#SHAREorg

SHARE
Technology · Connections · Results

Data Warehousing on System z: Best Practices with DB2 for z/OS

Willie Favero
IBM Silicon Valley Lab
Data Warehousing on System z Swat Team
(DB2 SME)

Thursday, August 9, 2012
1:30 PM – 2:30 PM
Session Number: 11954

my TWITTER
#zdwdb2

SHARE in Anaheim 2012

Please Note:

IBM's statements regarding its plans, directions, and intent are subject to change or withdrawal without notice 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.

Acknowledgements and Disclaimers:

Availability. References in this presentation to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates.

The workshops, sessions and materials have been prepared by IBM or the session speakers and reflect their own views. They are provided for informational purposes only, and are neither intended to, nor shall have the effect of being, legal or other guidance or advice to any participant. 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. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, this presentation or any other materials. 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 the applicable license agreement governing the use of IBM software.

All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics may vary by customer. Nothing contained in these materials is intended to, nor shall have the effect of, stating or implying that any activities undertaken by you will result in any specific sales, revenue growth or other results.

What Do "I" Mean by Best Practices

- This presentation consist of
  - Stuff I have been asked about
  - Things I have tried that have worked
  - Techniques I've read about that made sense
- Please consider this a starting point, not the conclusion
- All put in one place for you to decide what's good and what isn't

Here's My "Top 10" Best Practices

- Partitioning
- Parallelism
- Data compression
- Index Compression (maybe not)
- Star (Snowflake) Schema
- Query Processing
- Manage statistics
- Maintain accounting and statistical data
- Are backups necessary
- DSNZPARMs
- Buffer Pools

Partitioning

- Large objects
  - Manageability
  - Potential performance improvements
    - Parallelism
      - # CPs and # of Partitions
    - Scan only necessary partitions
  - Should almost always be considered for fact table
    - Use date key for fact table partitioning if available
  - You could have up to 4096 partitions

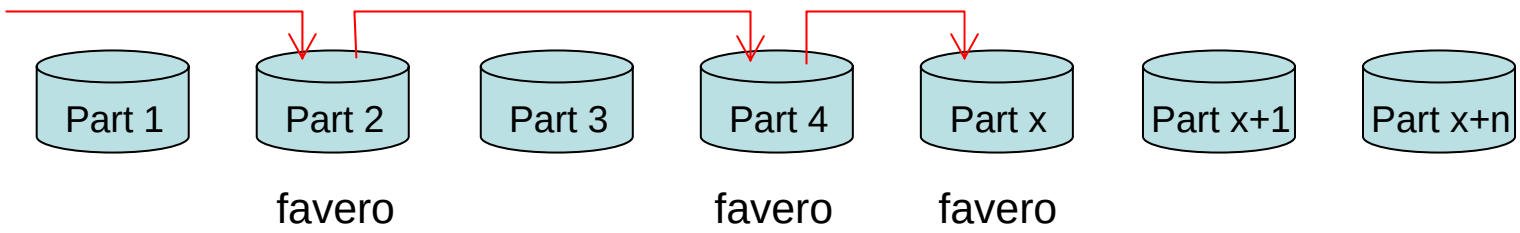
Partitioning Key

- Date column if available
  - Year? Quarter? Month? Day even?
  - Sometimes daily partitions can be attractive option
    - Could be easier to manage
- Surrogate key if not a usable date column
  - Use integer; short key
  - Use BIGINT; if more values are necessary

Partition Pruning

```
SELECT
    FROM part_table
         WHERE LASTNAME = ‘favero’
```

Partition by date



What About Universal Table Spaces?

- Introduced in DB2 9
- Improved in DB2 10
- The best of segmented and partitioning in one object
  - Partition-by-growth
  - Range-partition
  - Both can have to 128 TB of data
  - Mass DELETE

Table Space Definition

- No or low INSERT/UPDATE rate
  - FREEPAGE 0
  - PCTFREE 0
- SECQTY set to -1
- PRIQTY consider setting to -1
- LOCKSIZE
- CLOSE
- NOT LOGGED

Table Space Definition

- SEGSIZE
  - If > 0, range-partitioned universal table space if NUMPARTS is also specified
  - 0 and NUMPARTS specified, legacy partitioned table space
- DSSIZE
  - Defines maximum size of each partition
  - 1G, 2G, 4G, 8G, 16G, 32G, 64G, 128G, 256G
- NUMPARTS
  - Indicates partitioned table space
  - 1 to 4086 partitions can be specified
  - Page size and DSSIZE drive max NUMPARTS value

What About Dimension Tables?

