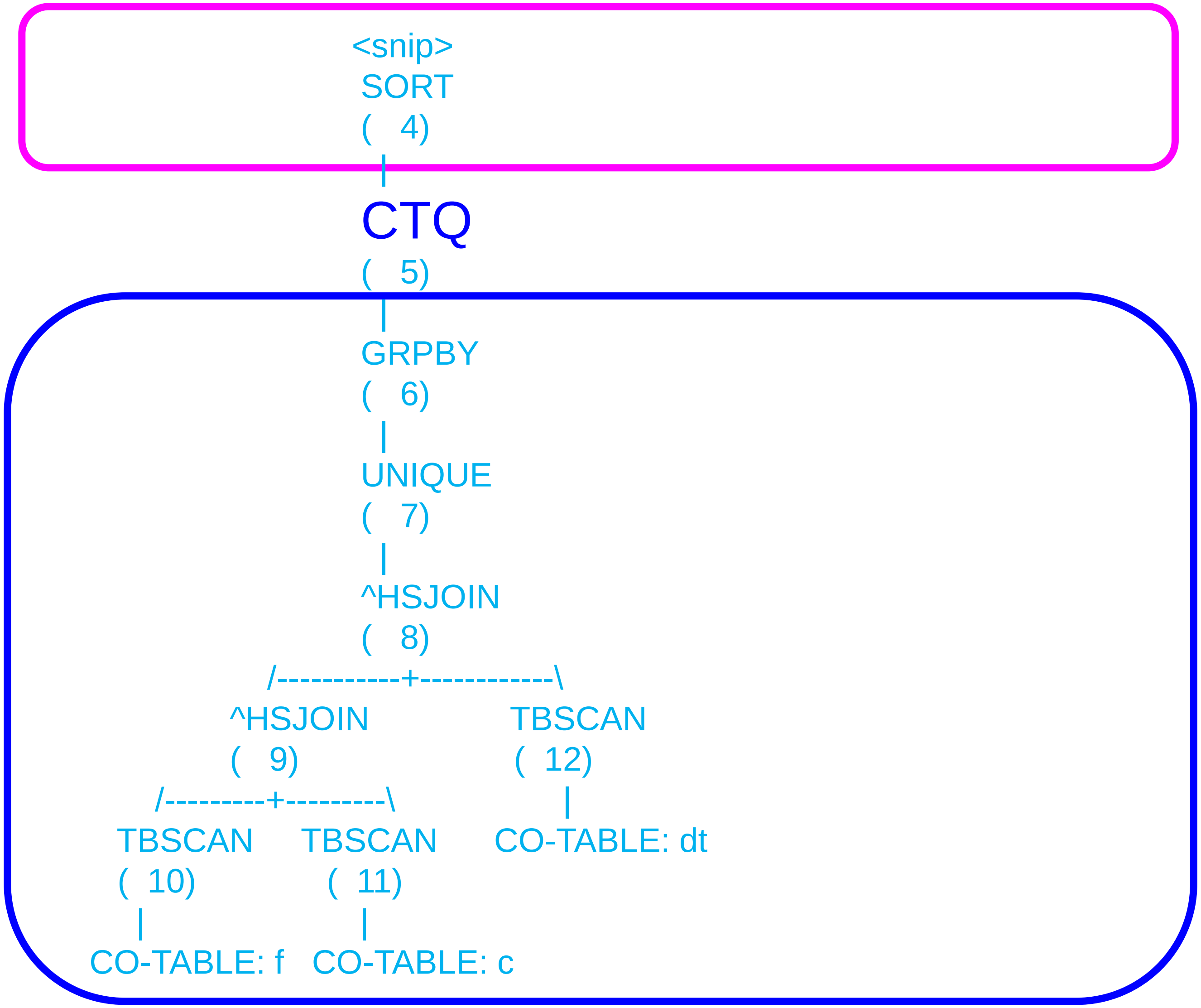
- New operator in execution plans: CTQ
- CTQ indicates transition between column-organized and row-organized data processing
- Leverage the usage of OQWT to analyze the workload well in advance

Sample Query

```
SELECT c.trading_name
FROM f, c, dt
WHERE f.client_dim_key = c.client_dim_key
  AND f.trade_dt = dt.dt_dim_key
  AND f.is_cancelled = 0
GROUP BY c.trading_name, dt.year
ORDER BY c.trading_name
```

Let’s review the execution plan of this query….

