Control your own destiny with Optimization Hints

Patrick Bossman
IBM Silicon Valley Lab

December 13, 2006

Columbia, MD
Agenda

- Overview
- Reasons for using
- Environment setup
- Preparation
- Sample cases
- Verifying hint used
- Limitations
- Future work --> Visual plan hint
Optimization hints overview

• What are optimization hints?
  – Feature added in V6
  – Uses PLAN_TABLE as INPUT
  – Allow user to specify desired access path to optimizer
  – Design point - support "fallback" to previous access path
  – Experienced / daring users can design their own access path
Optimization hints overview

• When should optimization hints be used?
  – 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 not viable
    • For transactional SQL REOPT too expensive
    • Limitation of optimizer
    • Providing more accurate statistics not viable, or does not solve problem
Alternative uses

- User feedback on use of OPTHINTS...
  - **Lock in access path**
    - Stabilize desired static access path
  - **Excessive prepare time**
    - Repeatedly execute complex dynamic SQL
    - Known desirable access path
    - Prepare cost very expensive
      - Complex join can be several minutes
      - Significant CPU / memory consumption
    - Provide optimizer hint which is same path that it normally chooses
  - When successful, OPTHINT only considers the access path you've provided
    - Very streamlined prepare
Why optimization hints?

- Why optimization hints over other tuning methods?
  - **Hints directly address problem SQL statement**
    - Will not adversely affect other SQL statements
    - More stable than other query tricks
    - Perform better than other query tricks
  - **Avoid statistics seeding**
    - Can adversely affect other SQL statements
    - Difficult to maintain (RUNSTATS overrides)
  - **Avoid query tricks**
    - Eg. 0=1, 0<>0, CONCAT "," + 0, etc
    - Trick may work today, but query still costed
    - Optimizer can still choose alternate access path tomorrow
Tips to generate optimization hints

• Comfortable with process of statistics seeding?
  – Get the benefit of statistics seeding with less risk
    • Possibly regress other SQL
    • RUNSTATS replaces your statistic trick
    • Future cost change, trick may not work
  – If you're comfortable with statistics seeding...
  – Use statistics seeding to generate desired access path
    • Seed statistic
    • Generate desired access path
    • Convert access path to hint
    • Undo seeding --> other SQL no longer at risk
More tips

• Comfortable with query tricks?
  – Get the benefit of the trick (good path), with less risk, better performance
  – Use query trick to generate the hint
    • Code SQL with query trick
    • Generate desired access path
    • Convert to optimization hint
    • Undo query trick
  – Benefit of good access path
    • No "extra" predicates
    • Sargability (when predicate processed) not degraded
    • Risk of future regression reduced
Environment setup

- Setup for 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 LEAST 49-column format
    - Added columns OPTHINT, HINT_USED, PRIMARY_ACCESSTYPE
    - Really, migrate to most current format...
      - Tip you’ll eventually have UNICODE explain tables…
Crisis planning

• Question:
  – Would you wait until a site disaster occurs to develop an offsite recovery plan?
    • Hopefully not, because it’s too late at that point.
  – Don’t wait for a query crisis to learn how to implement optimization hints
    – Optimization hints are another tool in the toolbox. Being able to quickly implement them may save you in a crisis.
Planning for a crisis

• Planning for crisis
  – Store critical explain information
    • For use as input to future OPPTHINTS
    • Bind static applications with EXPLAIN(YES)
    • Store access path for desired (critical) dynamic SQL statements
  – What if you don't have "previously good" access path?
    • Do you know what you want?
      – Manually change plan_table
      – Use seed / trick methods to generate desired access path

• No hint, no idea about what good path was?
  – Back to tuning....
Using and validating hints

- **Static SQL**
  - Enable previous access path
  - Validate hint is used
  - Use QUERYNO clause to overcome changing statement number problem