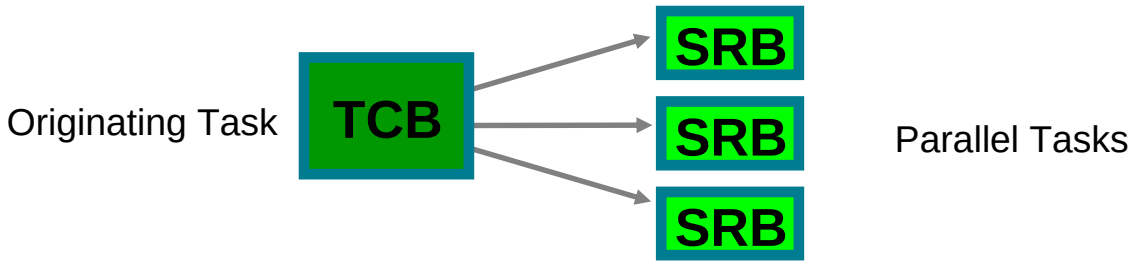
- They have design considerations also
- Use integer surrogate keys if date not available
- Using integer prevents fact table from unnecessary width

Parallelism

CP Parallelism (Behind the Scenes)

Multi-Tasking - How does DB2 do it?

- Spawning parallel tasks: z/OS preemptable SRBs are used for work done in parallel. Originating Task (TCB) handles SRB creation, cleanup and data merging.

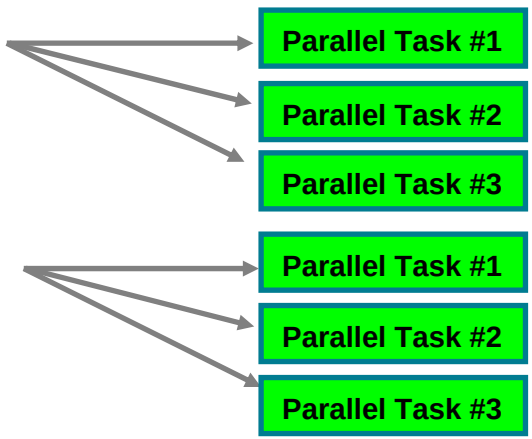


- Preemptable SRBs:
  - Synchronize originating and parallel tasks
  - Introduced with Enclave Services (MVS 5.2)
  - Inherit dispatching priority of allied address space. Therefore all work is done at the same priority (goodness)
- Originating task does not control scheduling or which CP an SRB is run on – z/OS handles scheduling.
- DB2 handles synchronization through suspending and resuming tasks

CP Parallelism (Behind the Scenes)

- Parallel tasks are started at OPEN CURSOR*
  - Application might be able to take advantage of this to achieve inter-query parallelism:

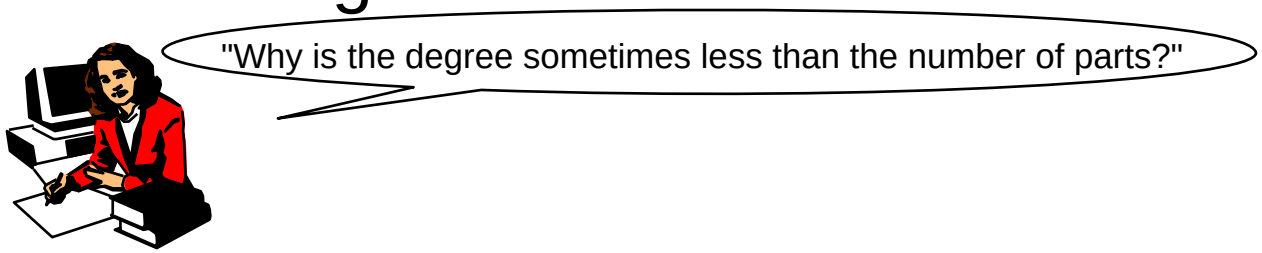
```
DECLARE CURSOR C1 FOR SELECT COUNT(*) FROM ORDERS
WHERE INVOICE_AMT > 4000.00
DECLARE CURSOR C2 FOR SELECT PARTNAME FROM PARTS
WHERE INVENTORY_AMT > 200

OPEN CURSOR C1
OPEN CURSOR C2
...
...
...
FETCH C1
FETCH C2
```



*Exception if RID sort, but no data sort, then //ism starts at first fetch (same as without //ism)

CP Parallelism (Behind the Scenes)

