



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_ACESSTYPE
 - 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 OPTHINTS
 - 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

QUERYNO	METHOD	TNAME	BIND_TIME
100	0	EMP	2002-12-01-...
100	1	EMPPROJECT	2002-12-01-...
100	3		2002-12-01-...
100	0	EMP	2003-06-01-...
100	4	EMPPROJECT	2003-06-01-...
100	3		2003-06-01-...

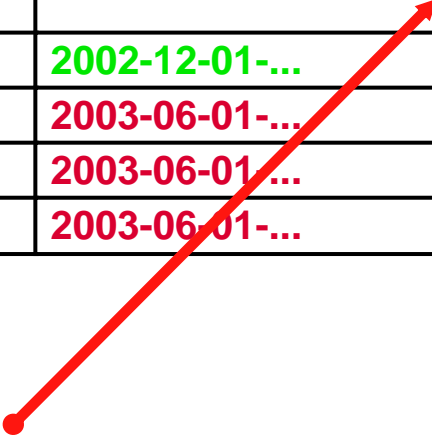
old good path

new bad path

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

Update PLAN_TABLE

QUERYNO	METHOD	TNAME	BIND_TIME	OPTHINT
100	0	EMP	2002-12-01-...	MYHINT
100	1	EMPPROJECT	2002-12-01-...	MYHINT
100	3		2002-12-01-...	MYHINT
100	0	EMP	2003-06-01-...	
100	4	EMPPROJECT	2003-06-01-...	
100	3		2003-06-01-...	



Update PLAN_TABLE

Set OPTHINT = 'MYHINT'

WHERE QUERYNO = 100

AND DATE(BIND_TIME) BETWEEN '2002-12-01' AND '2002-12-02'

- (Further qualify by PROGRAMNAME, 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

QUERYNO	METHOD	TNAME	BIND_TIME	OPHINT	HINT_USED
100	0	EMP	2002-12-01-...	MYHINT	
100	1	EMPPROJECT	2002-12-01-...	MYHINT	
100	3		2002-12-01-...	MYHINT	
100	0	EMP	2003-06-01-...		
100	4	EMPPROJECT	2003-06-01-...		
100	3		2003-06-01-...		
100	0	EMP	2003-06-01-...		MYHINT
100	1	EMPPROJECT	2003-06-01-...		MYHINT
100	3		2003-06-01-...		MYHINT

Check access path settings



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:

```
SELECT *  
FROM EMP E, EMPPROJECT EPA  
WHERE ...  
;
```

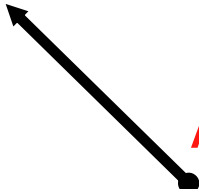
Add QUERYNO clause and explain

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

Explain to get
access path

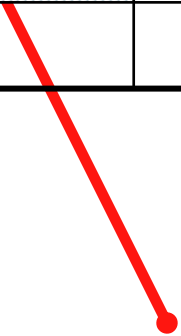


Add queryno clause to map
dynamic SQL to specific
QUERYNO.



Resulting explain


QUERYNO	METHOD	TNAME	PREF	BIND_TIME	OPTHINT
712	0	EMP		2002-12-01...	
712	4	EMPPROJECT	L	2002-12-01...	
712	3			2002-12-01...	



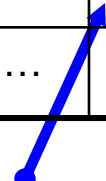
- Notice bad join method
 - Compare to previous explain
 - Your analysis indicates hybrid is inefficient in this case
 - Poor performance

Update PLAN_TABLE

QUERYNO	METHOD	TNAME	PREF	BIND_TIME	OPTHINT
712	0	EMP		2002-12-01...	DYNHINT
712	1	EMPPROJECT	L	2002-12-01...	DYNHINT
712	3			2002-12-01...	DYNHINT



**UPDATE PLAN_TABLE
SET METHOD = 1
WHERE TNAME = 'EMPPROJECT';**



**UPDATE PLAN_TABLE
SET OPTHINT = 'DYNHINT'
WHERE QUERYNO = 712;**

TIPS:

1. Need to set OPTHINT 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

QUERYNO	METHOD	TNAME	PREF	BIND_TIME	OPTHINT	HINT_USED
712	0	EMP		2002-12-01...	DYNHINT	
712	4	EMPPROJACT	L	2002-12-01...	DYNHINT	
712	3			2002-12-01...	DYNHINT	
712	0	EMP		2002-12-01...		DYNHINT
712	1	EMPPROJACT	L	2002-12-01...		DYNHINT
712	3			2002-12-01...		DYNHINT



Check access path
Settings



Verify HINT_USED column

Hmmm, what about prefetch?

QUERYNO	METHOD	TNAME	PREF	BIND_TIME	OPTHINT	HINT_USED
712	0	EMP		2002-12-01...	DYNHINT	
712	4	EMPPROJACT	L	2002-12-01...	DYNHINT	
712	3			2002-12-01...	DYNHINT	
712	0	EMP		2002-12-01...		DYNHINT
712	1	EMPPROJACT	L	2002-12-01...		DYNHINT
712	3			2002-12-01...		DYNHINT

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

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

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

If you forget QUERYNO
You'll probably get
SQLCODE 000, hint
not found, not used



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

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:
 - PROGRAMNAME = DSQCESQL
 - EXECUTE:
 - PROGRAMNAME = DSQCFSQL
 - Before execution, change PROGRAMNAME 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 ophint
 - 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)

Queries List

Select the query source. Then specify how you want to view the queries by selecting a view. To create a custom view, click **New**.

Query source:

View name:

Advisors Tools

All of the rows are displayed. The number of rows is 9.

STMT_ID	CPU	STAT_ELAP	STAT
3180	30112031861845	0.00940714124955383	12
3181	2259567157895	0.008941718380825192	19
3182	05188015863	0.01665736455835548	31
3183	1394883768231	0.025475797247068555	11
3184	0701841586653508392	0.027903352867738873	18
4702	10	0.036362697374002606	60
4703	20	0.013225768696443707	20
4704	30	0.031796031635181576	30

Tools menu:

- Query Annotation
- Access Plan Graph
- Visual Plan Hint**
- Query Reports
- Gather Service Information

Visual Plan Hint

The screenshot displays a SQL query optimizer interface. At the top, the 'Query Blocks' dropdown is set to 'Query block 1'. The main workspace shows a visual plan hint diagram with nodes: SYSDATABASE, SYSTABLESPACE, SYSTABLES, SYSINDEXES, and SYSKEYS. SYSDATABASE is connected to both SYSTABLESPACE and SYSTABLES. SYSTABLESPACE and SYSTABLES are connected to SYSINDEXES, which is connected to SYSKEYS. The right-hand toolbar includes 'Create Join Node', 'Default Join Sequence', 'Delete Selected Nodes', and 'Clear All'. The bottom toolbar includes 'Add Hint Criteria', 'Edit Hint Criteria', and 'Delete Hint Criteria'. Below the toolbar is a table with the following columns:

ID	QBLOCKNO	Tabno	Table Creator	Table Name	Correlation Name	Access Type	Access Creator	Access Name	Prefetch	PAGE_RANGE	Join Method	SORTN_JOIM	SORTC_JOIM	Paralle Mod
----	----------	-------	---------------	------------	------------------	-------------	----------------	-------------	----------	------------	-------------	------------	------------	-------------

Focus on join graph

- Query block selection
- Show local predicates
- Join predicates
- Zoom in / out

Join graph

Join graph

The screenshot displays a SQL query optimizer interface. The main window shows a join graph with five nodes: SYSDATABASE, SYSTABLESPACE, SYSTABLES, SYSINDEXES, and SYSKEYS. The nodes are connected as follows: SYSDATABASE is connected to SYSTABLESPACE and SYSTABLES; SYSTABLESPACE is connected to SYSTABLES; SYSTABLES is connected to SYSINDEXES; and SYSINDEXES is connected to SYSKEYS. The interface includes a toolbar with options like 'Validate Hints', 'Deploy Hints', 'Zoom In', 'Zoom Out', 'Show Local Predicates', 'Show Join Predicates', 'Create Join Node', 'Default Join Sequence', 'Delete Selected Nodes', and 'Clear All'. Below the graph is a table with columns for ID, QBLOCKNO, Tabno, Table Creator, Table Name, Correlation Name, Access Type, Access Creator, Access Name, Prefetch, PAGE_RANGE, Join Method, SORTN_JOIN, SORTC_JOIN, and Parallel Mod.

