



# Performance Enhancements for DB2 UDB for z/OS Version 8

Part Two

Tom Moulder

November 4, 2004



Performance Enhancements for DB2 UDB for z/OS Version 8

© Lightyear Consulting, Ltd. 2004

# Topics for Today

- Trigger Changes
- Runstats Changes
- Sort Changes
- Volatile Tables
- Multi-Row Operations

# 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

# Work File Changes

- Example Trigger

```
CREATE TRIGGER DOG_TRIG
AFTER INSERT ON DOGS
REFERENCING NEW AS NROW
FOR EACH ROW MODE DB2SQL
WHEN (NROW.NAME = 'FIDO' AND NROW.POUNDS =
      20)
INSERT INTO SPECIAL_PETS(COL1,COL2,COL3,COL4)
VALUES (0, 1, NROW.NAME, 'INSERTED FOR FIDO');
```

# Work File Changes

- **Example Inserts**

```
INSERT INTO DOGS(ID,NAME,POUNDS,C4,C5,C6,C7,C8,C9,C10)
VALUES (1, 'BRUISER', 10, '001',4, 2, 2, 4342, 'PURINA DOG CHOW',
'TOM')
```

```
INSERT INTO DOGS(ID,NAME,POUNDS,C4,C5,C6,C7,C8,C9,C10)
VALUES (2, 'DUKE', 9, '001', 4, 2, 2, 3023, 'KAL KAN', 'DICK')
```

```
INSERT INTO DOGS(ID,NAME,POUNDS,C4,C5,C6,C7,C8,C9,C10)
VALUES (3, 'FIDO', 12, '001', 4, 2, 2, 1000, 'KIBBLES', 'HARRY')
```

# Work File Changes

- No Trigger for Insert 1 and 2
  - No Work File at all, did not fire
- Trigger for Insert 3
  - Buffer accommodates
    - Transition Variables
    - All Three Rows
  - No Work File created

# Work File Changes

- Significant Performance Enhancement
  - Before
    - the work file would have been created three times
    - CPU, Memory and I/O resources needlessly used
  - V8
    - No Work File created at all
    - Small Buffer uses memory, no CPU or I/O
- No Coding Changes Required

# Statistics for the Optimizer

- **Distribution Statistics**
  - Index Columns Only
- **Non-Uniform Distribution Statistics**
  - Leading Index Columns Only
- **DSTATS**
  - Downloadable Offering for Previous Versions
  - [DSTATS Download Link](#)



# 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**

# New Keywords

- **COLGROUP**
- **MOST**
- **LEAST**
- **BOTH**
- **SORTNUM**
- **SORTDEVT**

# Sort Considerations

- SORTDEVT = DASD device type to use
- SORTNUM = the number of sort data sets
- Sizing Calculation
  - $2 * (\text{maximum record length of SYSCOLDISTATS} * \text{number of columns} * (\text{Frequent Values Count} + 2) * \text{number of indexes})$
  - Total Size for all sort data sets

# Example 1

```
RUNSTATS TABLESPACE DSN8D81A.DSN8S81E  
TABLE(DSN8810.EMP)  
COLGROUP(EDLEVEL,JOB,SALARY)
```

- Columns are not a part of the Index
- Cardinality Statistics stored in the Catalog
- Optimizer costs are more accurate

# Example 1

## SYSIBM.SYSCOLDIST

Name	Type	Numcolumns	Colgroupcolno	Cardf
Edlevel	C	3	00090008000C	33

## SYSIBM.SYSCOLDISTSTATS

Partition	Name	Type	Numcolumns	Colgroupcolno	Cardf
1	Edlevel	C	3	00090008000C	32
2	Edlevel	C	3	00090008000C	0
3	Edlevel	C	3	00090008000C	10
4	Edlevel	C	3	00090008000C	0
5	Edlevel	C	3	00090008000C	0