Parallel Degree Determination



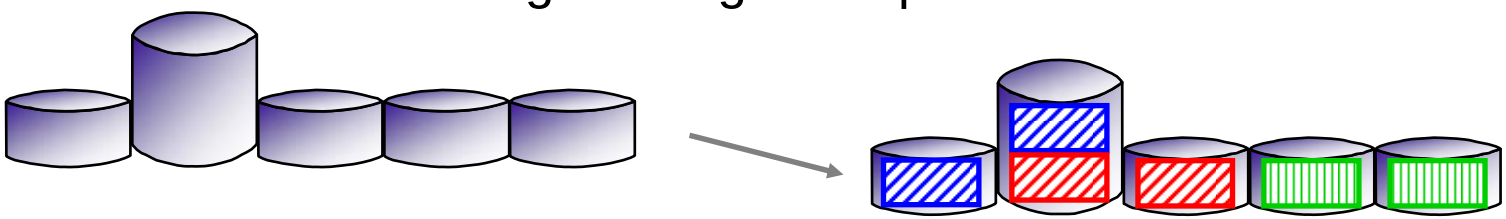
- Optimal degree for parallel group is determined at BIND/PREPARE time
  - Also called "Planned BIND degree" - shown in EXPLAIN output
- Optimal degree determined by considering:
  - Number of table space partitions
  - Estimated I/O cost of largest partition
  - Estimated CP cost considering:
    - Processing cost
    - MIPS rating of machine
    - Number of CPs on-line (used for CP parallelism only)
- Degree determination deferred if access path dependent on host variable

Determining the degree of parallelism

- DB2 chooses the smallest degree that will still deliver the best possible elapsed time
  - With the shared data model, DB2 has the flexibility to choose the degree of parallelism
  - I/O-intensive: Degree of parallelism approaches the number of partitions

  - Processor-intensive: Degree of parallelism approaches the number of processors (for DB2 9 - times 4, for DB2 10 – times 2)

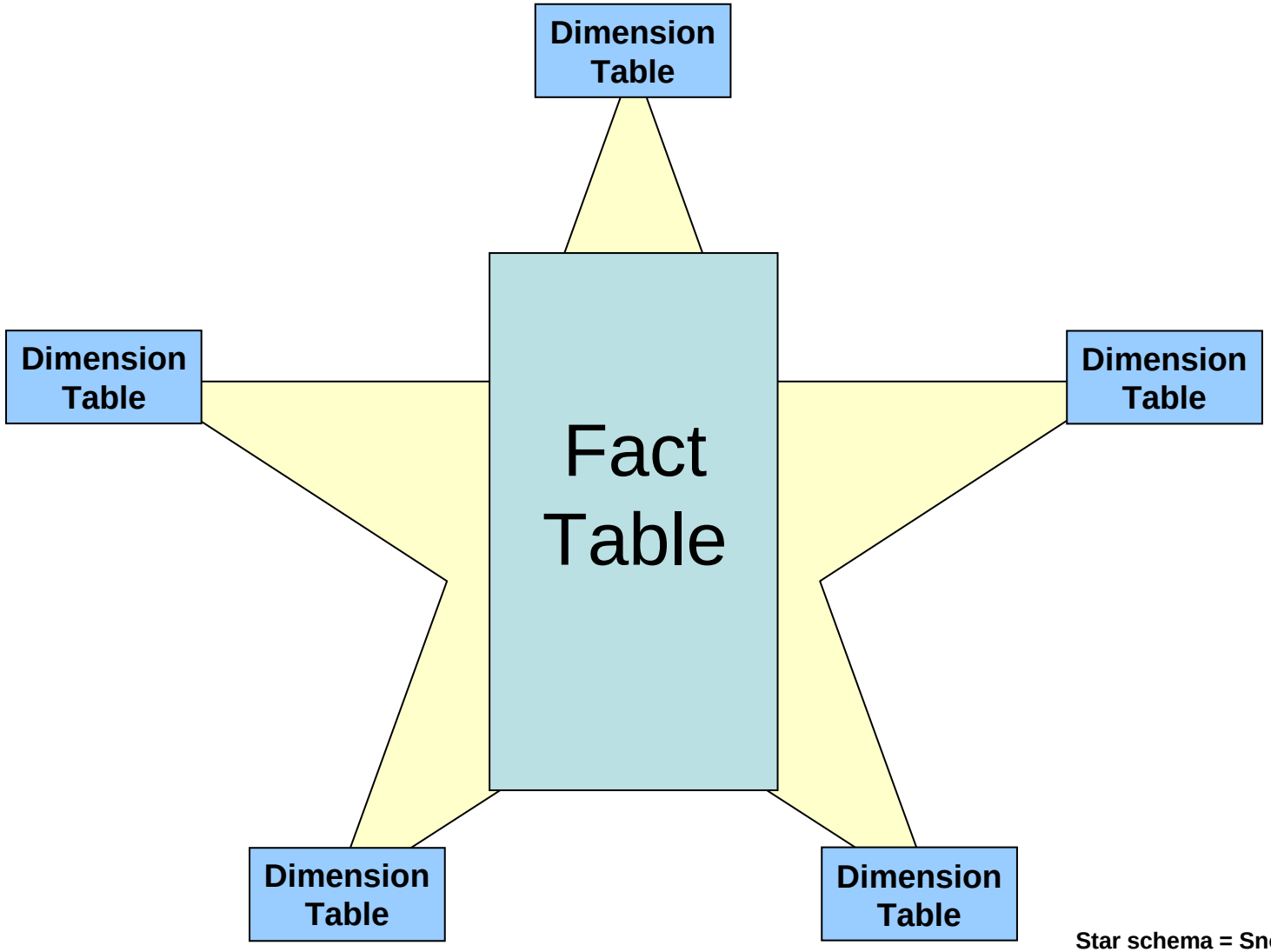
  - Additionally, skews in the data organization can be detected and compensated for in choosing the degree of parallelism



