Optimal query access plans are essential for good data server performance, and it is the query optimizer's job to choose the best one. However, occasionally no amount of statistics or tuning is enough to get the access plan you want. Or your application is effectively down, due to a poorly performing query, and there isn't time to implement best practices. When these situations arise, Db2 optimization profiles and guidelines can be used to correct the access plan and get your application performing well again, quickly. Optimization profiles can specify various aspects of an access plan, can control automatic query rewriting or can control the optimizer's plan search space. Guidelines can be specified with or without modifying your application. This presentation will show you what options are available and how to use them effectively.

Session Objectives

Tridex
A DB2 USER GROUP

- Optimization profile structure and syntax
- Access plan optimization guidelines
- Query rewrite optimization guidelines
- Installation and activation
- Optimization guidelines embedded in SQL
- Diagnosing optimization profile problems

Optimization profiles

Tridex
A DB2 USER GROUP

- More commonly known as 'hints'
- Ability to specify access plan details
  - Index scan, join method, join order, etc.
- Ability to control statement optimization
  - Can control both query rewrite optimization and access plan optimization
- Can be put into effect without editing application code
  - Compose optimization profile, add to DB, rebind targeted packages
- **Should only be used after all other tuning options exhausted**

3

The Db2 data server supports an even more direct way to influence access plans using optimization profiles. Optimization profiles allow you to specify access plan details such as base access and join methods and join order. For example, you can specify that access to a particular table should use a particular index or you can specify that two tables should be joined using the hash join method. Optimization profiles also allow you to control query rewrite optimizations such transforming certain types of subqueries to joins. You can specify the base table access methods, join methods, and join order for the entire access plan, or just a subset of the access plan.

Optimization profiles are a powerful tool for controlling access plans; however, they should be used with caution. Optimization profiles prevent access plans from adjusting to changes in your data and your environment. While this does result in more stable access plans, it may be a bad approach when used for extended periods of time, because the performance improvements resulting from better access plans will never be realized. Optimization profiles are best used for exceptional situations when the tuning actions described previously in the presentation are unsuccessful in improving or stabilizing access plans.

Optimization profiles: anatomy

Tridex
A DB2 USER GROUP

- An **XML document** stored in a special system table
- Elements and attributes understood as explicit optimization guidelines/hints
- Composed and validated with an XML schema
  `sqllib/misc/DB2OptProfile.xsd`
- **Profile Header** (exactly one)
  - Meta data and processing directives
- **Global optimization guidelines** (at most one)
  - Applies to all statements for which profile is in effect
  - E.g. eligible MQTs guideline defining MQTs to be considered for routing
- **Statement-level optimization guidelines** (zero or more)
  - Applies to a specific statement for which profile is in effect
  - Specifies aspects of desired access plan

4

An optimization profile is specified as an XML document that you create and store in the SYSTOOLS.OPT_PROFILE table.

An optimization profile contains optimization guidelines that specify the access plan details. An optimization profile can contain optimization guidelines for one or more SQL statements. The SQL statement text is stored in the optimization profile along with the optimization guidelines. When an optimization profile is in effect for your application, each SQL statement compiled by your application will be matched to the SQL statements specified in the optimization profile. When a matching SQL statement is found in the optimization profile, the SQL compiler will use the optimization guidelines for that SQL statement while optimizing it.

Profiles vs. guidelines

Tridex
A DB2 USER GROUP

- Optimization profile:
  - Refers to the entire XML document that can contain various types of optimization guidelines for one or more SQL statements
  - Identified by the <OPTPROFILE> element
  - Stored in SYSTOOLS.OPT_PROFILE
- Optimization guidelines
  - Refers to a portion of the optimization profile that provides hints or guidelines for specific aspects of query optimization
  - Identified by the <OPTGUIDELINES> element
  - Can be specified as an SQL statement comment

5

The term ‘optimization profile’ refers to the entire XML document that is stored in SYSTOOLS.OPT_PROFILE. An ‘optimization guideline’ refers to sections of the optimization profile represented by the <OPTGUIDELINES> element. An optimization guideline can exist on its own when it is specified as a comment on an SQL statement. More on this later.