# Example 2

**RUNSTATS TABLESPACE DSN8D81A.DSN8S81E  
TABLE(DSN8810.EMP)**

**COLGROUP(EDLEVEL, JOB, SALARY)**

**FREQVAL COUNT 10 MOST**

- Add the 10 Most Frequent Values
- Stored in SYSIBM.SYSCOLDIST

# Example 2

## SYSIBM.SYSCOLDIST

Name	Type	Numcolumns	Colgroupcolno	Cardf	Colvalue	Frequencyf
Edlevel	C	3	00090008000C	33	Null	Null
Edlevel	F	3	00090008000C	10	Unprintable	95
Edlevel	F	3	00090008000C	10	Unprintable	47
Edlevel	F	3	00090008000C	10	Unprintable	24
Edlevel	F	3	00090008000C	10	Unprintable	24
Edlevel	F	3	00090008000C	10	Unprintable	24
Edlevel	F	3	00090008000C	10	Unprintable	24
Edlevel	F	3	00090008000C	10	Unprintable	24
Edlevel	F	3	00090008000C	10	Unprintable	24
Edlevel	F	3	00090008000C	10	Unprintable	24
Edlevel	F	3	00090008000C	10	Unprintable	24
Edlevel	F	3	00090008000C	10	Unprintable	24
Edlevel	F	3	00090008000C	10	Unprintable	24
Edlevel	F	3	00090008000C	10	Unprintable	24

# Example 2

- CARDF is meaningless when Type = 'F'
- FREQUENCYF is meaningless when Type = 'C'
- FREQUENCYF is a percentage of rows
- COLVALUE and COLGROUPOCOLNO are
  - VARCHAR and FOR BIT DATA
  - Sometimes unprintable

# Trade Offs?

- **Runstats Impact**
  - More CPU and Elapsed Time for execution
  - More Data in the Catalog Tables
- **Optimizer Impact**
  - More accurate data for costing formulas
  - Improves access path selection
    - Non-Uniform data distributions
    - Non-Index Columns as predicates
    - Non-Leading Index Columns as predicates

# Sort Changes

- V8 Introduces Cost Based Parallel Sort
  - OPTOPSE DSNZPARM
    - Default is ON
    - Disabled if
      - Sort data is < 2MB
      - Sort Data per parallel degree , 100KB
  - Elapsed time improvements
  - More use of Sort work files and Storage

# On or Off?

- **OPTOPSE**
  - OFF will work like V7
  - ON and optimizer will make a cost based choice
- **Plan\_Table** explains the optimizer choice
  - Examine the Parallel Group ID columns
    - SORTC\_PGR\_ID for composite table sorts
    - SORTN\_PGR\_ID for new table sorts

# Work Data Sets

- Affects Sort Performance
- Encourage Parallel Sorts by
  - Allocating more data sets in DB07
  - If necessary, make smaller to maintain same space usage
- Optimizer will create more runs and greater parallelism
- Decreased elapsed time

# 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

# SQL Specifications

- Create Table Volatile/Not Volatile Cardinality
- Volatile
  - Use an Index Whenever possible
- Not Volatile
  - Based on Statistics
  - The Default
- Cardinality
  - Not used, just for LUW compatibility

# SAP usage

- Cluster Tables

- Group of rows that must be processed together in the same sequence
- Matching and non-matching index access cause problems when executed concurrently
- VOLATILE will encourage use of matching index
- Locking and contention is reduced

# SAP Usage

- V7 used NPGTHRSH DSNZPARM
  - Subsystem wide affect
- VOLATILE is preferred in V8
  - Specified at the Table level
  - Encourages similar access for all table usage

# An Example

- Table Columns for Primary Key
  - First\_Name, Last\_Name, Sequence Number
  - Sequence\_Number is an Identity column

First_Name	Last_Name	Sequence_Number	City
Harry	Fox	1	New York
Harry	Fox	2	Los Angeles
Harry	Fox	3	Dallas
Mary	Lamb	1	Chicago
Mary	Lamb	2	San Francisco
John	Doe	1	Houston

# An Example

- Access Path Differences
  - Select \* uses a scan
  - Select Where First\_Name = “Harry” and Last\_Name = “Fox” uses primary index with List Prefetch
  - Deadlocks could occur

# An Example

- Changing to a VOLATILE table
  - Both Select statements use index access with no list Prefetch
  - Deadlock possibilities are reduced
  - Rebind required to change access path selection

# Alter Implications

- After “Alter Table ... Volatile”
  - Table is defined as Volatile no data is affected
  - Dependent Plans/Packages not invalidated
    - Column “VALID” contains an “A”
    - Rebind is recommended
    - Execution exceptions are –
      - List Prefetch, Hybrid Join and Multiple Index Access are disabled
  - Rebind required to change other access paths

