What’s New with BLU in Db2 11.5 (and beyond!)

John Hornibrook
(Chris Drexelius)
IBM

Db2 LUW

IDUG
Leading the Db2 User Community since 1988

Db2 with BLU Acceleration provides industry leading performance for analytical applications running complex queries through its advanced column-organized table technology. Db2 11.5 continues to extend the reach of BLU capability to more Db2 features so that its superior performance can be applied to a broader range of applications. Additionally, BLU storage and compression technology has been enhanced to make it more effective and even easier to use. Attend this session to learn how these enhancements can allow BLU to be used for even more applications within your business.

Please note :

- IBM's statements regarding its plans, directions, and intent are subject to change or withdrawal without notice and at IBM's sole discretion.
- Information regarding potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision.
- The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code or functionality. Information about potential future products may not be incorporated into any contract.
- The development, release, and timing of any future features or functionality described for our products remains at our sole discretion.
- Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon many factors, including considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve results similar to those stated here.

Notices and disclaimers

•Information in these presentations (including information relating to products that have not yet been announced by IBM) has been reviewed for accuracy as of the date of initial publication and could include unintentional technical or typographical errors. IBM shall have no responsibility to update this information. **This document is distributed "as is" without any warranty, either express or implied. In no event, shall IBM be liable for any damage arising from the use of this information, including but not limited to, loss of data, business interruption, loss of profit or loss of opportunity.** IBM products and services are warranted per the terms and conditions of the agreements under which they are provided.

•IBM products are manufactured from new parts or new and used parts. In some cases, a product may not be new and may have been previously installed. Regardless, our warranty terms apply."

•**Any statements regarding IBM's future direction, intent or product plans are subject to change or withdrawal without notice.**

•Performance data contained herein was generally obtained in a controlled, isolated environments. Customer examples are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual performance, cost, savings or other results in other operating environments may vary.

•References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business.

•Workshops, sessions and associated materials may have been prepared by independent session speakers, and do not necessarily reflect the views of IBM. All materials and discussions are provided for informational purposes only, and are neither intended to, nor shall constitute legal or other guidance or advice to any individual participant or their specific situation.

•It is the customer's responsibility to insure its own compliance with legal requirements and to obtain advice of competent legal counsel as to the identification and interpretation of any relevant laws and regulatory requirements that may affect the customer's business and any actions the customer may need to take to comply with such laws. IBM does not provide legal advice or represent or warrant that its services or products will ensure that the customer follows any law.

Agenda

- Insert, Update, Delete Performance Improvements
- Large objects (LOBs)
- Spatial Support
- Temporal Tables
- Triggers
- Referential Integrity
- Indexing

Insert/Update/Delete Performance Enhancements

- Db2 11.5 greatly expands core-friendly parallelism for SQL-based IUD operations on columnar tables
  - KIWI: Kill It With Iron
  - Maximize CPU cache, cacheline efficiency
- Critical to maximize ETL/ELT batch performance
- Many general improvements, but primary focus on bulk operations

Core-friendly parallelism is one of the core ingredients of BLU's Secret Sauce.

- Make sure we are able to fully utilize all aspects of modern processors including SMP.
- Leverage CPU cache properties for performance.

Our business is data, so ETL/ELT processing is key.

All IUD operations were previously treated the same, regardless of what type they were.

General improvements include:

- Path-length improvements for insert codepath
- Logging enhancements to reduce the size, complexity, and in some cases need of log records.

Bulk Insert/Update Examples 1

- Insert-Subselect:

```
INSERT INTO MYSCHEMA.BIG_TABLE (SELECT * FROM
MYETLSCHEMA.STAGING_TABLE)
```

- External Table Load:

```
CREATE EXTERNAL TABLE EXTTBL_NAME
(first_col int, second_col int, third_col int) USING
(DATAOBJECT('/path/to/data/exttbl.csv') DELIMITER ',’)

INSERT INTO MYSCHEMA.BIG_TABLE
  (SELECT * FROM EXTTBL_NAME)
```


Bulk/trickle decision ultimately up to the optimizer.

ET load also new for v11.5, but covered in another session.

Bulk Insert/Update Examples 2

- Update:
```
UPDATE TABLE MYSCHEMA.BIG_TABLE SET THIRD_COL =
THIRD_COL + 1
```

- CREATE TABLE AS (CTAS):
```
CREATE TABLE NEWSCHEMA.NEW_TABLE AS
 (SELECT * FROM MYSCHEMA.BIG_TABLE)
WITH DATA ORGANIZE BY COLUMN
```

IDUG Leading the Db2 User Community since 1988

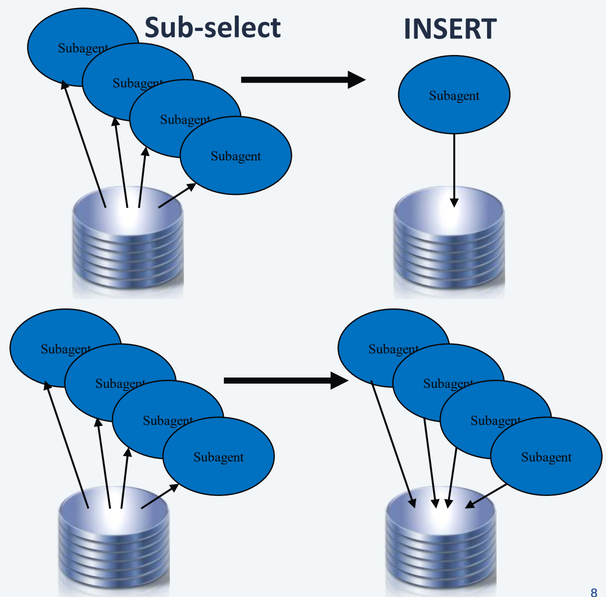
IDUG Db2 Tech Conference
Rotterdam, Netherlands | October 20-24, 2019

#IDUGDb2

Parallel Insert/Update/Delete

- BLU query processing leverages core-friendly parallelism
  - Excellent scalability for large SMPs
  - Combine SMP and MPP scaleout
- BLU bulk IUD now provides similar parallelism
  - Parallel insert available in v11.1.1.2
  - Parallel update and delete v11.5

```
INSERT INTO table2 SELECT *
FROM table1
```



8

BLU's query processing engine already heavily exploits core-friendly parallelism.