Sample Execution Plan



```
                    <snip>
                     SORT
                     (   4)
                       |
                     CTQ
                     (   5)
                       |
                     GRPBY
                     (   6)
                       |
                     UNIQUE
                     (   7)
                       |
                     ^HSJOIN
                     (   8)
                /-----------+-----------\
         ^HSJOIN                  TBSCAN
         (   9)                   (  12)
     /---------+---------\           |
 TBSCAN      TBSCAN            CO-TABLE: dt
 (  10)       (  11)
    |            |
CO-TABLE: f  CO-TABLE: c
```

Operators above CTQ use DB2’s regular row-based processing

Operators below CTQ are optimized for column-organized tables

Here: All table scans, hash joins, and grouping are performed in columnar query runtime. (Good.)

Execution Plans

- Runtime operators are optimized for row- and column-organized tables
- CTQ operator transfers data from column- to row-organized processing
- Operators that are optimized for column-org tables below CTQ include:
  - Table scan
  - Hash-based join, optionally employing a semi-join
  - Hash-based group by. Potentially faster without the sort
  - Hash-based unique
- Aim is to “push down” most operators below CTQ
- Some operations cannot be pushed down such as:
  - SORT
  - SQL OLAP function (e.g., rank())
  - Inequality join (join predicate that is NOT equality)

Typical characteristics of a…

…good plan

- One or few CTQ operators
- Few operators above CTQ
- Operators above CTQ work on few rows
- Few rows flow through the CTQ

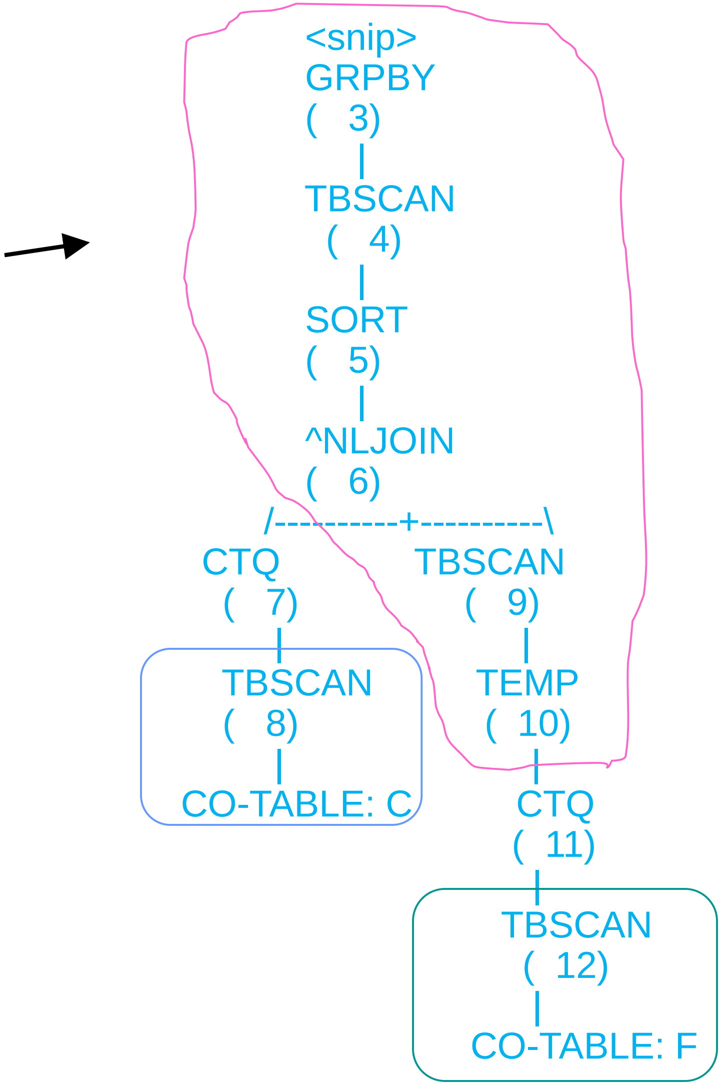
…a suboptimal plan

- Many CTQs
- Many operators above CTQ
- Operators above CTQ work on many rows
- Many rows flow through the CTQ

Disclaimer: While these are the most common indicators of good vs suboptimal plans with column-organized tables, there can be exceptions.

Example of a Suboptimal Execution Plan (1)

Operators above CTQ use DB2's regular row-based processing



```
        <snip>
        GRPBY
        (   3)
          |
        TBSCAN
         (   4)
          |
        SORT
        (   5)
          |
        ^NLJOIN
        (   6)
   /----------+----------\
CTQ                    TBSCAN
 (   7)                 (   9)
   |                      |
  TBSCAN                 TEMP
  (   8)                 (  10)
   |                      |
CO-TABLE: C              CTQ
                         (  11)
                          |
                        TBSCAN
                         (  12)
                          |
                       CO-TABLE: F
```

```
SELECT c.trading_name
FROM f,c
WHERE f.client_dim_key >
      c.client_dim_key
GROUP BY c.trading_name
```