ID	QBLOCKNO	Tabno	Table Creator	Table Name	Correlation Name	Access Type	Access Creator	Access Name	Prefetch	PAGE_RANGE	Join Method	SORTN_JOIN	SORTC_JOIN	Parallel Mod
----	----------	-------	---------------	------------	------------------	-------------	----------------	-------------	----------	------------	-------------	------------	------------	--------------

Query block selection

The screenshot shows a database query editor interface. At the top, there is a toolbar with buttons for 'Validate Hints', 'Deploy Hints', 'Zoom In', and 'Zoom Out'. Below the toolbar, there are buttons for 'Show Local Predicates', 'Show 1', 'Create Join Node', 'Default Join Sequence', and 'Delete Selected Nodes'. The main area of the editor displays a query plan with a 'Join' node. A dropdown menu is open, showing three options: 'Query block 1', 'Query block 1', and 'Query block 2'. The first 'Query block 1' option is highlighted with a dashed border. Below the dropdown, there is a diagram showing 'SYSTABLESPACE' and 'SYSDATABASE' nodes. At the bottom, there is a table with columns for 'ID', 'QBLOCKNO', 'Tabno', 'Table Creator', 'Table Name', 'Correlation Name', 'Access Type', 'Access Creator', 'Access Name', 'Prefetch', 'PAGE_RANGE', 'Join Method', 'SORTM_JOIM', 'SORTC_JOIM', and 'Paralle Mod'. The table is currently empty.

Query Blocks: Query block 1

Validate Hints Deploy Hints Zoom In Zoom Out

Show Local Predicates Show 1 Create Join Node Default Join Sequence Delete Selected Nodes

Join

Query block 1

Query block 1

Query block 2

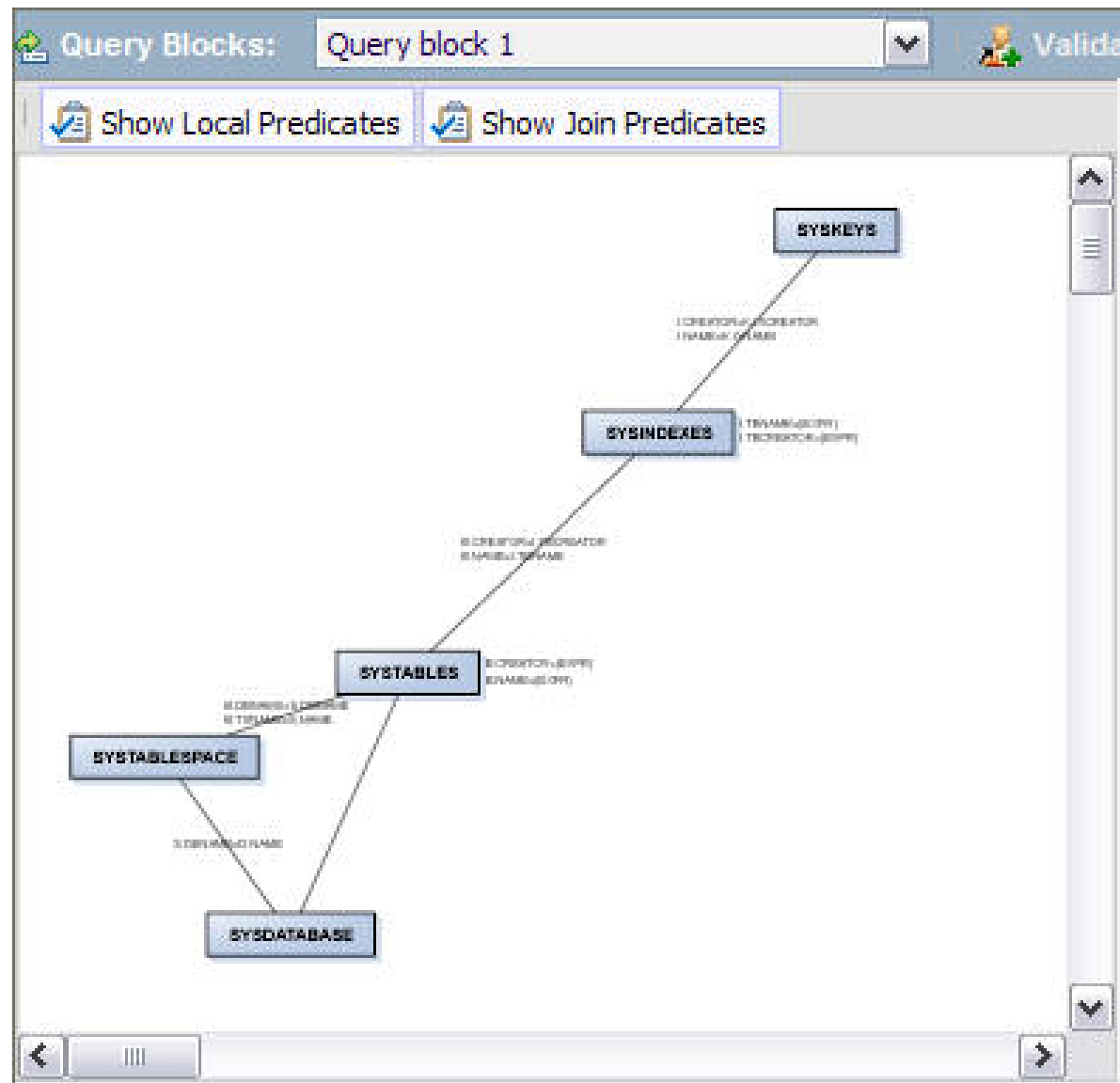
SYSTABLESPACE

SYSDATABASE

Add Hint Criteria Edit Hint Criteria Delete Hint Criteria

ID	QBLOCKNO	Tabno	Table Creator	Table Name	Correlation Name	Access Type	Access Creator	Access Name	Prefetch	PAGE_RANGE	Join Method	SORTM_JOIM	SORTC_JOIM	Paralle Mod
----	----------	-------	---------------	------------	------------------	-------------	----------------	-------------	----------	------------	-------------	------------	------------	-------------

Query block selection



Query block selection

Query Blocks: Query block 1

Validate Hints Deploy Hints Zoom In Zoom Out

Show Local Predicates Show Join Predicates Create Join Node

Join graph

Zoom In Zoom Out

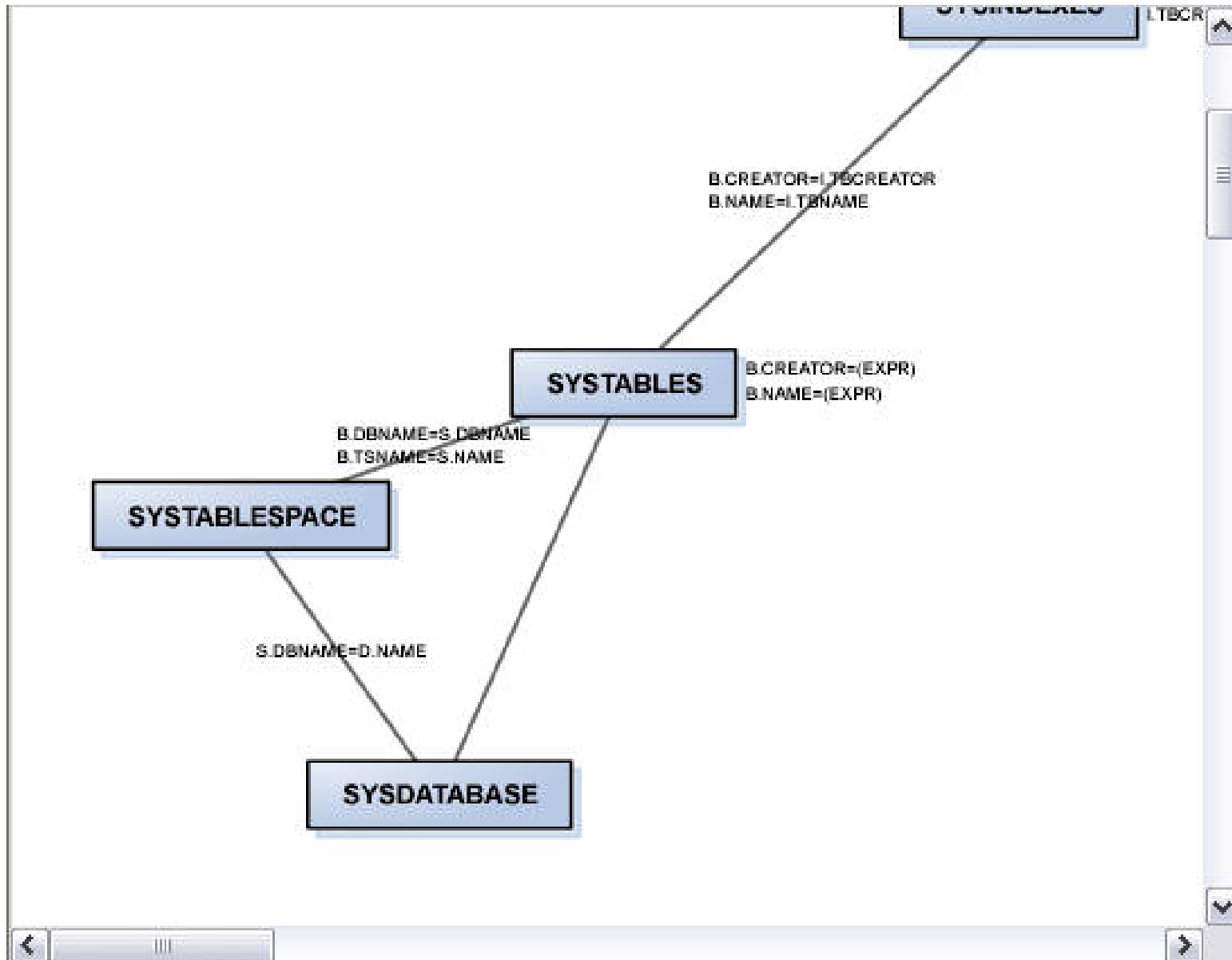
```
graph TD; SYSTABLESPACE --- SYSDATABASE; SYSTABLESPACE --- SYSTABLES; SYSDATABASE --- SYSTABLES;
```

