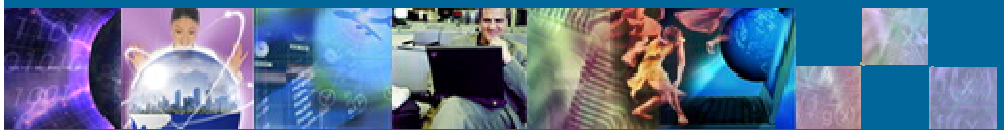


Using DB2 Viper Compression to Cut Costs

DEMAND BUSINESS™

DB2 9 (formerly known as Viper) from 10,000 ft

- Information as a Service
  - True Native XML
  - Native and Mixed XML/SQL Integrated in Single Infrastructure
- Secure and Resilient
  - Powerful New Security Mechanisms
  - Limits? What Limits?
  - New Flexibility in Recovery
- Manage your Business not your Database
  - Automatic Storage Management
  - Automatic Memory Management
  - Advanced Compression
  - Partitioning Advancements
- Agile Development
  - .NET and Visual Studio
  - Eclipse
  - Deployment Advancements

www.ibm.com/db2/viper

Today's Focus

- Information as a Service
  - True Native XML
  - Native and Mixed XML/SQL Integrated in Single Infrastructure
- Manage your Business not your Database
  - Automatic Storage Management
  - Automatic Memory Management
  - **Advanced Compression**
  - Partitioning Advancements

- Secure and Resilient
  - Powerful New Security Mechanisms
  - Limits? What Limits?
  - New Flexibility in Recovery
- Agile Development
  - .NET and Visual Studio
  - Eclipse
  - Deployment Advancements

www.ibm.com/db2/viper

Storage growth resulting in increasing costs

"The networked and attached data storage market is growing 5 to 9 percent faster than the server and personal computer markets, according to a Merrill Lynch industry survey of 75 U.S. and 25 European CIOs."

"Storage hardware is steadily increasing as a percentage of IT budgets"

"Backup and recovery was the second most important driver of spending in 2006—and is the No. 1 storage issue keeping CIOs awake at night"

DB2 Viper Compression significantly reduces both online storage costs and backup storage costs

What is DB2 9 Row Compression?

Agenda

- DB2 Compression
- The Nature of Data
- Compression Concepts
- Data Row Compression in DB2
- Enablement and Management
- Compression Estimation
- Use Cases
- Statistics
- Large RIDs
- Backup Compression
- Limitations
- Proof points
- Summary
- Testimonials

DB2 Compression

- V8 GA - NULL and Default Value Compression
  - No disk storage consumed for NULL column values, zero length data in variable length columns and system default values
- V8 GA - Multidimensional Clustering
  - Significant index compression can be achieved through block indexes
    - One key per thousands of records (vs one key per record with traditional indexes)
- V8 FP4 - Database Backup Compression
  - Smaller backup images; compress index and lf/lob tablespaces
- **DB2 9 - Data Row Compression**

The ‘Nature’ of Data

- As data continues to grow, the cardinality of the data drops. As it turns out there just are not that many truly "unique" things in the world. Now, they may be unique when used in combination, like DNA, but the basic elements themselves are not all that varied. Consider the Periodic Table of Elements - everything in our world is made up of this rather small set of basic elements in combination. Apply the concept to data, and you find the same is true.
- For instance, there may be about 300M people in the US according to the last census, but there are only approximately 78,800 unique last names, producing very low cardinality, with huge "clumps" in certain name sets. First name is even worse, coming in at about 6,600 unique first names (4,400 for the females, 2,200 for males).
- The names of cities and streets and their "normalized", address corrected adornments (Street, Avenue, etc.) are also very low cardinality.
- The English language has about 64,000 words, and those in most common use count less than 10,000.
- Product names, descriptions and attributes ("Dolby Digital") also tend to be highly redundant and low cardinality.

➢ Hence, symbol based compression works even better over very large domains of data, since the data within the domain is statistically not that variant.

There is statistical redundancy in data – Zipf’s Law, Pareto Principle (80:20 rule), …

DB2 Row Compression Concepts

- Dictionary based - symbol table for compressing/decompressing data records
  - Lempel-Ziv (LZ) based algorithm (static dictionary) utilizing 12bit symbols
  - Dictionary per table stored within the permanent table object (~75KB in size; disk+memory)
- Data resides compressed on pages (both on-disk and in bufferpool)
  - Significant I/O bandwidth savings
  - Significant memory (bufferpool) savings
  - CPU costs
    - Rows must be decompressed before being processed for evaluation
- Log data from compressed records in compressed format
- Does not compress rows where no storage saving is realized for that row

  “red” -> ‘#1#2#3’ symbol #1=“r” symbol #2=“e” symbol #3=“d”

  “red” == 3 bytes ‘#1#2#3’ == 12bits+12bits+12bits (or 4.5 bytes)

Compression Basics

- It is substitutional: replaces longer strings with 12 bit symbols e.g.,

  "Once upon a time …… The End"

  "Once" -> [symbol # 0]

- Uses a static dictionary: [string, symbol #] list is pre-built and stored in the table e.g., a sample dictionary