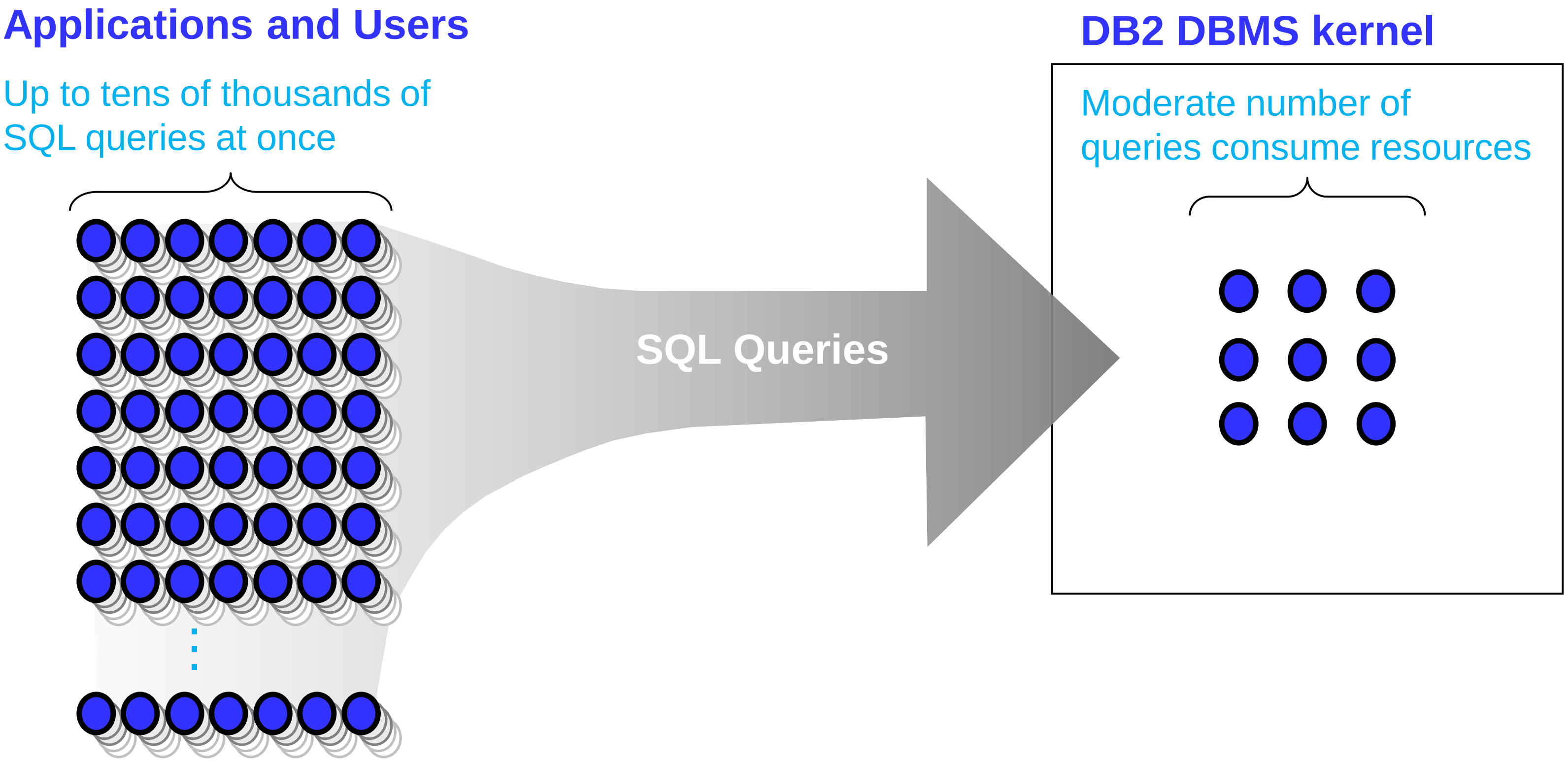
Here:

Two table scans deliver potentially many rows into the CTQs.

Subsequent join is executed with regular row-based processing. Does not benefit from column-optimized join techniques.

