IBM Software Group

IBM Data Management Technical Conference

Z37: ERP benefits in DB2 Version 8

Dr. Jim Teng (jteng@us.ibm.com)

DB2. Data Management Software

e business software

March 1-5, 2004

Disclaimer

- ✓ Some of the information in this presentation is preliminary. Any references to future plans are for planning purposes only, and IBM reserves the right to change those plans without any notice. Any reliance on such information by you is solely at your risk, and no commitment is made by IBM to provide additional information in the future.
- ✓ The materials in this presentation are subject to
  - ➡ Enhancements at some future date
  - ➡ a new release of DB2 or
  - ➡ a Programming Temporary Fix
- ✓ The information contained in this presentation has not been submitted to any formal IBM review and is distributed on an 'as is' basis without any warranty either expressed or implied.

DB2 V8 - The Biggest Release Ever

DBA

- Online schema change
- Backup and Recovery enhancements
- Automated space management
- Fast and automatic cached statements invalidation
- Transaction/end-user based accounting and workload management
- Enhancing RUNSTATS with DSTATS
- Long-running, non-committing readers detection
- Lock escalation reporting improvement
- Providing cached statement id in IFCID 124
- Improved LPL recovery
- DRDA and JCC tracing and diagnostics
- . . .

Continuous Operations

Performance

- Lifting virtual storage constraints
- Piece wise LOB insert
- More stage 1 predicates
- Index only access path for VARCHARs
- Fast retrieval of the most recent value
- Eliminating lock contention on special purpose tables
- Option to release locks at cursor close
- Allowing updating partitioning key column without partition locks
- Dissasociating clustering from partitioning key
- DPSI
- Up to 4096 partitions
- Data shraing improvements
- DRDA performance improvements
- . . .