  {("Once", 0), ("upon a", 1), …("The End", 4095)}

- Compression is just the process of matching a byte string (data row) into the longest sub-strings present in the tree, and substituting the corresponding symbol #s

- Expansion is even easier: given a compressed row

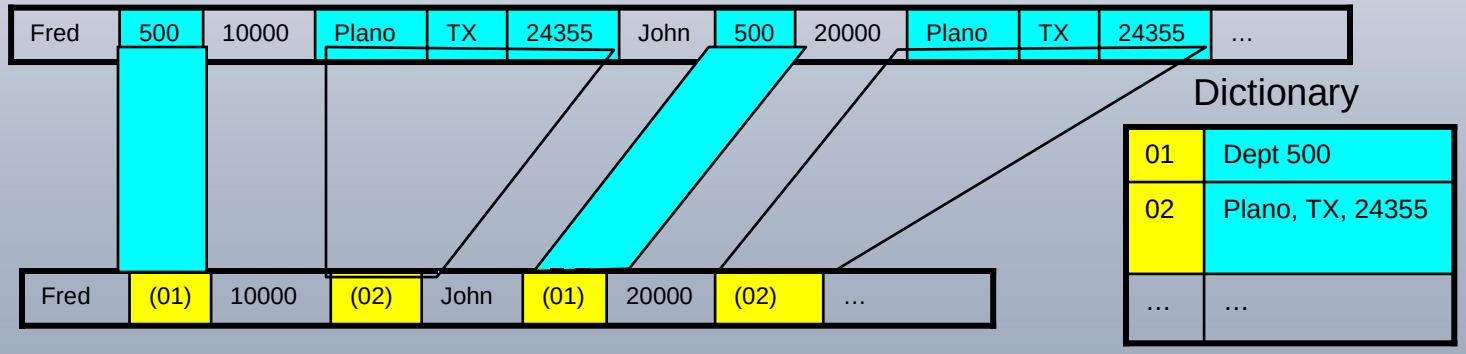
  [symbol #][symbol #] …

  simply do a look up and substitute each [symbol #] with the corresponding byte string

DB2 Data Row Compression

- Repeating patterns within the data (and just within each row) is the key to good compression. Text data tends to compress well because of reoccurring strings as well as data with lots of repeating characters, leading or trailing blanks

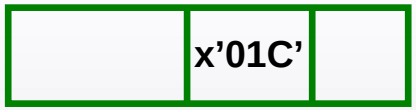
Name	Dept	Salary	City	State	ZipCode
Fred	500	10000	Plano	TX	24355
John	500	20000	Plano	TX	24355



DB2 Data Row Compression

Uncompressed Row

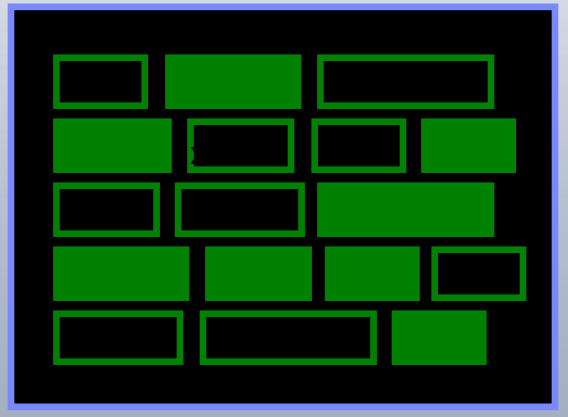
Compressed Row



Common sequences of consecutive bytes in row replaced with 12 bit symbol

Data page with uncompressed rows

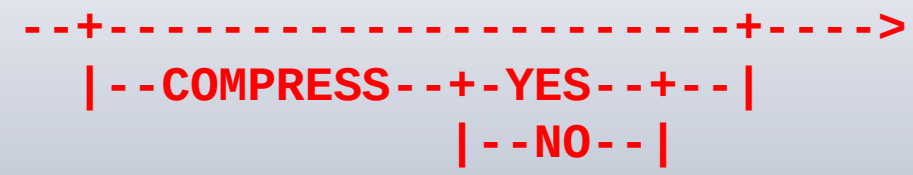
Data page with compressed rows



Enablement - Table DDL

- **CREATE TABLE <table name> --->**

- **ALTER TABLE <table name> --->**



- Compression is enabled at the table level via either the CREATE or ALTER TABLE statements
- Compression will be in effect once a table dictionary is built