Automatic Workload Management

- Built-in and automated query resource consumption control
- Enabled automatically when `DB2_WORKLOAD=ANALYTICS`
- Many queries can be submitted, but limited number get executed concurrently


Applications and Users
Up to tens of thousands of SQL queries at once
SQL Queries
DB2 DBMS kernel
Moderate number of queries consume resources

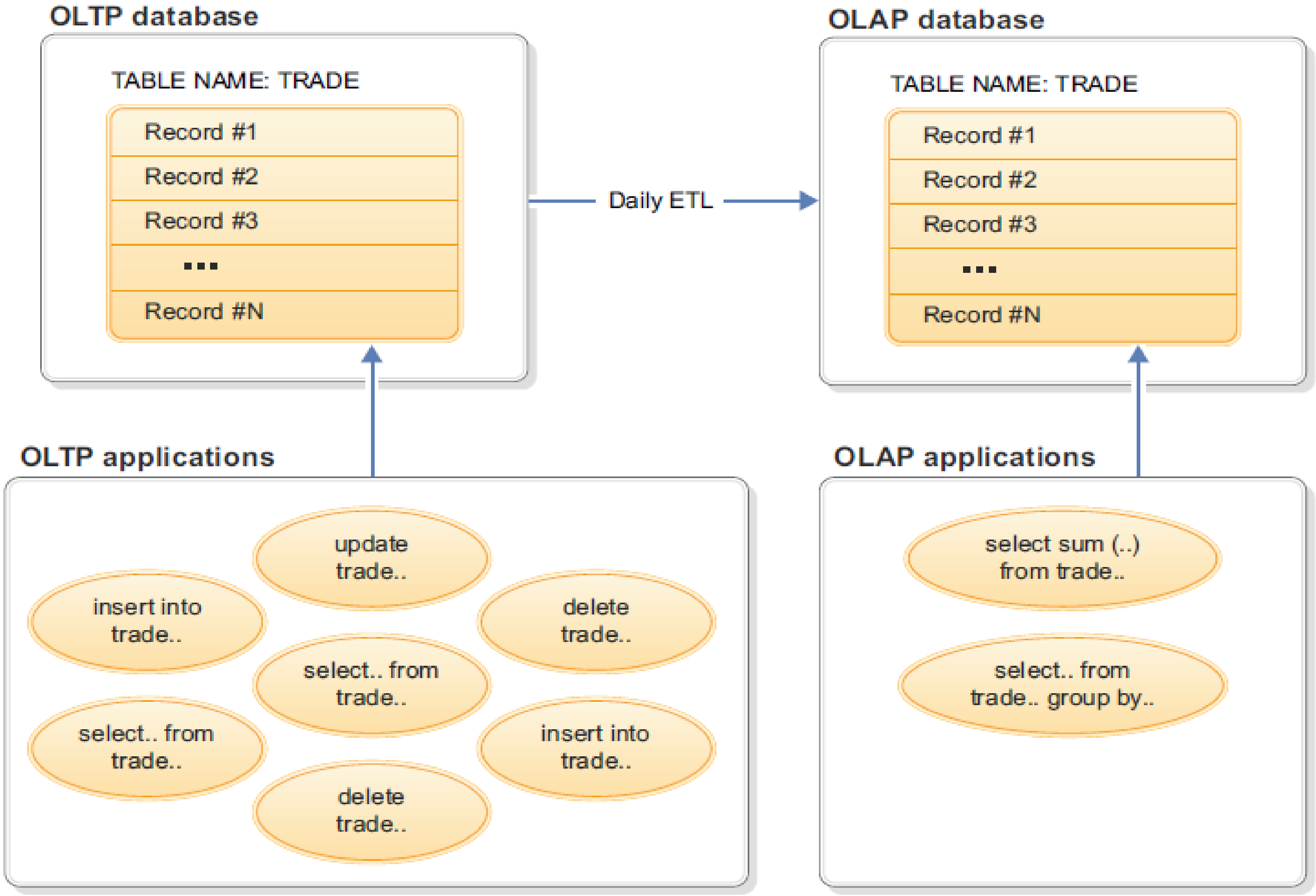
Module Content

- Value of BLU Acceleration
- Getting started with BLU Acceleration
- LOAD & Compression
- Query Execution and Workload Management
- **Shadow Table and HADR support**
- Key Benefits of DB2 BLU