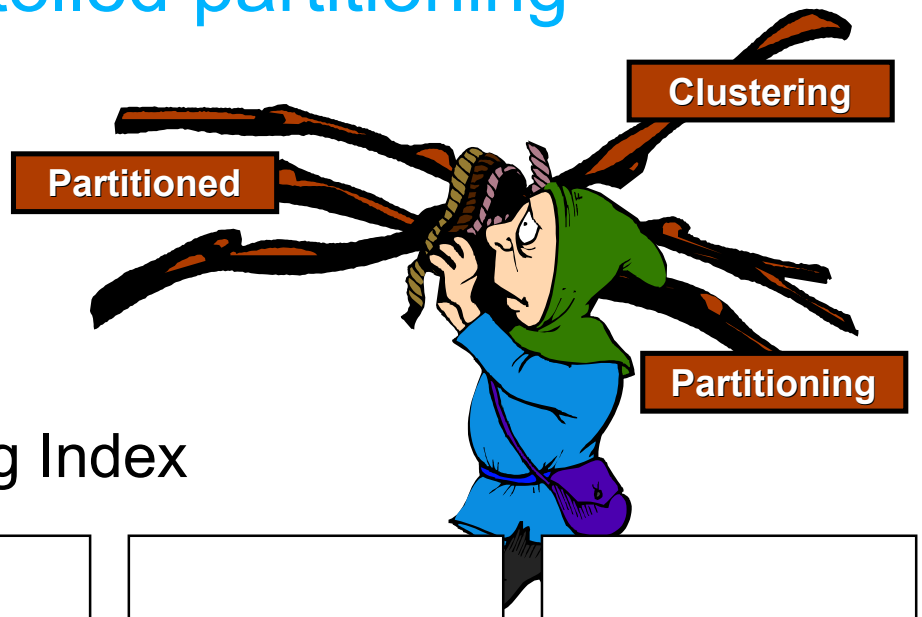
Scalability

SQL

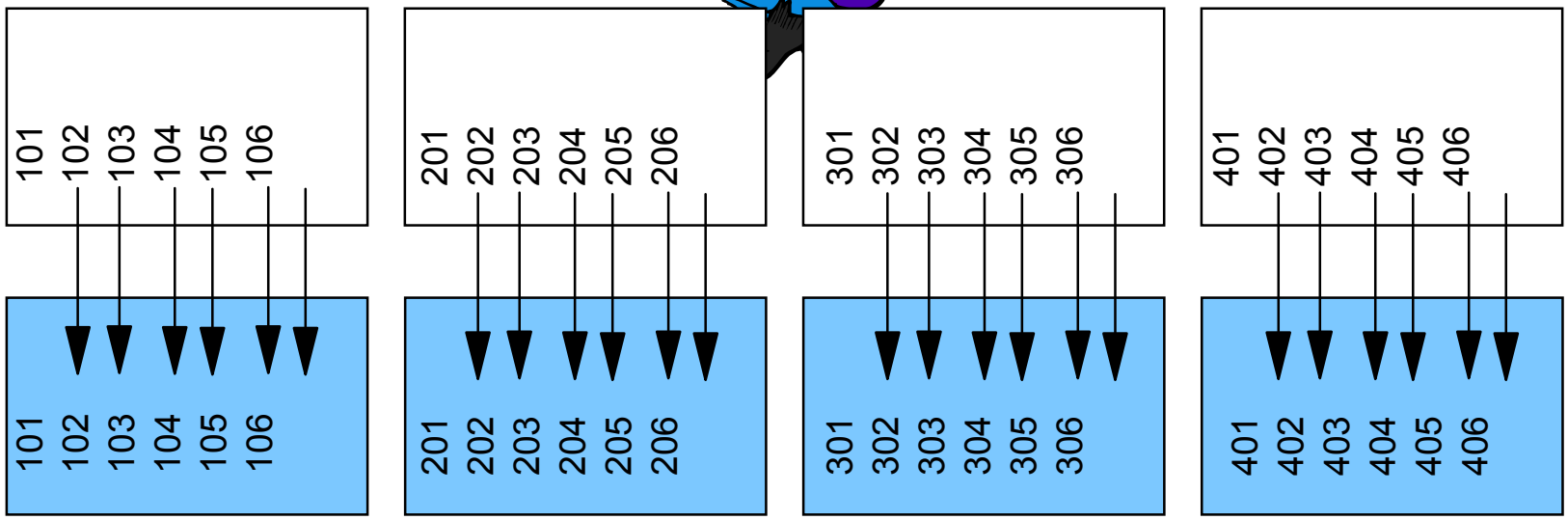
- Array inserts and fetches
- Sparse indexes
- Reducing negative impact of host variables at access path selection REOPT(ONCE)
- Transparent ROWID for tables containing LOBs
- Full Unicode support
- Lifting database object names length limits
- Up to 255 tables in FROM
- Materialized Query Tables (a.k.a. Automatic Summary Tables)
- Common Table Expressions
- Recursive SQL
- Multiple DISTINCTs
- DB2 Connect 64-bit client for Linux on zSeries
- Allowing comments in dynamic SQL
- . . .