- SMP + MPP (DPF) is an especially effective combination.

Parallelism of 4 selected for this example, but may be higher.

All table types supported.

Parallel update and delete new for v11.5.

Vectorized Insert/Update 1

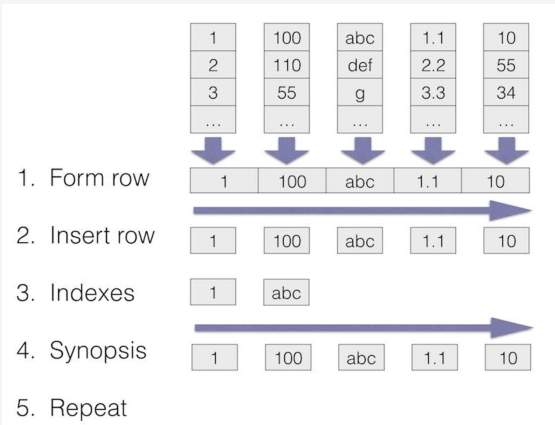
- BLU query processing leverages vectors of columnar tuplets
  - Enables bulk processing on columns instead of row by row
  - Maximizes cache and cacheline efficiency
- Bulk insert/update operations benefit from similar access pattern

Vectorized insert/update new for v11.5.

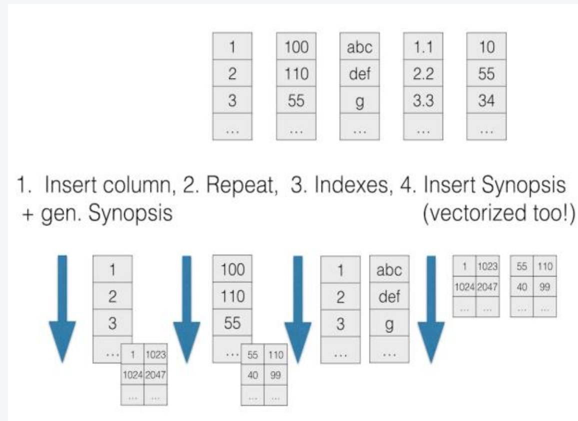
CPU cache and cacheline efficiency are also critical for IU operations.

Vectorized Insert/Update 2

Pre-v11.5 approach



v11.5 approach



We leverage vectors during bulk IU operations in much the same way.

Optimized Bulk Insert Codepath

- Cache and cacheline efficiency by processing vectors
- More efficient encoding and data page filling
  - Buffer incoming tuplets in memory instead of writing directly to data pages
- Page compression costs greatly reduced
  - No need to merge written pages
- Batch index updates

New insert codepath optimized for bulk inserts/updates in v11.5.

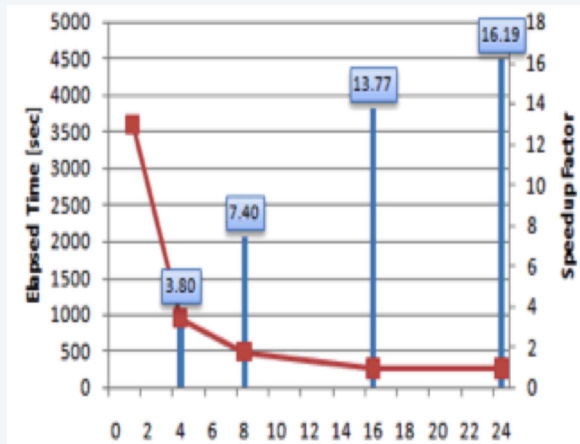
- Encode many tuplets before processing.
- Buffer encoded tuplets in memory
- Detect exactly when a page is filled without any unnecessary page writes.
- Efficient index updates.

IDUG Db2 Tech Conference
Rotterdam, Netherlands | October 20-24, 2019
#IDUGDb2

ETL Performance Example

- Data ingest rate
  - 1 TB/hour before enhancements
  - ***Now ~5 TB/hour (IIAS)***
- >10 TB data
- Table remains online
- Combined features
  - External Table load
  - Parallel insert
  - Vectorized insert
  - Optimized bulk insert codepath

Parallel Insert Degree/Time/Speedup



Degree

12

Now able to match or exceed Netezza’s speed in many cases!

This also applies to on-prem installations, though hardware limitations may be a limiting factor.

Compression Enhancements

- BLU leverages extreme compression
- Before v11.5, best practice to use LOAD utility
  - Creates high-quality evolved dictionary
  - Maximizes amount of encoded data
- SQL insert codepath enhanced in v11.5
  - Optimize evolved dictionary creation for bulk inserts
  - Enable dictionary creation for table with Z-lock and DGTT inserts
- Optimized encoding method
- New automatic REORG recompress option recompresses uncompressed tuplets

Extreme compression is another one of the core ingredients of BLU's Secret Sauce.

