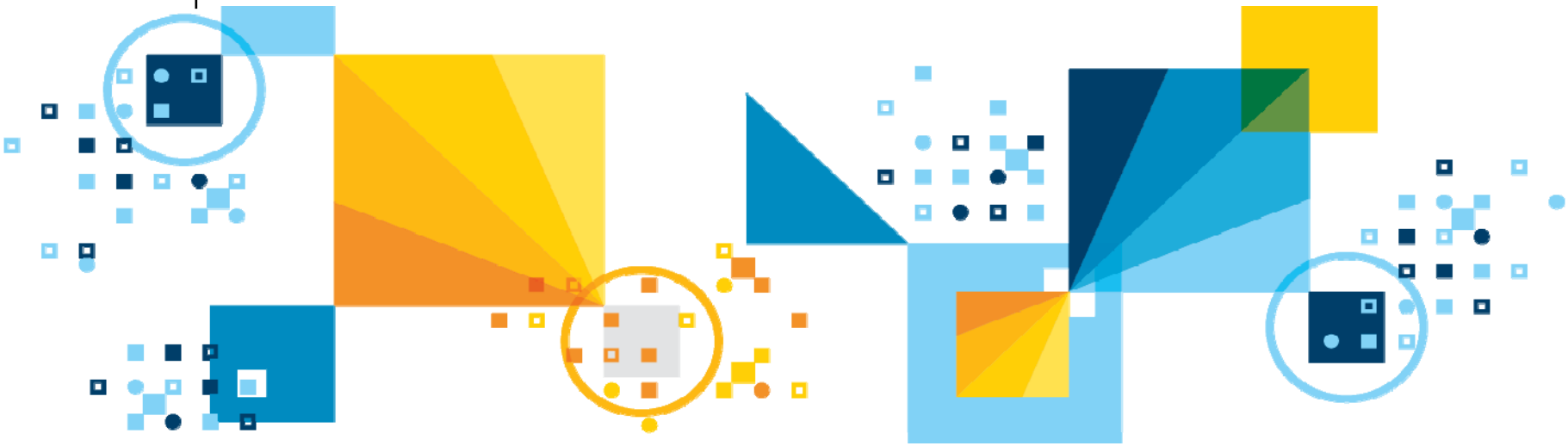
DB2 - Storage Optimizations

Module ID	10107
Length	35 minutes + 1 hour hands on lab



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

Disclaimer

Module Information

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

- After completing this module, you should be able to:
  - Describe the storage optimizations available in DB2
  - Explain the benefits of compression
  - Perform the following tasks:
    - Compress database tables
    - Determine the amount of compression is possible for a table

Topics

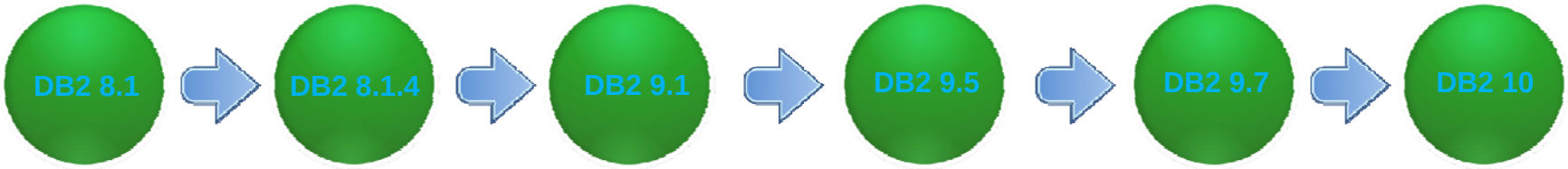
- Compression Overview
- Storage Optimization in DB2
  - Value Compression
  - Row Compression
  - Compression in BLU
  - Temporary Table Compression
  - XML Compression
  - Index Compression
  - Backup Compression
- Compression Benefits
- Client Success

Compression Overview

- Requires less storage space and memory
  - Usually the most expensive component in a database solution
- Helps manage storage growth
  - Saves on power and cooling
- Helps improve database performance
  - Fewer I/O operations required
- Reduce administration costs and increase productivity of the IT team
- Facilitates lower cost of ownership

Storage Optimization in DB2

- Provides storage compression services to optimize the performance and footprint of your data
- Basic compression features included in several editions
- Advanced compression features including Adaptive Compression and classic row compression are included in the following versions of DB2:
  - DB2 Advanced Workgroup Sever Edition
  - DB2 Advanced Enterprise Server Edition
  - DB2 Developer Edition



- Value Compression

- Backup Compression

- Row Compression*

- Automatic Dictionary Creation (ADC)*

- XML compression*
- Temporary table compression*
- Index compression*
- LOB inlining

- Adaptive compression*

** Included in DB2 Advanced Editions (and Developer Edition) as of DB2 10.5*

Value Compression

- Removes duplicate entries for a value, only storing one copy
- Can offer savings if your table:
  - has many rows with columns that contain the same value, such as a city name
  - has columns that contain the default value for the data type of the column
- Can be used in combination of either adaptive row compression or classic row compression
- Enabled with VALUE COMPRESSION option in CREATE TABLE
- Compress system default column values with clause COMPRESS SYSTEM DEFAULT

```
CREATE TABLE EMPLOYEE_SALARY
        (DEPTNO CHAR(3) NOT NULL,
         EMPNO CHAR(6) NOT NULL,
         SALARY DECIMAL(9,2) NOT NULL WITH DEFAULT COMPRESS SYSTEM DEFAULT)
        VALUE COMPRESSION;
```

Row Compression

- Also known as **deep compression**
- Uses a dictionary-based compression algorithm to replace recurring strings with shorter symbols within rows
- Continuous enhancement since it was introduced in DB2 9.1