# Multi-Row Operations

- Enabling This Option
  - “With Rowset Positioning”
  - New Block on Cursor
- Impacts
  - Local Unit of Works
  - Remote Unit of Works

# An Example

```
Declare Cursor_One Cursor
  With Rowset Positioning
  For
  Select Name,Dept,Title
  From Employees;
Open Cursor_One;
Fetch First Rowset from Cursor_One
  For 3 Rows
  Into :Name:NI,:Dept:DI,:Title:TI;
```

# COBOL Code

## 01 OUTPUT-VARS.

05 NAME OCCURS 3 TIMES.

49 NAME-LEN PIC S9(4) USAGE COMP.

49 NAME-TEXT PIC X(40).

05 DEPT OCCURS 3 TIMES.

49 DEPT-LEN PIC S9(4) USAGE COMP.

49 DEPT-TEXT PIC X(10).

05 TITLE OCCURS 3 TIMES.

49 TITLE-LEN PIC S9(4) USAGE COMP.

49 TITLE-TEXT PIC X(30).

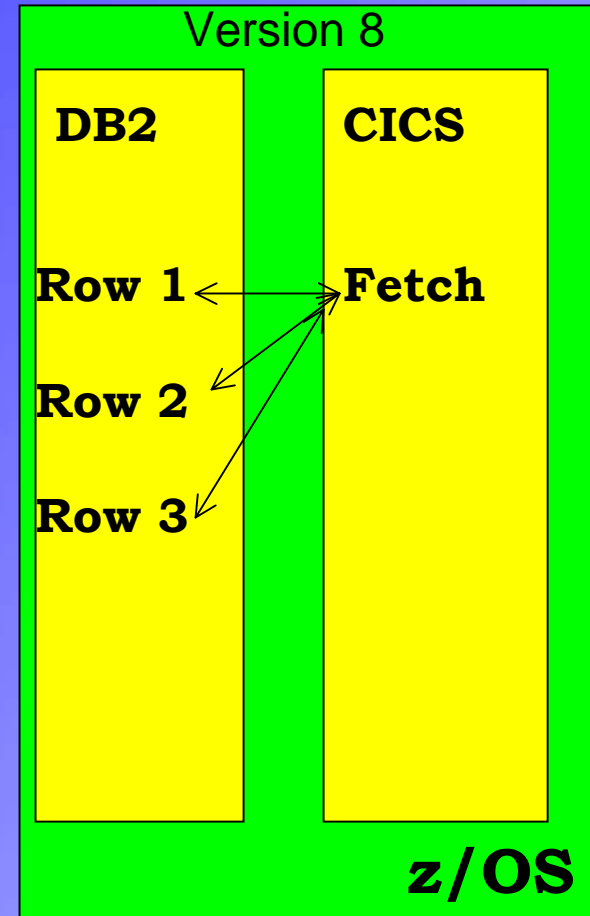
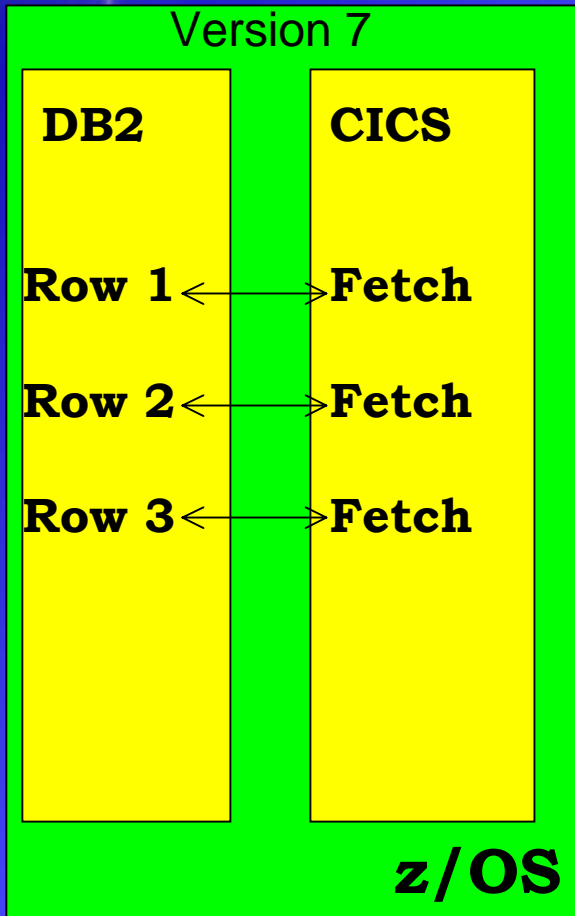
## 01 IND-VARS.