Previously, best practice for effective dictionary creation is the load utility.

+ Excellent dictionary

+ Once dictionary created, as much data as possible encoded.

+ Excellent performance

- Table offline
- May require workload changes

Goal is to enhance SQL-based insert to exceed Netezza performance and overcome load limitations.

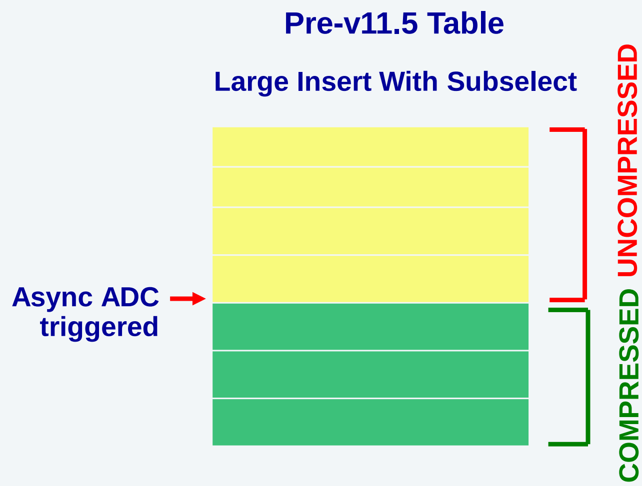
- All previous items discussed contribute to this goal
- Focus is bulk operations, but also enable dictionary creation for other special cases

Optimized encoding

- Hash-based encoding method
- Datatype- and dictionary size-dependent

Automatic Dictionary Creation

- Goals of ADC
  - High quality evolved dictionary
  - Build it fast
  - Minimal impact on inserts
- Increased ADC threshold
  - When to launch ADC
  - How much data to sample



14

Achieve highest quality dictionary by using actual data to build histograms.
Build it fast to minimize amount of uncompressed data.

Previous ADC threshold was too low, leading to an ineffective dictionary.
- New per-database partition threshold is 500,000 for single database partition or 1,000,000 / # database partitions for multiple partitions

Asynchronous ADC (pre 11.5)

- Executes in the ***background*** as ***separate transactions***
- Scans data pages on disk and ***in memory***
- Collects histograms from other database partitions
- Builds and distributes an evolved dictionary

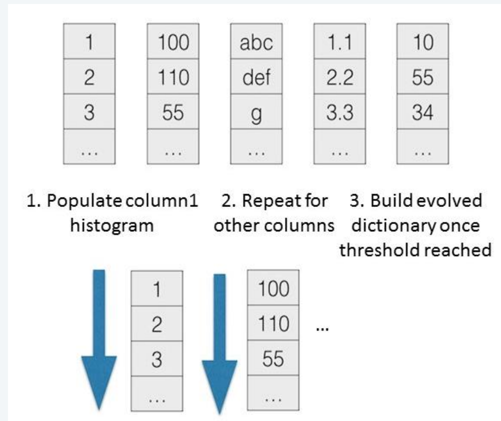
Original ADC method was asynchronous and has been available in BLU for several releases.

NOTE **bold** words.

- Re-reading in-memory pages requires extra serialization – actually quite expensive.
- Executing in the background means inserts continue full speed ahead.
- Separate transactions mean that certain scenarios where there is a Z-lock on the table or DGTTs will not evolve dictionaries.

Vectorized ADC

- Optimized for bulk insert
- Executes within insert threads, even across streams
- Maximizes cache and cacheline efficiency
- Once dictionary build starts, delay insert threads.



ADC method designed for bulk insert – new for v11.5.

Insert threads work together even across insert streams to populate histograms.

Leverages columnar tuplet vectors.

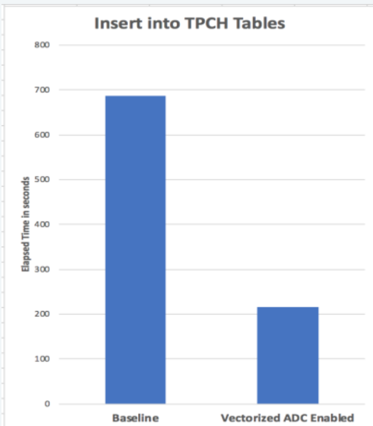
Delaying inserts is non-intuitive, but key point is we want to maximize amount of encoded data.

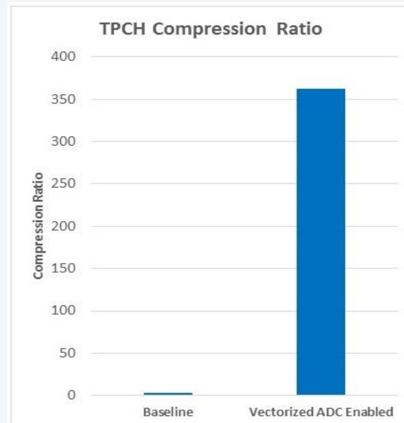
IDUG Leading the Db2 User Community since 1988

IDUG Db2 Tech Conference
Rotterdam, Netherlands | October 20-24, 2019

#IDUGDb2

Vectorized ADC Performance





- Insert ET data into tables using the 100 GB TPCH data set. Run in a single node.
- 3.2x insert speed and huge compression ratio difference observed with Vectorized ADC enabled!

17

Dramatic performance results!!

IIAS-like system used for TPCH.