DB2 Shadow Tables - Reporting on Transactional Data

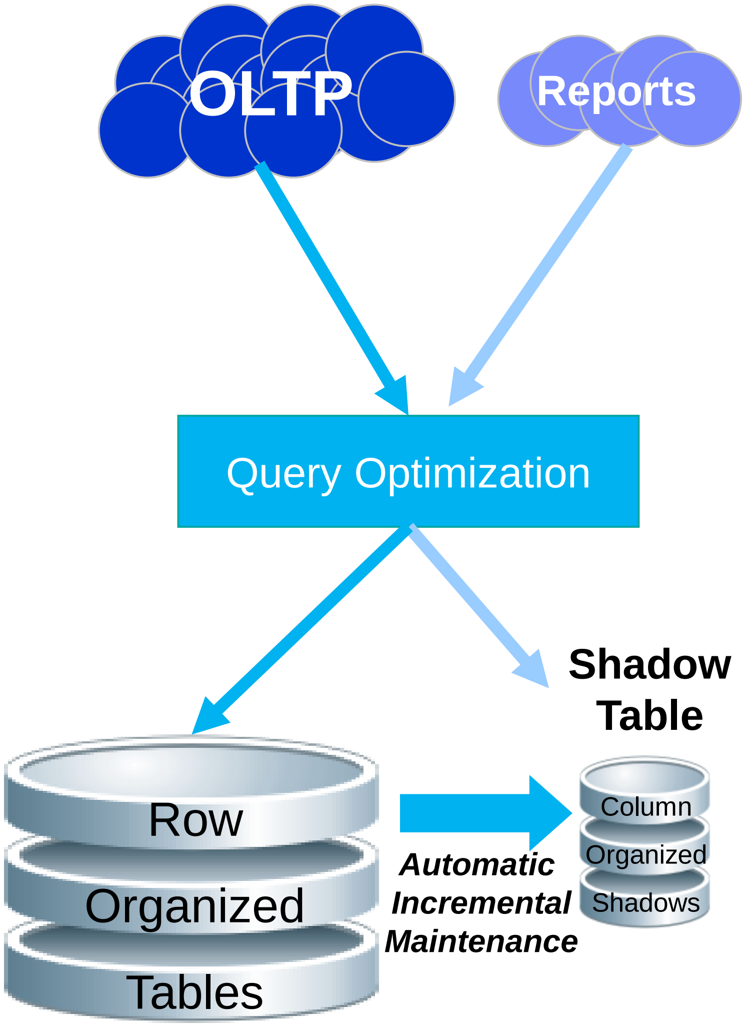
- Traditional analytic workloads are in many cases implemented in data warehouse, data mart as OLAP
- Data normally originates from OLTP and replicated to OLAP system
  - Disadvantage: Complex ETL setups, massive data movement, data time lag
- In order to improve business agility, organizations need to perform analytics processing of real-time data on OLTP system
- Reporting on transactional data (or Analytics on OLTP data) is the solution for the above business need