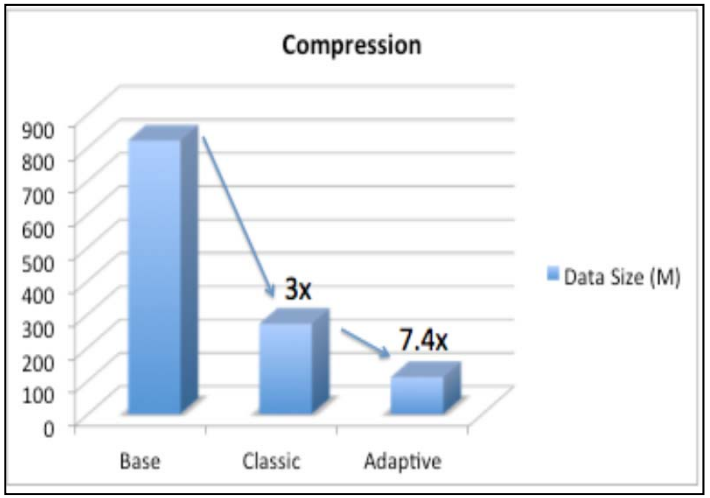
- Row Compression*

- Automatic Dictionary Creation (ADC)*

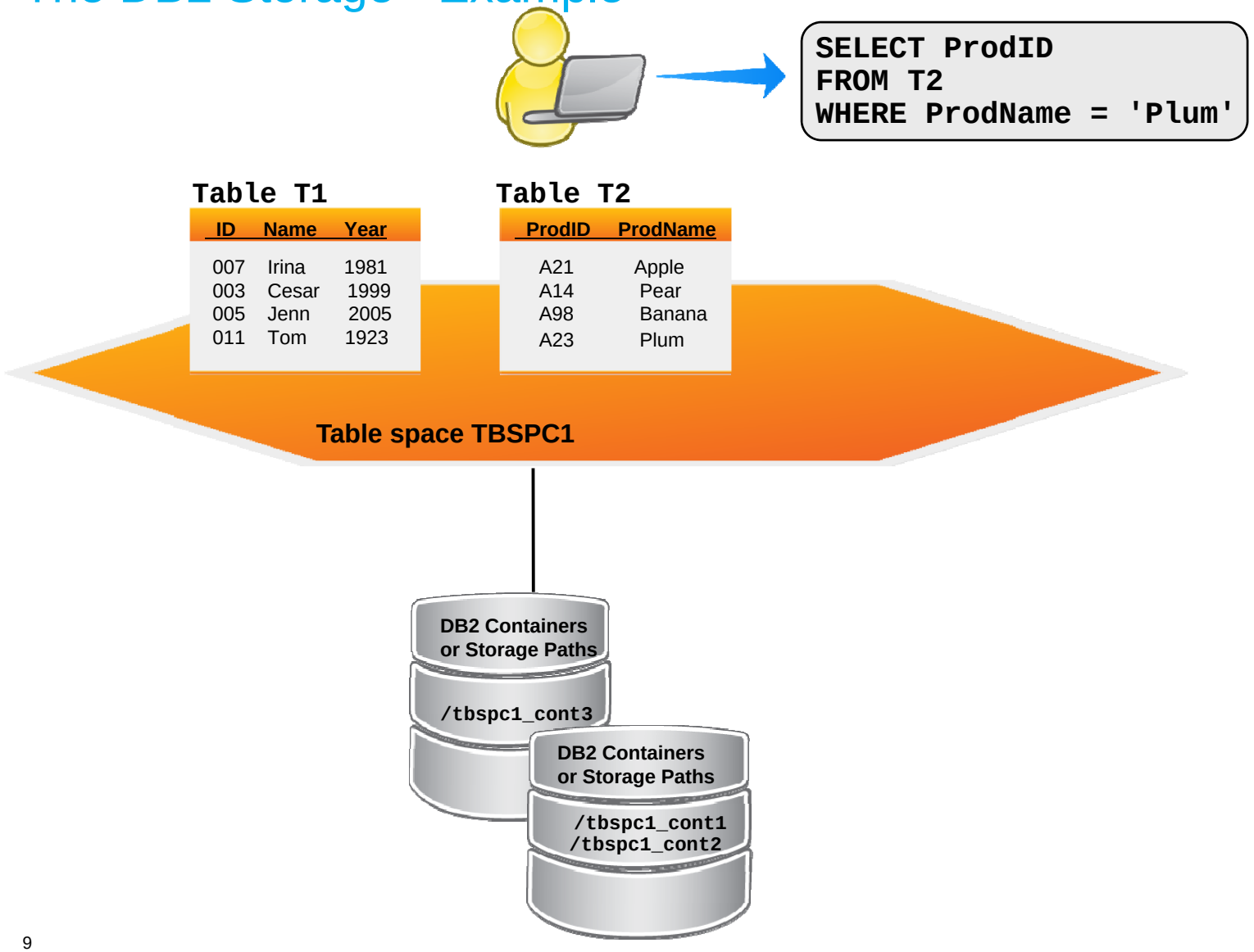
- XML compression*
- Temporary table compression*
- Index compression*
- LOB inlining

- Adaptive compression*

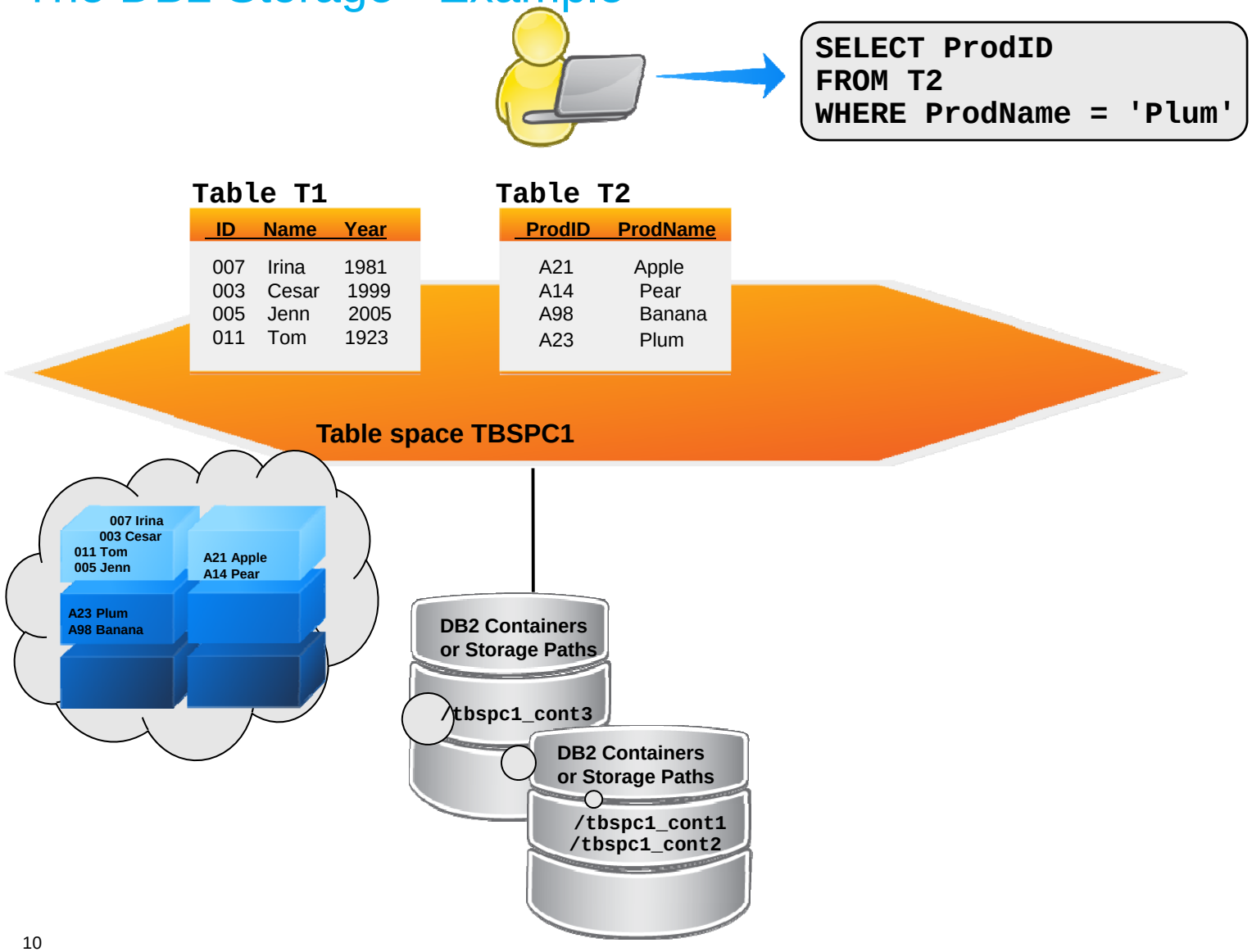
- Two types available:
  - **Classic** (static) row compression
  - **Adaptive** row compression
    - An enhancement to classic row compression to provide extra storage savings



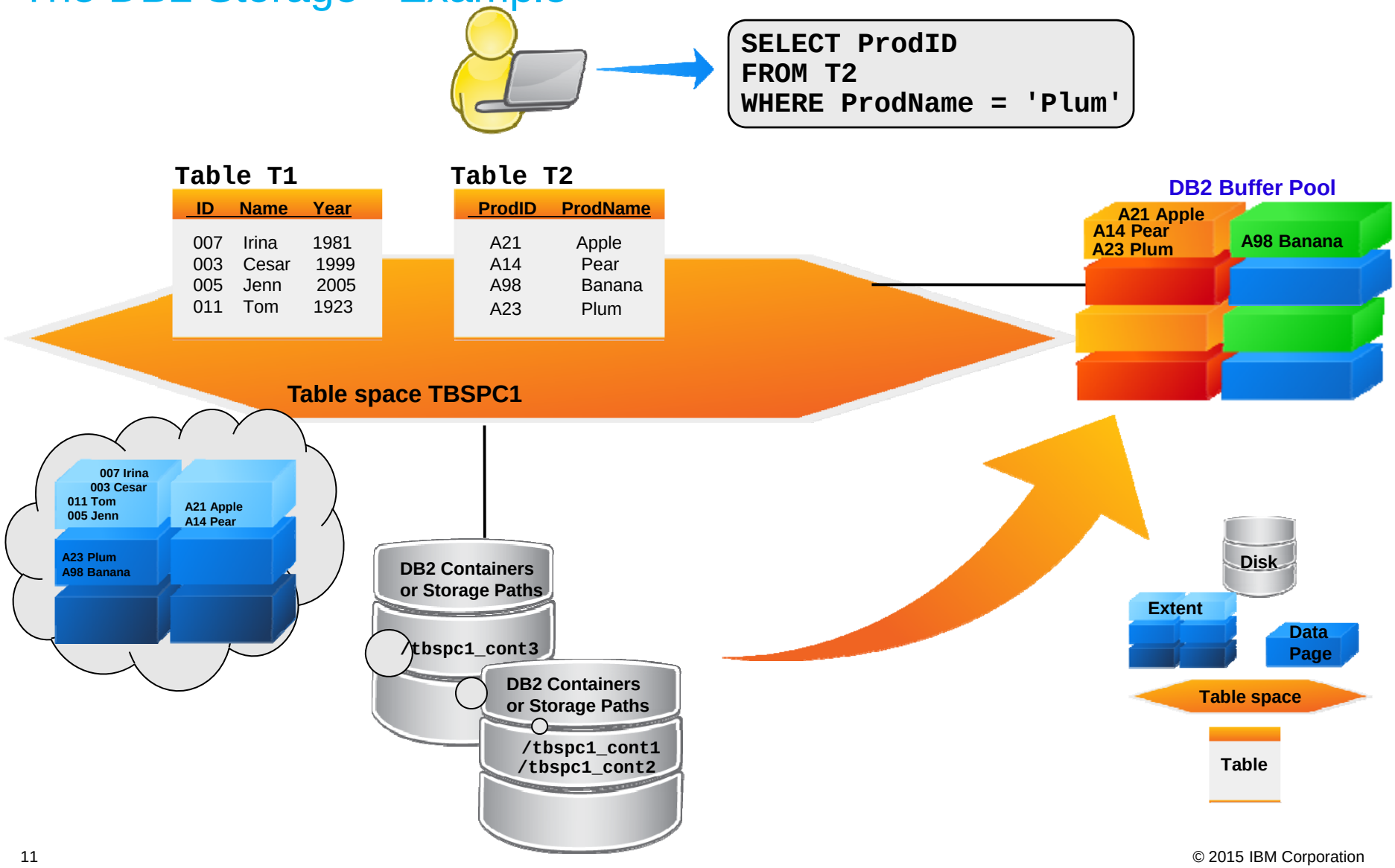
Understanding the Basics
The DB2 Storage - Example