- **Dynamic SQL**
  - No statement cache
  - Statement cache

- **Troubleshooting**
  - Most common mistakes
Static example

<table>
<thead>
<tr>
<th>QUERYNO</th>
<th>METHOD</th>
<th>TNAME</th>
<th>BIND_TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>0</td>
<td>EMP</td>
<td>2002-12-01-...</td>
</tr>
<tr>
<td>100</td>
<td>1</td>
<td>EMPROJACT</td>
<td>2002-12-01-...</td>
</tr>
<tr>
<td>100</td>
<td>3</td>
<td></td>
<td>2002-12-01-...</td>
</tr>
<tr>
<td>100</td>
<td>0</td>
<td>EMP</td>
<td>2003-06-01-...</td>
</tr>
<tr>
<td>100</td>
<td>4</td>
<td>EMPROJACT</td>
<td>2003-06-01-...</td>
</tr>
<tr>
<td>100</td>
<td>3</td>
<td></td>
<td>2003-06-01-...</td>
</tr>
</tbody>
</table>

- **Access path changed**
  - Previous good plan was nested loop join
  - New access path is hybrid
  - Runtime performance degraded
Update PLAN_TABLE

```sql
Update PLAN_TABLE
Set OPTHINT = 'MYHINT'
WHERE QUERYNO = 100
AND DATE(BIND_TIME) BETWEEN '2002-12-01' AND '2002-12-02'
```

- (Further qualify by PROGNAME, APPLNAME, COLLID, VERSION, etc)
Bind PACKAGE or PLAN

• If using packages, bind at package level:

REBIND PACKAGE (MYLOCATION.MYCOLLID.MYPACKAGE) -
EXPLAIN(YES) - <--- Want to validate hint used!
OPTHINT (MYHINT) -
VALIDATE(BIND)

• Otherwise bind at plan level:

BIND PLAN(MYPLAN) -
EXPLAIN(YES) - <--- Want to validate hint used!
OPTHINT(MYHINT) -
VALIDATE(BIND)

• *** Bind should have SQLCODE +394, Optimization hint used
Validate PLAN_TABLE

<table>
<thead>
<tr>
<th>QUERYNO</th>
<th>METHOD</th>
<th>TNAME</th>
<th>BIND_TIME</th>
<th>OPTHINT</th>
<th>HINT_USED</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>0</td>
<td>EMP</td>
<td>2002-12-01-...</td>
<td>MYHINT</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>1</td>
<td>EMPPROJACT</td>
<td>2002-12-01-...</td>
<td>MYHINT</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>3</td>
<td></td>
<td>2002-12-01-...</td>
<td>MYHINT</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>0</td>
<td>EMP</td>
<td>2003-06-01-...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>4</td>
<td>EMPPROJACT</td>
<td>2003-06-01-...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>3</td>
<td></td>
<td>2003-06-01-...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>0</td>
<td>EMP</td>
<td>2003-06-01-...</td>
<td>MYHINT</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>1</td>
<td>EMPPROJACT</td>
<td>2003-06-01-...</td>
<td>MYHINT</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>3</td>
<td></td>
<td>2003-06-01-...</td>
<td>MYHINT</td>
<td></td>
</tr>
</tbody>
</table>

Check access path settings
Verify HINT_USED column

Verify HINT_USED column
Troubleshooting Static

- **SQLCODE is critical**
  - SQLCODE = 000 means no hint found, used
    - Check PLAN_TABLE columns:
      - QUERYNO, APPLNAME, PROGNAME, VERSION, COLLID, OPTHINT same?
      - Updated OPTHINT column for ALL rows?
  - SQLCODE +394 means hint found, used
    - You should **STILL** validate explain output
      - Optimizer can add necessary sorts
      - Determine matching columns, multi-index path, etc.
      - Compare OPTHINT plan with HINT USED plan
Troubleshooting Static

• SQLCODE is critical (cont.)
  – SQLCODE +395 means hint found, not used
    • Look up +395 in Messages and Codes
    • Reason code identifies cause which disabled hint
      – Eg. Reason code 26: Table is missing
Troubleshooting Static

• SQLCODE is critical (cont.)
  – Multiple query block SQL statements
    • Optimization hints are used at a query block level
    • OPTHINT may be used in one query block, invalid in another
  – So what SQLCODE gets returned???
    • Priority: +395 → +394 → 000
    • If any query block finds hint / fails to use +395 is returned
    • If partial hint provided
      – Hint provided for only one QBLOCK out of many, hint used?
      – +394 provided
      – Plan_table will show which query block has hint used.
QUERYNO tip

• Use QUERYNO in static SQL to freeze the queryno
  – Without QUERYNO in static SQL
    • If program changes, STMTNO may change, also changes QUERYNO
  – Could result in OPTHINT no longer being found.
  – Use QUERYNO clause within the SQL to assign a specific QUERYNO which will not change even with application coding changes
    • STMTNO still changes, QUERYNO does not
Dynamic example