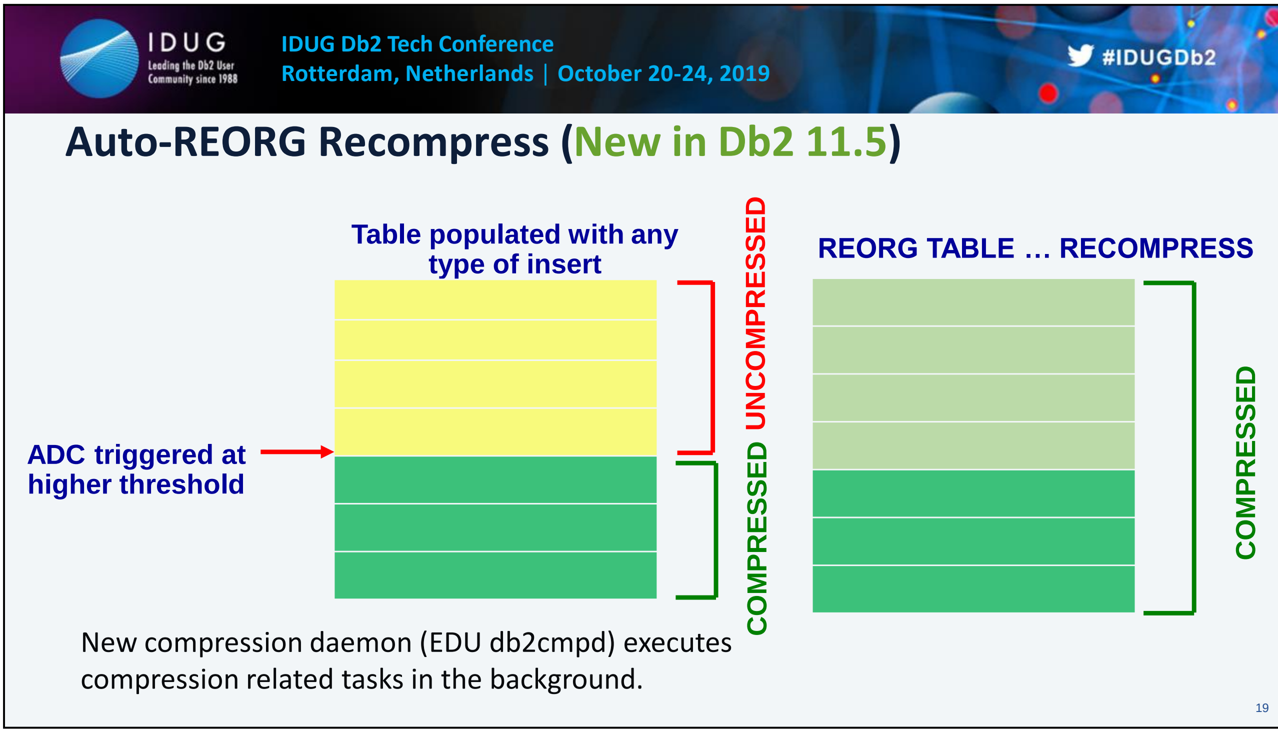
Financial customer on IIAS:
Raw data size: 712 GB
Compressed Db2 for z/OS size: 162 GB
Db2 BLU without vectorized ADC: 160+ GB, multiple hours
Db2 BLU with vectorized ADC: 57 GB, 25 minutes

Synchronous ADC

- Supports dictionary evolution for special non-bulk insert cases
  - Z-lock on permanent table
    - CTAS
    - NLI
    - Insert into uncommitted table
  - DGTT tables

Enhancement for non-bulk insert/update scenarios – new for v11.5.

Insert threads populate histograms.
Merge with master database partition-level histogram occasionally to control memory usage.
- Merge requires heavyweight synchronization.



Auto-REORG recompress is new for v11.5.

BLU query engine performance depends on encoded data.

- Hopefully unencoded data (yellow) is tiny.
- Even small amount of data at the beginning of a table could make a notable difference.

Auto-REORG recompress executes in the background once we evolve the dictionary.

Also able to handle existing tables from previous releases that have not yet been recompressed.

If an insert operation is used to populate a table, automatic creation of the compression dictionaries begin once a threshold count of rows are inserted into the table. This design ensures that a large enough sample of rows exists to build dictionaries that yield an adequate compression ratio. Rows that are inserted before the compression dictionaries are created populated and are not initially compressed. Rows inserted after dictionary creation will be compressed. The Automatic Recompress feature for tables uses the new compression daemon to asynchronously check for tables that contain rows that have not been compressed at the front of the table. It then recompresses those rows in place which may leave empty extents in the table. These empty extents can be reclaimed later. Only the values at the front of the table that were inserted before the dictionary was created will be recompressed. The Automatic Recompress feature runs online, concurrent with other insert, update, delete, and select transactions. In addition to improving space utilization, it will also improve query performance which is more efficient processing encoded data. As a result, queries will run faster on a table where the front part of the table has been recompressed with the dictionary. The Automatic Recompress feature requires no user intervention and uses the REORG TABLE command with a new RECOMPRESS option to recompress the rows. The user may notice the REORG TABLE ... RECOMPRESS command while monitoring the table. For simplicity, this feature is fully automated and manual execution of this RECOMPRESS option is not supported