Sample optimization profile

Tridex
A DB2 USER GROUP

```
<?xml version="1.0" encoding="UTF-8"?>
<OPTPROFILE VERSION="11.5.0">
<!--
    Global optimization guidelines section.
    Optional but at most one.
-->
<OPTGUIDELINES>
    <MQT NAME="DBA.AvgSales"/>
    <MQT NAME="DBA.SumSales"/>
</OPTGUIDELINES>
<!--
    Statement profile section.
    Zero or more.
-->
<STMTPROFILE ID="Guidelines for TPCD Q9">
    <STMTKEY SCHEMA="TPCD">
        <![CDATA[SELECT S.S_NAME, S.S_ADDRESS, S.S_PHONE, S.S_COMMENT FROM PARTS P, SUPPLIERS S, PARTSUPP PS
WHERE P_PARTKEY = PS.PS_PARTKEY AND S.S_SUPPKEY = PS.PS_SUPPKEY AND P.P_SIZE = 39
AND P.P_TYPE = 'BRASS' AND S.S_NATION = 'MOROCCO' AND S.S_NATION IN ('MOROCCO', 'SPAIN')
AND PS.PS_SUPPLYCOST = (SELECT MIN(PS1.PS_SUPPLYCOST) FROM PARTSUPP PS1, SUPPLIERS S1
WHERE P.P_PARTKEY = PS1.PS_PARTKEY AND S1.S_SUPPKEY = PS1.PS_SUPPKEY AND S1.S_NATION = S.S_NATION)]]>
    </STMTKEY>
    <OPTGUIDELINES>
        <IXSCAN TABID="Q1" INDEX="I_SUPPKEY"/>
    </OPTGUIDELINES>
</STMTPROFILE>
</OPTPROFILE>
```

Use CDATA to prevent > and < from being interpreted by the XML parser

6

This sample shows an entire optimization profile document.

The first <OPTGUIDELINES> element specifies which MQTs the optimizer should consider. It doesn't mean that these MQTs will be forced to be used, but they will be the only ones considered by the optimizer.

The <STMTPROFILE> section represents a guideline for a specific SQL statement. The SQL statement text is included in the <STMTKEY> element because it will be matched to an SQL statements that are compiled when this optimization profile is in effect. The <OPTGUIDELINES> element following <STMTKEY> represents an access path hint that will be applied to this SQL statement.

Embedded optimization guidelines

Tridex
A DB2 USER GROUP

- Optimization guidelines can be specified along with the SQL statement in an SQL comment
- Only one set of guidelines can appear in a /* */ comment after the entire SQL statement
- Supported for static and dynamic SQL and SQL/PL procedures
- Enabled by default in Db2 11.1
  - Must set registry variable DB2_OPTPROFILES=ON in prior releases

```
SELECT  S.S_NAME FROM  TPCD.SUPPLIER S WHERE S_NAME = 'XYZ CORP'
/* <OPTGUIDELINES><IXSCAN  TABLE='S'  INDEX='S_IX1'/></OPTGUIDELINES>  */
```

- This presentation uses embedded guidelines in most examples

7

https://www.ibm.com/docs/en/db2/11.5?topic=guidelines-embedded-optimization

General rules when using embedded optimization guidelines:

Embedded optimization guidelines can only be applied to Data Manipulation Language (DML) statements: the SELECT, INSERT, UPDATE, DELETE, and MERGE commands. The optimizer will ignore such comments on other types of statements. No error or warning will be provided.

The embedded optimization guideline must be provided after the SQL portion of the statement. They cannot appear inside subselects. However, other types of comments can be provided at the end of the statement before or after the optimization guideline.

The optimizer will look for one embedded optimization guideline comment for every DML statement. If there are multiple embedded optimization guideline comments, all of them are ignored and a warning is produced.

The optimization guideline must be written in well-formed XML. It cannot include extraneous text.

Optimization Guidelines (1|2)

Tridex
A DB2 USER GROUP

- Access plan guidelines
  - Base access request
    - Method to access a table e.g. TBSCAN, IXSCAN
  - Join request
    - Method and sequence for performing a join e.g. HSJOIN, NLJOIN, MSJOIN
    - IXAND star joins
- Query rewrite guidelines
  - IN-list to join
  - Subquery to join
  - NOT EXISTS subquery to anti-join
  - NOT IN subquery to anti-join

8

There are 3 main types of optimization guidelines – those that specify the access plan and those that control query rewrites/transformations.

The 3rd type is next …

Optimization Guidelines (2|2)

Tridex
A DB2 USER GROUP

- General optimization guidelines
  - REOPT (ONCE/ALWAYS/NONE)
    - Same as REOPT bind option
  - DEGREE
    - Intra-partition query parallelism degree
  - QRYOPT
    - Query optimization level
  - RTS
    - Real-time statistics, enable/disable, timeout time
  - MQT choices
    - Materialized query table options

9

The 3rd type of optimization guideline is ‘general’. It specifies options that control the query optimizer, such as the degree of parallelism and the query optimization level.