Portability

V7 - Index-contolled partitioning



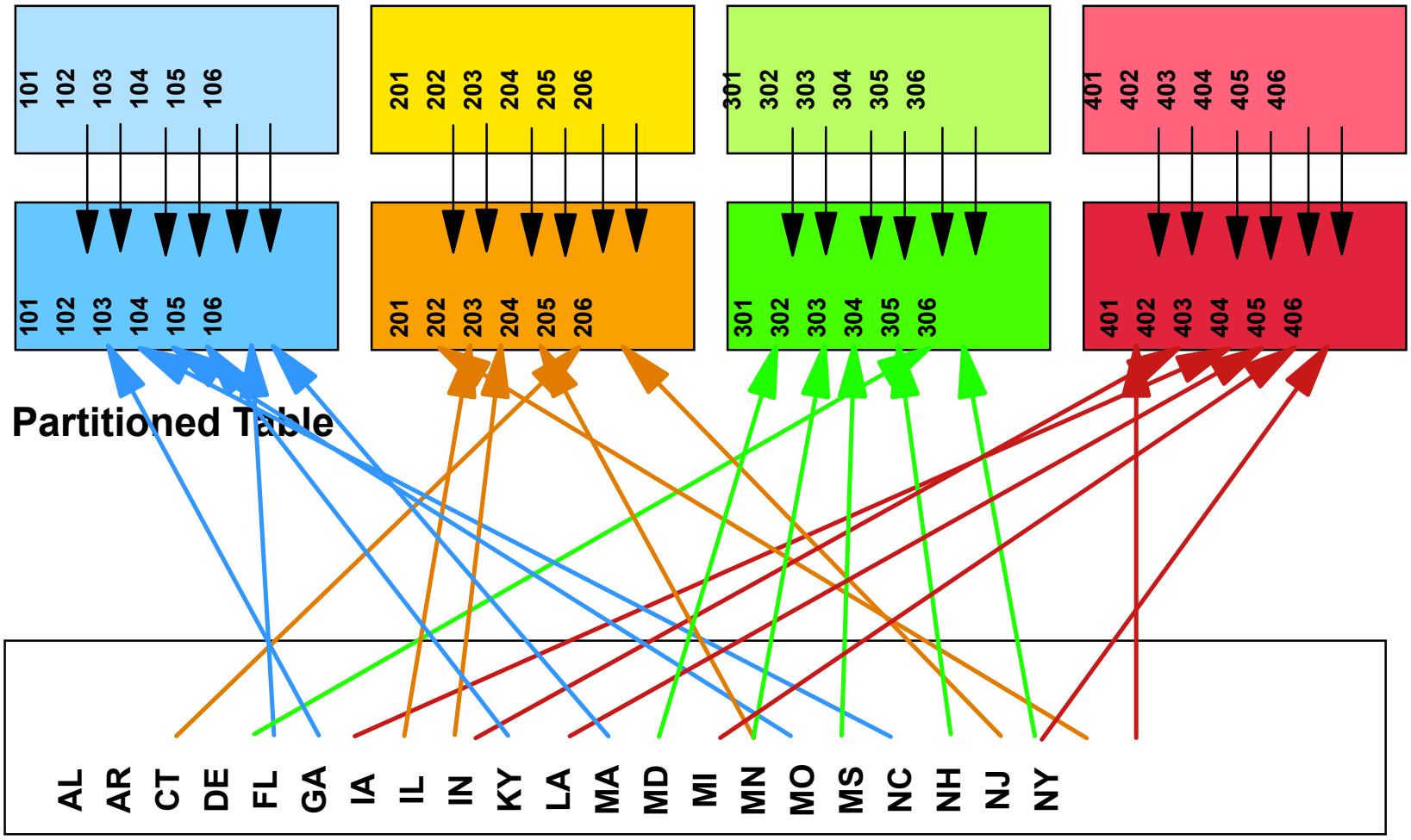
Partitioning Index



Partitioned Table

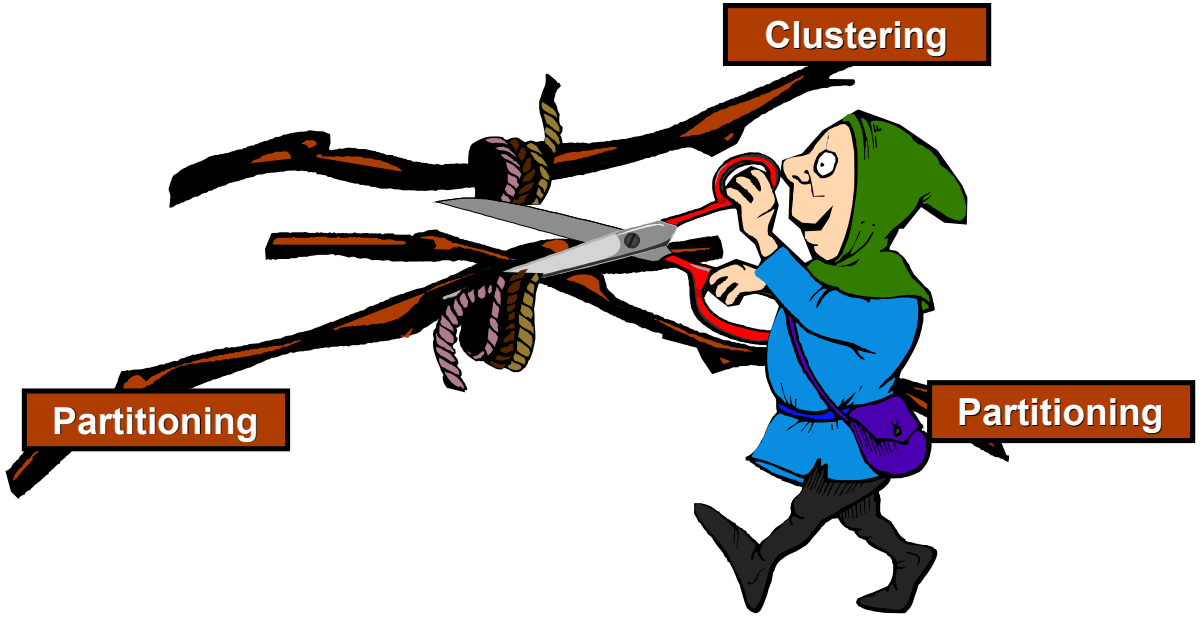
V7 PI and NPI/NPSI

Partitioning Index -- both logically and physically partitioned



NPI/NPSI -- logically partitioned

V8 - table-controlled partitioning



No indexes are required for partitioning!!