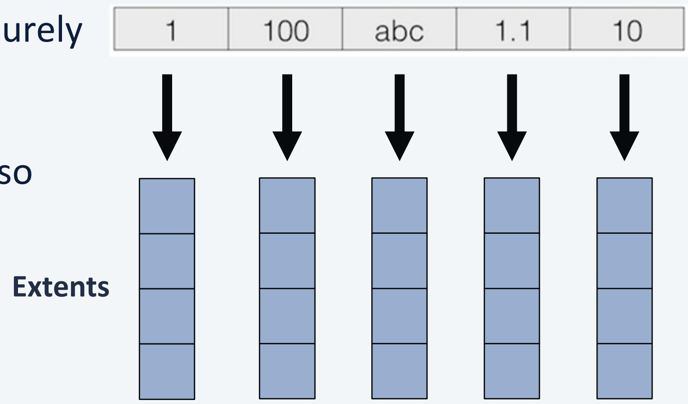
Post v11.5 BLU Storage and Compression Enhancements

- Trickle insert enhancements
- Improved compression of string datatypes
- Improved compression of numeric datatypes
- Automatic handling of data fragmentation due to updates/deletes
- Improved extent usage for small/medium tables

Trickle Insert Enhancements 1

- Not all BLU use cases are purely warehousing
  - IDAA
- OLTP-like workloads are also critical

v11.5 approach

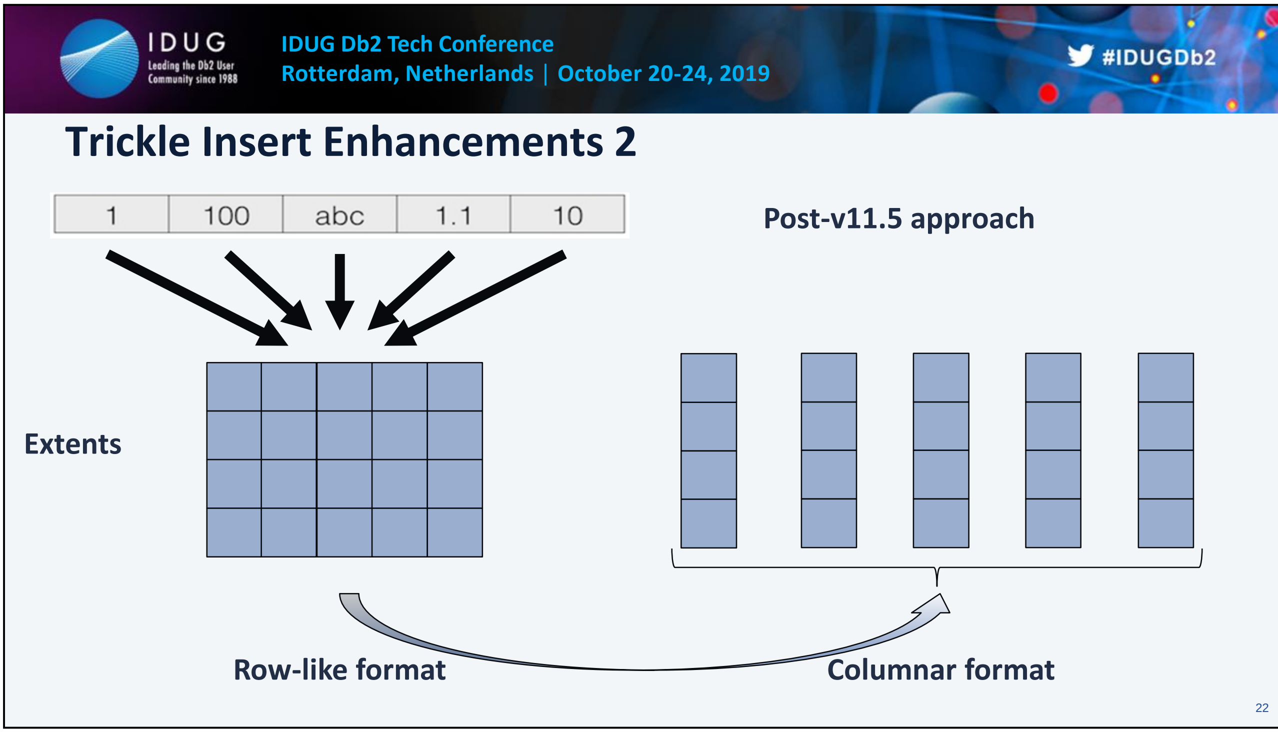


IDAA applies insert statements from Db2 for z/OS to Db2 BLU.

- OLTP inserts may be batched together, but not guaranteed.
- Critical customer offering

When a single row is inserted, each column in separate extent, so update all pages, multiple PMI entries, etc.

- Not scalable!

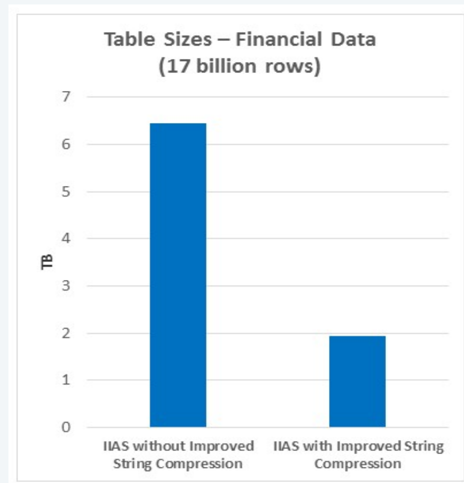


Solution: Use a more row-like format for trickle inserts

- Insert multiple columns into same page, more similar to row format
- Convert to traditional BLU data pages (one column group per extent) with extreme compression synchronously or asynchronously.
- Enable query engine to interpret new format.

Improved Compression of String Datatypes

- Frequency-based compression difficult for some string datasets
- String data dominates storage cost
- Add another level of pattern-based page compression



Frequency-based compression (even with our existing prefix compression) unable to provide coverage for some datasets.