Catalog Changes – SYSIBM.SYSTABLES

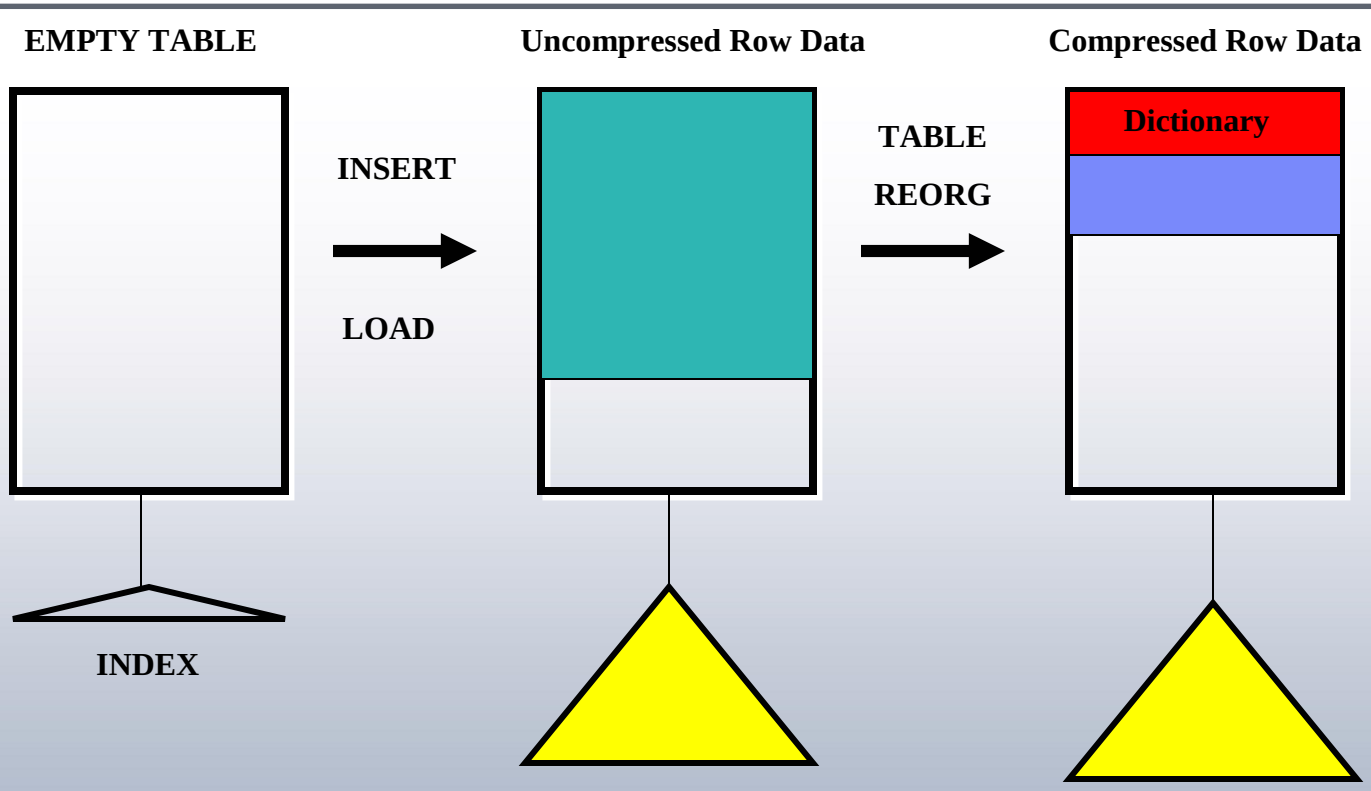
- COMPRESSION column value

Value	Meaning
N	No compression is set
V	Only Value compression is set
R	Only Row compression is set
B	Both Value and Row compression are set

Dictionary Building – 'Offline' Table REORG

- When the dictionary is being built a temporary in-memory buffer of 10MB is required
  - This memory will be allocated from the utilities heap
- All the data rows that exist in a table will participate in the building of the compression dictionary

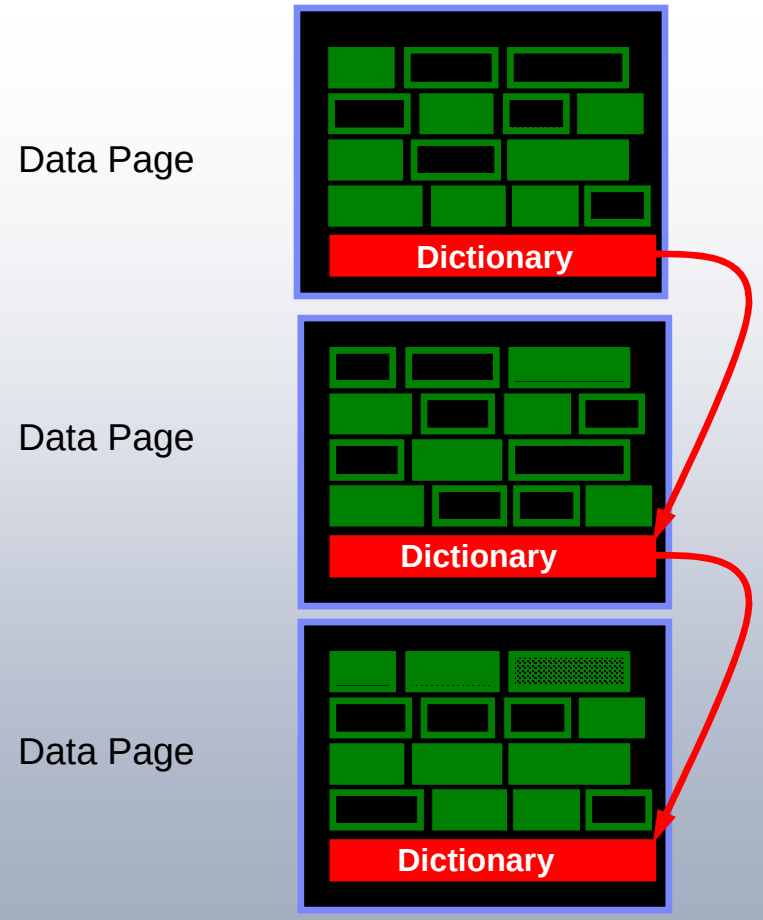
```
>--REORG--<table name>--+-----------------------+---->
                        '--INDEX--<index name>--'
   .-ALLOW READ ACCESS-.
>--+-------------------+--+---------------+--+-----------+->
   '-ALLOW NO ACCESS---'  '-USE-<tbspace>-'  '-INDEXSCAN-'
                     .-KEEPDICTIONARY---.
>--+-------------+-+--------------------+-+-->
   '-LONGLOBDATA-' '-RESETDICTIONARY---'
```