SYSTABLESPACE SYSDATABASE SYSTABLES

+ Add Hint Criteria Edit Hint Criteria Delete Hint Criteria

ID	QBLOCKNO	Tabno	Table Creator	Table Name	Correlation Name	Access Type	Access Creator	Access Name	Prefetch	PAGE_RANGE	Join Method	SORTM_JOIM	SORTC_JOIM	Paralle Mod
----	----------	-------	---------------	------------	------------------	-------------	----------------	-------------	----------	------------	-------------	------------	------------	-------------

Closer look at predicates



Focus on hint

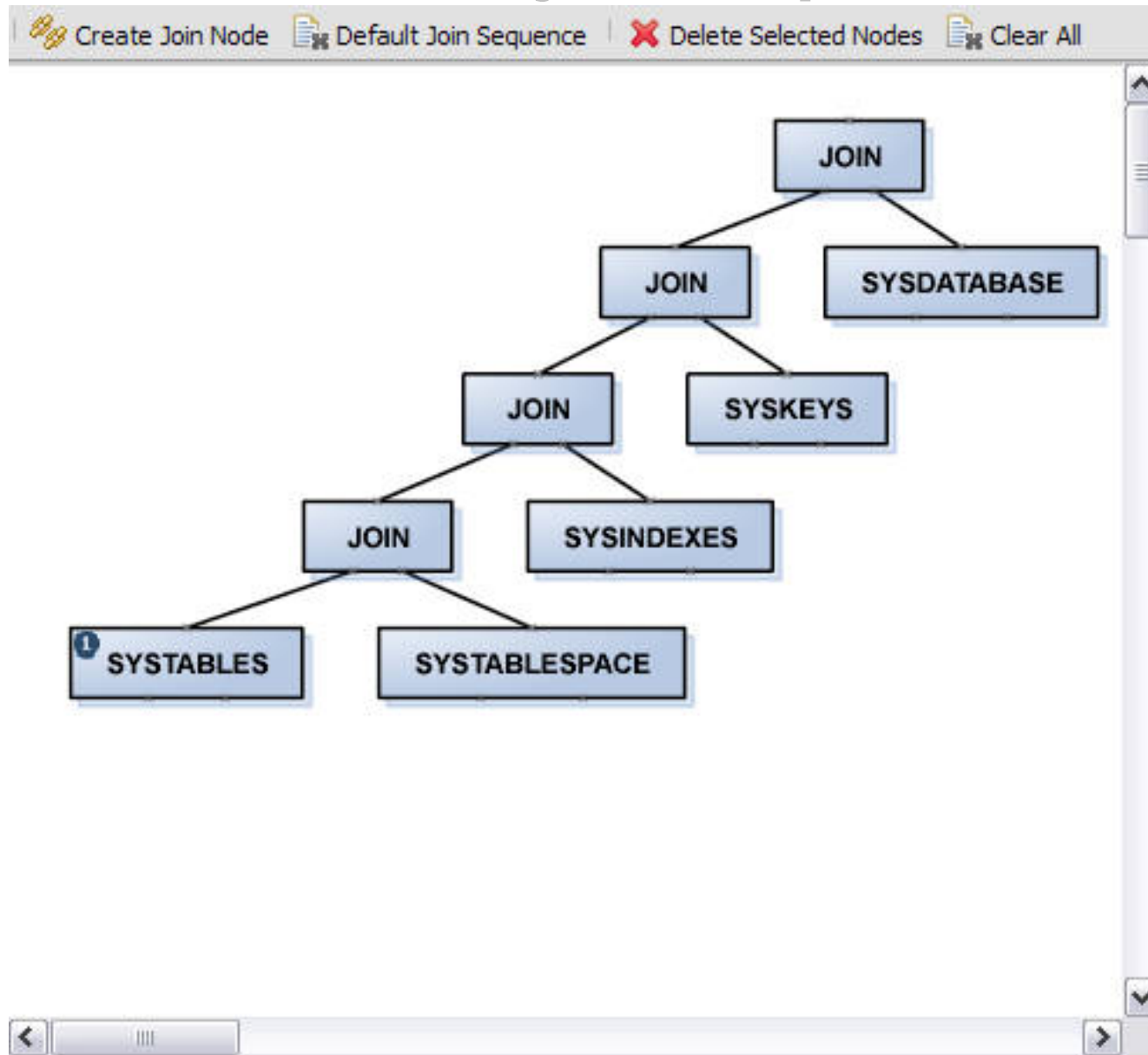
- Default to current access path
- Modify join sequence
- Change access method
- Change join method

Visual Plan Hint

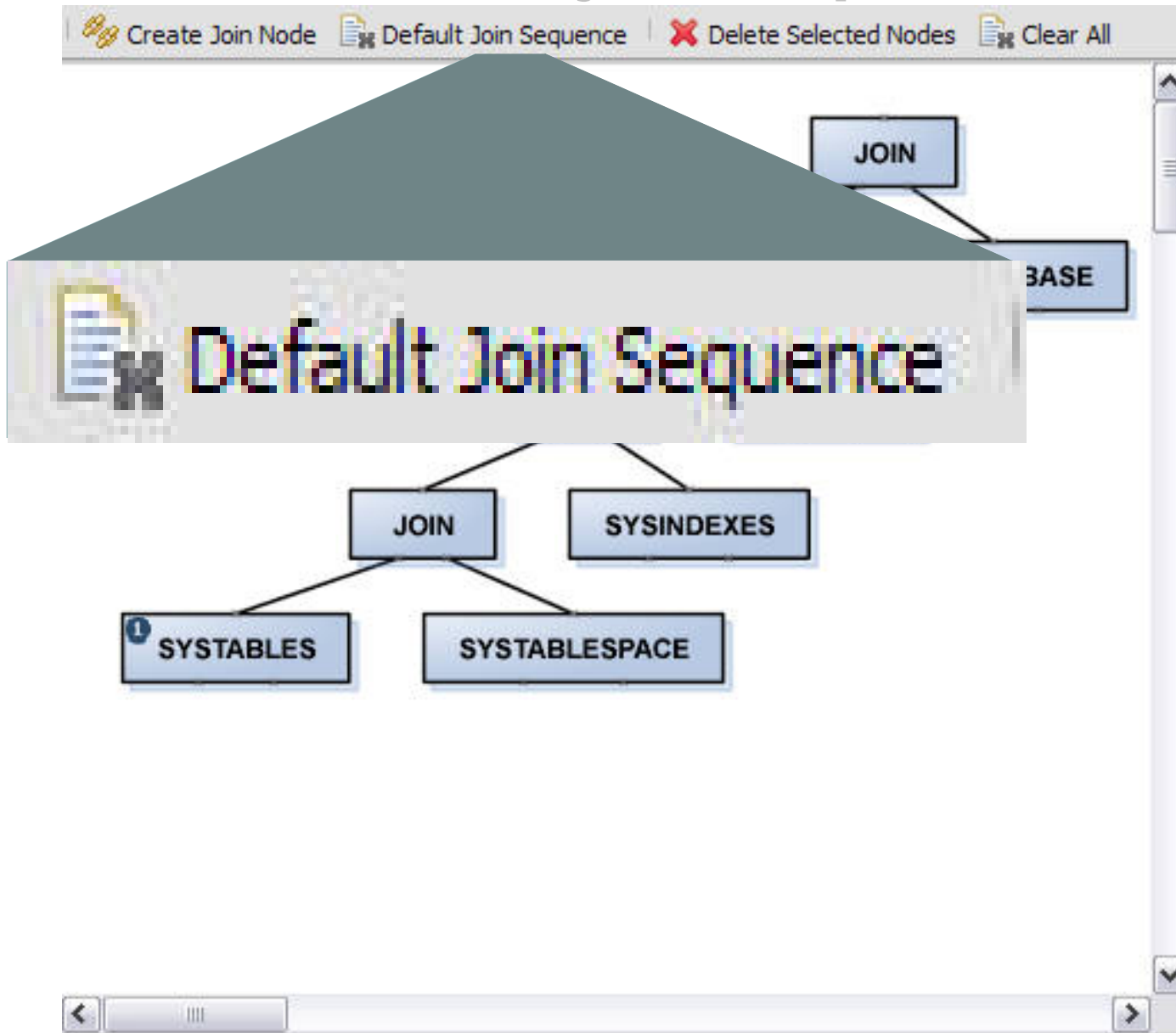
The screenshot displays a SQL query optimizer interface. On the left, a visual plan shows a tree structure of nodes: SYSDATABASE at the bottom, connected to SYSTABLESPACE and SYSTABLES. SYSTABLES is connected to SYSINDEXES, which is connected to SYSKEYS. On the right, a red-bordered box highlights the 'Hint area', which contains a toolbar with 'Create Join Node', 'Default Join Sequence', and 'Delete Selected Nodes' buttons, and a 'Clear All' button below. The main window title is 'Query Blocks: Query block 1'. At the bottom, there are buttons for 'Add Hint Criteria', 'Edit Hint Criteria', and 'Delete Hint Criteria', and a table with columns for query execution details.