Star (Snowflake) Schema

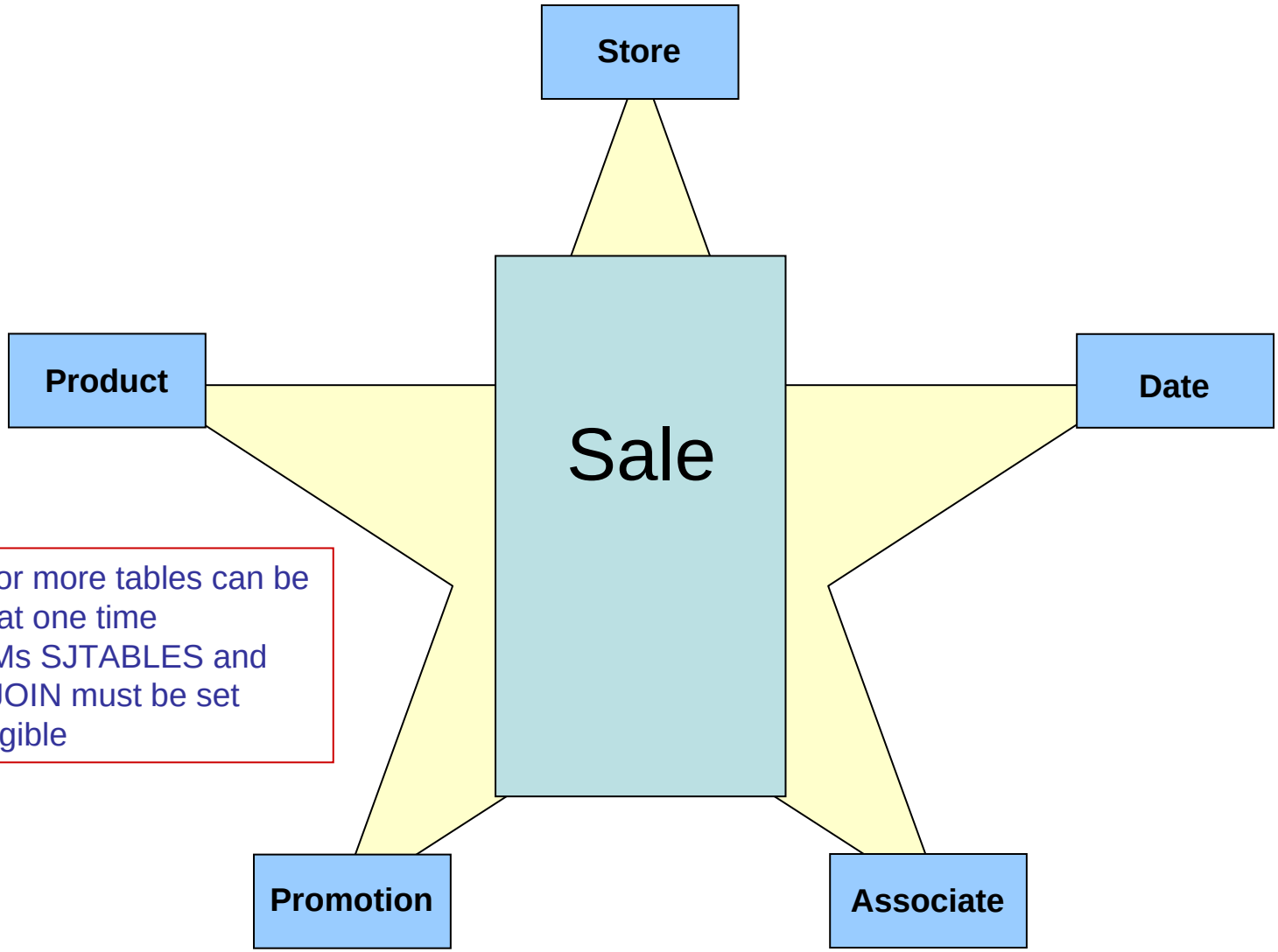
- Star (snowflake) schema = a relational database schema for representing multidimensional data
- Sometimes graphically represented as a ‘star’ or ‘snowflake’
  - Data is stored in a central fact table
  - Surrounding additional dimension tables hold information about each perspective of the data
  - Example: store "facts" of the sale (units sold, price, ..) with product, time, customer, and store keys in a central fact table. Store full descriptive detail for each keys in surrounding dimension tables. This allows you to avoid redundantly storing this information (such as product description) for each individual transaction
- Complex star schema parallel queries include the acts of joining several dimensions of a star schema data set (like promotion vs. product).
- Two specific DSNZPARMs must be setup accordingly: STARJOIN and SJTABLES.
- Proper index design must be present in the star schema tables.