Compression Dictionary Build in DB2 9

Row Compression Dictionary

- Compression dictionary
  - Stores common sequences of consecutive bytes in a row
    - Such sequences can span consecutive columns
  - A table must have a compression dictionary before rows can be compressed
- Compression dictionary storage
  - Directly in table partition
  - In special, internal, non-selectable rows which are linked together
  - Typically on the order of 75KB
- Compression dictionary creation
  - In initial release, requires non-inplace REORG, INSPECT



Dictionary Management

- The dictionary is deleted only during table truncation, if the COMPRESS table attribute is turned off
  - If compress attribute is set, dictionary is preserved
- Operations such as TABLE REORG or LOAD/REPLACE or INSERT/REPLACE drive table truncation - dictionary can be removed
- No other alters or updates are performed against the dictionary, or any other dictionary records – the dictionary is static

Dictionary Management by Table REORG

RESETDICTIONARY

Table COMPRESS Attr	Dictionary Exists	Result Outcome
YES	YES	Build new dictionary; rows compressed
YES	NO	Build new dictionary; rows compressed
NO	YES	Remove dictionary; all rows uncompressed
NO	NO	No effect

KEEPDICTIONARY

Table COMPRESS Attr	Dictionary Exists	Result Outcome
YES	YES	Preserve dictionary; rows compressed
YES	NO	Build dictionary; rows compressed
NO	YES	Preserve dictionary; all rows uncompressed
NO	NO	No effect

Compression Estimator (Advisor) - INSPECT

- This tool looks at all the rows of the table data, and builds a compression dictionary from it. This dictionary will then be used to test compression against the records contained in the sample. Results include:
  - Estimate of compression savings
  - Dictionary size
  - Will insert the dictionary if COMPRESS YES is set
    - Allows for online dictionary creation/insertion
    - Future rows inserted/updated are compressed
    - Does not address existing rows (REORG to be used)

```
>>-INSPECT-+-| Check Clause |--------------------+------------->
           '-| Row Compression Estimate Clause |-'
…
|--ROWCOMPESTIMATE-TABLE--+-NAME--<table-name>--+-----------------------+-+-
|
                          |                     '-SCHEMA-- <schema-name>-' |
                          '-TBSPACEID-- <tbspc id> --OBJECTID-- <obj id>---'
```