• Poorly performing SQL:

```sql
SELECT * FROM EMP E, EMPPROJACT EPA WHERE ... ;
```
Add QUERYNO clause and explain

EXPLAIN ALL FOR
SELECT *
FROM EMP E, EMPPROJECT EPA
WHERE ...
QUERYNO 712
;

Explain to get access path

Add queryno clause to map dynamic SQL to specific QUERYNO.
### Resulting explain

<table>
<thead>
<tr>
<th>QUERYNO</th>
<th>METHOD</th>
<th>TNAME</th>
<th>PREF</th>
<th>BIND.TIME</th>
<th>OPPTHINT</th>
</tr>
</thead>
<tbody>
<tr>
<td>712</td>
<td>0</td>
<td>EMP</td>
<td></td>
<td>2002-12-01...</td>
<td></td>
</tr>
<tr>
<td>712</td>
<td>4</td>
<td>EMPROJECT</td>
<td>L</td>
<td>2002-12-01...</td>
<td></td>
</tr>
<tr>
<td>712</td>
<td>3</td>
<td></td>
<td></td>
<td>2002-12-01...</td>
<td></td>
</tr>
</tbody>
</table>

- Notice bad join method
  - Compare to previous explain
  - Your analysis indicates hybrid is inefficient in this case
    - Poor performance
Update PLAN_TABLE

<table>
<thead>
<tr>
<th>QUERYNO</th>
<th>METHOD</th>
<th>TNAME</th>
<th>PREF</th>
<th>BIND_TIME</th>
<th>OPPTHINT</th>
</tr>
</thead>
<tbody>
<tr>
<td>712</td>
<td>0</td>
<td>EMP</td>
<td></td>
<td>2002-12-01…</td>
<td>DYNHINT</td>
</tr>
<tr>
<td>712</td>
<td>1</td>
<td>EMPPROJACT</td>
<td>L</td>
<td>2002-12-01…</td>
<td>DYNHINT</td>
</tr>
<tr>
<td>712</td>
<td>3</td>
<td></td>
<td></td>
<td>2002-12-01…</td>
<td>DYNHINT</td>
</tr>
</tbody>
</table>

TIPS:
1. Need to set OPPTHINT for ALL rows in query block, so use multiple updates!!!
2. Double check to ensure access path UPDATES to PLAN_TABLE update only intended rows.
Use explain to validate hint

SET CURRENT OPTIMIZATION HINT = 'DYNHINT';

EXPLAIN ALL FOR
SELECT *
FROM EMP E, EMPPROJACCT EPA
WHERE ...
QUERYNO 712;

Dynamic SQL uses Special register

First validation:
SQLCODE = +394, Optimization hint used?
Validate PLAN_TABLE

<table>
<thead>
<tr>
<th>QUERYNO</th>
<th>METHOD</th>
<th>TNAME</th>
<th>PREF</th>
<th>BIND_TIME</th>
<th>OPTHINT</th>
<th>HINT_USED</th>
</tr>
</thead>
<tbody>
<tr>
<td>712</td>
<td>0</td>
<td>EMP</td>
<td></td>
<td>2002-12-01…</td>
<td>DYNHINT</td>
<td></td>
</tr>
<tr>
<td>712</td>
<td>4</td>
<td>EMPPROJACT</td>
<td>L</td>
<td>2002-12-01…</td>
<td>DYNHINT</td>
<td></td>
</tr>
<tr>
<td>712</td>
<td>3</td>
<td>EMP</td>
<td></td>
<td>2002-12-01…</td>
<td>DYNHINT</td>
<td></td>
</tr>
<tr>
<td>712</td>
<td>0</td>
<td>EMP</td>
<td></td>
<td>2002-12-01…</td>
<td>DYNHINT</td>
<td></td>
</tr>
<tr>
<td>712</td>
<td>1</td>
<td>EMPPROJACT</td>
<td>L</td>
<td>2002-12-01…</td>
<td>DYNHINT</td>
<td></td>
</tr>
<tr>
<td>712</td>
<td>3</td>
<td>EMP</td>
<td></td>
<td>2002-12-01…</td>
<td>DYNHINT</td>
<td></td>
</tr>
</tbody>
</table>

Check access path Settings
Verify HINT_USED column
Hmmm, what about prefetch?