Understanding the Basics
The DB2 Storage - Example



Understanding the Basics
The DB2 Storage - Example

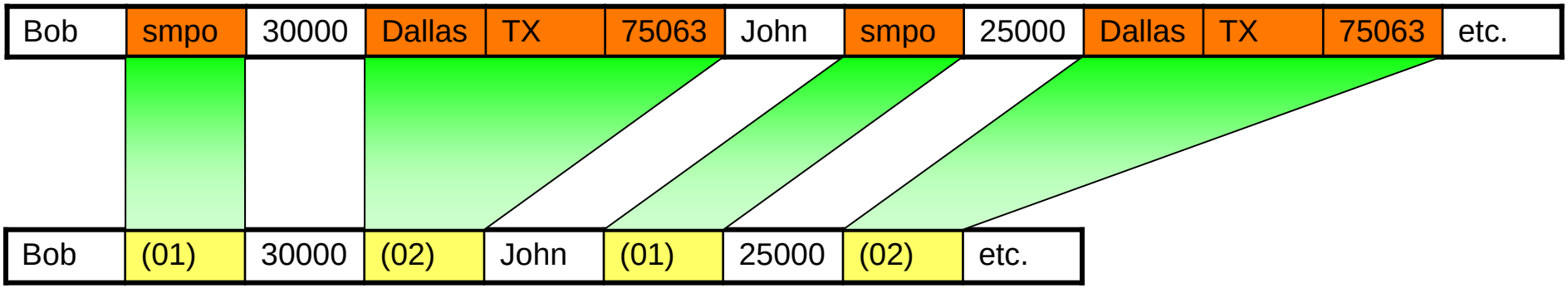


Row Compression – Classic

- Also referred to as static row compression
- Uses a table-level compression dictionary (1 dictionary per table) to compress data by row, across multiple columns
- Dictionary is used to map repeated byte patterns to smaller symbols. These smaller symbols replace long patterns in table rows.
- After dictionary is created, data is compressed as it is inserted/updated in the table.
  - DB2 automatically creates the dictionary when enough the table has enough data for sampling

Name	Dept	Salary	City	ST	ZIP
Bob	smpo	30000	Dallas	TX	75063
John	smpo	25000	Dallas	TX	75063

Dictionary	
(01)	smpo
(02)	Dallas, TX, 75063

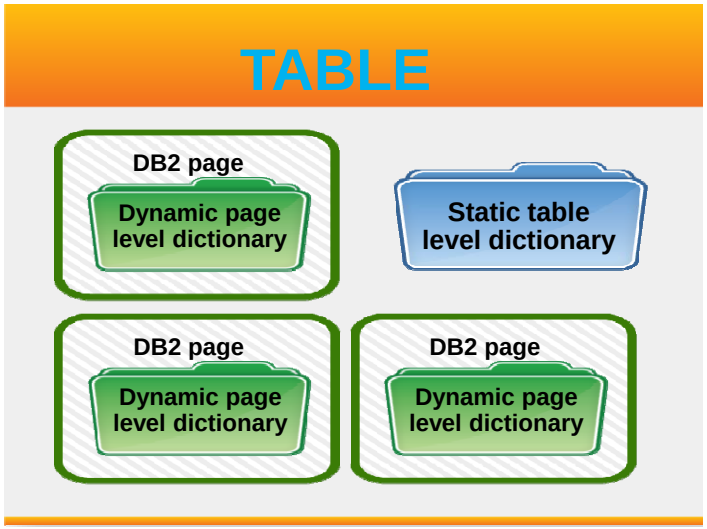


Why Choose Adaptive Row Compression

- Classic row compression works very well
  - Very fast and robust
  - Not sensitive to data clustering and ordering
- But it has limitations
  - Dictionary requires classic table reorganization to refresh
  - Not sensitive to data clustering and ordering
  - Dictionary capacity limits compression ratios for some large tables
- DB2 10 introduces Adaptive Compression
  - An enhancement to classic row compression
  - Industry-leading compression technology for row stores
- Reduces TCO, providing extra compression savings:
  - Classic row compression: Typically saves ~40%-75%
  - Adaptive row compression: Typically saves ~75%-85%
  - Adaptive typically saves 30% over classic row compression

Row Compression – Adaptive

- Adaptive Compression is an enhancement to the Classic Row Compression
  - Compress rows by using a combination of two types of algorithms
    - Globally recurring byte sequences
    - Locally recurring byte sequences



- Benefits
  - Page level dictionaries adapt to data skew over a period of time
  - No REORGs required to maintain high compression ratio
  - Less disk space for data and logs
  - Reduced I/O. Fewer pages to process

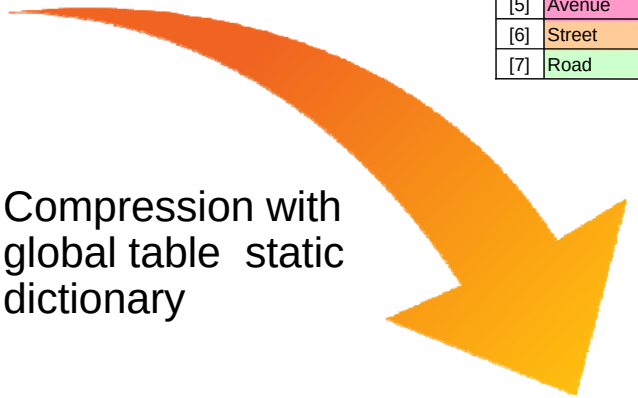
- **Better compression ratios than Classic Row Compression**
- **Over time reduces need of a REORG as page-level dictionaries adapt to data skew over time (i.e. maintain compression rates)**
- **Default row compression method in DB2 10**

How Does Adaptive Compression Work?