ID	QBLOCKNO	Tabno	Table Creator	Table Name	Correlation Name	Access Type	Access Creator	Access Name	Prefetch	PAGE_RANGE	Join Method	SORTN_JOIM	SORTC_JOIM	Paralle Mod
----	----------	-------	---------------	------------	------------------	-------------	----------------	-------------	----------	------------	-------------	------------	------------	-------------

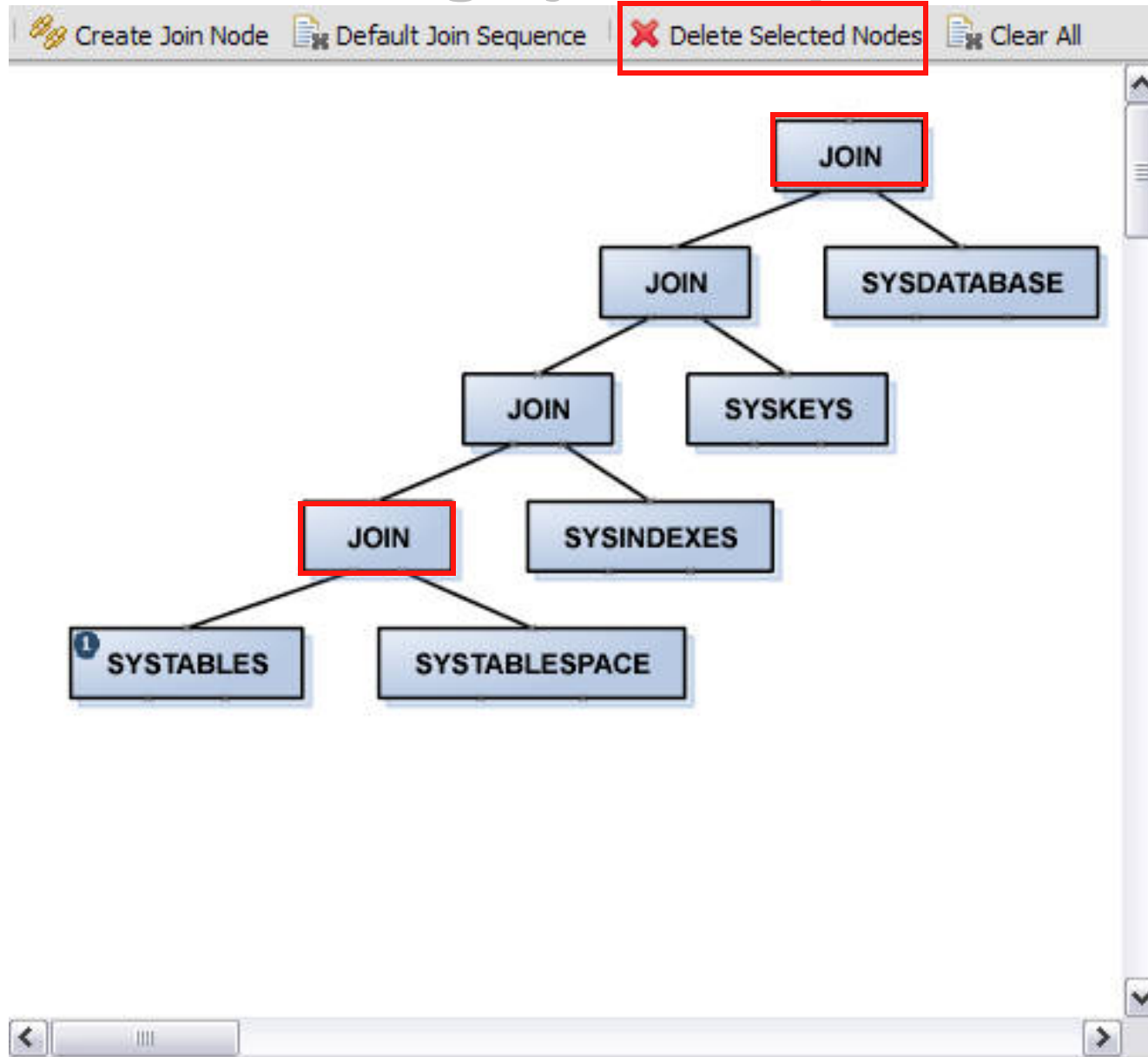
Default join sequence



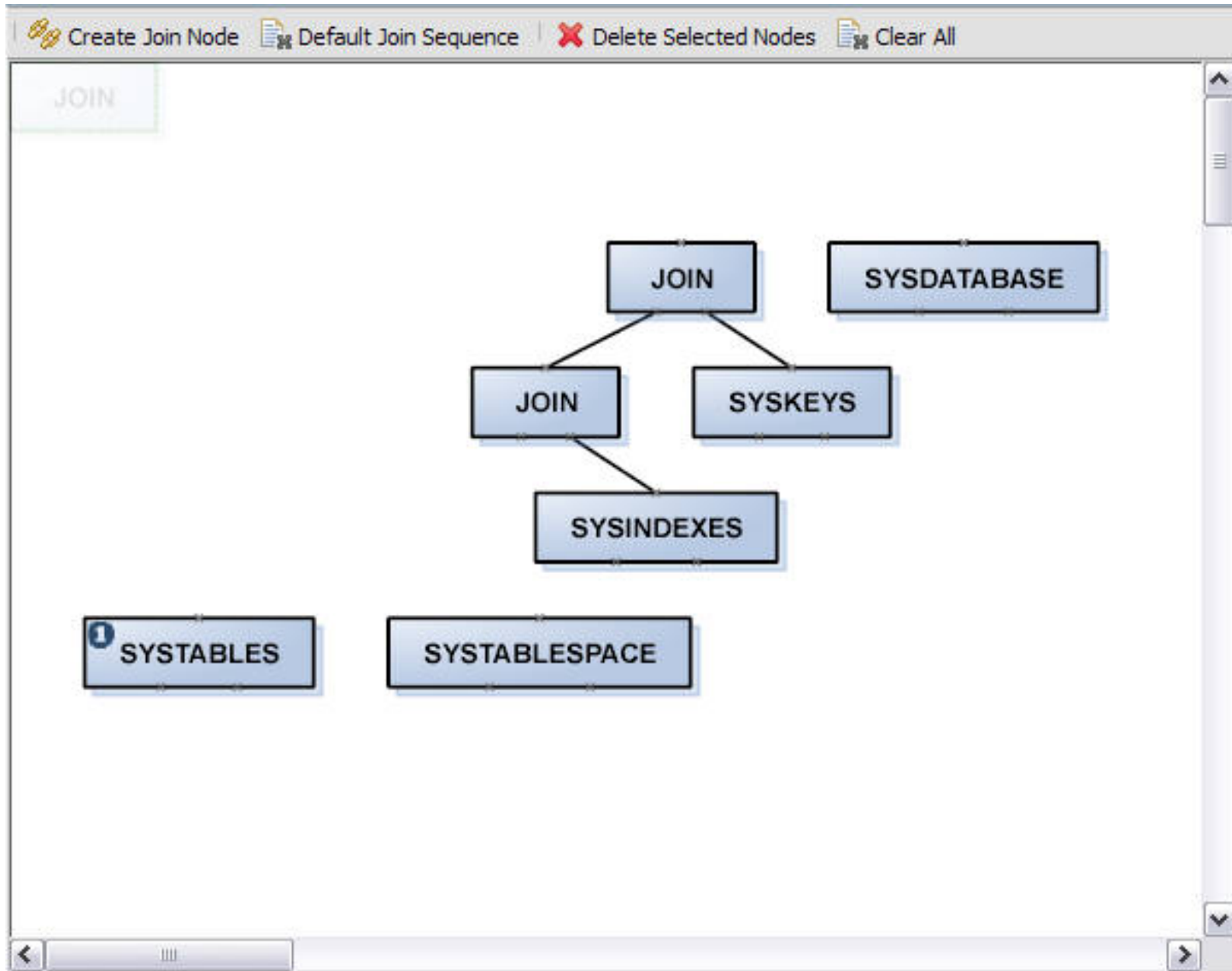
Default join sequence



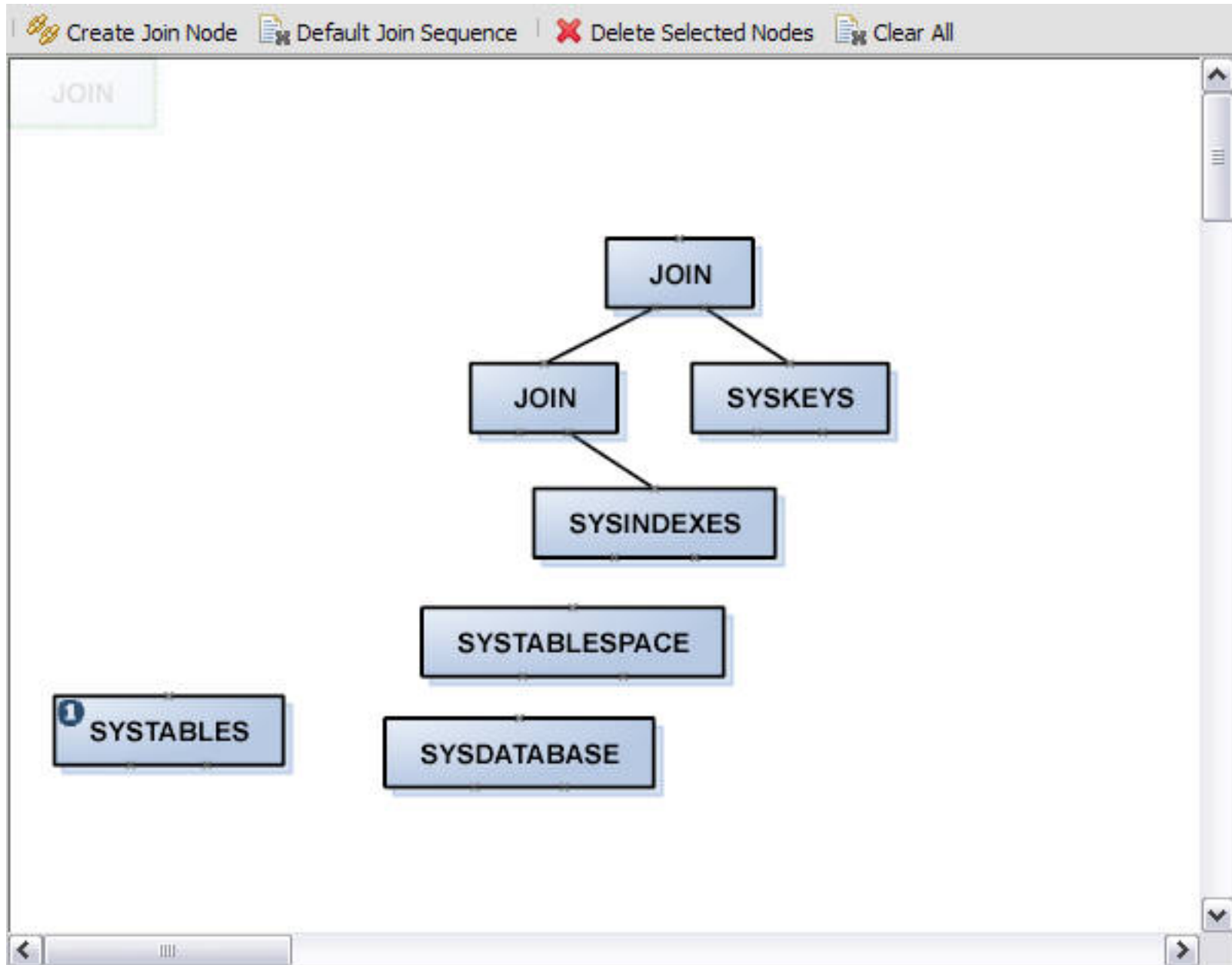
Change join sequence



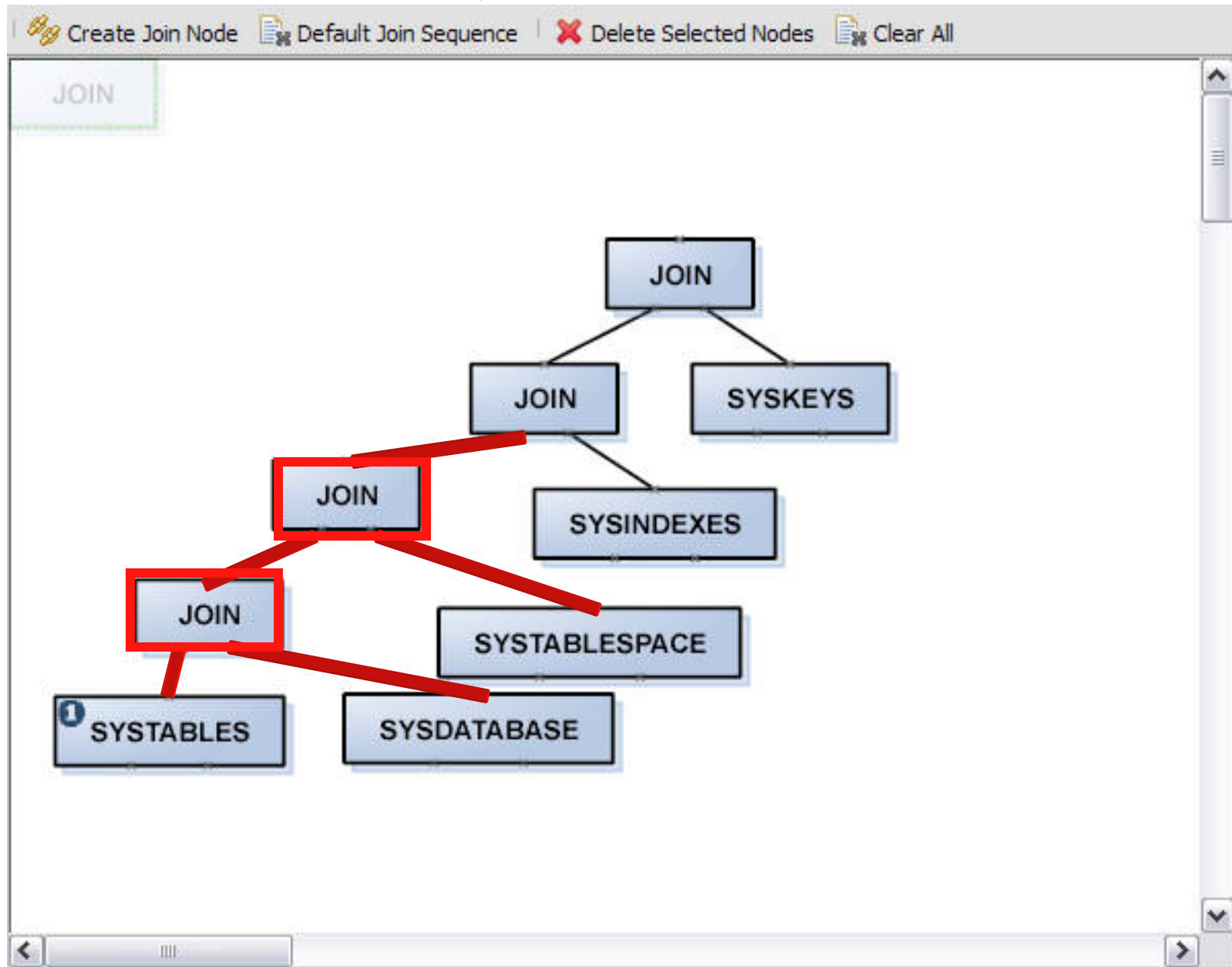
Join connectors removed



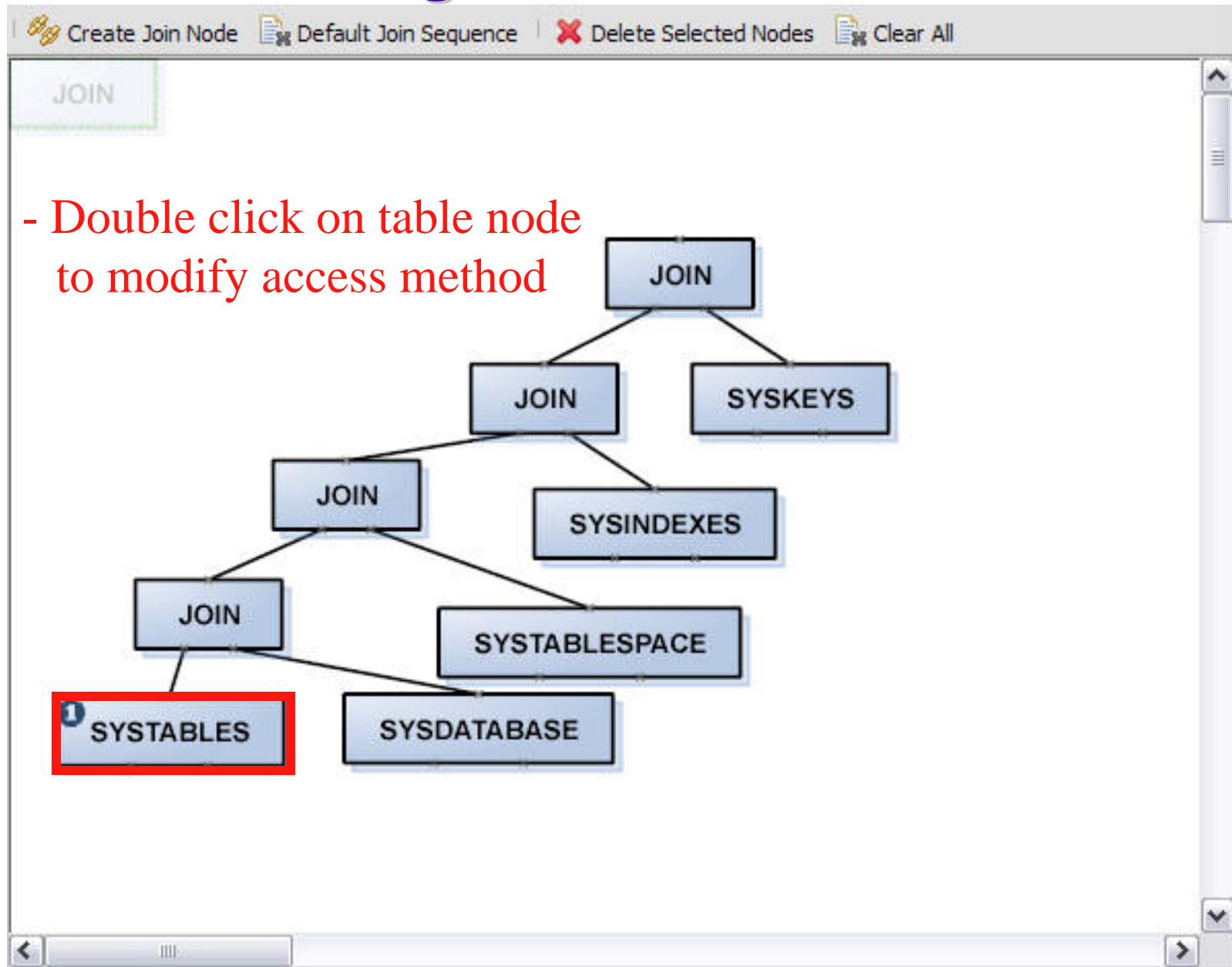
Move the tables around



Create, connect nodes



Change access method



Change access method

Hint Customization Rule

Table Creator	SYSIBM	
Table Name	SYSTABLES	
Correlation Name	B	
Access Type	INDEX	NULL
Access Creator	SYSIBM	NULL
Access Name	DSNDTX01	NULL
Prefetch		NULL
PAGE_RANGE		NULL
Join Method	NULL	NULL
SORTN_JOIN	N	NULL
SORTC_JOIN	N	NULL
Parallelism Mode	NULL	NULL
Access Degree	NULL	NULL
Join Degree	NULL	NULL
PRIMARY_ACESSTYPE		NULL
WHEN_OPTIMIZE		NULL

Leading table

OK Cancel

Change access method

Hint Customization Rule

Table Creator	SYSIBM
Table Name	SYSTABLES
Correlation Name	B
Access Type	INDEX
Access Creator	SYSIBM
Access Name	DSNDTX01
Prefetch	
PAGE_RANGE	
Join Method	NULL
SORTN_JOIN	N
SORTC_JOIN	N
Parallelism Mode	NULL
Access Degree	NULL
Join Degree	NULL
PRIMARY_ACESSTYPE	
WHEN_OPTIMIZE	