Access Plan Optimization Guidelines

Tridex
A DB2 USER GROUP

- **Example:**

```
SELECT S.S_NAME, S.S_ADDRESS, S.S_PHONE, S.S_COMMENT
FROM "Tpcd".PARTS, "Tpcd".SUPPLIERS S, "Tpcd".PARTSUPP PS
WHERE
    P_PARTKEY = PS.PS_PARTKEY AND  S.S_SUPPKEY = PS.PS_SUPPKEY AND  P_SIZE = 39 AND P_TYPE = 'BRASS'
    PS.PS_SUPPLYCOST =
          (SELECT MIN(PS1.PS_SUPPLYCOST)
           FROM "Tpcd".PARTSUPP PS1, "Tpcd".SUPPLIERS S1
           WHERE "Tpcd".PARTS.P_PARTKEY = PS1.PS_PARTKEY AND
              S1.S_SUPPKEY = PS1.PS_SUPPKEY AND  S1.S_NATION  = S.S_NATION)
ORDER BY S.S_NAME
/* <OPTGUIDELINES><IXSCAN TABLE='S' INDEX='I_SUPPKEY'/></OPTGUIDELINES> */
```

- Choose an index access using index 'I_SUPPKEY' for access to SUPPLIERS table in main sub-select
  - **Don't specify an index qualifier/schema! (It will not be recognized)**
- Table is referenced using correlation name 'S'
  - TABLE attribute must reference the 'exposed' name

10

This example shows an optimization guideline that specifies to use index I_SUPPKEY to access "Tpcd".SUPPLIERS S in the outer sub-select. The TABLE attribute is used to refer to the table. Tables are referenced using their 'exposed' name as described by the SQL standard.

Access Plan Optimization Guidelines

Tridex
A DB2 USER GROUP

- **Example:**

```
SELECT S.S_NAME, S.S_ADDRESS, S.S_PHONE, S.S_COMMENT
FROM "Tpcd".PARTS, "Tpcd".SUPPLIERS S, "Tpcd".PARTSUPP PS
WHERE P_PARTKEY = PS.PS_PARTKEY AND S.S_SUPPKEY = PS.PS_SUPPKEY AND
   P_SIZE = 39 AND P_TYPE = 'BRASS'
   PS.PS_SUPPLYCOST =
     (SELECT MIN(PS1.PS_SUPPLYCOST)
      FROM "Tpcd".PARTSUPP PS1, "Tpcd".SUPPLIERS S1
      WHERE "Tpcd".PARTS.P_PARTKEY = PS1.PS_PARTKEY AND
         S1.S_SUPPKEY = PS1.PS_SUPPKEY AND
         S1.S_NATION  = S.S_NATION)
ORDER BY S.S_NAME
/* <OPTGUIDELINES>
   <NLJOIN>
         <IXSCAN TABLE='"Tpcd".Parts'/>
         <IXSCAN TABLE="PS"/>
   </NLJOIN>
</OPTGUIDELINES> */
```

Index name not provided so optimizer chooses based on cost

- Join requests contains 2 elements – inner and outer
- Elements can be base accesses or other join requests

11

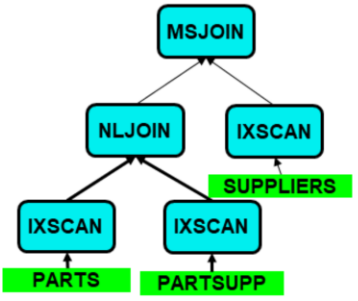
This example shows a join optimization guideline. The tables to be joined are contained within the <NLJOIN> element. The first table is the outer of the join and the second is the inner.

Nesting join requests

Tridex
A DB2 USER GROUP

- **Example**

```
<OPTGUIDELINES>
  <MSJOIN>
    <NLJOIN>
      <IXSCAN TABLE='"Tpcd".Parts'/>
      <IXSCAN TABLE="PS"/>
    </NLJOIN>
    <IXSCAN TABLE='S'/>
  </MSJOIN>
</OPTGUIDELINES>
```



- Nested loop join is on outer of merge scan join
- Nested access request elements inside a join request must reference tables in the **same FROM clause** of the optimized statement
  - (more on this later)
- Query rewrite optimization guidelines, as well as general optimization guidelines, can affect the optimized statement.

12

Join elements can be nested within join elements, provided that the tables to be joined are within the same sub-select in the 'optimized' or transformed/rewritten SQL. This is because the access plan is built based on an automatically rewritten version of the original statement, which could be very different than the original. More on this later.

Forming table references

Tridex
A DB2 USER GROUP

- 2 methods
  - Reference 'exposed' name in the original SQL statement
    - Use **'TABLE'** attribute
    - Rules for specifying SQL identifiers apply to 'TABLE' attribute
  - Reference correlation name in the optimized SQL statement
    - Use **'TABID'** attribute
    - 'Optimized' SQL is the semantically equivalent version of the statement after is has been optimized by query rewrite
    - Use the explain facility to get the optimized SQL statement
    - NOTE: There is no guarantee that correlation names in the optimized SQL statement are stable across new releases
- Table references must refer to a single table or they are ignored
  - i.e. no ambiguous references
- Unqualified table references are implicitly qualified by the current schema
- If both 'TABLE' and 'TABID' are specified, they must refer to the same table or they are ignored.
- Use 'TABID' to reference derived tables

13

The term table reference is used to mean any table, view, table expression, or the table which an alias references in an SQL statement or view definition. An optimization guideline can identify a table reference using either its exposed name in the original statement or the unique correlation name that is associated with the table reference in the optimized statement.