- Step 1: Compression with static table level dictionary

Christine	Haas	(408) 463-1234	555 Bailey Avenue	San Jose	California	95141
John	Thompson	(408) 463-5678	555 Bailey Avenue	San Jose	California	95141
Jose	Fernandez	(408) 463-1357	555 Bailey Avenue	San Jose	California	95141
Margaret	Miller	(408) 463-2468	555 Bailey Avenue	San Jose	California	95141
Bruce	Kwan	(408) 956-9876	4400 North 1st Street	San Jose	California	95134
James	Geyer	(408) 956-5432	4400 North 1st Street	San Jose	California	95134
Linda	Hernandez	(408) 956-9753	4400 North 1st Street	San Jose	California	95134
Theodore	Mills	(408) 927-8642	650 Harry Road	San Jose	California	95134
Susan	Stern	(408) 927-9630	650 Harry Road	San Jose	California	95134
James	Polaski	(415) 545-1423	425 Market Street	San Francisco	California	94105
John	Miller	(415) 545-5867	425 Market Street	San Francisco	California	94105
James	Walker	(415) 545-4132	425 Market Street	San Francisco	California	94105
Elizabeth	Brown	(415) 545-8576	425 Market Street	San Francisco	California	94105
Sarah	Johnson	(415) 545-1928	425 Market Street	San Francisco	California	94105

[1]	California 9
[2]	San
[3]	Jose
[4]	Francisco
[5]	Avenue
[6]	Street
[7]	Road

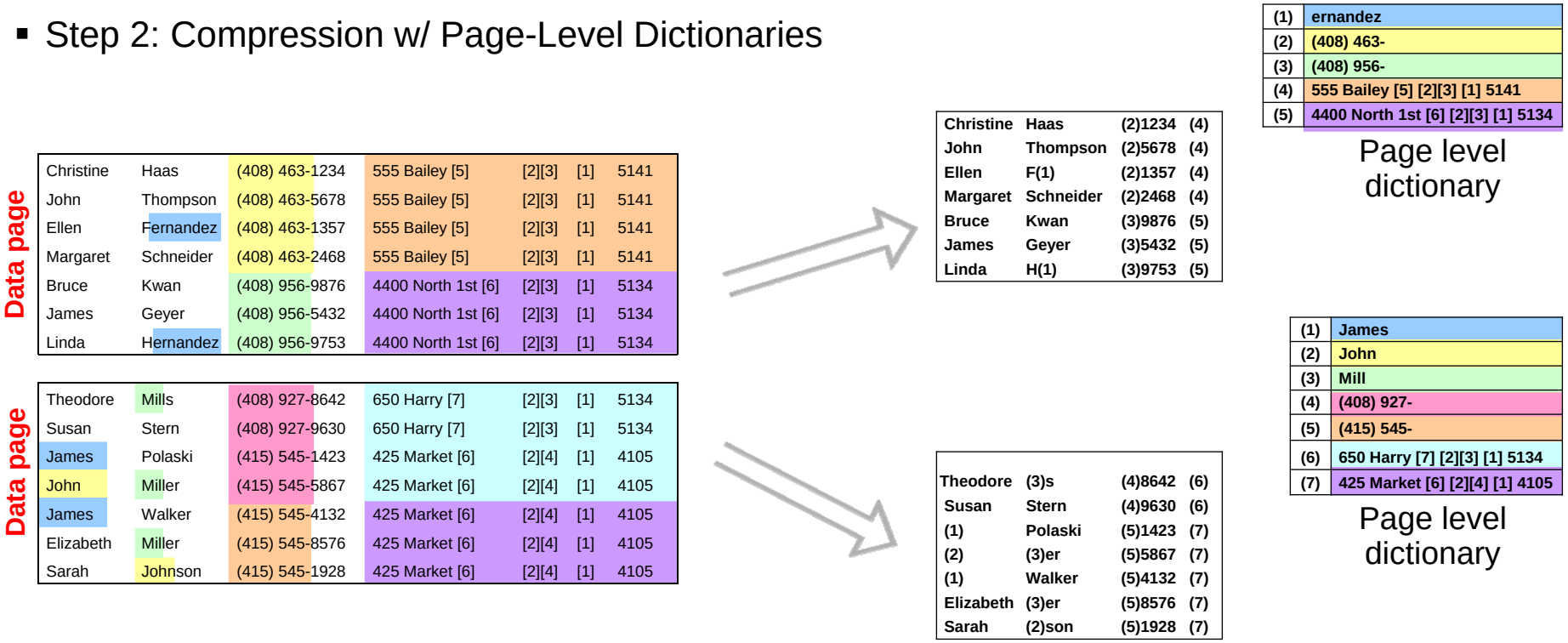


Christine	Haas	(408) 463-1234	555 Bailey [5]	[2][3]	[1]	95141
John	Thompson	(408) 463-5678	555 Bailey [5]	[2][3]	[1]	95141
[3]	Fernandez	(408) 463-1357	555 Bailey [5]	[2][3]	[1]	95141
Margaret	Schneider	(408) 463-2468	555 Bailey [5]	[2][3]	[1]	95141
Bruce	Kwan	(408) 956-9876	4400 North 1st [6]	[2][3]	[1]	95134
James	Geyer	(408) 956-5432	4400 North 1st [6]	[2][3]	[1]	95134
Linda	Hernandez	(408) 956-9753	4400 North 1st [6]	[2][3]	[1]	95134
Theodore	Mills	(408) 927-8642	650 Harry [7]	[2][3]	[1]	95134
Susan	Stern	(408) 927-9630	650 Harry [7]	[2][3]	[1]	95134
James	Polaski	(415) 545-1423	425 Market [6]	[2][4]	[1]	94105
John	Miller	(415) 545-5867	425 Market [6]	[2][4]	[1]	94105
James	Walker	(415) 545-4132	425 Market [6]	[2][4]	[1]	94105
Elizabeth	Miller	(415) 545-8576	425 Market [6]	[2][4]	[1]	94105
Sarah	Johnson	(415) 545-1928	425 Market [6]	[2][4]	[1]	94105

  - Table level compression symbol dictionary containing globally recurring patterns
  - Table-level dictionary can only be rebuilt during Classic Table REORG
    - Involves re-compressing all data

How Does Adaptive Compression Work?

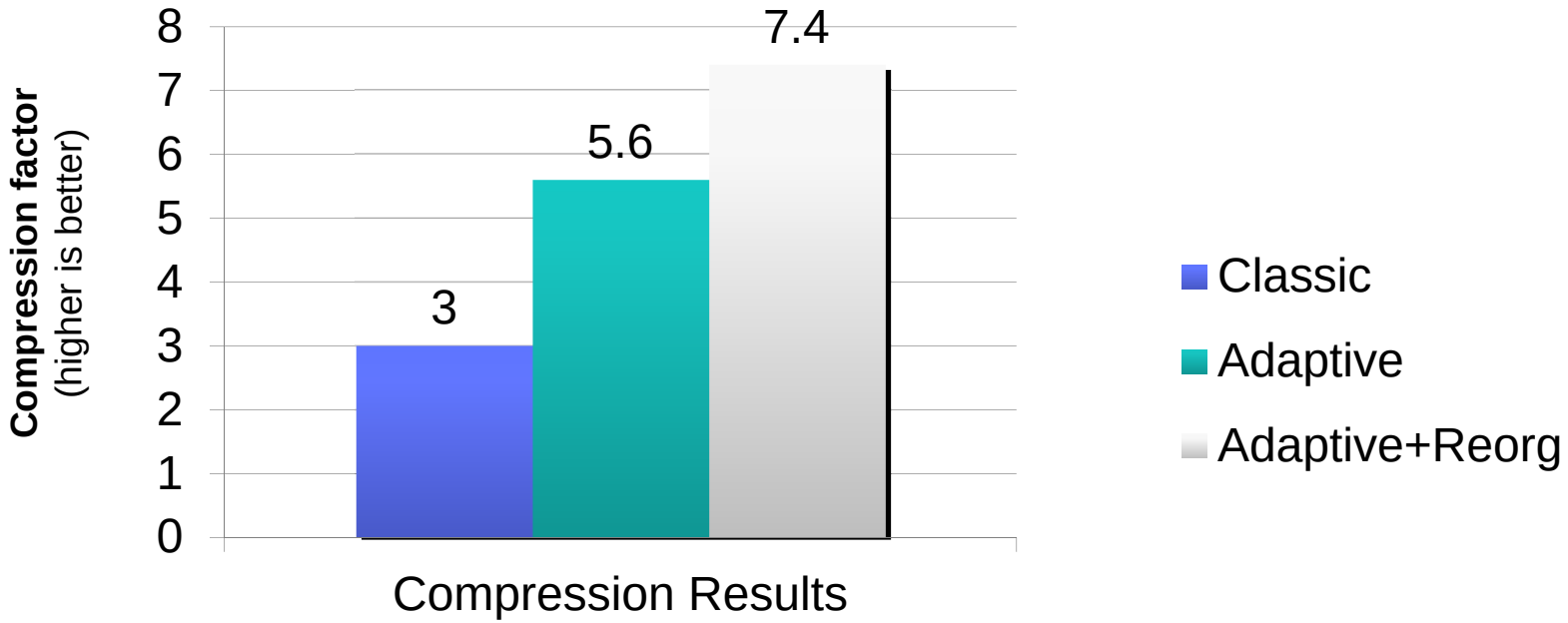
- Step 2: Compression w/ Page-Level Dictionaries



