IBM Software Group

PLAN Stability

Fen-Ling Lin

DB2 for z/OS Technical Conference
October 5-6, 2009
Taipei, Taiwan

ON DEMAND BUSINESS

Disclaimer

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 express or implied. The use of this information is a customer responsibility.

In addition, the material in this presentation may be subject to enhancements or Programming Temporary Fixes (PTFs) subsequent to general availability of the code.

The Problem

- Unpredictable, unstable performance of queries due to changes in access paths:
  - Affects Static SQL (REBIND/BIND REPLACE) and Dynamic SQL (PREPARE)
  - Regressions are typically seen when customers migrate to a new release.
  - Also seen during the lifetime of a release when applications are bound/rebound due to maintenance fixes, changes in data and/or schema, or changes in the applications.
- Customers have to do a lot of “prep work” before REBINDing to protect themselves.
- Pinpointing the exact source of such regressions can be hard. When regressions do happen, there are usually no quick fixes.

Why the Access Path Change?

- Estimated cost of multiple plans are closed
  - Easily to be affected by environment and other factors
- Optimizer estimate cost inaccurately
  - Unknown data skew
    - Distribution Stats (V8)
    - Histogram Stats (V9)
  - Lack of correlation statistics
    - Correlation Stats (V8)
  - Predicate estimation with host variables or parameter markers

Possible Solutions

- Ability to backup/restore access path
  - V9 Plan Stability
- Optimization Hints
  - Full Hints
  - CTE Hints
- Avoid conflicting access path choices
  - By design
  - Or, statistically

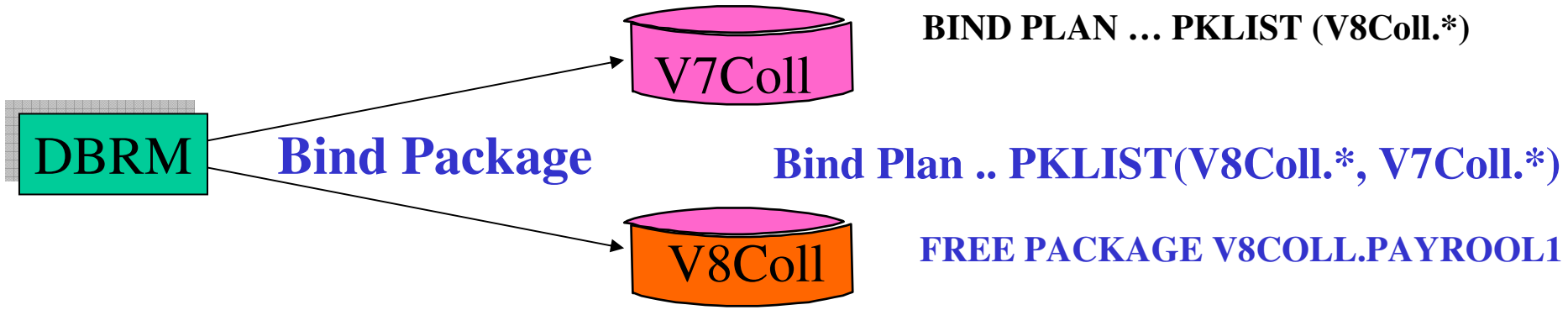
Access Path Backup

Provide an unprecedented level of stability of query performance achieved by stabilizing access paths:

- Dummy Collection (V8)
- Static SQL (V9)
  - Relief from REBIND regressions
  - Remove the fear of REBINDing!
- Dynamic SQL (Vx)
  - Remove the unpredictability of PREPARE
  - Extend Static SQL benefits to Dynamic SQL