High Level Architecture: Traditional OLAP

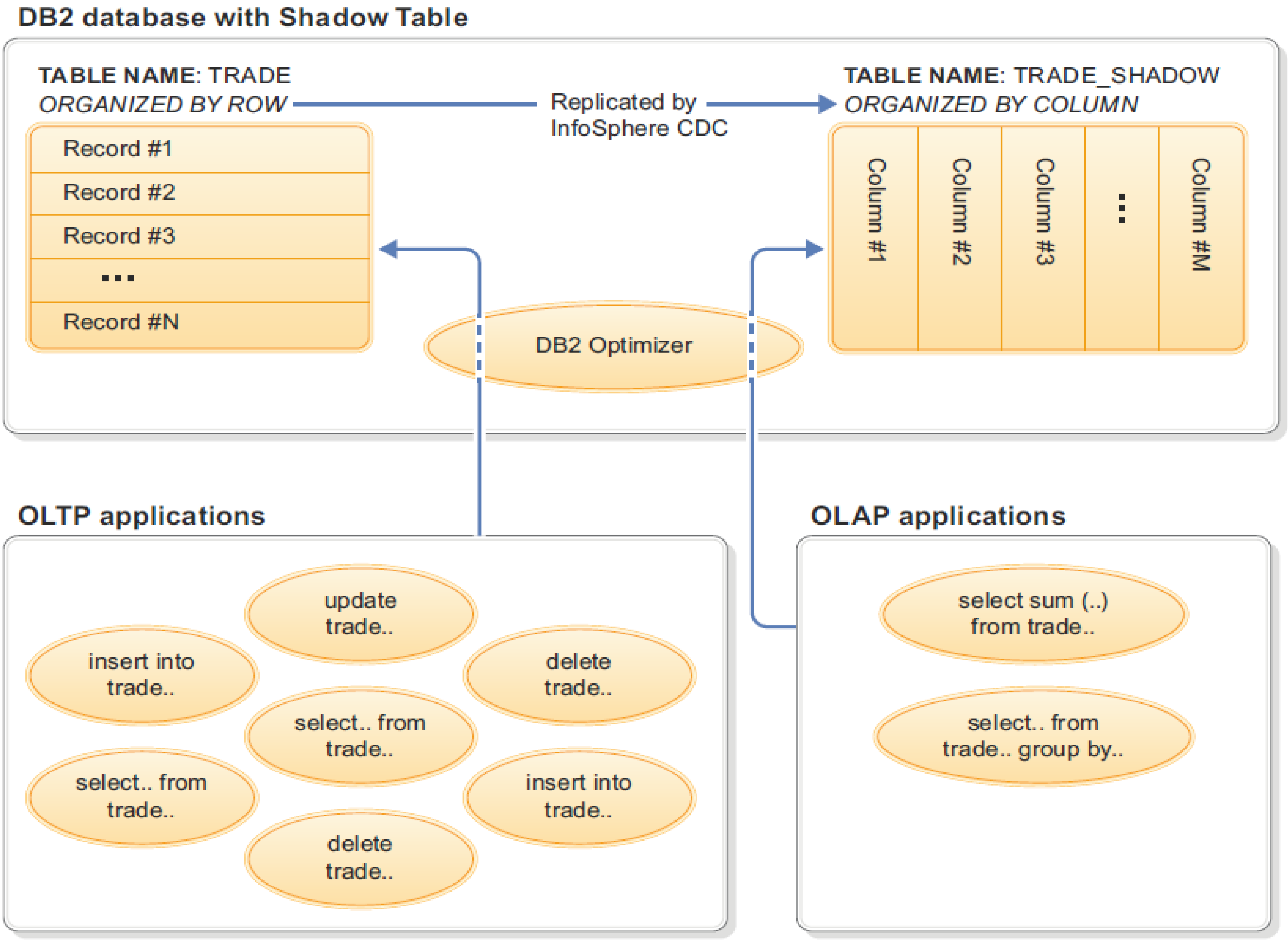


What Is a Column-Organized Shadow Table

- Transparent BLU “Shadow Table”
  - A new way to implement an MQT as a columnar data store
- Powered by DB2 BLU Acceleration
  - Queries only perform I/O on the columns and values that match query. Work performed directly on columns
- Smart: Analytical queries issued against the normal row based table automatically routed to shadow tables to take advantage of BLU Acceleration
- Improved performance vs. traditional MQT



High Level Architecture: Shadow Tables – DB2 View

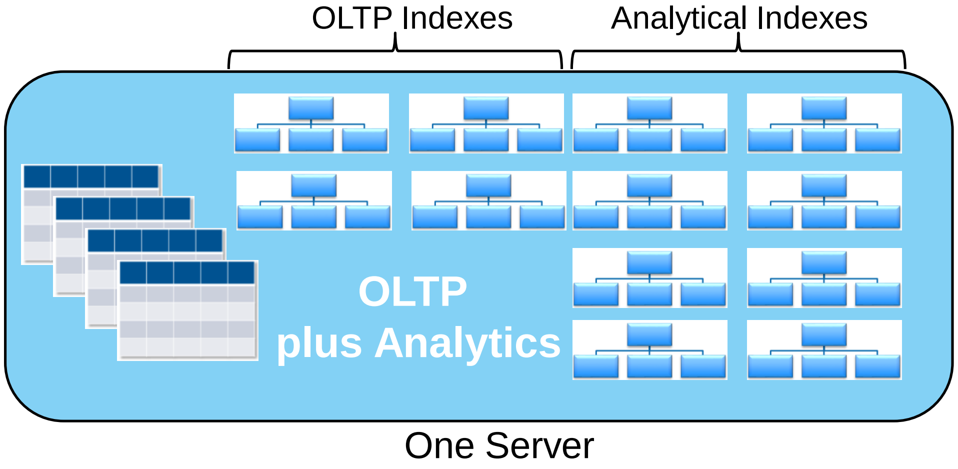


Shadow Tables: Key Facts

- OLTP transactions access the row-organized tables
- OLAP transactions access the copies of the same tables, but column-organized. These are called Shadow Tables
- Both types reference the same table names. No change is required to OLAP queries
- OLAP transactions are much faster when accessing BLU, column-organized tables than row-organized tables
- Shadow Tables, based on BLU technology, do not require any index other than primary key index
- Shadow Tables simplify administration and boost performance
- IBM InfoSphere CDC is used to maintain Shadow Tables and is included in the DB2 10.5 Advanced Enterprise Server Edition (AESE) or Advanced Workgroup Server Edition (AWSE) offerings (as well as Developer Edition (DE))