- Page-level compression dictionaries contain locally frequent patterns
- Page-level compression dictionary building & rebuilding is fully automatic
- Algorithm optimized for compressing data already compressed by table-level dictionary
- Page-level compression dictionaries are stored as special records in each page

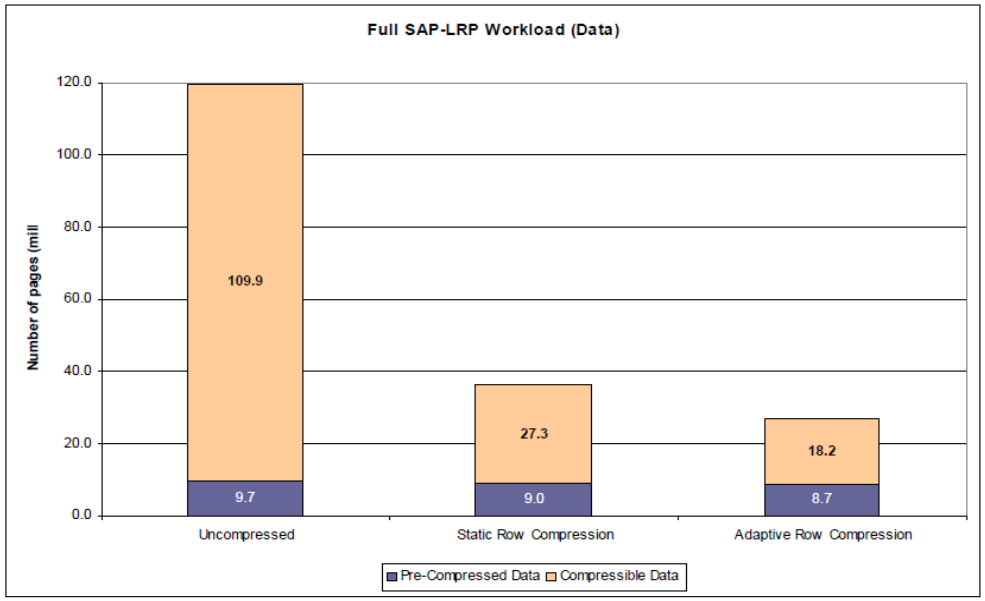
Data Warehouse Compression Results
230GB raw size - Most of the data in a single table

- Graph – Storage Savings
- Increase in savings by Adaptive Compression
  - 3x Compression with Static Compression using reorg
  - 5.6x Compression with Automatic dictionary creation and Adaptive Compression
  - 7.4x Compression with Adaptive Compression and full reorg



Additional Storage Savings

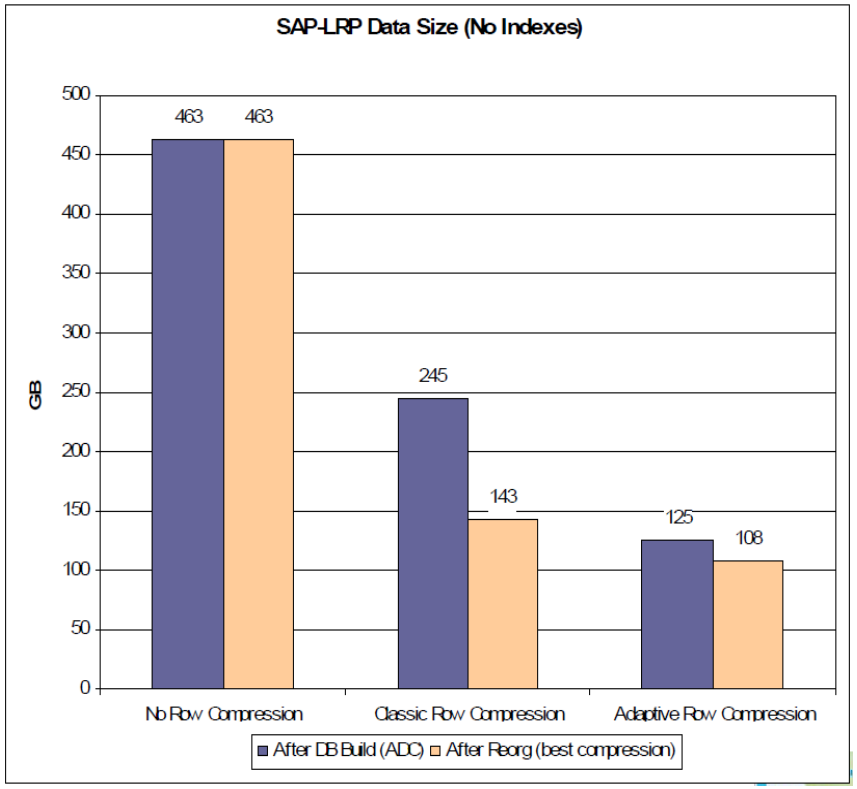
- SAP-LRP workload, ~2TB data size before compression



- Considering the 1425 largest tables (~99.5% of the total data), after Classic Table Reorg
- 33% savings for normal tables, compared to Classic Row Compression
- Almost no storage savings for tables with pre-compressed data (SAP cluster tables, certain tables with LOB fields)

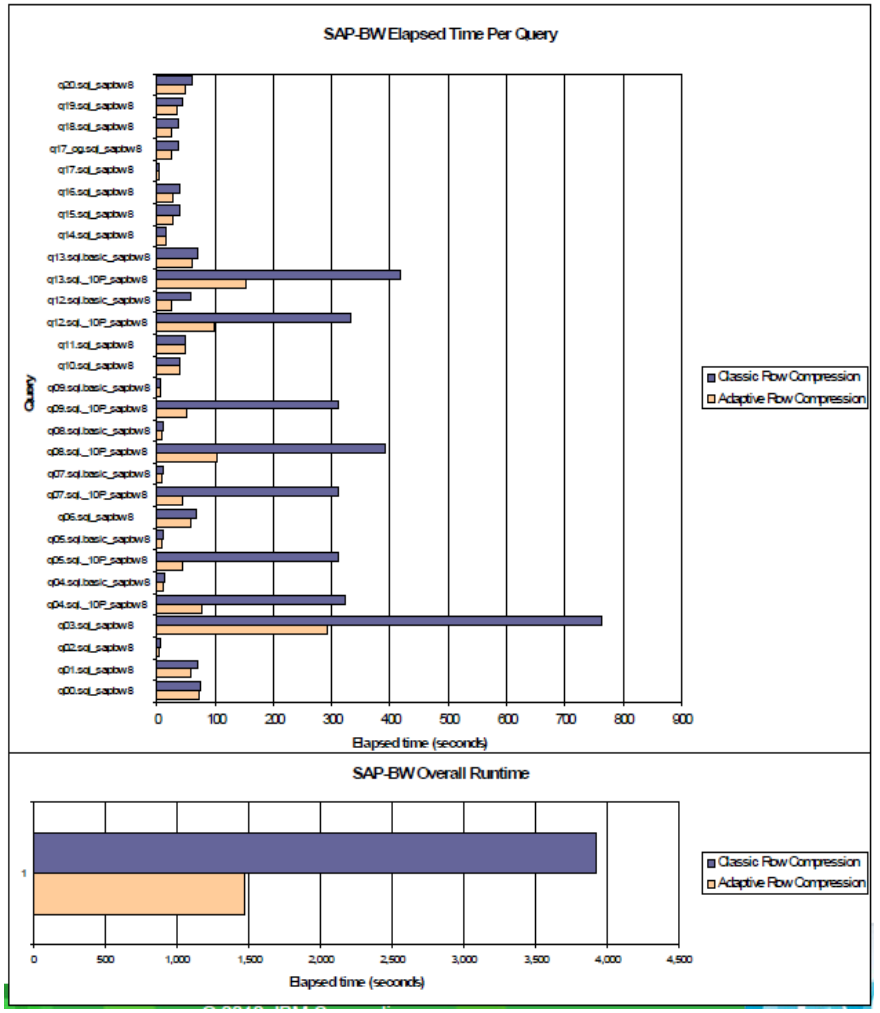
Reorg Avoidance – ADC vs. Classic Table Reorg