DB2 INSPECT – Compression Evaluation

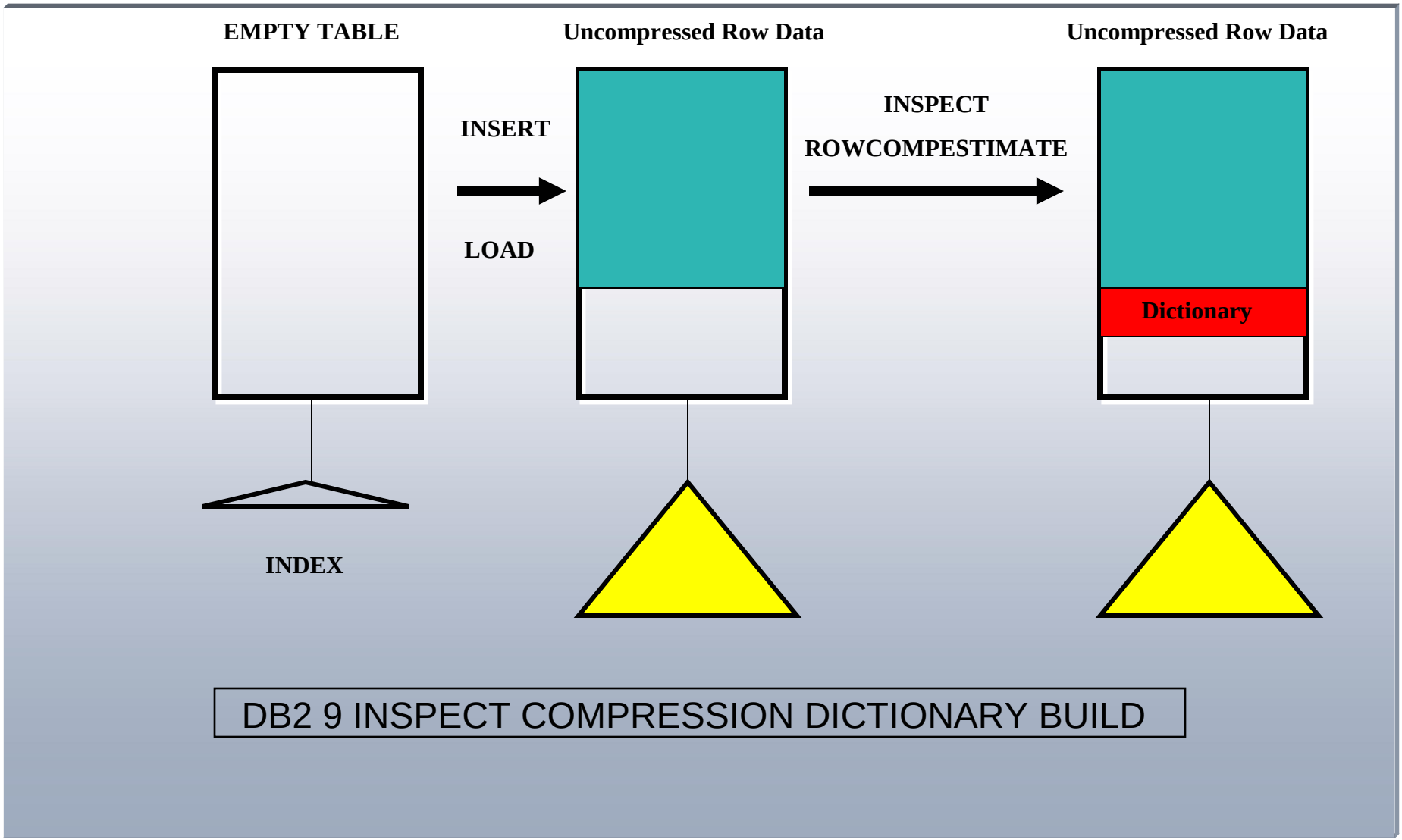
- **INSPECT ROWCOMPESTIMATE TABLE NAME <tbname>**
  SCHEMA <schema> RESULTS KEEP <filename>

  - In the db2dump directory, you will see <filename>
  - **db2inspf <filename> <outfile>**
  - <outfile> will contain the results of compression estimation

```
Action: ROWCOMPESTIMATE TABLE
Schema name: MIKEW
Table name: EMPLOYEE
Tablespace ID: 2  Object ID: 6
Result file name: emp

    Table phase start (ID Signed: 6, Unsigned: 6; Tablespace ID: 2) : MIKEW.EMPLOYEE

      Data phase start. Object: 6  Tablespace: 2
      Row compression estimate results:
      Percentage of pages saved from compression: 46
      Percentage of bytes saved from compression: 46
      Percentage of rows ineligible for compression due to small row size: 0
      Compression dictionary size: 13312 bytes.
      Expansion dictionary size: 10240 bytes.
      Data phase end.
    Table phase end.
```



DB2 9 INSPECT COMPRESSION DICTIONARY BUILD

Row Compression: New Table with Small Temp

- Create Table that is Eligible for Compression

```
CREATE TABLE Sales COMPRESS YES
```

- Get Representative Data Sample

```
LOAD FROM filesmall OF DEL REPLACE INTO Sales
```

- Creates dictionary on sample data

```
REORG TABLE Sales
```

- Load respects dictionary

```
LOAD FROM filerest OF DEL INSERT INTO Sales
```

Row Compression: New Table Partitions

- Makes table eligible for compression

```
ALTER TABLE Sales COMPRESS  YES
```

- New Sales Data Range

```
CREATE TABLE NewSales … COMPRESS YES
```

- Load Data into New Partition

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

- Data Now Gets Compressed

```
REORG TABLE NewSales RESETDICTIONARY
```

- New Partition is Added to the Table

```
ALTER TABLE Sales
  ATTACH PARTITION Mar05
  STARTING '03/01/2005'
  ENDING '03/31/2005'
  FROM TABLE NewSales
```

Administrative Table Function

```
db2 describe "select * from
   table(sysproc.admin_get_tab_info('MIKEW','STAFF')) as t"
```