V8 Plan Stability – Dummy Collection

- Bind package into a new collection ID



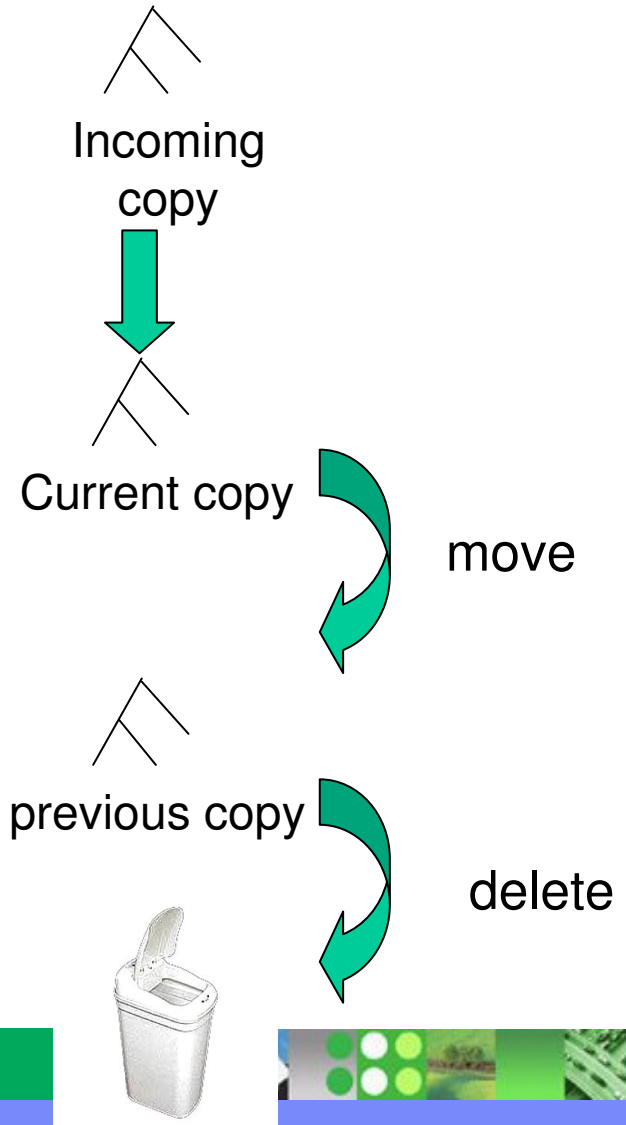
- BIND PLAN with new collection ID Added
- Free regressed packages from the new collection
  - To allow fallback to prior package

V9 Plan Stability

- Ability to backup your static SQL packages
- At REBIND
  - Save old copies of packages in Catalog/Directory
  - Switch back to previous or original version
- Two flavors
  - BASIC
    - 2 copies: Current and Previous
  - EXTENDED
    - 3 copies: Current, Previous, Original
  - Default controlled by a ZPARM (PLANMGMT)
  - Also supported as REBIND options

Plan Stability - BASIC support

REBIND … PLANMGMT(BASIC)



REBIND … SWITCH(PREVIOUS)