101	102	103	104	105	106
201	202	203	204	205	206
301	302	303	304	305	306
401	402	403	404	405	406

Partitioned table

Table-controlled partitioning

Problem

Columns (C1, C2) are ideal for partitioning table T but cannot be used because of any of the following:

- Clustering index (C3) on T improves performance of critical queries
- Adding an additional index (for partitioning reasons only) comes with significant overhead

Solution

Table-controlled partitioning; partitioning index is no longer needed

```
CREATE TABLESPACE TS NUMPARTS n

CREATE TABLE T (C1,C2,C3,...)
      PARTITION BY (C1,C2)
            PARTITION 1 ENDING
bnd11,bnd21)
            PARTITION n ENDING
bnd1n,bnd2n)

CREATE INDEX X ON T (C3)
      CLUSTER
```

To transform existing table, create a DPSI with DEFER YES and then drop it. Alternatively, just drop the partitioning index.

Converting to table controlled partitioning

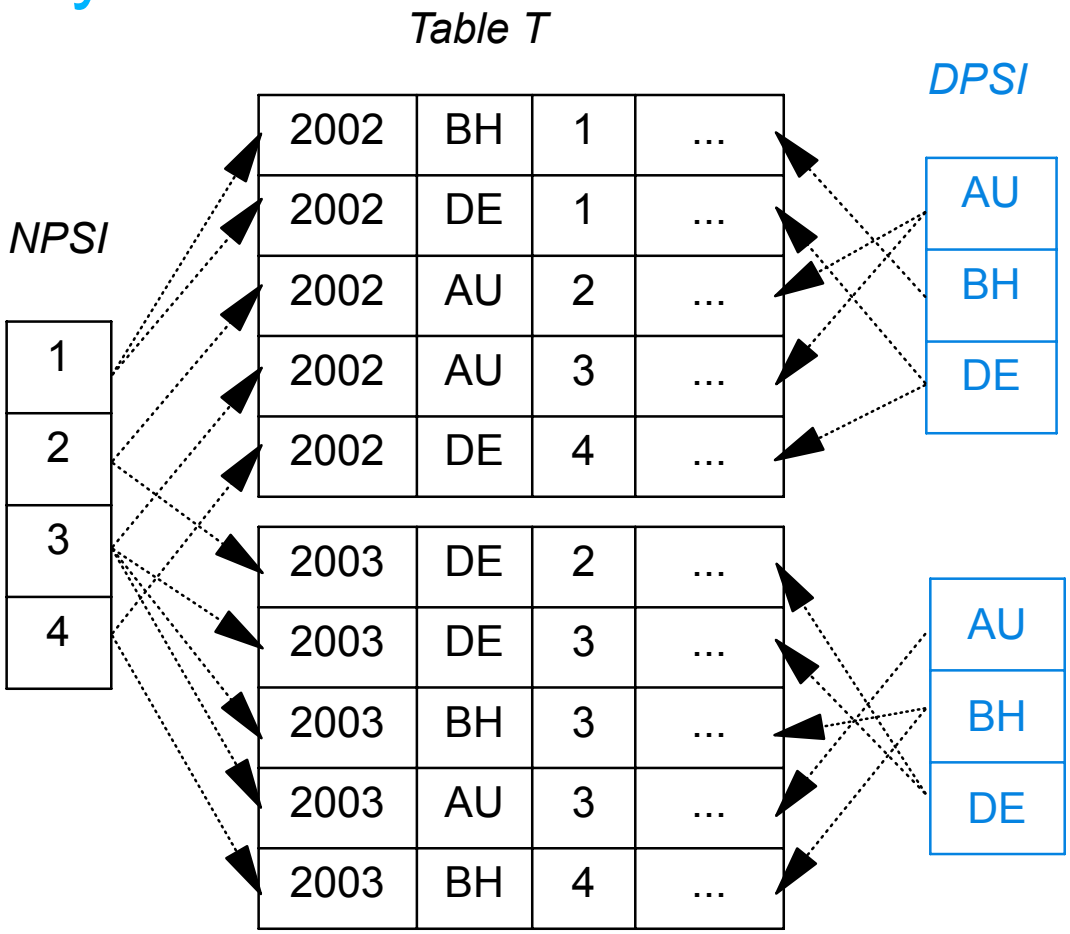
- Not necessary to drop and recreate the table
- Start with an existing table with a single PI
- Use any new function and a conversion is triggered from index-controlled to table-controlled partitioning:
  - Drop the partitioning index (partitioning by table)
  - ALTER partitioning index NOT CLUSTER
  - Create index PARTITIONED (DPSI or PI)
  - Add a partition
  - Rotate partitions
  - Create INDEX VALUES but no CLUSTER keyword
  - Add a partitioning key