- Geospatial data
- URLs
- Comments
- Etc.

String datatypes = CHAR, VARCHAR, GRAPHIC, VARGRAPHIC, BINARY, and VARBINARY

Enormous space savings possible for many such datasets!

Large Objects in Column-Organized Tables

- Datatypes supported: CLOB, BLOB, DBCLOB, NCLOB
- BLU user can now avoid storing large objects in row-organized tables.
- LOB meta-data and in-lined data are stored in the column-organized data page
  - LOBs are stored in same storage space as row-organized tables
- This results in more query processing occurring in the column-organized runtime
  - Better performance!

Allows for storing large string or binary values within the column-organized table directly. No more need for side row tables.

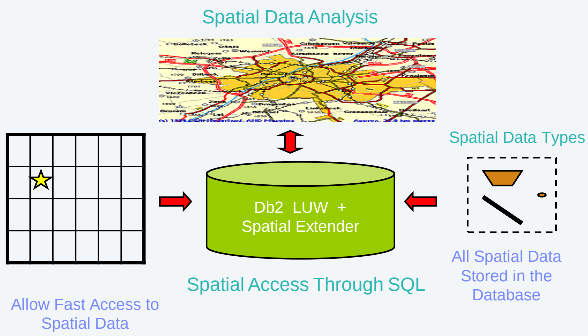
LOB Storage itself is shared with row engine. Column-organized data page stores the LOB meta-data and inlined data but large values stored in the same tablespace as row-organized LOBs.

Column-organized though inlining can be up to approximately the page size regardless of other columns in table.

Spatial Analytics

- Spatial support for analytic applications using BLU technology
  - Supports both row and column-organized tables
- Similar functionality as existing Db2 Spatial Extenders
- Some external and architectural differences
- Tech preview in Db2 11.5

Db2 Spatial Extender Support Review (1|2)



- On-Premise (single-node, MPP, pureScale)
- Db2 Warehouse on Cloud (public, dedicated, local)
- IIAS

- Implements
  - **Open Geospatial Consortium (OGC) Simple Features for SQL using Types and Functions**
  - **ISO SQL/MM part 3: spatial**
  - **Geography Markup Language (GML) for geometries**
  - **Well-known Text/Binary (WKT/WKB) for geometries**
  - **Shapefile import/export (de-facto)**
  - **Well-known Text (WKT) for coordinate systems (>5000 predefined)**

```
SELECT a.road_id, a.time, i.id, db2gse.ST_Distance(a.loc, i.loc,'METER') as distance
FROM accidents a, intersections i
WHERE db2gse.ST_Distance(a.loc,i.loc,'METER') < 10000 AND a.weather = 'RAIN'

SELECT a.name, a.type
FROM highways a, floodzones b
WHERE db2gse.ST_Intersects(a.location,b.location) = 1 AND b.last_flood > 1980
```

Db2 Spatial Extender Support Review (2|2)

- Schema DB2GSE with ~350 functions/ methods
- Spatial Metadata
  - DB2GSE catalog using triggers, constraints
  - MPP: SRS replicated across nodes via materialized query tables (MQTs)
- Abstract Data Types
  - Hierarchical geometries
  - Structured Type (ST_Geometry)
  - Geometries stored as BLOBs
- Spatial indexes via **Index Extensions**
- Spatial Extender enabled in custom install path
- db2se command line executable

DB2® Spatial Extender provides the database with several resources to enable it for support of spatial operations.

These resources are:

Stored procedures. Spatial operations such as commands to import spatial data, invoke one of these stored procedures to perform the operation.

Spatial data types. You must assign a spatial data type to each table or view column that is to contain spatial data.

DB2 Spatial Extender's catalog. Certain operations depend on this catalog. For example, before you can access a spatial column from the visualization tools, the tool might require that the spatial column be registered in the catalog.

A spatial grid index. You can define grid indexes on spatial columns.

Spatial functions. You use these to work with spatial data in a number of ways such as determining relationships between geometries and to generate more spatial data.

Definitions of coordinate systems.

Default spatial reference systems.

Two schemas: DB2GSE and ST_INFORMTN_SCHEMA. DB2GSE contains the objects just listed: stored procedures, spatial data types, the DB2 Spatial Extender catalog, and so on. Views in the catalog are available also in ST_INFORMTN_SCHEMA to conform with the SQL/MM standard..

Spatial Analytics Tech Preview

- Dedicated data type to hold shapes
  - LOB-based datatype SYSIBM.ST_GEOMETRY allows storing large geometries
  - Dedicated subtypes for points, linestrings, polygons, etc
- Pre-loaded spatial catalog data with SQL procedures for customization
  - E.g. add custom coordinate systems
- SQL functions based on the SQL/MM and OGC standards:
  - Construct and maintain/modify shapes
  - Determine relations between shapes
  - Get properties
    - `SELECT ST_DISTANCE(ST_CENTROID(GEOM1), GEOM2) FROM TAB1, TAB2`
- Same feature/interfaces for column and row-organized tables
- Enabled via SYSINSTALLOBJECTS procedure

Default spatial reference system = 4326

Spatial Processing in Db2

	Spatial Extender	Spatial Analytics
Processing Method	In-Database	In-Database
Data Organization	Row-Store	Column-Store Row-Store
Index Type	Spatial Grid	N/A
Spatial Joins	Yes	Yes
Function Type	Planar (with few exceptions)	Planar (with few exceptions)
Support for custom Coordinate Systems	Yes	Yes
Support for Spatial Reference Systems	Yes, default = 0, undefined	Yes, default = 4326, WGS84
Maximum Shape Size (compressed)	4 MB	4 MB