<table>
<thead>
<tr>
<th>QUERYNO</th>
<th>METHOD</th>
<th>TNAME</th>
<th>PREF</th>
<th>BIND_TIME</th>
<th>OPThINT</th>
<th>HINT_USED</th>
</tr>
</thead>
<tbody>
<tr>
<td>712</td>
<td>0</td>
<td>EMP</td>
<td></td>
<td>2002-12-01…</td>
<td>DYNHINT</td>
<td></td>
</tr>
<tr>
<td>712</td>
<td>4</td>
<td>EMPPROJACT</td>
<td>L</td>
<td>2002-12-01…</td>
<td>DYNHINT</td>
<td></td>
</tr>
<tr>
<td>712</td>
<td>3</td>
<td></td>
<td></td>
<td>2002-12-01…</td>
<td>DYNHINT</td>
<td></td>
</tr>
<tr>
<td>712</td>
<td>0</td>
<td>EMP</td>
<td></td>
<td>2002-12-01…</td>
<td>DYNHINT</td>
<td></td>
</tr>
<tr>
<td>712</td>
<td>1</td>
<td>EMPPROJACT</td>
<td>L</td>
<td>2002-12-01…</td>
<td>DYNHINT</td>
<td></td>
</tr>
<tr>
<td>712</td>
<td>3</td>
<td></td>
<td></td>
<td>2002-12-01…</td>
<td>DYNHINT</td>
<td></td>
</tr>
</tbody>
</table>

Hybrid always uses list prefetch, we changed from HYBRID to Nested Loop Join, but didn’t change the prefetch flag… (oops!)

Let’s be careful out there… (check prefetch, sort flags, etc)
Ready to execute

```sql
SET CURRENT OPTIMIZATION HINT = 'DYNHINT';

SELECT *
FROM EMP E, EMPPROJACCT EPA
WHERE ...
QUERYNO 712;
```

Final validation:
SQLCODE = +394, Optimization hint used.

Already used explain and plan_table to validate how the hint is used. For the truly paranoid, use PERFORMANCE TRACE CLASS(30) IFCID 22, 63 to see execution time access path.

If you forget QUERYNO
You’ll probably get SQLCODE 000, hint not found, not used
Dynamic SQL Troubleshooting

• Static trouble shooting still applies…
• Other typical dynamic problems
  – Forget to add QUERYNO ==> SQLCODE 000
    • Hint never found, not used
  – Generate / test hint with one application, use another to execute
    • Optimizer finds the hint by matching on ALL of these columns
      – PROGNAME, APPLNAME, COLLID, VERSION
      – QUERYNO, QBLOCKNO, OPTHINT
QMF considerations

- QMF uses different programs for EXPLAIN versus EXECUTE
  - **EXPLAIN:**
    - PROGNAME = DSQCESQL
  - **EXECUTE:**
    - PROGNAME = DSQCFSQL
  - Before execution, change PROGNAME to DSQCFSQL so optimizer will find the hint
  - See informational APAR: II13347
Dynamic statement cache (historical)

• BEFORE PQ89083
  1. Could use REOPT(VARS) to avoid use of statement cache
  OR
  2. Use circumvention to CACHE opthint
     • Statement text must not be cached
     • Change PLAN_TABLE so optimizer can find hint:
       – SET QUERYNO = 0
       – SET COLLID = ‘DSNDYNAMICSQLCACHE’
     • Access path determined by hint will be cached. Used by executions REGARDLESS of OPTHINT setting.

• Circumvention #2 is no longer necessary and will NO LONGER WORK after PQ89083!!!
Dynamic statement cache

- Post-PQ89083
  - OPTHINT is used as part of statement cache matching
    - Same SQL text can be cached multiple times, with different access paths
    - No hint / different hint name → considered different SQL
    - Eliminates need to flush statement cache
    - Allows greater flexibility
  - Repeated uses of same SQL with same hint
    - Initial prepare uses optimization hint, caches the SQL statement and hint
    - Repeated executions of same SQL text, same hint will obtain cache hit
Hint limitations

• Cannot undo query transformation
  – Subquery to join
  – More aggressive merge
    • Was an issue with V5 -> V6 with outer join
    • Transformation differences uncommon otherwise

• Optimizer determines how index used
  – One fetch, in-list, etc
  – Matchcols
  – Optimizer WILL honor single index prefetch setting
    • Blank, ‘S’, ‘L’
Hint limitations (cont.)