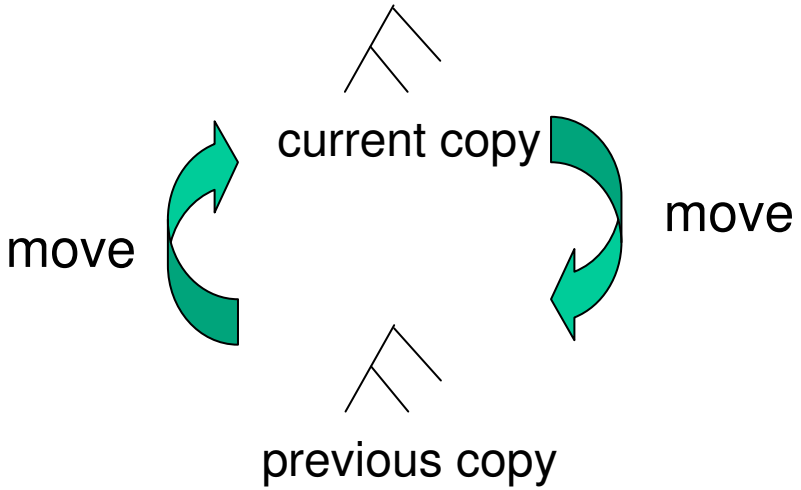
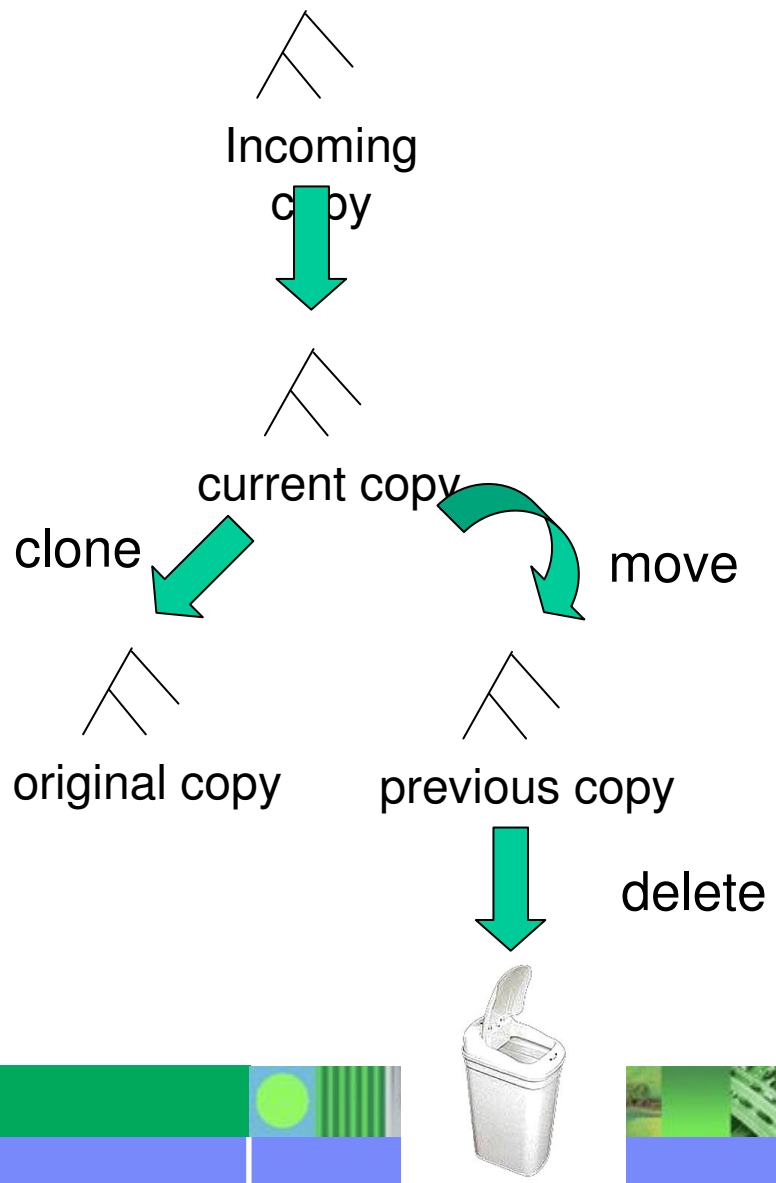


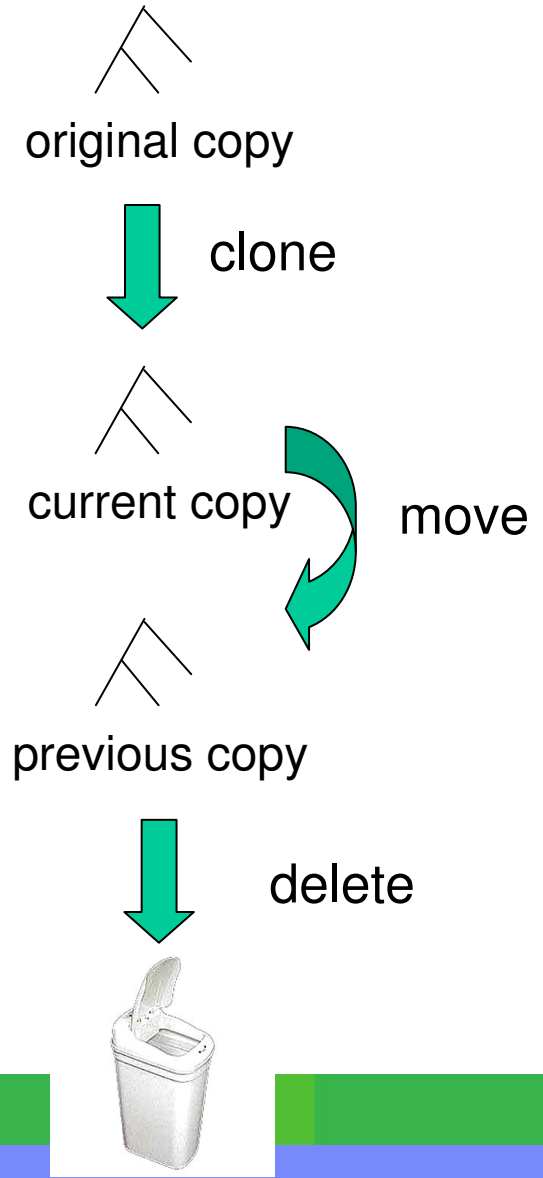
Chart is to be read from bottom to top

Plan Stability - EXTENDED support

REBIND … PLANMGMT(EXTENDED)