Spatial Analytics – Data types and Catalogs

- Flat Spatial Types in SYSIBM schema
  - Simple Types (ST_Geometry, ST_Point, ST_Polygon, etc.)
    - With soft hierarchy
  - Metadata as prefix to shape data
  - Geometry type based on BLOB (4190000) with weak type rules

- Catalogs
  - Schema SYSGEO for tables and views
  - Coordinate system table with pre-populated content (>5000)
    - For reference only
  - Spatial reference systems table with some (< 10) predefined entries
    - Definitions kept independent of coordinate systems table

Spatial Analytics Catalog Views

Name (SYSGEO schema)	Purpose
ST_COORDINATE_SYSTEMS	Co-ordinate systems
ST_GEOMETRY_COLUMNS	Details about spatial columns Based on SYSCAT.COLUMNS and SYSCAT.COLOPTIONS
ST_SIZINGS	Supported spatial variables and their maximum length.
ST_SPATIAL_REFERENCE_SYSTEMS	Registered spatial reference systems
ST_UNITS_OF_MEASURE	Spatial units of measures

Spatial Analytics – Functions and Procedures

- Spatial functions (schema SYSPROC)
  - Compatibility with row-store function signatures (SQL/OGC standards)

```
SELECT DB2GSE.ST_DISTANCE(DB2GSE.ST_POINT(10,20),POINTS)) FROM GSETAB
SELECT SYSPROC.ST_DISTANCE(ST_POINT(10,20), POINTS)) FROM BLUTAB
```

- Spatial procedures (schema SYSPROC)
  - Register geometry columns
  - Manage spatial reference systems
  - Import/export shapes (implementation shared with Spatial Extender)
  - Future: Manage Units of measure

- Note: No separate command line interface (db2se)

Spatial Analytics Stored Procedures

Name (SYSPROC schema)	Purpose
ST_ALTER_SRS	Update spatial reference system definition in SYSGEO.ST_SPATIAL_REFERENCE_SYSTEMS
ST_CREATE_SRS	Create a spatial reference system
ST_DROP_SRS	Drop a spatial reference system
ST_REGISTER_SPATIAL_COLUMN	Register a spatial column and associate a spatial reference system (SRS) with it.
ST_UNREGISTER_SPATIAL_COLUMN	Remove the registration of a spatial column

Spatial Functions Example (1|2)

```
CREATE TABLE TEST.POINTS (ID INTEGER NOT NULL PRIMARY KEY, POINT ST_POINT) ORGANIZE BY COLUMN
CREATE TABLE TEST.POLYS (ID INTEGER NOT NULL PRIMARY KEY, POLY ST_POLYGON) ORGANIZE BY COLUMN
INSERT INTO TEST.POINTS VALUES (1, ST_POINT(8, 9, 4326))
(…)
INSERT INTO TEST.POLYS VALUES
(1, ST_POLYGON('POLYGON ((10.15 20.01, 11.23 19.45, 11.01 20.12, 10.15 20.01))',4326))

SELECT ID, ST_AREA(POLY) AS DEFAULT_AREA, ST_AREA(POLY, 'FATHOM') AS UNITS_AREA
   FROM TEST.POLYS

ID          DEFAULT_AREA                           UNITS_AREA
----------- ------------------------ ------------------------
          1   +3.00200000000000E-001   +1.03893618288258E+009

1 record(s) selected.
```

Spatial Functions Example (2|2)

```
SELECT A.ID, B.ID, ST_DISTANCE(A.POLY, B.POINT) FROM TEST.POLYS A, TEST.POINTS B

ID          ID          3
----------- ----------- ------------------------
1           1    +1.09377968531144E+001
1           2    +4.61686040508049E+000
1           3    +0.00000000000000E+000

3 record(s) selected.
```

```
SELECT A.ID, B.ID, ST_CONTAINS(A.POLY, B.POINT) AS CONTAINS,
  SUBSTR(ST_ASTEXT(A.POLY),1,50) AS POLY, SUBSTR(ST_ASTEXT(B.POINT),1,50) AS POINT
  FROM TEST.POLYS A, TEST.POINTS B

ID    ID    CONTAINS POLY                                               POINT
----- ----- -------- -------------------------------------------------- --------------------------------
1     1     0        POLYGON ((10.150000000 20.010000000, 11.230000000  POINT (8.000000000 9.000000000)
1     2     0        POLYGON ((10.150000000 20.010000000, 11.230000000  POINT (10.000000000 15.000000000)
1     3     1        POLYGON ((10.150000000 20.010000000, 11.230000000  POINT (10.200000000 20.010000000)

3 record(s) selected.
```

System-Maintained Temporal Tables for BLU

- Recap Db2 11.1.3.3 support:
  - Create column-organized **user-maintained:**
    - System-period temporal tables (STT)
    - Application-period temporal tables (ATT)
    - Bi-temporal tables (BTT)
  - Execute **read-only** temporal queries on column-organized temporal tables.
  - UPDATE and DELETE FOR PORTION OF BUSINESS TIME clause and row-splitting feature is not supported.
  - User must maintain history themselves
- Db2 11.5++ will add support for **system-maintained** temporal tables
- Equivalent support to row-organized tables

36

Db1 11.1.3.3 Support:

Creating User Maintained Temporal Tables organized by Column

```
CREATE TABLE policy_info( policy_id    CHAR(4) NOT NULL,coverage     INT NOT NULL, sys_start
TIMESTAMP(12) NOT NULL , sys_end      TIMESTAMP(12) NOT NULL, PERIOD SYSTEM_TIME (sys_start,
sys_end) maintained by user) organize by column;

CREATE TABLE hist_policy_info LIKE policy_info organize by column;

ALTER TABLE policy_info ADD USER VERSIONING USE HISTORY TABLE hist_policy_info;
```

Note: the absence of the column options GENERATED ALWAYS AS ROW_BEGIN and GENERATED ALWAYS AS ROW_END for the system_time period columns and absence of transaction start-ID column.