- Optimizer determines multi-index access operations
  - User can only indicate they want a multi-index access path, optimizer determines the operations
  - PK07550 (V8) – Optimizer will limit multi-index plan based on customer supplied indexes

- Order of merge join columns based on order coded in SQL statement

- **Ensure APAR PK07750/UK07760 is applied for V8**
  - Fixes several V8 optimization hint issues
  - Available since 10/2005
Visual plan hint

• GUI interface to generate optimization hint
• PLAN_TABLE update barrier to use
  – SQL to update PLAN_TABLE cumbersome
  – Minimize typographical errors
    • Eliminate typos
    • Minimize other mistakes
    • Forget to change plan number
  – Don't set OPTHINT for all rows
  – ....
Why visual plan hint?

• Visual plan hint improves situation
  – GUI interface easier to use
  – Focus on what should change, rather than on mechanics of change
  – Make verification of hint easy and mandatory
    • Avoid early declaration of success
  – Provide some basic consistency checking
    • Eg. Join method = 0 for outer table
  – Highlight differences in access path
    • Optimizer is still allowed to make changes
      – Eg. Matchcols, sort flags, multi-index access, etc.
Visual plan hint limitations

- Doesn't catch all problems
  - Can still pick an inefficient access path
  - GUI does not catch all illegal access paths
    - Parallelism limitations
    - List prefetch limitations (multi-index access)
- Allow easy, fast validation / compare
- Focus is on easier interface
Capabilities

• Start with existing access path
• Allow incremental changes to access path
• Implement and verify hint works

• Easiest to just take a tour...
Optimization Service Center
(brief overview)

• New tooling to replace Visual Explain
  – Includes Visual Explain capabilities
  – Includes much more
    • Visual Plan Hint
    • Query annotation
    • Statistics Advisor
    • Workload Statistics Advisor
    • Improved query reports
    • Textual Explain

• OSC is it’s own double session presentation (see you next year?)

• Take a tour... Of VPH within OSC
Launch Visual Plan Hint (VPH)
Visual Plan Hint
Focus on join graph

• Query block selection
• Show local predicates
• Join predicates
• Zoom in / out
Join graph
Query block selection
Query block selection

![Diagram showing query block selection with options to show local and join predicates.](image_url)

- Show Local Predicates
- Show Join Predicates

[Diagram: A hierarchical network of database schema elements, including 'SYSKEYS', 'SYSINDEXES', 'SYSTABLES', 'SYSTABLESPACE', and 'SYSDATABASE'.]
Query block selection
Query block selection

Join graph
Closer look at predicates
Focus on hint

• Default to current access path
• Modify join sequence
• Change access method
• Change join method
Visual Plan Hint

Hint area
Default join sequence
Default join sequence
Change join sequence
Join connectors removed
Move the tables around
Change access method

- Double click on table node to modify access method
Change access method

<table>
<thead>
<tr>
<th>Table Creator</th>
<th>SYSIBM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table Name</td>
<td>SYSTABLES</td>
</tr>
<tr>
<td>Correlation Name</td>
<td>B</td>
</tr>
<tr>
<td>Access Type</td>
<td>INDEX</td>
</tr>
<tr>
<td>Access Creator</td>
<td>SYSIBM</td>
</tr>
<tr>
<td>Access Name</td>
<td>DSNDTX01</td>
</tr>
<tr>
<td>Prefetch</td>
<td>NULL</td>
</tr>
<tr>
<td>PAGE_RANGE</td>
<td>NULL</td>
</tr>
<tr>
<td>Join Method</td>
<td>NULL</td>
</tr>
<tr>
<td>SORTN_JOIN</td>
<td>N</td>
</tr>
<tr>
<td>SORTC_JOIN</td>
<td>N</td>
</tr>
<tr>
<td>Parallelism Mode</td>
<td>NULL</td>
</tr>
<tr>
<td>Access Degree</td>
<td>NULL</td>
</tr>
<tr>
<td>Join Degree</td>
<td>NULL</td>
</tr>
<tr>
<td>PRIMARY_ACESSTYPE</td>
<td>NULL</td>
</tr>
<tr>
<td>WHEN_OPTIMIZE</td>
<td>NULL</td>
</tr>
</tbody>
</table>