Data Partitioned Secondary Indexes

```
CREATE TABLE T
      (YEAR,COUNTRY,TYPE,...)
      PARTITION BY YEAR
         PARTITION 1 ENDING 2003
         PARTITION 2 ENDING 2004

CREATE INDEX NPSI
      ON T (TYPE ASC)
      CLUSTER

CREATE INDEX DPSI
   ON T (COUNTRY ASC)
   PARTITIONED
```



Data Partitioned Secondary Indexes (DPSIs)

- A new V8 index type
  - Physical partitions like table
  - DPSI = physically partitioned secondary index
  - #parts(DPSI) = #parts(table)
  - Keys in part 'n' of DPSI refer only to rows in part 'n' of table
- 3 kinds of indexes now:
  - Partitioning Index (PI).
    - As today, except optional in V8 and may or may not be partitioned
  - New Data Partitioned Secondary Index (DPSI).
  - Non Partitioned Secondary Index (NPSI) As today's NPI

Data Partitioned Secondary Indexes (DPSIs)

- Benefits:
  - Full Partition Independence
  - Eliminate REORG BUILD2 phase
  - Eliminate LOAD PART contention
  - Parallel Utilities (REORG, LOAD, RECOVE)
  - Partition scope operations (add, rotate, reset)
  - Data affinity in data sharing
  - Potentially more efficient partitions pruning
- Potential impact to query performance
  - Many partitions to search if partition key is not specified
  - Not allowed for unique index

Updating Partitioning Columns

Problem

Column C1 is ideal for partitioning table T but cannot be practically used because it can be updated. Updating is allowed by setting zparm PARTKEYU to YES, but it comes with a heavy price in concurrency.

If updating C1 results in moving the updated row from one into another partition, DB2 blocks concurrent access to entire ranges of affected partitions and indexes.

Solution

If PARTKEYU is set to YES, no drain processing takes place and the concurrent activity continues without any negative effect.

Changing Clustering Index

Problem

Index X1 is the clustering index for table T, but the queries would be better performing if index X2 was the clustering index.
Changing the clustering index can be quite obtrusive:

- Drop index X1
- Create index X1 as non-clustering
- Drop index X2
- Create index X2 as clustering

If any index ensures uniqueness, then the access to table T needs to be stopped during the index recreation. In any case, indexes are unavailable during the entire procedure.

Solution

An index can be altered to specify clustering attribute

```
ALTER INDEX X1 NOT CLUSTER
ALTER INDEX X2 CLUSTER
```

After altering X1 it is still used as the implicit clustering index.
After altering X2 the rows are inserted according to the new clustering order.
The preexisting rows are reordered at the next reorganization.

Backup/Restore System

Problem

- 10000s of tablespaces and indexes per system make image copy based backup/recovery procedures very difficult to manage
- Existing volume-level backup requires suspending log writes during taking the backup – in the best case that‘s several seconds which is still too high for some customers
- Recovery to a prior point in time is complex, error prone and, if not optimized by usage of sophisticated procedures, very time consuming: many hours

Solution

- New utility BACKUP SYSTEM provides automated and entirely non-disruptive (no need to suspend log) volume-level backup procedure. It creates backups that guarantee recovery to any point in time.
- Recovery to the time when the backup was taken includes restoring the backup (by DFSMShsm) and starting DB2.
- Recovery to any other point in time is completely automated by the new RESTORE SYSTEM utility.
- Use in system cloning and DB2 Tracker

Large VSAM Control Interval Support

Problem

To address the exposure of creating inconsistent pages during FlashCopy, Split Mirror, GDPS FREEZE the following is enforced or should be observed:

- Taking image copies of 32K-page tablespaces if FlashCopy (and similar) is used when DB2 system is not stopped.
- No VSAM striping for >4K-page size objects.
- No concurrent copy for >4K-page tablespaces
- Taking image copies of >4K-page tablespaces to recover broken pages caused by a DB2 crash

Solution

In addition to 4K Control Interval (CI), DB2 now supports 8K, 16K and 32K CIs that match the supported DB2 tablespace page sizes. If the page size and CI match, the exposures listed on the left side do no apply.