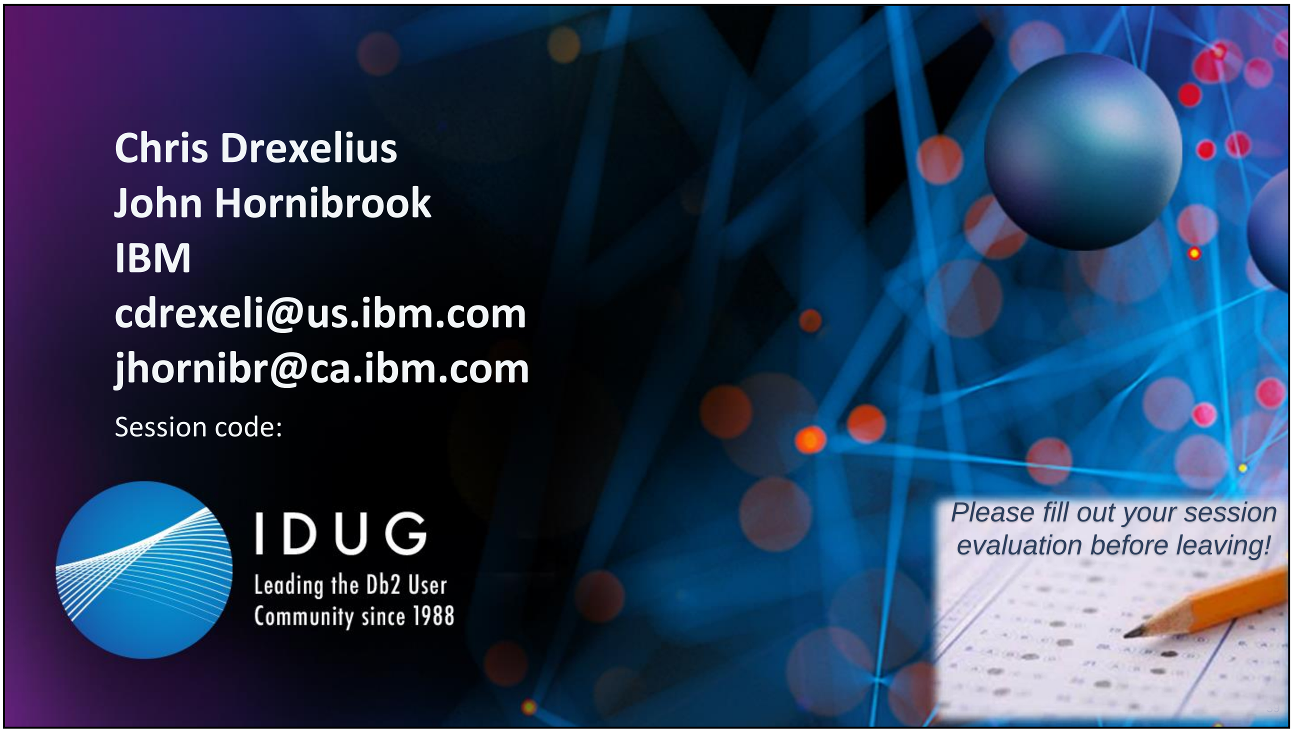
Note: any modification to POLICY_INFO table will not automatically trigger appropriate modification to HIST_POLICY_INFO since these modifications are done by the user or application, not done automatically by the DB2 system. That is the reason to include USER versioning while linking the HIST_POLICY_INFO.

Trigger and Referential Integrity Support

- Not supported in 11.5 GA, but on roadmap for later mod paks
- Architecturally, they require common infrastructure
  - Ability to flow data from insert, update, delete operations to query operations
    - Join, filtering
  - Requires precise sequencing of operations
- Goal is to provide similar performance to row-organized tables
  - Other than existing differences for columnar “decomposed” updates
    - UPDATE=DELETE+INSERT, for each column
- Triggers will likely be available before RI

BLU Index Support

- Support was added in Db2 11.1.3.3, with some restrictions
- Some of these will be gradually lifted over the lifespan of Db2 11.5 (in priority order)
  1. Searched UPDATE or DELETE, for > 1 row
  2. INCLUDE columns
  3. CREATE INDEX and REORG INDEX .. REBUILD mode will allow concurrent readers but not concurrent writers.
  4. Index advisor
  5. DGTTs
  6. Jump scan
  7. Expression-based indexes
  8. EXCLUDE NULL KEY option



Chris is a Senior Software Engineer who is product owner for storage and compression for Db2 with BLU Acceleration. This technology is part of Db2 for Linux, UNIX and Windows, Db2 Warehouse, Db2 on Cloud, IBM Integrated Analytics System (IIAS) and Db2 Big SQL.

John is a Senior Technical Staff Member responsible for relational database query optimization on IBM's distributed platforms. This technology is part of Db2 for Linux, UNIX and Windows, Db2 Warehouse, Db2 on Cloud, IBM Integrated Analytics System (IIAS) and Db2 Big SQL. John also works closely with customers to help them maximize their benefits from IBM's relational DB technology products.