- SAP-LRP system with ~450GB of data in uncompressed form
- ~24,000 tables, 99.7% of the data in 1,500 tables
- Comparison between database build (ADC) and full Classic Table Reorg
- Classic Row Compression with ADC yields 71% larger data size than after Reorg
- Adaptive Compression with ADC is only 16% larger than after Reorg
- Adaptive Compression with ADC is 14% smaller than Classic Row Compression after Reorg (i.e. best possible compression)



Query Performance – SAP-BW Benchmark (1TB)

- Additional CPU consumption for compression & expansion operations
  - Row compression design paradigm: Compression under logical I/O
  - Rule of thumb: Compression is beneficial in I/O-bound situations
- Performance measurements done from two angles:
  - 1. Focus on additional CPU cost (to minimize overhead)
  - 2. Elapsed time studies in benchmark scenarios
- Most long-running queries experience dramatic speed-up
- Overall runtime reduced by 62%
- Median query speedup of 43%



Row Compression – Enablement & Tools

- How to enable row compression?
  - Must have DB2 Storage Optimization Feature
  - To enable classic row compression

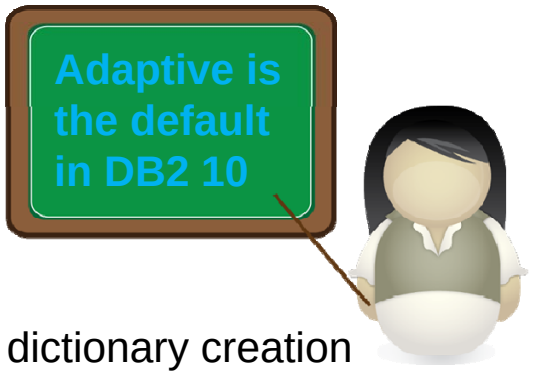
```
CREATE TABLE / ALTER TABLE … COMPRESS YES STATIC
```

  - To enable adaptive row compression

```
CREATE TABLE / ALTER TABLE … COMPRESS YES
```

  - To disable compression

```
CREATE TABLE / ALTER TABLE … COMPRESS NO
```



- Data is compressed after the table dictionary is created.
  - INSERT/UPDATE/LOAD/IMPORT can trigger the automatic dictionary creation
  - Classic REORG with RESETDICTIONARY option will always generate a new dictionary and compress all table data

Row Compression - Example Scenarios

1) Compressing data for new table

```
CREATE TABLE Sales (<columns definition>) COMPRESS YES
```

Automatic Dictionary Creation (ADC) will kick off and create compression dictionary with the load command. Once the dictionary is built, new data put into the table is compressed:

```
LOAD FROM file OF DEL REPLACE INTO NewSale
```

2) Compressing data in existing tables

```
ALTER TABLE Sales COMPRESS YES
```

Existing data is still un-compressed. Explicitly compress existing data via REORG:

```
REORG TABLE Sales
```

3) Recreating the dictionary to optimize compression

(Classic Row Compression) Data has changed a lot so current dictionary is not so effective anymore. Use REORG to recreate dictionary and re-compress data:

```
REORG TABLE Sales RESETDICTIONARY
```

Adaptive Compression greatly reduces the need for REORGs to maintain the compression ratio.

4) Uncompressing your data

Disable compression:

```
ALTER TABLE Sales COMPRESS NO
```

Uncompress data:

```
REORG TABLE Sales
```

Row Compression – Enablement & Tools

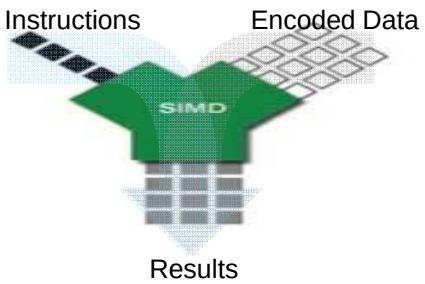
- Estimating storage savings
  - ADMIN_GET_TAB_COMPRESS_INFO
  - ADMIN_GET_TAB_DICTIONARY_INFO
  - ADMIN_GET_TAB_COMPRESS_INFO_V97 ← **Deprecated in DB2 10!**

```
SELECT SUBSTR(TABNAME,1 ,10) tabname, OBJECT_TYPE, ROWCOMPMODE,
PCTPAGESSAVED_CURRENT current, PCTPAGESSAVED_STATIC with_static,
PCTPAGESSAVED_ADAPTIVE with_adaptive
FROM TABLE(SYSPROC.ADMIN_GET_TAB_COMPRESS_INFO('DB2INST1','CUSTOMERS')) AS T;
```

```
TABNAME    OBJECT_TYPE  ROWCOMPMODE  CURRENT WITH_STATIC WITH_ADAPTIVE
---------- ------------ ------------ ------- ----------- -------------
CUSTOMERS  DATA                    S       60          68            81
CUSTOMERS  XML                     S       58          62            62
```

BLU uses Multiple Compression Techniques

- Approximate Huffman-Encoding ("frequency-based compression"), prefix compression, and offset compression
- For column-organized tables, COMPRESSION cannot be enabled/disabled
- Frequency-based compression: Most common values use fewest bits
  - Exploiting skew in data distribution improves compression ratio
  - Very effective since all values in a column have the same data type
  - Maps entire values to dictionary codes
- Actionable compression: Order-preserving encoding allows predicates to be evaluated on compressed data
- SIMD (Single Instruction Multiple Data) parallelism used for fast predicate evaluation on multiple compressed values
- Avoiding decompression during predicate evaluation provides significant query performance gains



Temporary Table Compression

- Enabled automatically with the DB2 Storage Optimization Feature
- Only classic row compression is used for temporary tables
- System temporary tables
  - DB2 Optimizer analyzes storage savings and performance impact, if it is worthwhile to compress, classic row compression is used automatically
- User temporary tables
  - Created global temporary tables (CGTTs) and declared global temporary tables (DGTTs) are always enabled for compression
- Use db2pd to see whether the optimizer used compression for system temporary tables

```
db2pd –db <database> -temptable
```

Index Compression

- Especially useful for large OLTP and data warehouse environments
- Index is compressed by default for compressed tables
- Can specify COMPRESS YES option in CREATE INDEX
- Different algorithms implemented by the DB2 engine for:
  - RID list compression
  - Prefix compression
  - Variable slot directory

```
CREATE INDEX <index_name> COMPRESS YES

ALTER INDEX <index_name> COMPRESS YES
REORG INDEXES ALL FOR TABLE <table_name>
```

Index Compression – Monitoring

- How much space could I save by compressing the indexes on table T1 ?

```
SELECT index_name, pages_saved_percent, compress_attr, index_compressed
FROM TABLE(SYSPROC.ADMIN_GET_INDEX_COMPRESS_INFO ('T', 'myschema', 'T1', NULL, NULL)) AS T
```

```
INDEX_NAME      PERCENT_PAGES_SAVED      COMPRESS_ATTR      IN
--------------  -----------------------  -----------------  --------------------
INDEX1          57                       N                  N
```

Index Compression – Monitoring

- How much space did I save by compressing the indexes on table T1 ?