10 NI PIC S9(4) USAGE COMP OCCURS 3 TIMES.

10 DI PIC S9(4) USAGE COMP OCCURS 3 TIMES.

10 TI PIC S9(4) USAGE COMP OCCURS 3 TIMES.

# Local Connections



# Coding Changes

- New Error handling must be coded
- SQLCODE +100 has new meaning
- End of Result set could still return rows to process
  - SQLERRD(3) contains the number of rows returned by the fetch
  - “Get Diagnostics” will also return the same information

# Expectations

- One API trip as opposed to many
- CPU savings based on
  - Number of Rows Fetched
  - Number of Columns
  - Application Processing for each row
  - Up to 50% savings for Fetch
  - Up to 30% savings for Insert

# Insert Processing

- All or None processing Option by Default
  - Atomic Clause Defaults
- Savepoint created at the start of the insert
  - Insures undo of inserts if one fails
- Minimal Overhead

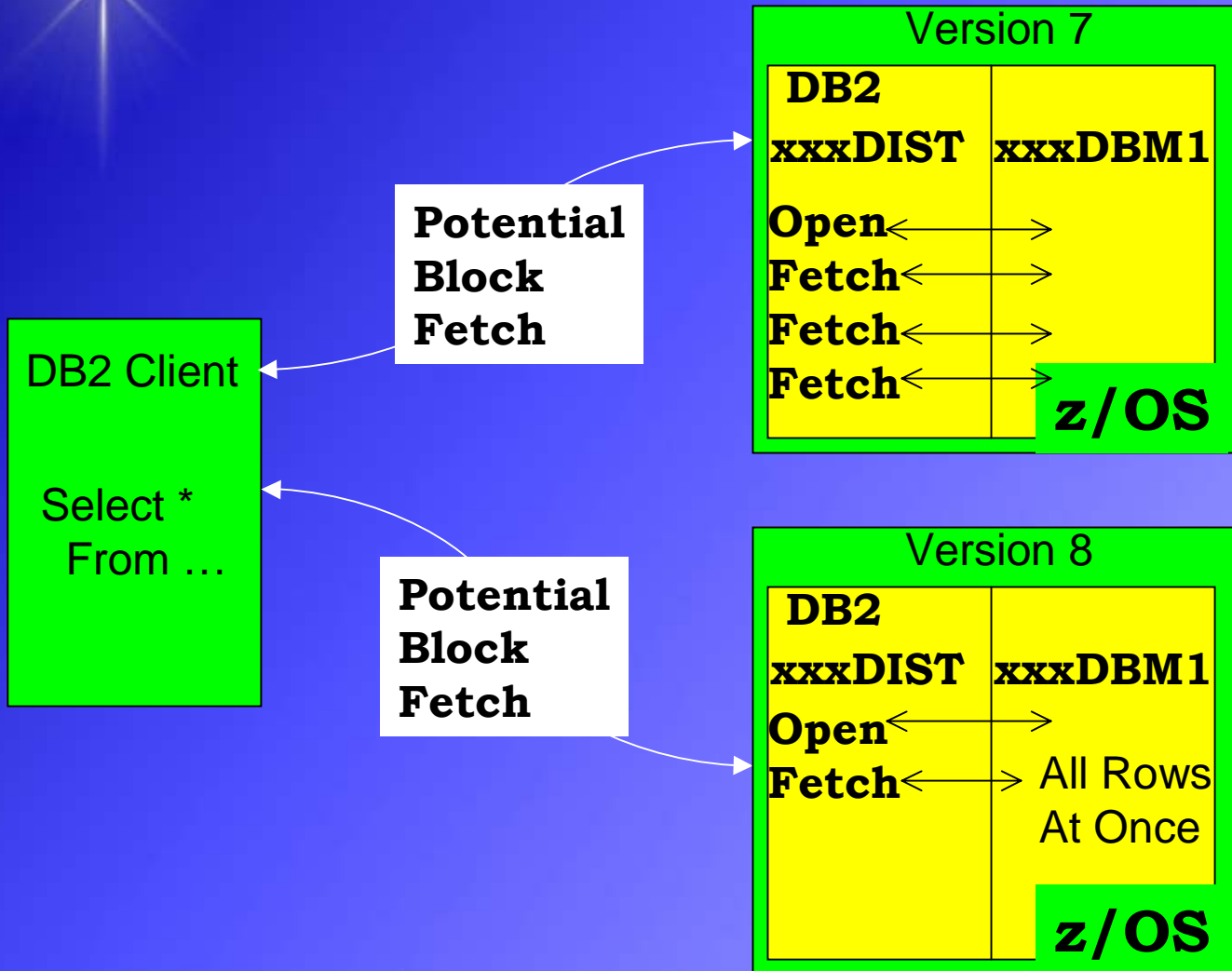
# Distributed Connections

- DB2 for z/OS to DB2 for z/OS
  - Effective when Block Fetch not possible
    - Fetch if it is not
      - Read Only or
      - Currentdata(No) and Ambiguous Cursor
    - Update or Delete with Cursor
    - Insert
  - Remember the locks you are holding
    - Commit Frequently

# Distributed Connections

- Client to DB2 for z/OS
  - Insert Processing
    - Currently all inserts bundled for network transmission
    - With V8, DB2 connect processes all inserts as Multi-Row
  - Savings
    - Eliminates the call to the DB2 API
      - Fetch, Insert, Update or Delete
    - Dynamic Scrollable Cursors
      - V8 Enables Multi-Row Fetch to save Network traffic

# DDF Usage



# Summary

- **Wow! A Lot of Information to Digest**
- **Points to Remember**
  - NCCR – No Coding Changes Required