Shadow Tables
Great Performance – Simplified Administration

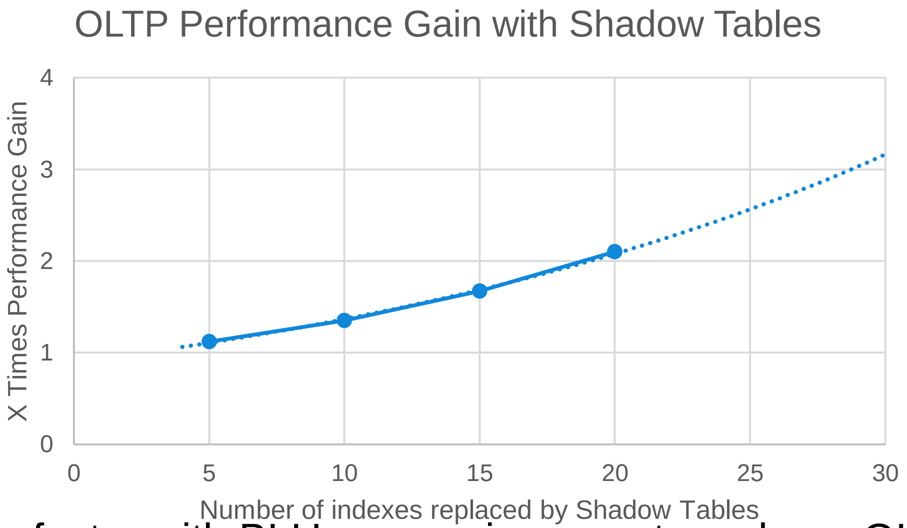
- OLTP system with OLTP indexes and several extra indexes to support reporting queries



- Shadow Tables simplify administration and boost performance
  – Greatly reduces indexes needed for performance – no analytical indexes
  – Greater than 10x faster reporting
  – No impact to OLTP throughput by replacing just 4 analytic indexes with Shadow Tables
    • In fact, when dropping 20 indexes OLTP performance improves by more than 2x

Shadow Tables
Performance Compared to a Single System

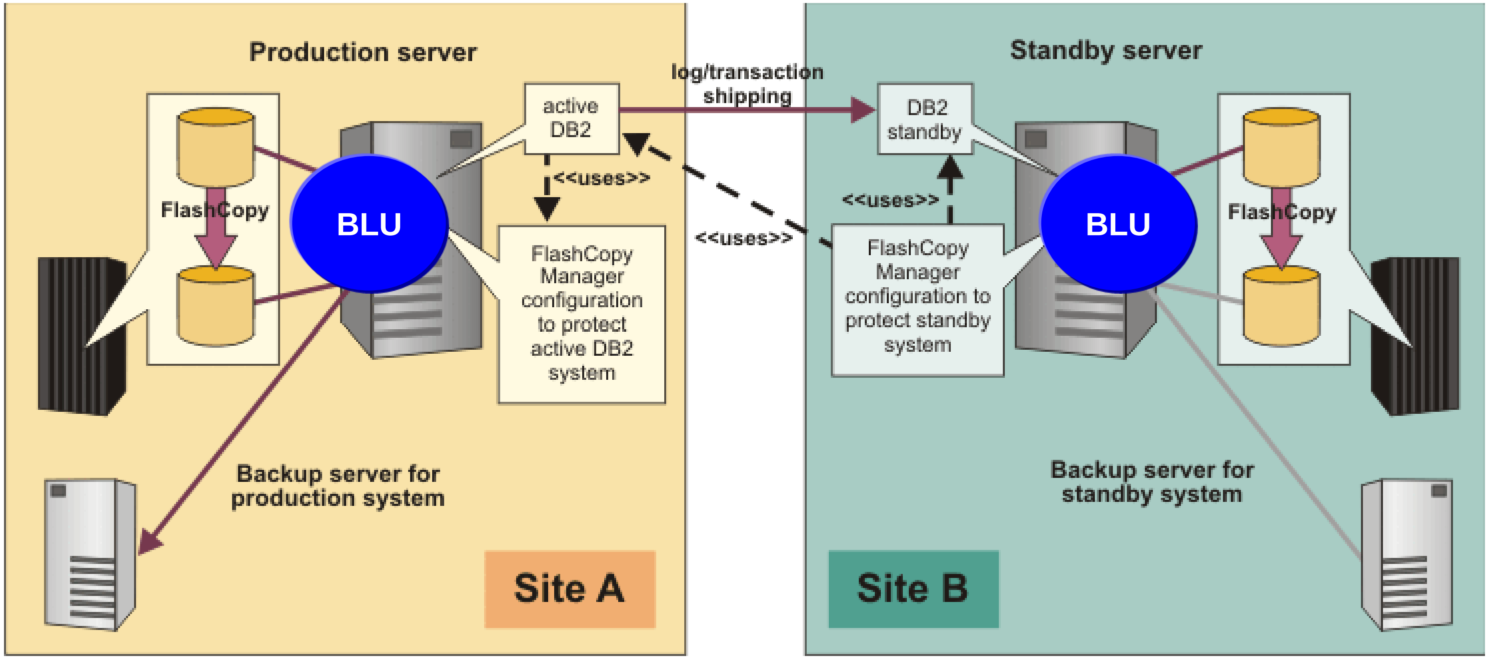
- With only 4 OLAP indexes replaced by BLU Shadow Tables, performance of OLTP is the same
  - No additional impact to OLTP when using Shadow Tables
  - Reduced memory consumed for indexes by 10x with Shadow Tables
  - Other vendors are talking about replacing 10-20 indexes (not typical with OLTP systems)