Star Schema



Star schema = Snowflake schema

Star Schema



- Three or more tables can be joined at one time
- ZPARMs SJTABLES and STARJOIN must be set
- zIIP eligible

Star schema = Snowflake schema

Referential Integrity

- Good idea… just be careful
- Apply after LOAD to improve load performance
- Integrity verses performance?
  - What are the tradeoffs
  - Maintain integrity
  - Increase cost of INSERT, DELETE, and UPDATE processing

Queries

- Include partitioning key as WHERE predicate when possible
- EXPLAIN, Accounting and Statistics traces
  - Improves limited partition scan
    - EXPLAIN's Page-Range column
  - Know when to STOP tuning
- Take advantage of technology
  - Table Expressions
  - Materialized Query Tables (MQT)
  - Common Table Expressions (CTE)

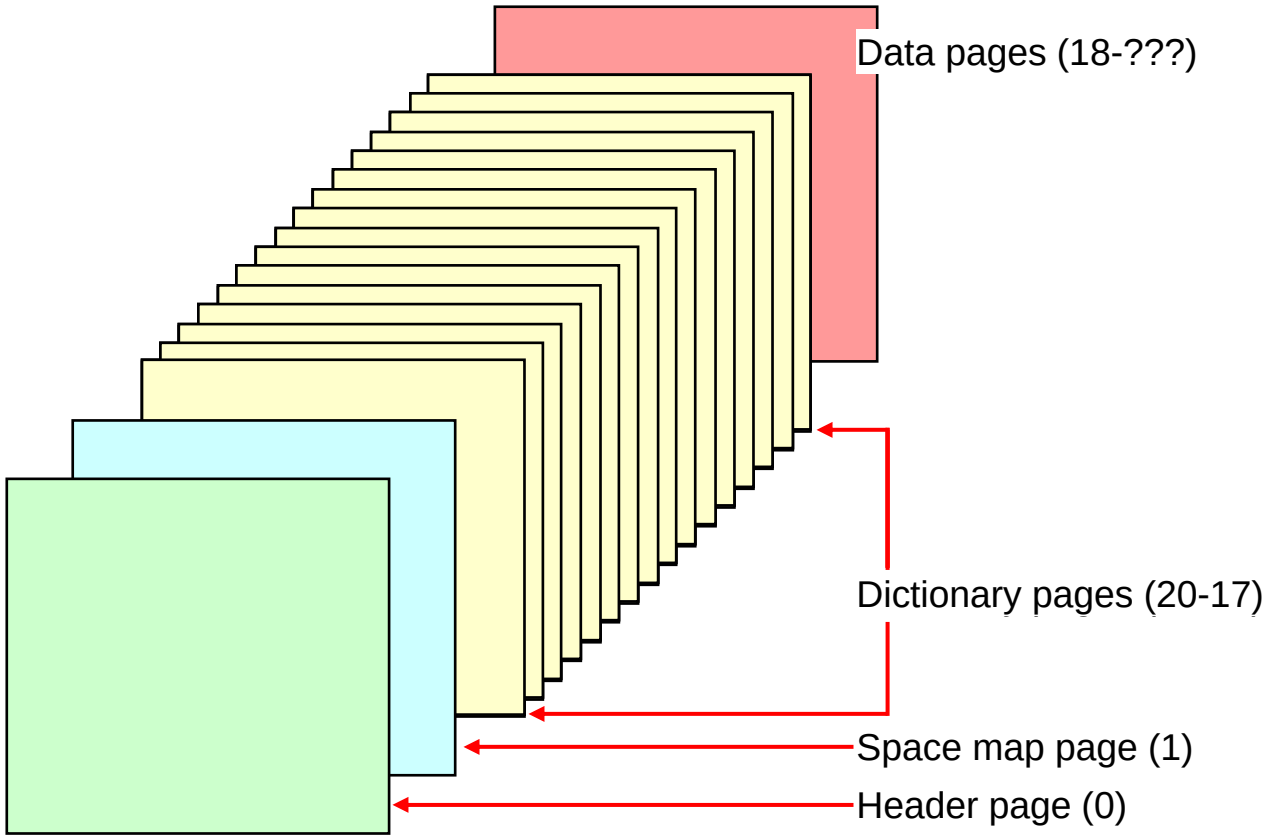
Manage Amount of Data Maintained

- If date partitioning key, roll off partitions
- Archive
  - Consider offloading partitions no longer accessed
    - Use a technique that removes data from fact table but still leaves it available to queries if necessary

DB2 Data Compression

- Compression should always be considered for a data warehouse
- Savings are usually greater than 50%
  - Have seen as high as 80% in certain situations
- Overhead on INSERT
  - minimal on SELECT
  - Warehouse queries dominated by sequential prefetch, which benefit from DB2 compression.
- Not all rows in a table spaces can be compressed
  - If the row after compression is not shorter than the original uncompressed row, the row remains uncompressed.
- Compression dictionary size
  - 64K (16 X 4K page) of storage in the DBM1 address space
  - Dictionary goes above the bar in DB2 Version 8 and later releases
- Faster hardware, faster compression
    - See hardware chart toward end of presentation

Dictionary – 4K Page Size



- The dictionary is created by the LOAD and/or REORG utilities only
  - It occupies:
    - 4K – 16 pages
    - 8K - 8 pages
    - 16K - 4 pages
    - 32K - 2 pages
- The compression dictionary follows the header and first space map pages (next slide)
- Dictionaries can be at the partition level (Careful, you could have 4096 partitions

Data Compression

- Rows are compressed on INSERT
- For an UPDATE
  - Expand, update, then re-compressed row
  - UPDATE has the potential to be expensive
- Changes (INSERT & UPDATE) are logged in compressed format
  - Possible reduced logging cost
  - Active log reductions carried over to the archive logs
- Larger page sizes may result in better compression.
  - Resulting rows after compression are variable length
  - You might be able to fit more rows with less wasted space in a larger page size.