- Leading table: checked
### Change access method

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table Creator</td>
<td>SYSIBM</td>
</tr>
<tr>
<td>Table Name</td>
<td>SYSTABLES</td>
</tr>
<tr>
<td>Correlation Name</td>
<td>B</td>
</tr>
<tr>
<td>Access Type</td>
<td>INDEX</td>
</tr>
<tr>
<td>Access Creator</td>
<td>SYSIBM</td>
</tr>
<tr>
<td>Access Name</td>
<td>DSNDTX01</td>
</tr>
<tr>
<td>Prefetch</td>
<td></td>
</tr>
<tr>
<td>PAGE_RANGE</td>
<td></td>
</tr>
<tr>
<td>Join Method</td>
<td>NULL</td>
</tr>
<tr>
<td>SORTN_JOIN</td>
<td>N</td>
</tr>
<tr>
<td>SORTC_JOIN</td>
<td>N</td>
</tr>
<tr>
<td>Parallelism Mode</td>
<td>NULL</td>
</tr>
<tr>
<td>Access Degree</td>
<td>NULL</td>
</tr>
<tr>
<td>Join Degree</td>
<td>NULL</td>
</tr>
<tr>
<td>PRIMARY_ACESSTYPE</td>
<td></td>
</tr>
<tr>
<td>WHEN_OPTIMIZE</td>
<td></td>
</tr>
<tr>
<td>Leading table</td>
<td>checked</td>
</tr>
</tbody>
</table>

Access Type Options:
- NULL
- INDEX
- RSCAN
- INLIST
- MULTI_INDEX

The selected option is INDEX.
Change access method

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table Creator</td>
<td>SYSIBM</td>
</tr>
<tr>
<td>Table Name</td>
<td>SYSTABLES</td>
</tr>
<tr>
<td>Correlation Name</td>
<td>B</td>
</tr>
<tr>
<td>Access Type</td>
<td>INDEX</td>
</tr>
<tr>
<td>Access Creator</td>
<td>SYSIBM</td>
</tr>
<tr>
<td>Access Name</td>
<td>DSNDTX01</td>
</tr>
<tr>
<td>Prefetch</td>
<td>NULL</td>
</tr>
<tr>
<td>PAGE_RANGE</td>
<td>NULL</td>
</tr>
<tr>
<td>Join Method</td>
<td>NULL</td>
</tr>
<tr>
<td>SORTN_JOIN</td>
<td>N</td>
</tr>
<tr>
<td>SORTC_JOIN</td>
<td>N</td>
</tr>
<tr>
<td>Parallelism Mode</td>
<td>NULL</td>
</tr>
<tr>
<td>Access Degree</td>
<td>NULL</td>
</tr>
<tr>
<td>Join Degree</td>
<td>NULL</td>
</tr>
<tr>
<td>PRIMARY_ACCESSTYPE</td>
<td>NULL</td>
</tr>
<tr>
<td>WHEN_OPTIMIZE</td>
<td>NULL</td>
</tr>
<tr>
<td>Leading table</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Change access method

- Double click on join node to modify join method
Change join method
Change join method

<table>
<thead>
<tr>
<th>Table Creator</th>
<th>SYSIBM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table Name</td>
<td>SYSDATABASE</td>
</tr>
<tr>
<td>Correlation Name</td>
<td>D</td>
</tr>
<tr>
<td>Join Method</td>
<td>NLJ</td>
</tr>
<tr>
<td>SORTN_JOIN</td>
<td>N</td>
</tr>
<tr>
<td>SORTC_JOIN</td>
<td>N</td>
</tr>
<tr>
<td>Join Degree</td>
<td>NULL</td>
</tr>
<tr>
<td></td>
<td>NULL</td>
</tr>
<tr>
<td></td>
<td>NLJ</td>
</tr>
<tr>
<td></td>
<td>SMJ</td>
</tr>
<tr>
<td></td>
<td>HYBRID</td>
</tr>
</tbody>
</table>

[Drop-down menu for Join Method with options: NULL, NULL, NLJ, SMJ, HYBRID]
Implement hint

- Validate hint
- Deploy hint
Validate hint
Validate hint
(will show screen shot)