  - Significant Performance Improvements
  - Many changes are easy to implement in V8
- **Questions Anyone**

# Next Steps with DB2 V8 and Lightyear

- Our series of detailed presentations on various DB2 V8 topics:

Pre-requisites	V7 to V8 Migration
Unicode	Utilities
Access Path Review	Highlights of New Functionality
Schema Evolution	In Depth Review of the "Top" 5 New Functions
Catalog Changes	Enhancements to SQL

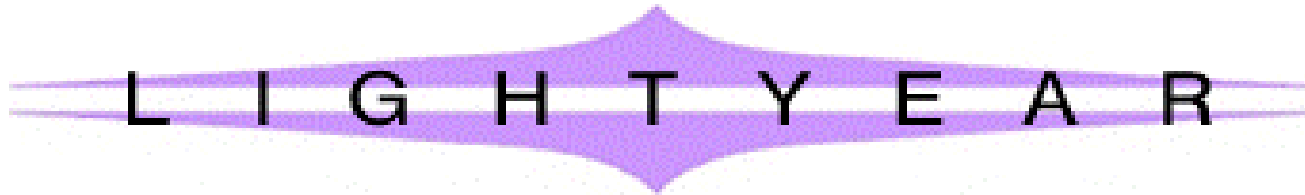
# Next Steps with DB2 V8 and Lightyear

- FREE presentations available NOW.
- These presentations will be given to just one customer at a time and will be tailored to that customer.
- Presentations will vary in length, dependent upon topic and tailoring.
- A 15 minute “prep” call will be required.

# Next Steps with DB2 V8 and Lightyear

- Access Path Review is a performance factor
- We offer a free on-site two-day review
  - Your systems
  - Your SQL
  - Predictive Report of potential problem SQL
- Watch our website for details on both of these exciting and FREE offers and register your interest.

# Services available from:



**Palo Alto – Fort Worth - Calgary - Laguna Beach**

**Scottsdale – Chicago – St. Louis – Boston**

**[www.lightyr.com](http://www.lightyr.com)**



- Database migration to *DB2*
- *VS/2* and *DL/2* software sales and related services (*sole North American distributor*)
- *CICS*, *DB2*, *IMS*, & *z/OS* software and tools, sales and upgrades
- Customized on-site technical seminars & education classes
- *WebSphere MQ* and application integration services
- *CICS* & *IMS* Web enabling design and implementation
- Database and online system performance analysis and tuning



# Questions ...