REBIND … SWITCH(ORIGINAL)



Plan Stability Notes

- REBIND PACKAGE …
  - PLANMGMT (BASIC)

  2 copies: Current and Previous
  - PLANMGMT (EXTENDED)

  3 copies: Current, Previous, Original

- REBIND PACKAGE …
  - SWITCH(PREVIOUS)

  Switch between current & previous
  - SWITCH(ORIGINAL)

  Switch between current & original

- Most bind options can be changed at REBIND
  - *But a few must be the same …*

- FREE PACKAGE …
  - PLANMGMTSCOPE(ALL) – Free package completely
  - PLANMGMTSCOPE(INACTIVE) – Free old copies

- Catalog support
  - SYSPACKAGE reflects active copy
  - SYSPACKDEP reflects dependencies of all copies
  - Other catalogs (SYSPKSYSTEM, …) reflect metadata for all copies

- Invalidation and Auto Bind
  - Each copy invalidated separately

Why Optimization Hints?

- Temporary fix to resolve immediate crisis
- Access path regresses from previously good path
  - Migrate to new release
  - RUNSTATS + REBIND
  - Environmental change (ridpool, bufferpool, zparm,…)
  - Maintenance upgrade
- Use OPTHINT to address known access path problem when other solutions aren't viable
  - For transactional SQL REOPT too expensive
  - Limitation of optimizer
  - Providing more accurate statistics not viable, or does not solve problem
- Lock in access path

Optimization Hints Overview

- Full Optimization Hints – (V6)
  - Uses PLAN_TABLE as INPUT
  - Allow user to specify desired access path to optimizer
  - Design Point
    - Support “Fallback” to previous access path
    - Experienced users can design their own access path
- Visual Plan Hints
- CTE Hints (V8)

Optimization Hints

- Set ZPARM to enable optimization hints
  - Specify YES in Optimization Hints field
  - Installation panel DSNTIP4
  - ZPARM can be changed on-line
- PLAN_TABLE must be migrated to at lease 49-column format

Static Example

QUERYNO	METHOD	TNAME	BIND_TIME	
100	0	EMP	2007-12-15	
100	1	PROJECT	2007-12-15	old
100	3		2007-12-15	
100	0	EMP	2008-12-16	
100	4	PROJECT	2008-12-16	new
100	0		2008-12-16	

Access Path changed from NLJ to Hybrid Join
performance degraded

Update Plan_Table

QUERYNO	METHOD	TNAME	BIND_TIME	OPTHINT
100	0	EMP	2006-12-15…	**MYHINT**
100	1	PROJECT	2006-12-15…	**MYHINT**
100	3		2006-12-15..	**MYHINT**
100	0	EMP	2007-12-16..	
100	4	PROJECT	2007-12-16	
100	3		2007-12-16..	

- **Update Plan_Table set OPTHINT = 'MYHINT'**
  WHERE QUERYNO = 100 and
  DATE(BIND_TIME) BETWEEN '2007-12-01' and '2002-12-02';

- Further qualify by PROGNAME, APPLNAME, COLLID, VERSION, …

Static Example

- Bind Package or Plan
- If using packages, bind at package level

```
REBIND PACKAGE (myloc.mycoll.mypkg)/PLAN
  EXPLAIN(YES)                          ← validate hint is used
  OPTHINT(MYHINT)
  VALIDATE (BIND)
```

*** BIND should have SQLCODE +394 to indicate Hint is used

Dynamic Example

- SET CURRENT OPTIMIZATION HINT = 'MYHNT';

```
EXPLAIN ALL FOR
  SELECT * FROM EMP, PROJECT
   WHERE ….
   QUERYNO 100;
```