- You cannot turn compression on for the catalog, directory, work files, or LOB table spaces
- Index compression does not use a dictionary

Possible Performance Gain

- When compression is on, data pages are brought into buffer pool in compressed state
  - Increasing the number of rows in the same size pool could increase buffer pool hit ratio
  - Increasing hit ratio could reduce I/O necessary to satisfy the same number of getpage requests.
- If Compression doubles the number of rows per page
  - When DB2 loads that page in a buffer pool, it will be loading twice as many rows.
- Less I/O is always a good thing.

DB2 Index Compression…..

- Index compression is new to DB2 9 for z/OS
- Page level compression
- Unlike data row compression:
  - Buffers contain expanded pages
  - Pages are decompressed when read from disk
  - Prefetch performs the decompression asynchronously
  - A buffer hit does not need to decompress
  - Pages are compressed by the deferred write engine
- Like data row compression:
  - An I/O bound scan will run faster
- DSN1COMP utility can be used to predict space savings

Index compression saves space, it's not for performance

Index Compression: Performance

- CPU cost is mostly inconsequential. Most of the cost is asynchronous, the exception being a synchronous read. The worst case is an index with a poor buffer hit ratio.

  Example: Suppose the index would compress 3-to-1. You have three options…..

  1. Use 8K buffer pool. Save 50% of disk. No change in buffer hit ratio or real storage usage.
  2. Use 16K buffer pool and increase the buffer pool size by 33%. Save 67% of disk, increase real storage usage by 33%.
  3. Use 16K buffer pool, with no change in buffer pool size. Save 67% of disk, no change in real storage used, decrease in buffer hit ratio, with a corresponding increase in synchronous CPU time.

.....DB2 Index Compression

- The CI Size of a compressed index on disk is always 4K
- A 4K expands into a 8K or 16K buffer, which is the DBA's choice. This choice determines the maximum compression ratio.
- Compression of key prefix and RID Lists
  - A Rid List describes all of the rows for a particular index key
  - An index with a high level of non-uniqueness, producing long Rid Lists, achieves about 1.4-to-1 compression
  - Compression of unique keys depends on prefix commonality

Do Not Ignore Statistics

- Needed only if you plan to run queries
- LOAD/REORG/REBUILD utilities with inline stats
  - Careful - Do not gather all stats you may need
- RUNSTATS
  - Use to gather stats for specific quer needs
  - Use Data Studio's Statistics Advisor to determine RUNSTATS control cards

Are Backups Necessary?