Additionally, queries on such tablespaces are likely to result in shorter elapsed times.

Providing a zparm is left at its default all the new tablespaces are created with matching CI sizes. The existing tablespaces can be reassigned to matching CIs via reorganization

Recovering To Any RBA

Problem

Option ENDLRSN in CRESTART is not supported for non-data sharing . ENDRBA must be used instead and it must be a multiple of 4096. The log is then truncated to last logical log record written before that RBA.

Appart from being imprecise, this restriction prevents synchronizing log points across a federation of database systems. Namely, the federation members can be synchronized via coordinated suspension of their log writes, but DB2 cannot be recovered via conditional restart exactly to the point when the log was suspended

Solution

CRESTART is enhanced to support option ENDLRSN for non-data sharing as well, which accepts the RBA of an logical log record. Option SYSPITR is supported in the same way.
The log will be truncated at the end of the log record.

This enables conditional restart and RECOVER SYSTEM to bring the system to a consistent state at the time the log write was suspended. Consequently, DB2 can be recovered to the same point as the rest of the federated environment.

Automated Space Management

Problem

Maximum number of extents (255) gets reached before the data sets grow to their maximum size, resulting in sqlcode -904.

Selecting primary and secondary space allocation that would prevent this causes inefficient space utilization.

E.g. To ensure that a data set grows to 2GB, the allocation quantity should be 8.5MB which results in wasting DASD space for small data sets.

The maximum size of data sets in packaged applications is unpredictable.

Solution

No primary nor secondary allocation needs to be specified at create time. DB2 determines the optimal values based on a combination of the DSSIZE, LARGE and PIECESIZE specifications, and system parameters MGEXTSZ, TSQTY and IXQTY.

The secondary allocation is automatically increased using a sliding scale until 127 extents and a constant size thereafter to avoid exhausting the maximum number of extents per data sets.

Cached Statements Invalidation Enhancements

Problem

- RUNSTATS TABLESPACE tsname UPDATE NONE
  is the most efficient way of invalidating an entry in the global statements cache when no catalog statistics refresh is needed.
  However, even with the minimum level of details the RUNSTATS can consume lots of resources depending on the tablespace size.
- CREATE INDEX should ideally invalidate all the cached statements refering to the associated table. The behaviour depends on environment:
  - data sharing: statements invalidated if there are no active cached entries
  - non-data sharing: statements not invalidated.

Solution

- RUNSTATS TABLESPACE tsname UPDATE NONE REPORT NO
  is now allowed options combination. If specified, only the associated cached entries will be invalidated without actual collecting of the catalog statistics.
- CREATE INDEX invalidates associated cached entries irrespective of the environment.
  No wait for active entries.

Enhanced RUNSTATS

Problem

```
SELECT * FROM MLST
      WHERE MANDT = '005'
      AND PSPNR IN ('2','4','9','7','1')
```

There is no match on PSPNR and using an appropriate index would be optimal.
There is such an index: (MANDT, PSPNR) and all the statistics available:

- **colcardf(MANDT)=colcardf(PSPNR)=1**
- **freqval(MANDT) value '005' only**
- **freqval(MANDT ||PSPNR) value '005'||'0' only**

As optimizer does not use multi-column frequency for IN and range predicates, the index is ignored. Resulting tablespace scan takes 5.6 seconds to return no row.
Running DSTATS collects the freqval(PSPNR) and helps the optimizer to pick the index scan.

Solution

New RUNSTATS options, inheridited from DSTATS, are added to facilitate collecting:

- frequency value distribution for non-indexed columns or groups of columns
- cardinality values for groups of non-indexed columns
- LEAST and MOST frequently occurring values for both index and non-indexed column distributions

Reducing Host Variables Impact on Optimzer

Problem

When parameter markers are used for dynamic SQL, Optimizer cannot take advantage of columns values distribution. A simple example is:

```
PREPARE ...
 SELECT ... FROM T WHERE C1 = ?
```

For a low card(C1) this most often results in tablespace scan which is very bad if the cursor is opened with a non-existing value of C1.
The same negative effect exist whenever there is a skew in column values distribution.
DB2 V7 remedy, REOPT(VARS) is often too strong medicine with significant downsides in performance, concurrency and monitoring.