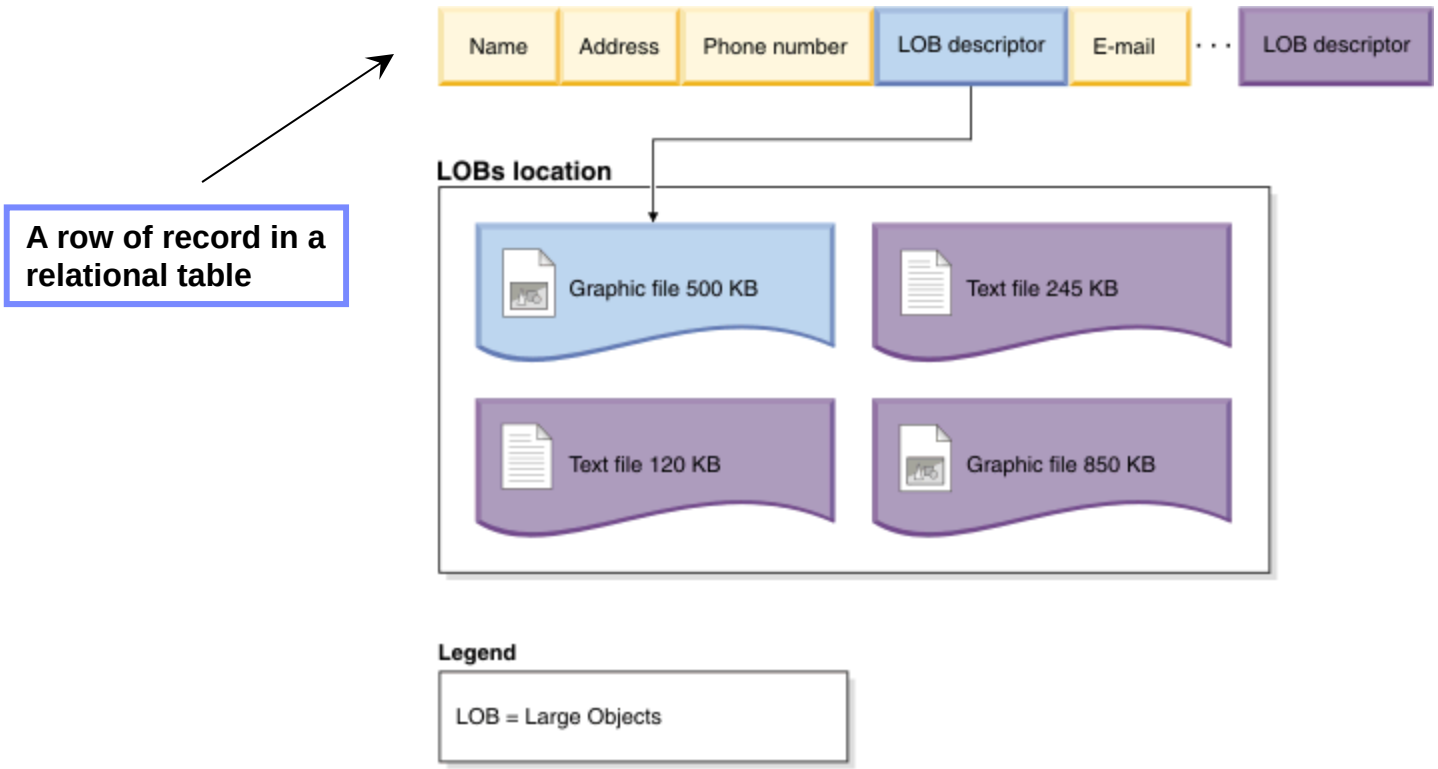
```
ALTER INDEX index1 COMPRESS YES
REORG INDEXES ON TABLE t1
RUNSTATS ON TABLE t1

SELECT index_name, pages_saved_percent, compress_attr, index_compressed
  FROM TABLE (SYSPROC.ADMIN_GET_INDEX_COMPRESS_INFO('T', 'myschema', 'T1', NULL, NULL)) AS T
```

```
INDEX_NAME          PERCENT_PAGES_SAVED   COMPRESS_ATTR          INDEX_COMPRESSED
------------------- --------------------- ---------------------  --------------------------
INDEX1              58                    Y                                 Y
```

actual savings

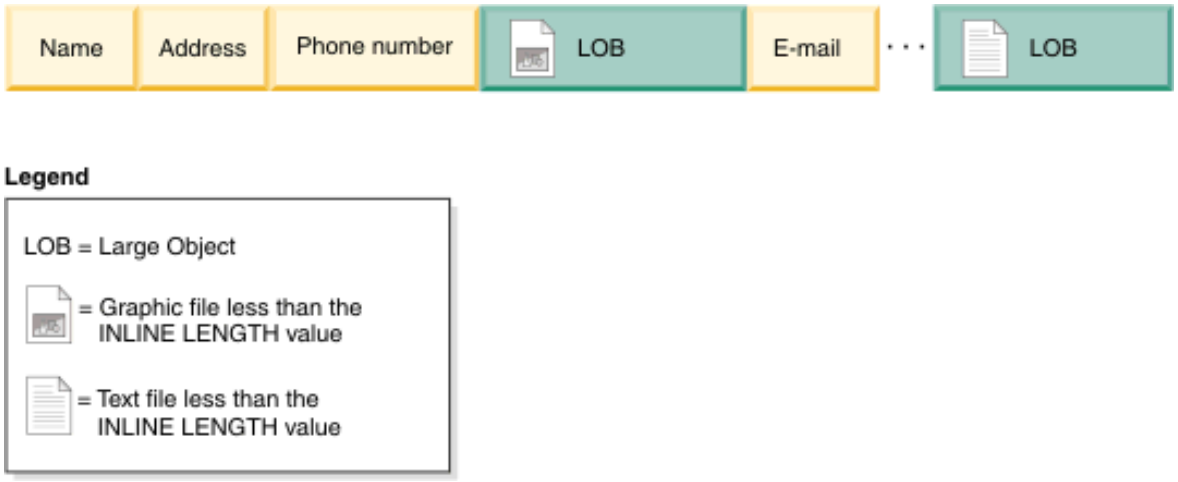
XML and LOB Inlining

- Default storage behavior
  - All XML documents stored in XML storage object, also referred to as XDA (XML Data Area)
  - All LOB (Large Objects) stored in LOB storage object
  - Descriptors in rows identify location of each object



XML and LOB Inlining

- Inlining
  - Allows small (<32KB for a 32KB page size) XML documents and LOBs to be stored in the base table row
  - Better performance and reduced storage needs




```
CREATE TABLE PROJECTS (PID INTEGER, PLAN XML INLINE LENGTH 300, STARTDATE DATE, …)

ALTER TABLE PROJECTS ALTER COLUMN PLAN SET INLINE LENGTH 1004
```

XML and LOB Compression

- XML (XDA) Compression
  - Available since DB2 9.7
  - Same dictionary approach used for Table Compression
  - Enablement is via the table COMPRESS YES option
  - Classic/'Offline' REORG table based
    - Use LONGLOBDATA option to compress XML data

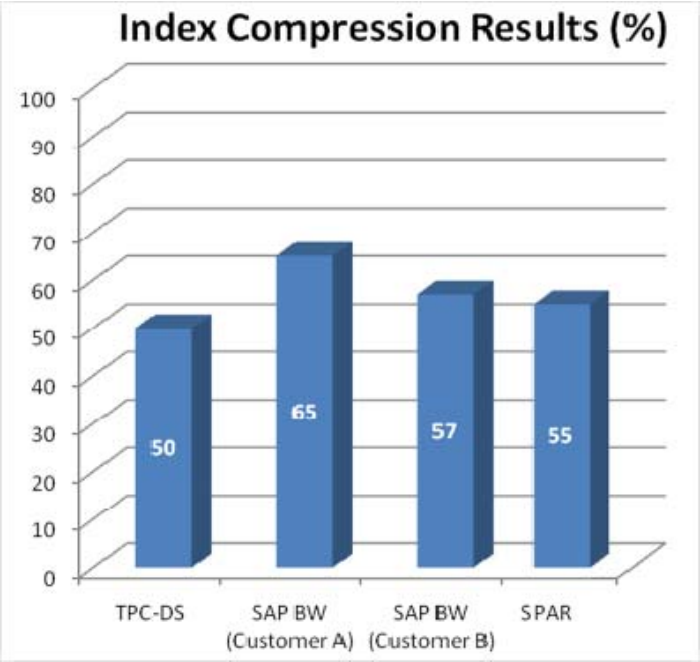
```
CREATE TABLE mytab1 COMPRESS YES
ALTER TABLE mytab1 COMPRESS YES
```