sqltype		sqllen	sqlname.data	sqlname.length
449	VARCHAR	128	TABSCHEMA	9
449	VARCHAR	128	TABNAME	7
453	CHARACTER	1	TABTYPE	7
501	SMALLINT	2	DBPARTITIONNUM	14
497	INTEGER	4	DATA_PARTITION_ID	17
453	CHARACTER	1	AVAILABLE	9
493	BIGINT	8	DATA_OBJECT_L_SIZE	18
493	BIGINT	8	DATA_OBJECT_P_SIZE	18
493	BIGINT	8	INDEX_OBJECT_L_SIZE	19
493	BIGINT	8	INDEX_OBJECT_P_SIZE	19
493	BIGINT	8	LONG_OBJECT_L_SIZE	18
493	BIGINT	8	LONG_OBJECT_P_SIZE	18
493	BIGINT	8	LOB_OBJECT_L_SIZE	17
493	BIGINT	8	LOB_OBJECT_P_SIZE	17
493	BIGINT	8	XML_OBJECT_L_SIZE	17
493	BIGINT	8	XML_OBJECT_P_SIZE	17
501	SMALLINT	2	INDEX_TYPE	10
453	CHARACTER	1	REORG_PENDING	13
449	VARCHAR	10	INPLACE_REORG_STATUS	20
449	VARCHAR	12	LOAD_STATUS	11
453	CHARACTER	1	READ_ACCESS_ONLY	16
453	CHARACTER	1	NO_LOAD_RESTART	15
501	SMALLINT	2	NUM_REORG_REC_ALTERS	20
453	CHARACTER	1	INDEXES_REQUIRE_REBUILD	23
453	CHARACTER	1	LARGE_RIDS	10
453	CHARACTER	1	LARGE_SLOTS	11
493	BIGINT	8	DICTIONARY_SIZE	15

Compression Statistics in SYSCAT.TABLES

- AVGROWSIZE
  - Average physical row length in a table, including both compressed and uncompressed rows
  - Used to determine if the row count is approaching the maximum number of rows per page
- PCTPAGESSAVED
  - Estimate of the percentage of pages saved by using compression
- PCTROWSCOMPRESSED
  - Percentage of the number of rows that are compressed to the total number of rows in the table
  - Used by the optimizer to help determine the decompression CPU cost
- AVGROWCOMPRESSIONRATIO
  - Average compression ratio for all compressed rows
  - Used by the optimizer to help determine the cost of expanding a compressed row
- AVGCOMPRESSEDROWSIZE
  - Contains the average physical (on-disk) length of all compressed rows
  - Used by the optimizer to help determine the cost of expanding a compressed row

Larger RIDs – More Records/Data Page

Page Size	REG TBSP Min Rec Length	REG TBSP Max Records	LARGE TBSP Min Rec Length	LARGE TBSP Max Records
4 KB	14	251	12	287
8 KB	30	253	12	580
16 KB	62	254	12	1165
32 KB	127	253	12	2335

Backup Compression and Data Row Compression

- Test backup compression in addition to tables with row compression
- Backup compression can be expensive and may not provide much added value in additional savings to backup image size
  - **Time/size/value depends on the percentage of table space content with row compression. E.g. Are all tables compressed? Are indexes or long data stored in the same table space?**

Scenario	Total User Time (seconds)	Pages Used	Table Space Size (GB)	Backup Image Size (GB)
No compression	468	1510400	11.57	12
Backup Compression Only	1028	1510400	11.57	4.2
Data Row Compression Only	198	610816	4.68	4.7
Data Row and Backup Compression	662	610816	4.68	4.2