Leading table

NULL
INDEX
RSCAN
INLIST
MULTI_INDEX

NULL
NULL
NULL
NULL
NULL
NULL
NULL
NULL
NULL

OK Cancel

Change access method

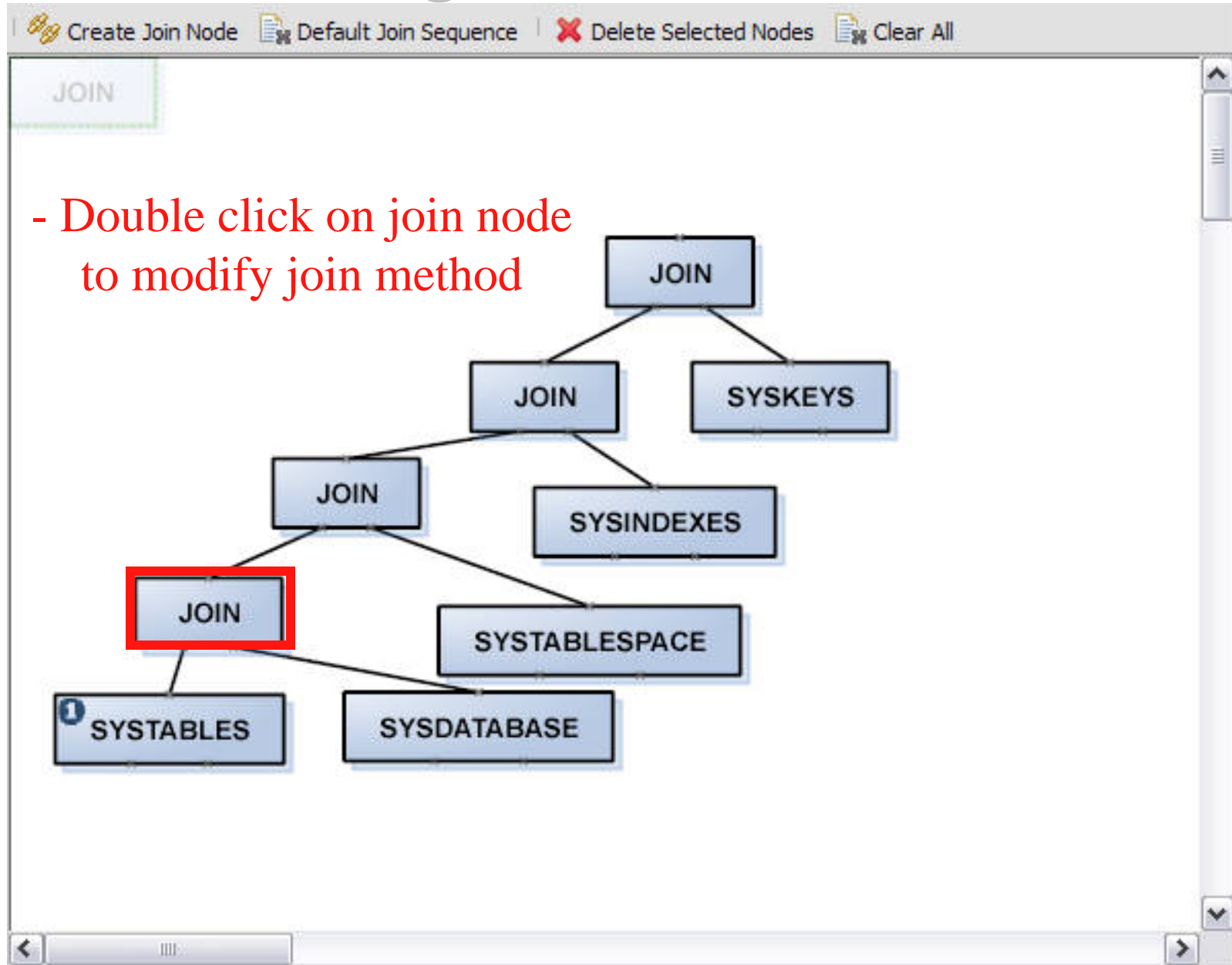
Hint Customization Rule

Table Creator	<input type="text" value="SYSIBM"/>	<input type="text" value="NULL"/>
Table Name	<input type="text" value="SYSTABLES"/>	<input type="text" value="NULL"/>
Correlation Name	<input type="text" value="B"/>	<input type="text" value="NULL"/>
Access Type	<input type="text" value="INDEX"/>	<input type="text" value="NULL"/>
Access Creator	<input type="text" value="SYSIBM"/>	<input type="text" value="NULL"/>
Access Name	<input type="text" value="DSNDTX01"/>	<input type="text" value="NULL"/>
Prefetch	<input type="text"/>	<input type="text" value="NULL"/>
PAGE_RANGE	<input type="text"/>	<input type="text" value="NULL"/>
Join Method	<input type="text" value="NULL"/>	<input type="text" value="NULL"/>
SORTN_JOIN	<input type="text" value="N"/>	<input type="text" value="NULL"/>
SORTC_JOIN	<input type="text" value="N"/>	<input type="text" value="NULL"/>
Parallelism Mode	<input type="text" value="NULL"/>	<input type="text" value="NULL"/>
Access Degree	<input type="text" value="NULL"/>	<input type="text" value="NULL"/>
Join Degree	<input type="text" value="NULL"/>	<input type="text" value="NULL"/>
PRIMARY_ACESSTYPE	<input type="text"/>	<input type="text" value="NULL"/>
WHEN_OPTIMIZE	<input type="text"/>	<input type="text" value="NULL"/>

Leading table





OK Cancel

Change access method



Change join method





Hint Customization Rule

Table Creator	<input type="text" value="SYSIBM"/>		
Table Name	<input type="text" value="SYSDATABASE"/>		
Correlation Name	<input type="text" value="D"/>		
Join Method	<input type="text" value="NLJ"/>		<input type="text" value="NULL"/>
SORTN_JOIN	<input type="text" value="N"/>		<input type="text" value="NULL"/>
SORTC_JOIN	<input type="text" value="N"/>		<input type="text" value="NULL"/>
Join Degree	<input type="text"/>		<input type="text"/>

OK Cancel

Change join method

Hint Customization Rule

Table Creator	<input type="text" value="SYSIBM"/>	
Table Name	<input type="text" value="SYSDATABASE"/>	
Correlation Name	<input type="text" value="D"/>	
Join Method	<input type="text" value="NLJ"/>	
SORTN_JOIN	<input type="text" value="N"/>	
SORTC_JOIN	<input type="text" value="N"/>	
Join Degree	<input type="text"/>	

