DB2 V8 Neat Enhancements that can Help You

Phil Gunning

September 25, 2008

DB2 V8 – Not NEW!

- General Availability March 2004
- DB2 V9.1 for z/OS announced March 2007
- Next release in the works and well along the way

Bufferpools

- Large bufferpools -- Up to 1TB per subsystem
- **<u>Bufferpool Enhancements: PGFIX(YES)</u>**:
- Version 8 introduces a new option to the **ALTER BUFFERPOOL command – PGFIX(YES).**
- By issuing the
- **–ALTER BUFFERPOOL(bpname) VPSIZE(vpsize) PGFIX(YES)**
  command, you will turn on long term page fixing for the particular Bufferpool. Page fixing bufferpools in memory can lead to significant cpu savings, with some customers seeing up to 8-10% cpu savings. Building on the savings that can be realized in V6 with the choice of page stealing algorithms (LRU vs FIFO), PGFIX(YES) puts the BP pages in memory and prevents the fixing and freeing instructions that need to be executed each time there is an I/O.
- One additional nice thing about PGFIX(YES), is that it is available *immediately* in Version 8 – i.e., in Compatibility Mode. You don’t have to wait for ENFM or NFM to make use of this feature and savings

Multi-Row Fetch

- New to Version 8 is DB2's ability to process multiple rows in a single FETCH statement. Prior to V8, a program had to issue multiple FETCH statements in order to read through a CURSOR's result set. Now, you have the ability to retrieve what is called a *rowset* – which is a predefined (on the FETCH statement) number of Rows. This function allows the application programming staff a much more robust SQL engine, and can lead to better performance and reduced network activity that comes from repeated trips between the program and the database engine. New syntax has been added to the DECLARE CURSOR syntax: WITH ROWSET POSITIONING- which will allow for multi-row FETCHING. Once the CURSOR is defined this way, issue:
- FETCH ROWSET FROM <cursor name> FOR 10 ROWS INTO …. ← this will return 10 rows (if available) at a time. Keep in mind, that you will need to have an array built for the expected result set.
- Note: Multi-Row processing is also available for INSERT processing as well.
- Available in New Function Mode (NFM)

Backward Index Scan

- Prior to Version 8, if you wanted to avoid a SORT by creating an index – and you wanted to be able to use both ORDER BY … ASC or ORDER BY … DESC, you would need to create two separate indexes. Version 8 introduces the ability of using the same index to do backward index scans. Backward index scans can not only allow you to drop some un-needed indexes, but it can dramatically improve performance by taking advantage of dynamic prefetching and all that those benefits entail.
- Available in Compatability Mode (CM)

Work File Changes

- Avoid Work File Creation
  - WHEN Clause is False
    - Trigger not invoked anyway, why create work file
    - Applies to both BEFORE and AFTER triggers
  - By using Buffers
    - Small Buffer created to stored Transition Variables and Table Rows
    - Avoids Work File completely when this is large enough

Volatile Tables

- Tables where cardinality varies significantly
- Static SQL affected by Statistics
  - Runstats when the table is near empty
    - Optimizer will chose table scans
  - Access Path is constant
  - Cardinality varies
  - Performance is inconsistent
  - Problems are sure to follow

Statistics for the Optimizer

- Distribution Statistics
  - Index Columns Only
- Non-Uniform Distribution Statistics
  - Leading Index Columns Only

Runstats Enhancements

- Distribution and Frequency Statistics
  - Any Column (Indexed or Not)
  - User-Defined Groups of Columns
  - Specified at the Table Level
- Cardinality for groups of columns
- Least Frequent as well as Most Frequent Values

Varying Length Index Keys

- Previous versions always padded variable length index keys to the full length
- V8 makes this padding an option
  - “Padded” and “Not Padded” expression of Create Index DML
- Index used for all access path choices
- Storage requirements are reduced

Padding

- No automatic conversion
- Alter Index to convert to “Not Padded”
- Default controlled through DSNZPARM
- PADIX on DSNTIPE panel at install

Conversion Issues

- Impact of Alter Conversion
- Alter Index Index_Name Not Padded
- Index placed in RBDP status
- Optimizer will NOT use the index
- Rebuild Index clears the status and use begins

Performance Impact

- Fewer Index pages and potentially fewer levels
- Therefore Fewer I/O for Index scans
- More CPU to pad for comparisons
- More CPU to find the start of the column
- YMMV, therefore test to determine the benefit

Mismatched Data types

- More common because of C/C++ and JAVA data types
- Performance Problem for previous versions
- V8 improves performance by making Stage 1 and possible indexable
  - "Column" operation "expression"
  - "Expression" operation "Column"
  - "Column" between "Expression" and "Expression"
  - "Column" in ("List")

Mismatched Data types

- Operation can be =, <, <=, >, >= or <>
- <> is made Stage 1, but can not be indexable
- "List" items must be
  - Constants
  - Host Variables
  - Special Registers
  - Session Variables
  - Parameter Markers
- List cannot be in a "When" clause of a trigger
- All items in list must be Stage 1 and Indexable

Numeric Mismatch

- EMP(NAME CHAR (20),
- SALARY DECIMAL (12,2),
- DEPTID CHAR (3));
- SELECT * FROM EMP

WHERESALARY > :HV_FLOAT;

–Stage 1 and indexable on column Salary in V8

–No Change to source code or schema

String Mismatch

- EMP(NAME CHAR (20),
- SALARY DECIMAL (12,2),
- DEPTID CHAR (3));
- SELECT * FROM EMP

WHERENAME = :HV_LENGTH_15;

  – Stage 1 and Indexable on column NAME
  – No Change to source code or schema

Summary

- DB2 V8 contains many new application related new features
- Implement post cutover after testing and refining
- Points to Remember
- No Coding Changes Required
- Significant Performance Improvements