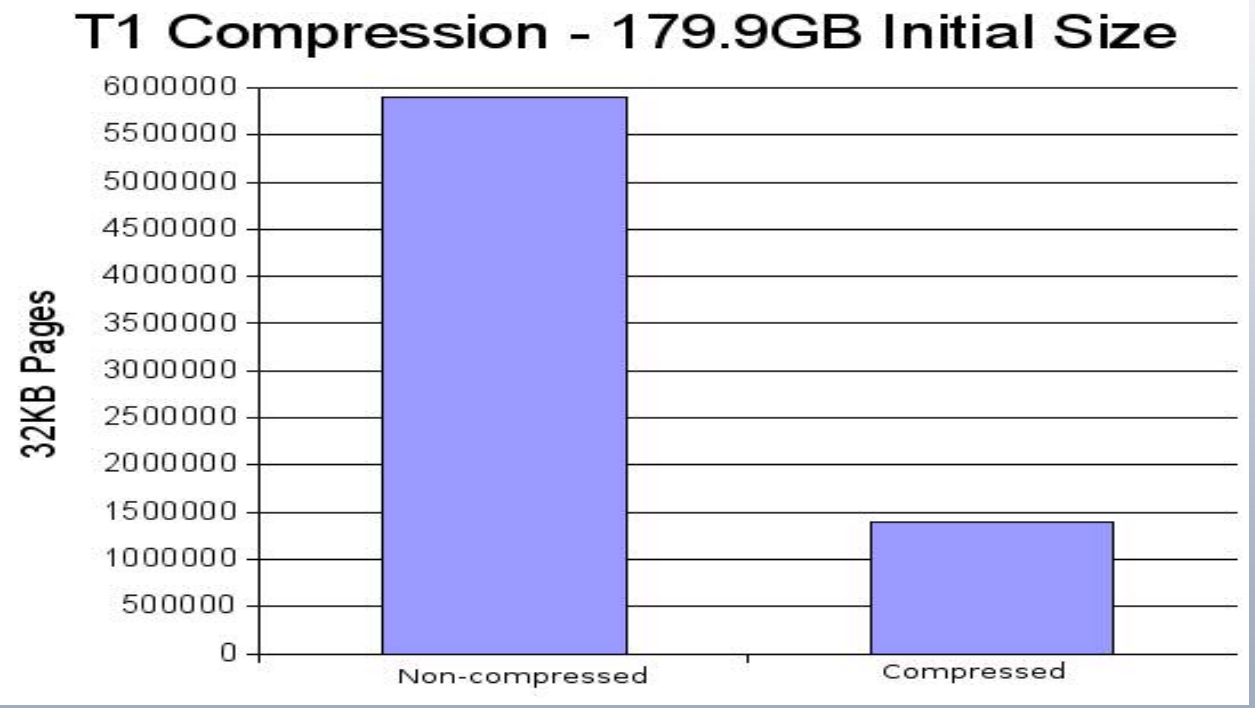
Limitations of Data Row Compression

- Row compression is only supported for tables supported by table REORG. The following tables cannot be compressed:
  - RCT tables (range-clustered tables)
  - Catalog tables
  - Declared global temporary tables
  - System temporary tables
- Row compression is only applicable to rows in data objects table space!! (the key to this is if it is stored with tables, or somewhere else)
  - Pure XML in separate object – XML data is not compressed
  - LOBs are stored in separate object – LOB data is not compressed
  - LONGs are stored in separate object - LONG data is not compressed
- Row compression support and table data replication support will not be compatible.
  - DATA CAPTURE CHANGES option is not compatible with the COMPRESS YES option. If the compress row attribute is NO for the table but the table has a dictionary, the DATA CAPTURE CHANGES option is also not allowed since there may be rows that are compressed in the table.
- Compression is not practical for a table smaller than 100KB, since the space savings may not offset the storage requirements for the dictionary.

Compression Ratios (Customer Financial Data)

Compression Type	32KB Page Count	Space Required on Disk
No compression	5893888	179.9GB
Row compression	1392446	42.5GB

% Pages Saved: 76.4%



Compression Savings – Customer Data Warehouse

Table	% of Table Sampled	Number of Rows	Sample Size (GB)	Sample Size Compressed (GB)	Compression Ratio	Total Estimated Disk Savings (GB)
T1	100	159259747	71.49	13.23	81.5%	58
T2	10	77390826	21.48	5.23	75.7%	162
T3	100	191362309	141.46	36.18	74.4%	105
T4	20	19474261	11.80	3.05	74.2%	43
T5	10	23370458	25.78	8.54	66.9%	172
T6	10	39573969	23.24	6.32	72.8%	169

Data warehouse estimated reduction in size from 2.873TB to 1.453TB

Query Performance

- Sample Warehouse Query Workload
  - I/O bound system: observed overall improvements in end-to-end workload execution time (20-30%)
  - CPU bound system: performance neutral or degradations on the order of 10% in end-to-end workload
- Compression trades I/O time for CPU time
  - If the system is CPU bound, or if an agent is already consuming an entire CPU on its own, then compression will likely have a negative impact
  - If there is CPU headroom and I/O waits are happening compression can be beneficial

In Summary

- The storage savings and performance impact of data row compression are intimately tied to the characteristics of the data within the database, the configuration of the database (layout and tuning) as well as application workload.
- In general terms, the net performance impact of compression depends on whether the database is CPU or I/O bound.
- Data row compression can significantly reduce table size at the expense of additional CPU utilization.
- Test drive the Beta code to a get sense for what impact you can expect with respect to both storage and performance.

Customer Feedback

- **"… is quite pleased with the Viper implementation of data compression - there is significant storage reduction, and yet is seems there is no significant overhead in CPU usage ; this is great news for anyone with large databases!"**

- **"I got about 156K pages of data in 2 tables compressed to about 32K pages. My elapsed times from db2batch were about 42% lower with the compressed data."**

  "The results match my expectations but may surprise others."

- **"Row compression saves more than 50% storage"**

  "Row compression has a little positive impact on <…> performance (throughput improved from 121.9/sec to 130.2/sec)"

Comparison of DB2 Viper Row Compression vs. the Competition

Comparison to Microsoft SQL Server 2005

- SQL Server 2005 has backup compression only available from 3rd parties
- They have no value compression
- They have no row compression

- In fact this note from Microsoft also recommends that you do not backup or store your database on compressed file systems
  - http://support.microsoft.com/kb/231347/en-us

Comparison to Teradata

- Teradata has dictionary based compression
- However,
  - Only at the column level for each column in the table
  - Only compresses 255 values per column
  - **DBA MUST SPECIFY THE VALUES TO COMPRESS ON CREATE TABLE**
  - Examples in their marketing material show 30% to 50% table compression

```
CREATE TABLE property (
   street VARCHAR(40),
   city CHAR(20) COMPRESS ('Chicago','Los Angeles','New York'),
   statecode CHAR(2)
)
```

Oracle Compression

- Oracle does allow tables or partitions of tables to be compressed
- However, Oracle compresses out common values **at the page level** (DB2 compression is at the table level)
  - This means that repeating values in a single page are replaced by a symbol in Oracle. The symbols are stored in the page header
  - Disadvantages of Oracle's approach
    - If there are consistent repeating values throughout the entire table, they will be stored multiple times in each page header
    - If the data is not sorted, there may be repeating patterns in the table but not on each page so Oracle will miss out on these compressions
    - Oracle only supports compression for bulk loads
      - DB2 supports compression for load, insert, and import