Solution

- Delay optimization until the first execute. At that time the parameter marker values are known and using them enables taking full advantage of their frequency distribution.
- Fix the access plan into the statements cache and take full advantage of caching.
- A bind option controls this behviour.
- Use the feature if the first set of values is more representative than the application agnostic set of default values which must be assumed by DB2 otherwise
- Explain is enhanced to show the actual access path details and not what they would be if the statement is reprepared with parameter markers

'Index-Always" Access Path

Problem

- **DB2 table is ‘abused‘ for storing non-relational, multi-row objects or it is volatile, i.e. its cardinality fluctuates (e.g. queue tables).**
- **Such tables need to be accessed in a specific pattern otherwise major performance and concurrency problems (deadlocks) are to be expected. That pattern (rule!) is basically using the index that provides most matching columns and avoid list prefetch regardless of the catalog statistics and other optimization factors.**
- **There are two approaches that come close to ensuring this access path:**
  - **Set zparm NPGTHRSH to -1: practically switches off cost based optimizer, therefore, this is strongly discouraged**
  - **Update catalog statistics which is unreliable and difficult to administer**

Solution

- New table attribute: VOLATILE
- Supported at CREATE and ALTER TABLE
- Main use for multi-row object tables and queues
- If specified, any statement accessing the table will use an index access path (if an index with *n* matching columns exist)
- SYSTABLES column SPLIT_ROWS is set to Y if the table is specified as VOLATILE
- Option to release locks at cursor close additionally helps reducing lock contention

Index-Only Access Path For VARCHARs

Problem

- Although index-only access is one of the most efficient access paths, it is not selected in many cases where varchar columns are involved in the query.
- All SAP character columns are defined as varchar.
- Padding varchars in the index prevents from retrieving the columns directly from the index.
- If enforced by zparm RETVLCFK, index-only access can retrieve incorrect output, therefore it is strongly discouraged.
- Typical example of incorrect output is:
  SELECT LENGTH(C1) ...
  and C1 is in an index. Let‘s say C1 is defined as VARCHAR(5) and the stored value is ‘A‘. The actual length is 1. Index-only access would return 5.

Solution

- No padding of variable length columns in indexes.
- The actual length is stored along the column value (like in the table)
- Expect major improvement for eligable SAP queries

Various DBA and Continuous Availability Enhancements

- Online schema change
  - Adding columns to index
  - Converting column types
  - Adding and rotating partitions
- SAP transaction/end-user based accounting and workload management
- Monitoring enhancements
  - Long-running, non-committing readers detection
  - Lock escalation reporting improvement
  - Providing cached statement id in IFCID 124
- DRDA and JCC tracing and diagnostics
  - Network statistics
  - DRDA ping
  - Flowing sqlseti information at statement boundary
  - Controlling trace outside JCC connection
  - Dedicated trace files for individual JCC connections

Various Performance and Scalability Enhancements

- Lifting virtual storage constraints
  - **64-bit addressing**
  - **Faster short prepares**
  - **Default contraction threshold (SPRMSTH) changed to 1MB**
- Array inserts and fetches
- Piece wise LOB insert
- IN list predicate access path performance enhancement
  - **PQ68662**
- Fast retrieval of the most recent value
- Up to 4096 partitions
- Improved LPL recovery
- Data sharing improvements
- DRDA performance improvements
- Sparse indexes
- More stage 1 predicates

Various SQL and Portability Enhancements

- Transparent ROWID for tables containing LOBs
- Unicode
  - Predicates
  - Keys
  - Unicode catalog and parser
  - Long SQL statements
- Lifting database object names length limits
- Up to 255 tables in FROM
- Materialized Query Tables (a.k.a. Automatic Summary Tables)
- Common Table Expressions
  - Recursive SQL
- Multiple DISTINCTs
- DB2 Connect 64-bit client for Linux on zSeries
- Allowing comments in dynamic SQL

References

- SG24-6871-00: DB2 UDB for z/OS V8 - Technical Preview
- Redbook: DB2 UDB for z/OS V8 Features That Benefit SAP
- ibm.com/software/d2zos