NULL

NULL

NLJ

SMJ

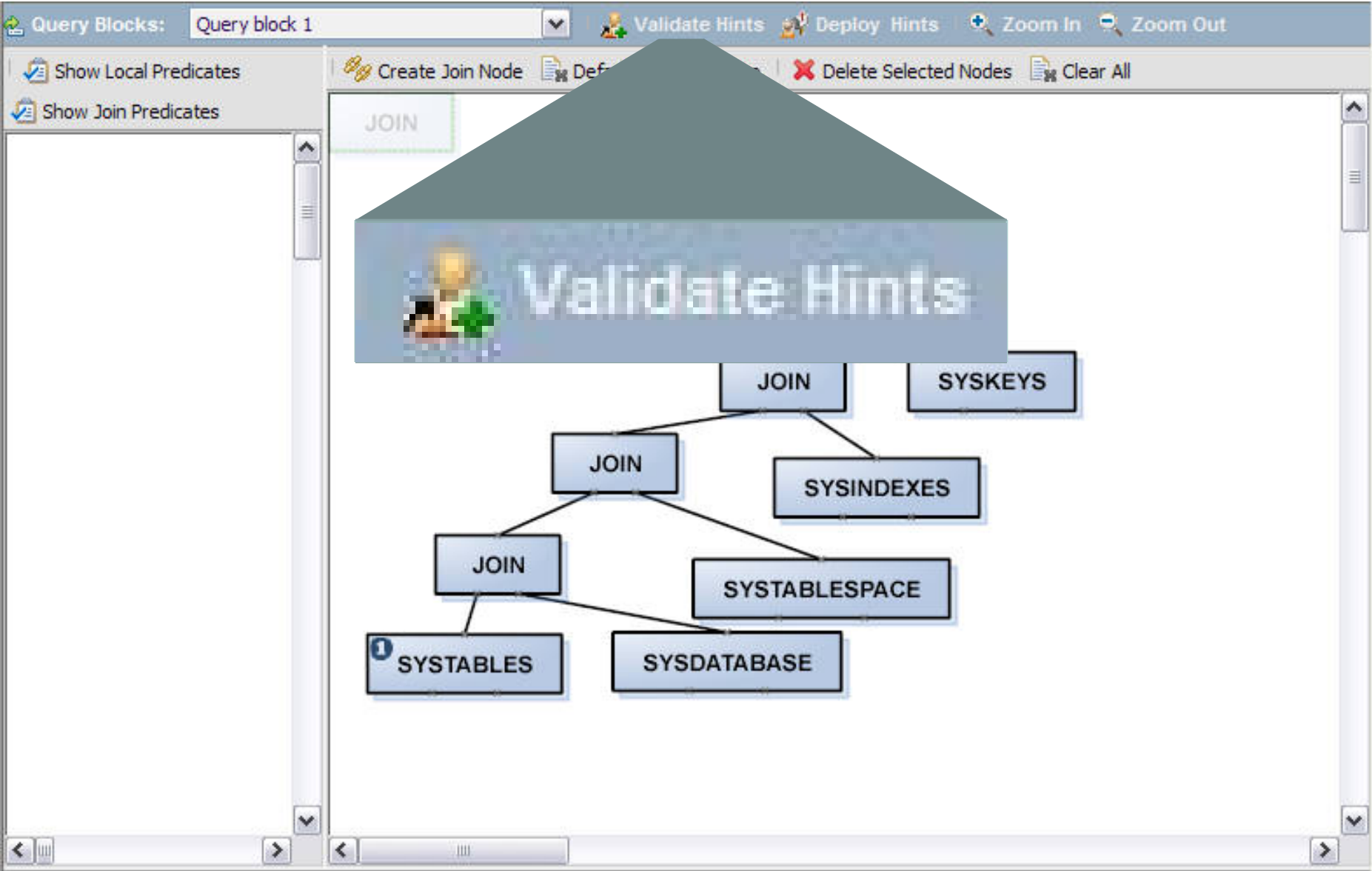
HYBRID

OK Cancel

Implement hint

- Validate hint
- Deploy hint

Validate hint

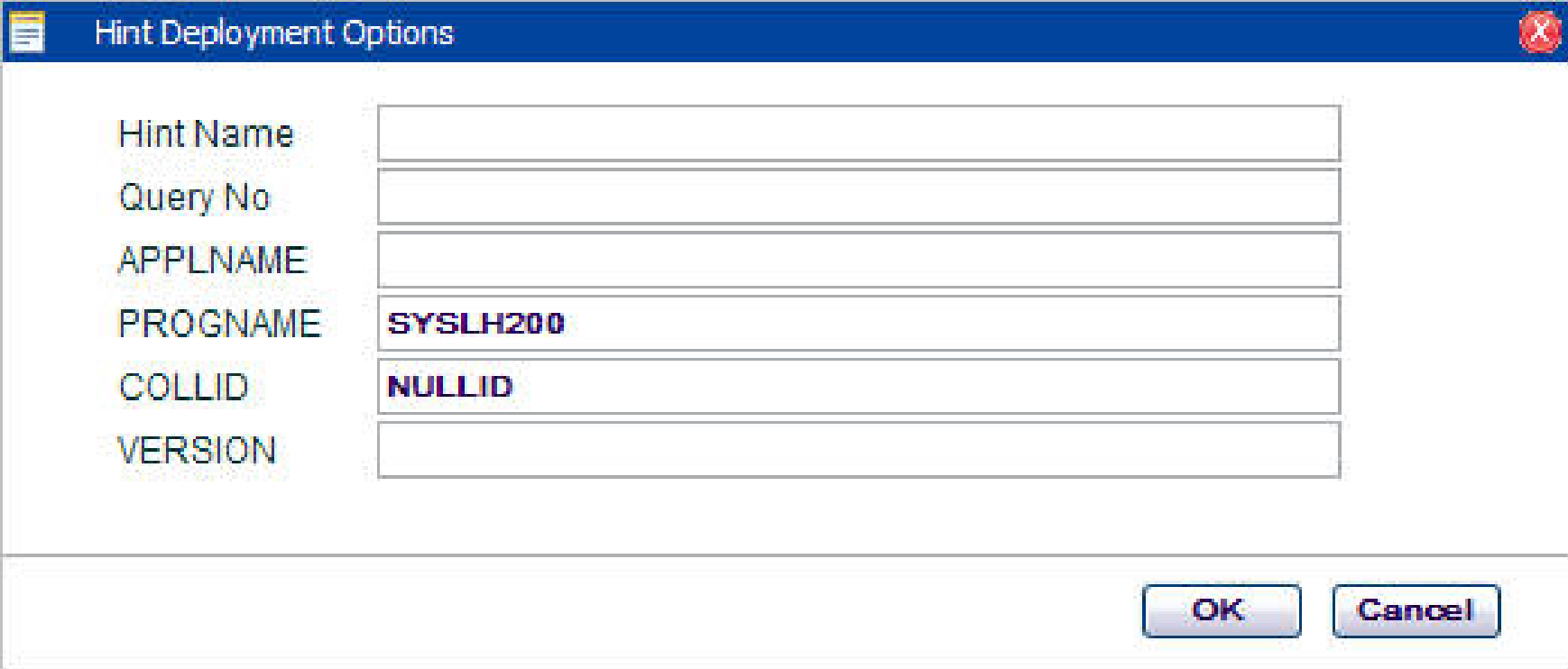


Validate hint (will show screen shot)

Plan table without using plan hint												
QUERYNO	QBLOCKNO	APPLNAME	PROGNAME	PLANNO	METHOD	CREATOR	TNAME	TABNO	ACCESSTYPE	MATCHCOLS	ACCESSCREATOR	ACCESSNAME
1	1		SYSLH200	1	0	SYSIBM	SYSTABLES	1	I	2	SYSIBM	DSNDTX01
1	1		SYSLH200	2	1	SYSIBM	SYSTABLESPACE	2	I	2	SYSIBM	DSNDSX01
1	1		SYSLH200	3	1	SYSIBM	SYSINDEXES	3	I	2	SYSIBM	DSNDXX03
1	1		SYSLH200	4	1	SYSIBM	SYSKEYS	4	I	2	SYSIBM	DSNDKX01
1	1		SYSLH200	5	1	SYSIBM	SYSDATABASE	5	I	1	SYSIBM	DSNDH01

Plan table after plan hint												