<table>
<thead>
<tr>
<th>QUERYNO</th>
<th>QBLOCKNO</th>
<th>APPLNAME</th>
<th>PRONAME</th>
<th>PLANNO</th>
<th>METHOD</th>
<th>CREATOR</th>
<th>TNAME</th>
<th>TABNO</th>
<th>ACCESTYPE</th>
<th>MATCHCOLS</th>
<th>ACCESSCREATOR</th>
<th>ACCESSNAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>SYSLH200</td>
<td>1</td>
<td>0</td>
<td>SYSIBM</td>
<td>SYSTABLES</td>
<td>1</td>
<td>I</td>
<td>I</td>
<td>2</td>
<td>SYSIBM</td>
<td>DSNDTX01</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>SYSLH200</td>
<td>2</td>
<td>1</td>
<td>SYSIBM</td>
<td>SYSTABLESPACE 2</td>
<td>I</td>
<td>2</td>
<td>SYSIBM</td>
<td>DSNSX01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>SYSLH200</td>
<td>3</td>
<td>1</td>
<td>SYSIBM</td>
<td>SYSINDEXES</td>
<td>3</td>
<td>I</td>
<td>2</td>
<td>SYSIBM</td>
<td>DSNDXX03</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>SYSLH200</td>
<td>4</td>
<td>1</td>
<td>SYSIBM</td>
<td>SYSKEYS</td>
<td>4</td>
<td>I</td>
<td>2</td>
<td>SYSIBM</td>
<td>DSNDXX01</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>SYSLH200</td>
<td>5</td>
<td>1</td>
<td>SYSIBM</td>
<td>SYSDATABASE</td>
<td>5</td>
<td>I</td>
<td>1</td>
<td>SYSIBM</td>
<td>DSNDDH01</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>QUERYNO</th>
<th>QBLOCKNO</th>
<th>APPLNAME</th>
<th>PRONAME</th>
<th>PLANNO</th>
<th>METHOD</th>
<th>CREATOR</th>
<th>TNAME</th>
<th>TABNO</th>
<th>ACCESTYPE</th>
<th>MATCHCOLS</th>
<th>ACCESSCREATOR</th>
<th>ACCESSNAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>15462</td>
<td>1</td>
<td>SYSLH200</td>
<td>1</td>
<td>0</td>
<td>SYSIBM</td>
<td>SYSTABLES</td>
<td>1</td>
<td>I</td>
<td>0</td>
<td>SYSIBM</td>
<td>DSNDTX02</td>
<td></td>
</tr>
<tr>
<td>15462</td>
<td>1</td>
<td>SYSLH200</td>
<td>2</td>
<td>1</td>
<td>SYSIBM</td>
<td>SYSDATABASE</td>
<td>5</td>
<td>I</td>
<td>1</td>
<td>SYSIBM</td>
<td>DSNDDH01</td>
<td></td>
</tr>
<tr>
<td>15462</td>
<td>1</td>
<td>SYSLH200</td>
<td>3</td>
<td>1</td>
<td>SYSIBM</td>
<td>SYSTABLESPACE 2</td>
<td>I</td>
<td>2</td>
<td>SYSIBM</td>
<td>DSNSX01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15462</td>
<td>1</td>
<td>SYSLH200</td>
<td>4</td>
<td>1</td>
<td>SYSIBM</td>
<td>SYSINDEXES</td>
<td>3</td>
<td>I</td>
<td>2</td>
<td>SYSIBM</td>
<td>DSNDXX03</td>
<td></td>
</tr>
<tr>
<td>15462</td>
<td>1</td>
<td>SYSLH200</td>
<td>5</td>
<td>1</td>
<td>SYSIBM</td>
<td>SYSKEYS</td>
<td>4</td>
<td>I</td>
<td>2</td>
<td>SYSIBM</td>
<td>DSNDXX01</td>
<td></td>
</tr>
</tbody>
</table>
Deploy hint

Input query level settings
Hint implemented
Summary

• Purpose and preparation
• Implementing optimization hints
  – Static, dynamic, special cases
• Validating hint used
• Common pitfalls
• Limitations
• Coming soon in OSC - Visual Plan Hint
Thank you for attending!!!

Control your own destiny with optimization hints

Patrick Bossman
E-mail: bossman@us.ibm.com