Key DB2 advantages over Oracle

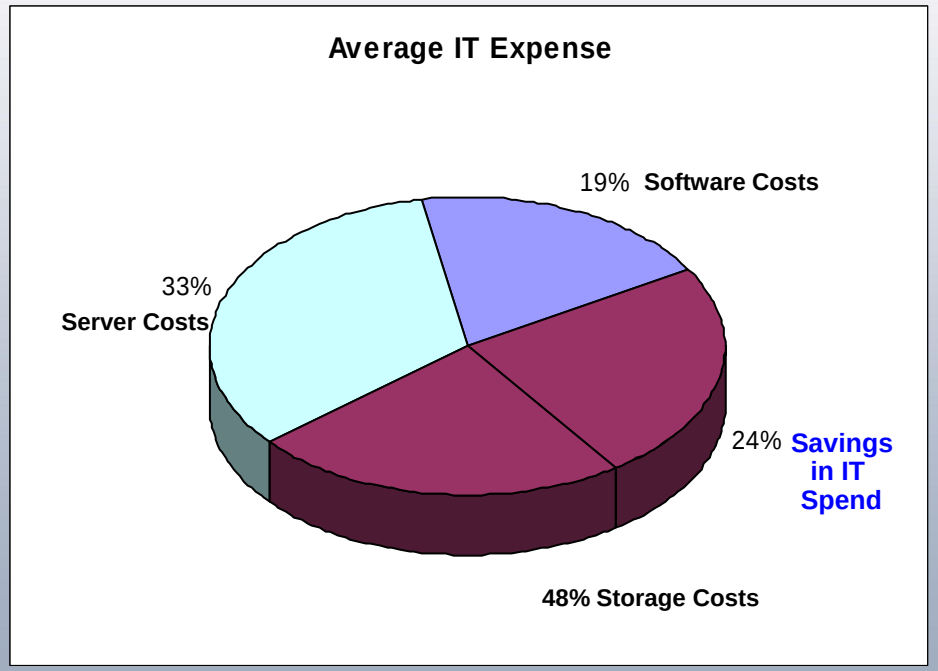
- Non technically speaking
  - How many people in your department share the same birthdate?
  - How many people in your entire company share the same birthdate?
  - DB2 looks at a much large population of data and therefore finds more patterns to compress

- In a research paper on compression written by Oracle engineers they state:“Due to its global optimality of compression a **table-wide dictionary approach can result in high compression factors**.”

Table	Compression Ratio	
	Oracle	**DB2**
LINEITEM	38%	**58% (1.5x better)**
ORDERS	18%	**60% (3x better)**

Cost savings with DB2 Viper Compression

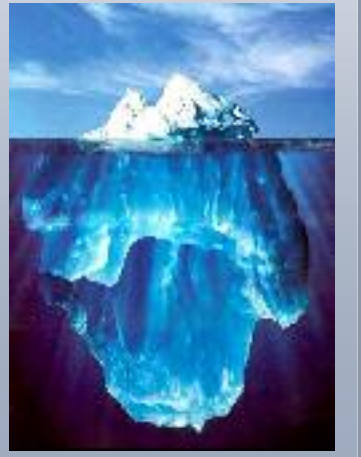
- Storage infrastructure is often the largest expense in large databases
- Cutting this cost in half can result in significant bottom line savings

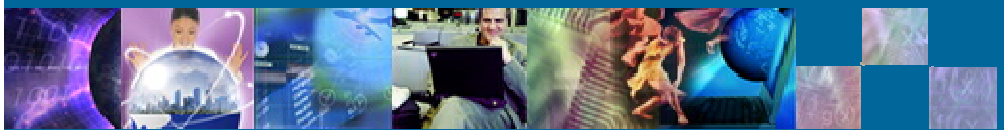


Based on large scale data warehouse benchmarks

(Some) Other DB2 9 (Viper) Highlights

- **Information as a Service**
  - New Native XML data store, complete with new storage, indexing and SQL and XQuery support
  - New application development tools and capabilities for more agile application development
  - Enhanced driver for JDBC and SQLJ
  - New light-weight CLI/ODBC driver
  - ….
- **Scalability & Performance**
  - Fast communication manager enhancements for improved performance, scaling & dynamic buffer mgt
  - IPV6 support
  - Generalized Row Compression
  - …
- **Total Cost of Ownership**
  - Self Tuning Memory Management
  - Automatic Storage Management
  - Automatic support for IO_SERVERS and AVG_APPLS
  - Better out of the box defaults
  - Inplace ALTER
  - New schema level operations (COPY, DROP,…)
  - New SQL administration capabilities (eg. comprehensive table state)
  - …..
- **High Availability & Resiliency**
  - Restartable Recovery
  - REBUILD Recovery
  - Asynchronous index rebuild after restart
  - Performing redirected restore automatically with scripts generated from backup images
  - .....
- **Security**
  - Label Based Access Control
  - A new security administrator level (SECADM)
  - A new RESTRICT option for CREATE DATABASE
  - …..





Using DB2 Viper Compression to Cut Costs

DEMAND BUSINESS™