- Reporting queries >10x faster with BLU vs. running reports on base OLTP tables

Simple HA and DR Solution for BLU

- BLU Acceleration with HADR availability
  - Highly available analytics
  - Use for both HA and DR
  - Includes most HADR capabilities
    - All synchronization modes, multiple standby, time delay, and log spooling
    - Read-on-Standby not supported in this release
  - Shadow Tables and HADR support added in Fix Pack 5


Production server
active DB2
<<uses>>
FlashCopy
BLU
FlashCopy Manager configuration to protect active DB2 system
Backup server for production system
Site A
log/transaction shipping
<<uses>>
Standby server
DB2 standby
<<uses>>
FlashCopy Manager configuration to protect standby system
BLU
FlashCopy
Backup server for standby system
Site B

DB2 10.5 with BLU Acceleration

- BLU Acceleration provides three key benefits:
  - Fast
    - Unprecedented performance for analytical workloads, often 8x to 25x faster.
    - Examples of workloads > 100x
    - Examples of individual queries > 1000x
  - Small
    - Stronger compression and less space required for auxiliary data structures.
    - 10x savings is versus uncompressed row-tables is common.
  - Simple
    - Much less tuning needed, more predictable and reliable performance
    - Tuning, statistics collection, space reclaim, workload management all tuned and automated right out of the box
    - Adapts automatically to your server’s memory and CPUs

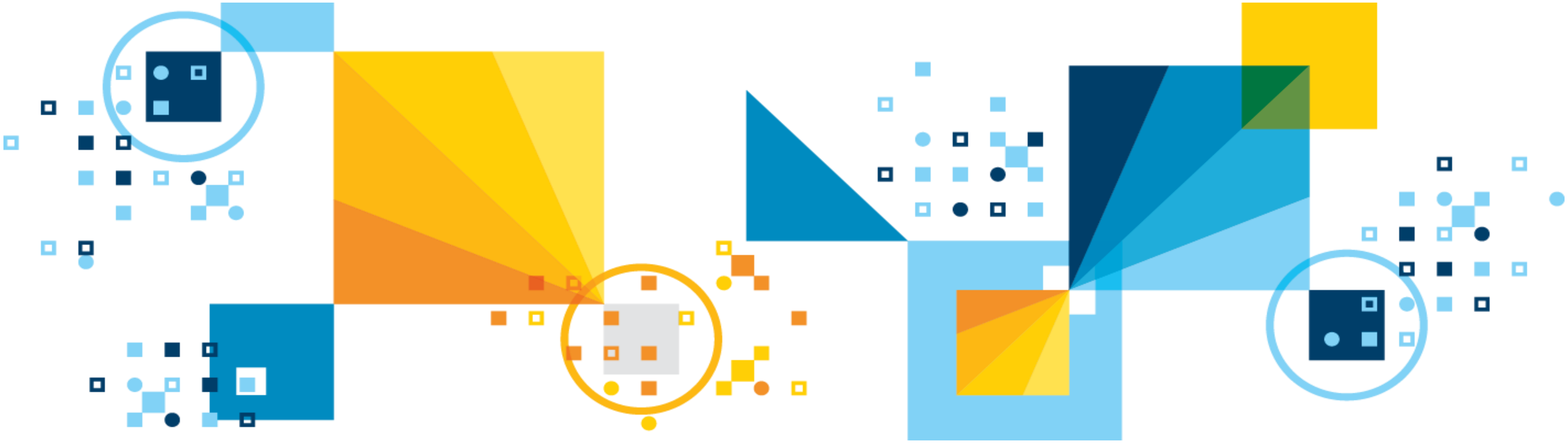
> *“Intel is excited to see* ***greater than 30x improvement*** *in query processing performance using DB2 10.5 with BLU acceleration over DB2 10.1. To achieve these amazing gains, IBM has taken advantage of the Advanced Vector Extensions (AVX) instruction set on Intel® Xeon® processor E5-based systems. Customers running this hardware can now immediately* ***realize dramatically greater performance boost at lower cost per query.***”
>
> *-Pauline Nist, Intel General Manager, Enterprise Software Alliances, Datacenter and Connected Systems Group*

The next steps…

The Next Steps…

- Complete the online quiz for this module
  - Log onto SKI, go to “My Learning” page, and select the “In Progress” tab.
  - Find the module and select the quiz
- Provide feedback on the module
  - Log onto SKI, go to “My Learning” page
  - Find the module and select the “Leave Feedback” button to leave your comments

Questions?
askdata@ca.ibm.com



January 30, 2015