- Does data need to be recovable?
- Can it be reloaded with last refresh?
- Is warehouse data included in DR plan

Plan on Saving Performance Data

- DB2 traces
  - Accounting records (SMF 100)
    - Classes 1,2 and 3
    - If using packages, classees 7 and 8
  - Statistics records (SMF 101)
    - Classes 1, 3, 4, 5, and 6
  - Design a performance database
    - See Omegamon
- EXPLAIN outputs

Is a Test System Necessary

- Have test mimic production warehouse in as many ways as possible
- Statistics
- Cache definitions – Buffer pools, sort pool, etc…
  - Profiles
- Number and Speed of processors
  - DSNZPARM

DSNZPARM Keywords

- The two biggies..
  - DSN6SPRM
  - DSN6SYSP
- But don't ignore
  - DSN6FAC
  - DSN6GRP (if you are using data sharing)
- To a lesser degree
  - DSN6ARVP
  - DSN6LOGP
- Finally
  - DSN6SPRC

DSNZPARMs

- **CDSSRDEF** (DSN6SPRM) Update Yes
  - Set to 1 or ANY, 1 is the default
    - Can affect your distributed SQL
- **PARAMDEG** (DSN6SPRM) Update Yes
  - Controls the max degree of parallelism
  - Set to 0 if DB2 should decide, otherwise 1-254
    - Pick a reasonable value based on the number of CPs
- **PARA_EFF** (DSN6SPRM) Update yes
  - 0-100, 50 default – how is parallelism effected
    - 1 less parallelism, 99 more parallelism
- **SPRMPTH** (DSN6SPRC) Update No
  - Default 120, may want a higher value

MXQBCE

- With system parameter MAX_OPT_STOR (default 40MB)
  - DB2 will try to cap the storage usage for bind/prepare at the value set
  - Attempting to avoid SQLCODE -904 or storage abends
- When DB2 gets close to the threshold it will engage in "emergency clipping“
  - Choosing the lowest cost access path at that point and continue processing it
  - Discarding all others
  - If the threshold is set too low, then DB2 may have only completed small% of the prepare which may not be enough to really distinguish which is the best plan

MXQBCE …cont…

- Better to use system parameter MXQBCE instead
  - Controls number of join combinations considered by the DB2 Optimiser
  - Predictive rather than reactive
  - Attempts to clip access paths early, so as to avoid “emergency clipping”
  - Use the formula (2**n)-1
  - Default of (2**15)-1=32767 prepare will take no worse than a 15 table join
  - For a 14 table join it is 6383, 13 table 8191, 12 table 4095, 11 table 2047 and 10 table 1023
  - Do not go below 1023

DNSZPARM

- **CACHEDYN** (DSN6SPRM) Update Yes
  - Set to YES (default)

- **EDMSTMTC** (DSN6SPRM) Update Yes
  - Installation defined based on the amount of dynamic SQL expected and how long cache will gather information before wrapping

DSNZPARM

- **STARJOIN*** (DSN6SPRM) Update Yes
  - Turn on star join and how to handle fact table
  - DISABLE, ENABLE, 1, 2 to 3276

- **SJTABLES*** (DSN6SPRM) Update Yes
  - Only consider star join when number of table is great than this value

* Profile tables can also be used to manage these values

DSNZPARM

- MXDTCACH (DSN6SPRM) Update Yes
  - Data caching per thread
  - Default is 20, might be low, try 128
- SRTPOOL (DSN6SPRM) Update Yes
  - Maximum 128000K, Default is 10000K
  - Max out if possible but be careful, this value is per thread
- MAXRBLK (DSN6SPRM) Update Yes
  - Default: 400000, Possible values are: 0, 128 to 10000000
  - Don’t pick something too small because running out of space fails over to the work files
  - MAXTEMPS_RID needs to be set to a value
    - Do not use NONE or NOLIMIT

DSNZPARM

- CTHREAD, IDBACK, MAXDBAT, and CONDBAT, all on the DSN6SYSP macro – Update Yes
  - Manage local and distributed threads
  - CTHREAD = Maximum number of allied (local) threads that can be concurrently allocated
  - MAXDBAT = Maximum number of concurrent DBATs or connections if CMTSTAT=ACTIVE
  - CONDBAT = Maximum number of concurrent connections
  - CMTSTAT =ACTIVE or INACTIVE governs whether DBATs remain active across commits
- Highly recommended to set **CMTSTAT = INACTIVE**

DSNZPARM

- **IDTHTOIN** (DSN6FAC) Update Yes
  - Thread timeout value
  - Default 120
  - Is default long enough? 0 = never time out