QUERYNO	QBLOCKNO	APPLNAME	PROGNAME	PLANNO	METHOD	CREATOR	TNAME	TABNO	ACCESSTYPE	MATCHCOLS	ACCESSCREATOR	ACCESSNAME
15462	1		SYSLH200	1	0	SYSIBM	SYSTABLES	1	I	0	SYSIBM	DSNDTX02
15462	1		SYSLH200	2	1	SYSIBM	SYSDATABASE	5	I	1	SYSIBM	DSNDH01
15462	1		SYSLH200	3	1	SYSIBM	SYSTABLESPACE	2	I	2	SYSIBM	DSNDSX01
15462	1		SYSLH200	4	1	SYSIBM	SYSINDEXES	3	I	2	SYSIBM	DSNDXX03
15462	1		SYSLH200	5	1	SYSIBM	SYSKEYS	4	I	2	SYSIBM	DSNDKX01

Deploy hint



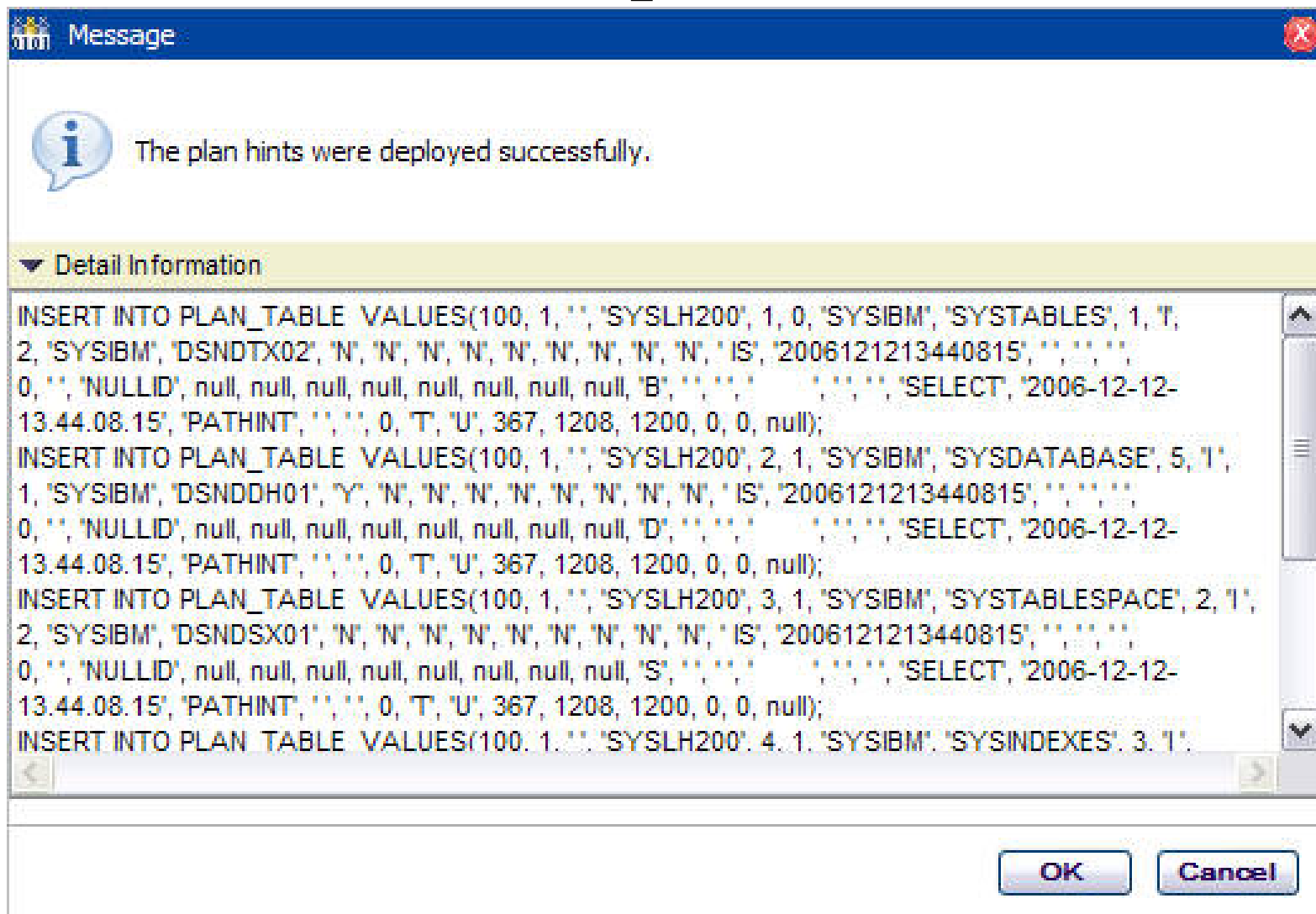
Hint Deployment Options

Hint Name	<input type="text"/>
Query No	<input type="text"/>
APPLNAME	<input type="text"/>
PROGNAME	<input type="text" value="SYSLH200"/>
COLLID	<input type="text" value="NULLID"/>
VERSION	<input type="text"/>

OK Cancel

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