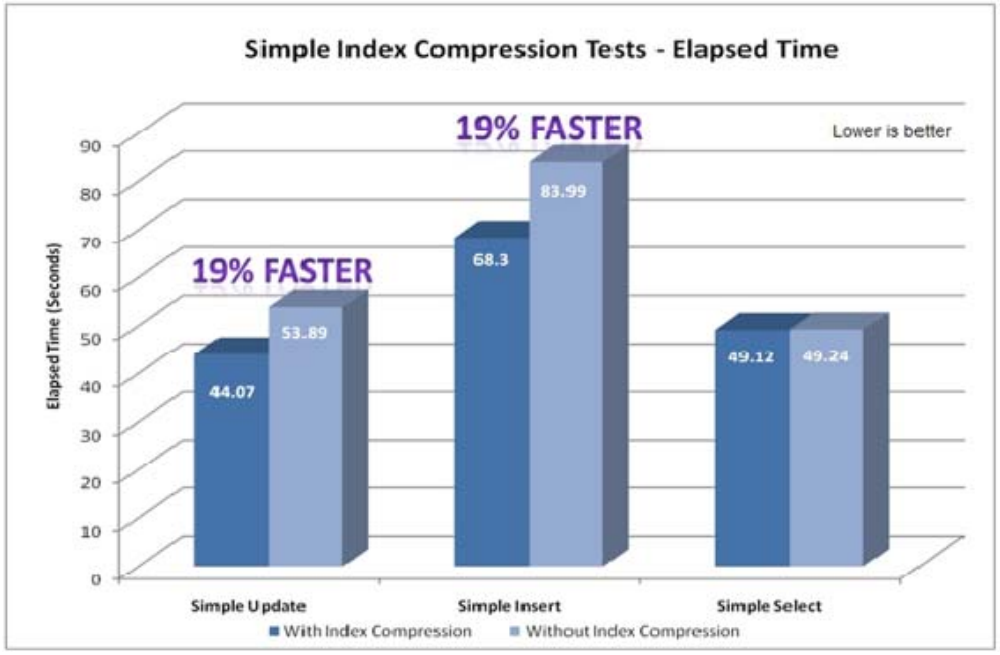
- Inlining
  - Inlined XML or LOB data can be compressed with both adaptive and classic row compression

New in DB2 10

Compression Benefits
Index Compression

- Fewer index levels
  - Fewer logical and physical I/Os for key search (insert, delete, select)
  - Better buffer pool hit ratio
- Fewer index leaf pages
  - Fewer logical and physical I/Os for index scans
  - Fewer splits
  - Better buffer pool hit ratio

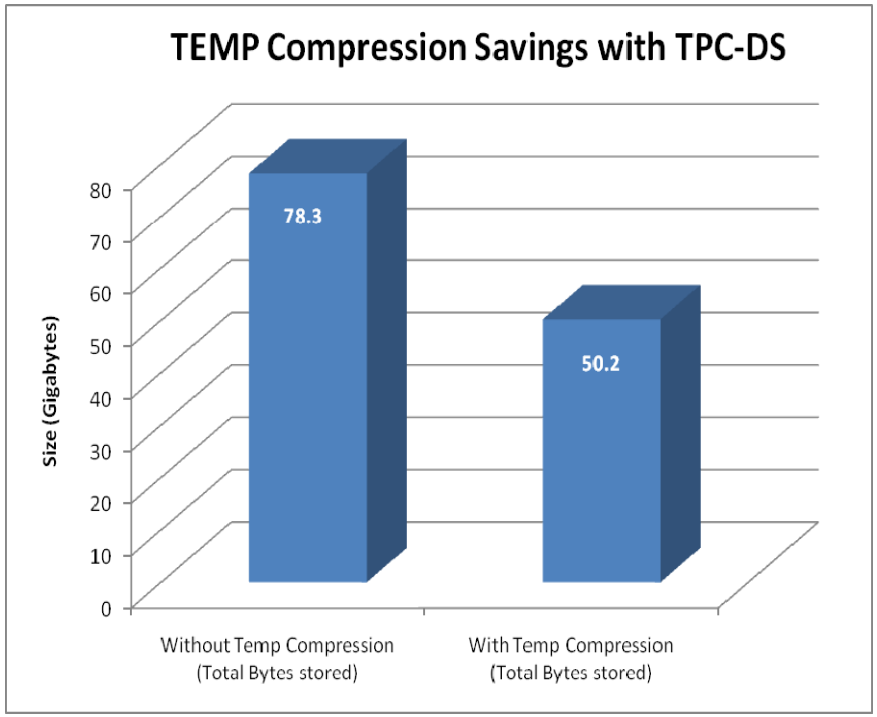




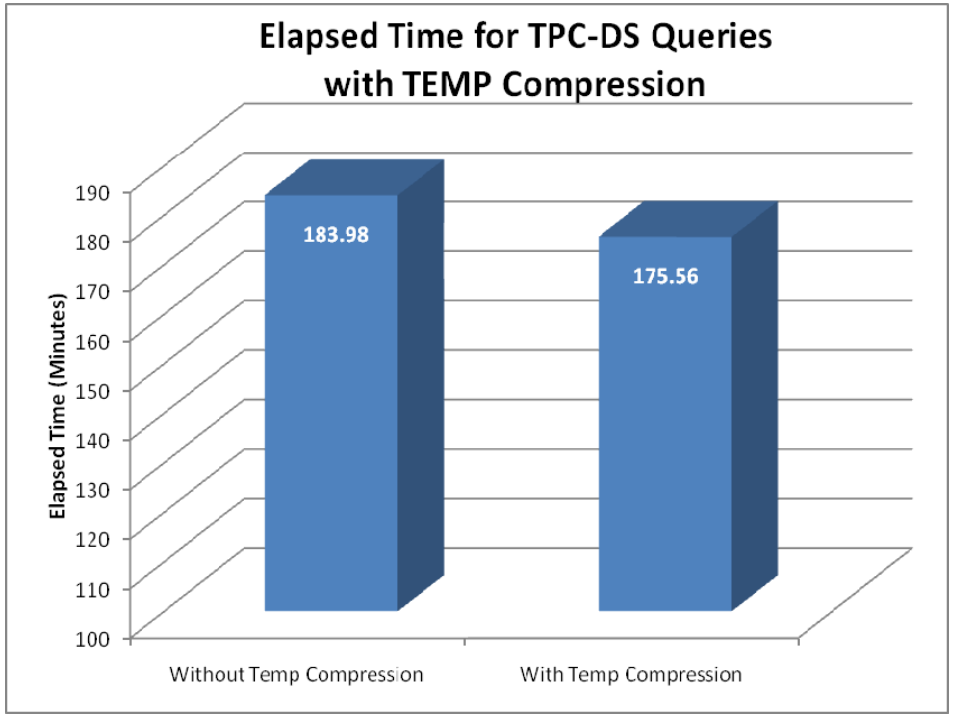
Compression Benefits
Temp Table Compression

- Only vendor in industry to compress temp tables
  – Internal (DB2 utilities, SORT) and external (DGTTs) compressed
- Savings and performance benefits require no DBA intervention

Space Savings with Temp Compression



Time Savings with Temp Compression



Backup Compression

- Reduce size of database backups in addition to other table-level compressions
- All of data in backup image is compressed, including
  - Catalog tables
  - Index objects
  - LOB objects
  - Auxiliary database files
  - Database meta-data
  - more
- Enable with COMPRESS option in BACKUP DATABASE

```
BACKUP DATABASE <database> … COMPRESS
```

Archive Log Compression

- New in DB2 10 – archive log compression reduces log archive storage
  - Automatically compresses log extents on-the-fly
  - Uses the same algorithm as (default) backup compression
- Simply turn it on and DB2 does the compression for you
  - Enabled by Logarchcompr1 and/or Logarchcompr2 database configuration parameter set to ON

Large SAP customer generates 60GB of log per day and they keep 8 weeks of archives

- Storing 3.3TB of archived log files
- Compression of **4x** results in storage of **only 825GB for 8 weeks**

Client Success
What Customers are Saying ...

- Customer experiences are consistent
  - Tables will compress in the range of 60% - 80%
  - Indexes compress around 50%
  - Temp space will compress around 35%
  - Overall database storage savings will be between 50% and 65%

That's 65% less disk space needed to support a DB2 10 database!

Summary

- Storage optimization through DB2 10 compression can save 55-70% of your database storage requirements
- You not only see your online database storage size shrink but often more importantly, your backup storage and disaster recovery storage is less as well
- In real customer examples storage savings are realized and performance benefits are apparent
- DB2 10 offers better compression ratios, improved ease of use and higher data availability
  - Adaptive compression provides better compression than classic row compression
  - Adaptive compression adapts to date skews over time – i.e. no REORG needed to maintain compression ratio

The next steps…

The Next Steps…

- Complete the Hands on Lab for this module
  - Log onto SKI, go to “My Learning” page, and select the “In Progress” tab.
  - Find the module
  - Download the workbook and the virtual machine image
  - Follow the instructions in the workbook to complete the lab
- 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

The Next Steps…

- The next set of modules to consider :
  - <Name the next set of modules that user should consider.>
- Additional Reading Material
  - Provide links etc (might be available for existing bootcamps)
  - Link 1
  - Link 2
- <Any other topics – such as applicable social media links>

Questions?
askdata@ca.ibm.com