- **TCPKPALV** (DSN6FAC) Update Yes
  - TCP/IP keep alive value
  - Default 120
  - Can enable, disable or set to a value
  - Coordinate with group that maintains TCP/IP

DSMAX

- **DSMAX** (DSN6SPRM) Update Yes
  - Default 20,000
  - Controls the maximum number of open datasets
  - If 99% of DSMAX reached
    - Async physical VSAM close of 3% of datasets
      - First CLOSE=YES on LRU scheme
      - Followed by CLOSE=NO on LRU scheme
  - Excessive dataset open/close activity will result in high DBM1 TCB Time
  - You need to be aware of the number of data sets you are dealing with the affects because of the use of percentages

CONTSTOR

- Drives thread storage contraction in DBM1 when CONTSTOR = YES
- Associated CPU overhead (typical < 1-2%)
- Design point is long running persistent threads with RELEASE(COMMIT)
- Compresses out part of Agent Local Non-System storage
- Does not compress
  - Agent Local System
  - Getmained Stack Storage
  - Local Dynamic Statement Cache
- Controlled by two hidden system parameters
  - SPRMSTH @ 1048576 (1MB)
  - SPRMCTH @ 10 (commits)
- Triggered at:
  - # Commits > SPRMCTH | (Agent Local Non-System > SPRMSTH & # Commits > 5)

MINSTOR

- With MINSTOR=NO (default), first fit algorithm is used
  - Fragmentation may happen with free space (leap frog effect)
- With MINSTOR=YES, best fit algorithm is used instead
  - Will go through all the chains across all the segments
  - Makes the storage denser
  - Observed CPU overhead < 1% for 3-4MB storage pools
  - Danger is that it masks storage leaks (makes them appear to go away)
  - It makes debugging storage leaks more difficult
  - Also degraded slower performance as the storage leak progresses
- Use MINSTOR=YES when
  - System is fully tuned and optimised for storage
  - You have determined there are no leaks
  - Out of other options and need the last ounce of storage

MGEXTSZ

- Added to enable Sliding Secondary Quantity for DB2 Managed Pageset where an explicit SECQTY value has been specified by the user and recorded in the DB2 Catalog
- Possible values: NO, YES
- Secondary extent allocations are to be optimised automatically by DB2 according to the respective sliding scale (Rule 4)
- DB2 will use the greater of the respective sliding scale and the secondary quantity in DB2 Catalog
- When allocating a new dataset for a pageset
  - DB2 uses SECQTY or DB2 calculated extent size instead of PRIQTY

DSNZPARM

- WFDBSEP (DSN6SPRM) Update No
  - Specify NO
    - DB2 **attempts** to direct declared global temporary table work to DB2-managed work file table spaces that are defined with a non-zero SECQTY.
    - DB2 **attempts** to direct sort work to DB2-managed work file table spaces that are defined with a zero SECQTY.
  - The operative word above is “Attempts”
    - If you specify YES, “**Attempts**” becomes “**Always**”

Buffer Pools

- Catalog/directory use BP0, BP8K0, BP16K0 and BP32K0
  - Do **NOT** use for any other purpose
- Minimum of 4 user BPs: user index (4K) and user data (4K) and work files (4K and 32K)
- Don’t be afraid to use 8K and 16K buffer pools
  - In many cases can improve rows per page
- Separate dimension tables from fact table
  - If dimension tables are not too large, may be able to pin in pool
  - Same for indexes on dimension tables

Buffer Pools

- Careful with the suggestions to Super Size your pools
  - Always make sure you never exceed real storage
- DB2 10 allocates pool space as needed
  - Be careful over allocating pool if you are using PGFIX = YES
- Consider taking advantage of 1 MB pages
- Requires PGFIX = YES

Buffer Pools

- DSNDB07 buffer pools
  - Need 4k and 32K work files
    - Lots of 32K, maybe more than 4k, monitor
  - VPSEQT = 98 for DSNDB07 pools
    - If sparse index is used, lower to 90-95
  - Go for LARGE pools if possible

- Work files
  - Many and smaller is better than few and large
  - For sort work files, always use zero secondary
  - If using DGTT, make sure you have a few with secondary greater than 0 (zero)

Shameless Self Promotion

Please Visit My DB2 for z/OS Blog

http://it.toolbox.com/blogs/db2zos/

Thank You For Attending!

SHARE
Technology · Connections · Results

Willie Favero

DB2 SME
Data Warehousing for System z Swat Team

IBM Silicon Valley Laboratory

My DB2 Blog
www.it.toolbox.com/blogs/db2zos/

http:www.WillieFavero.com

my TWITTER | my LINKEDIN | My slideshare profile | my FACEBOOK

SHARE supplied QR code for this session