Dynamic SQL uses Special Register

*** BIND should have SQLCODE +394 to indicate Hint is used

Validate Plan_Table

QUERYNO	METHOD	TNAME	BIND_TIME	OPTHINT	HINT_USED
100	0	EMP	2007-12-15	MYHINT	
100	1	PROJECT	2007-12-15	MYHINT	
100	3		2007-12-15	MYHINT	
100	0	EMP	2008-12-16		
100	4	PROJECT	2008-12-16		
100	3		2008-12-16		
100	0	EMP	2008-12-17		MYHINT
100	1	PROJECT	2008-12-17		MYHINT
100	3		2008-12-17		MYHINT

HINT IS USED

CTE Hints

- Available in V8
- Both Dynamic and Static Queries
- Enhance the Hint Usability
  - Attach the Hint with the SQL statement
  - Pass the plan hints into DB2 via a special Common Table Expression embedded within the statement
- Easy to use

CTE Hint

- Turn on OPTHINTS
- The name of CTE must be
  - **DSN_INLINE_OPT_HINT**
- The special CTE has to be the first CTE within the statement

- **WITH DSN_INLINE_OPT_HINT**
  (TABLE_NAME, ACCESS_TYPE, ACCESS_NAME, JOIN_SEQ) AS
  (VALUES (NULL, 'INDEX', NULL, NULL),
  ('T2', NULL, 'INDX1_T2', 1))
  SELECT * FROM T1, T2 WHERE T1.C= T2.C;
- **Host-variable not allowed**
- **SQLCODE +395 issued if hint accepted and DB2 will attempt to use it**

CTE Hint Attributes

Attributes	Comment
TABLE_CREATOR	Name of the table creator
TABLE_NAME	Table name in the query
CORRELATION_NAME	Qualifier to the TABLE_CREATOR and TABLE_NAME
ACCESS_TYPE	**RSCAN, INDEX, INLIST, MULTI_INDEX**
ACCESS_CREATOR	Creator of an INDEX
ACCESS_NAME	Name of an INDEX

CTE Hint Attributes

Attributes	Comment
JOIN_SEQ	Join sequence number
JOIN_METHOD	**NLJ, SMJ, HYBRID**
PARALLELISM_MODE	**CPU, IO, SYSPLEX**
ACCESS_DEGREE	Degree of Parallelism
JOIN_DEGREE	Degree of Parallelism
TABNO	Internal number assigned by DB2 to a table. In most cases, this attribute isn't needed
QBLOCKNO	Internal number assigned by DB2 to query block. In most cases, this attribute isn't needed

CTE Hint Example -1

- First row
  - Enforce to use index access on all tables if no other hint applies to a given table
- Second row
  - Enforce T2 to use index INX1_T2.
  - Force T2 as the leading table

```
WITH DSN_INLINE_OPT_HINT
     (TABLE_NAME, ACCESS_TYPE, ACCESS_NAME, JOIN_SEQ) AS
     (VALUES (NULL, 'INDEX', NULL, NULL),
             ('T2', NULL, 'INDX1_T2', 1))
     SELECT *  FROM T1, T2 WHERE T1.C= T2.C;
```

CTE Hint Example -2

- First row
  - Apply to all instances of T1
  - Enforce to use INX1_T1 for T1
  - Enforce Nested Loop Join
- Second row
  - Apply to all instances of T2
  - Enforce T2 to use index INX1_T2.
  - Force T2 as the leading table

```
WITH DSN_INLINE_OPT_HINT
    (TABLE_NAME,  ACCESS_NAME, JOIN_SEQ, JOIN_METHOD) AS
    (VALUES ('T1', 'INX1_T1', NULL, 'NLJ'),
                ('T2',  'INDX1_T2', 1, NULL))
    SELECT *  FROM T1, T2 WHERE T1.C= T2.C;
```

Hint Limitation

- SQLCODE = 000
  - Hint not found, not used
- SQLCODE +394
  - Hint found, used
  - Optimizer can add necessary sort
  - Optimizer might not be able to support the Hint.
    - Check Reason code
    - Cannot undo query transformation
    - Optimizer determine how index is used

Visual Plan Hints

- Manually altering PLAN_TABLE for hints is difficult
- GUI interface to generate Optimization Hint
  - Graphically alter the explain table
  - Part of Optimization Service Center