A table reference is identified by using the exposed name of the table. The exposed name is specified in the same way that a table would be qualified in an SQL statement.

The rules for specifying SQL identifiers also apply to the TABLE attribute value of an optimization guideline. The TABLE attribute value is compared to each exposed name in the statement. Only a single match is permitted in this Db2® release. If the TABLE attribute value is schema-qualified, it matches any equivalent exposed qualified table name. If the TABLE attribute value is unqualified, it matches any equivalent correlation name or exposed table name. The TABLE attribute value is therefore considered to be implicitly qualified by the default schema that is in effect for the statement.

Referencing derived tables

Tridex
A DB2 USER GROUP

- Example

```
SELECT  S.S_NAME,  PS.PS_AVAILQTY,  P.P_NAME
FROM   TPCD.SUPPLIER  S  INNER JOIN TPCD.PARTSUPP  PS
 ON S.S_SUPPKEY = PS.PS_SUPPKEY
 LEFT OUTER JOIN TPCD.PART  P
 ON P.P_PARTKEY  = PS.PS_PARTKEY
```

- This guideline is invalid because the inner join between SUPPLIER and PARTSUPP is in a separate derived table :

```
<OPTGUIDELINES>
   <HSJOIN>
     <IXSCAN TABLE="P"/>
     <HSJOIN>
       <IXSCAN TABLE="S" />
       <IXSCAN TABLE="PS"/>
     </HSJOIN>
   </HSJOIN>
</OPTGUIDELINES>
```

14

This is an example of an invalid optimization guideline because all the tables to be joined aren't in the same derived tables, or sub-select. But there are ways to handle this …

Referencing derived tables

Tridex
A DB2 USER GROUP

- Option 1: Reference derived table using TABID and optimized SQL correlation name
  - Doesn't require modifying the SQL text
  - But optimized SQL correlation names aren't stable across releases

```
<OPTGUIDELINES>
 <HSJOIN>
  <IXSCAN TABLE="P"/>
  <ACCESS TABID="Q3" />
 </HSJOIN>
 <HSJOIN>
  <IXSCAN TABLE="S" />
  <IXSCAN TABLE="PS"/>
 </HSJOIN>
</OPTGUIDELINES>
```

```
SELECT  Q3.S_NAME AS "S_NAME",  Q3.PS_AVAILQTY AS
"PS_AVAILQTY",  Q4.P_NAME AS "P_NAME"
FROM
 (SELECT  Q2.PS_AVAILQTY,  Q2.PS_PARTKEY,  Q1.S_NAME
  FROM
   TPCD.SUPPLIER  AS Q1,
   TPCD.PARTSUPP  AS Q2
  WHERE
   (Q1.S_SUPPKEY  = Q2.PS_SUPPKEY)
 ) AS Q3
 LEFT OUTER JOIN TPCD.PART AS Q4
 ON (Q4.P_PARTKEY  = Q3.PS_PARTKEY)
```

Tip: Get the optimized SQL from the explain facility

15

The preferred option for referencing derived tables is to use the TABID attribute to specify the optimized SQL correlation name.

Referencing derived tables

Tridex
A DB2 USER GROUP

- Option 2: Rewrite the SQL to use an inline view and reference the view in the guideline:

```
WITH VX AS
(SELECT S.S_NAME, PS.PS_AVAILQTY, PS.PS_PARTKEY
 FROM TPCD.SUPPLIER S INNER JOIN TPCD.PARTSUPP PS ON S.S_SUPPKEY = PS.PS_SUPPKEY)
SELECT VX.S_NAME, VX.PS_AVAILQTY, P.P_NAME
FROM VX LEFT OUTER JOIN TPCD.PART P ON P.P_PARTKEY = VX.PS_PARTKEY
/* <OPTGUIDELINES>
 <HSJOIN>
  <IXSCAN TABLE="P"/>
  <ACCESS TABLE="VX" />
 </HSJOIN>
 <HSJOIN>
  <IXSCAN TABLE="S" />
  <IXSCAN TABLE="PS"/>
 </HSJOIN>
</OPTGUIDELINES> */
```

Must also unnest the join guidelines

Caveat – works for this example, but it might not in more complex scenarios

16

An alternative to using TABID is to rewrite the SQL statement so that the derived table is an inline view. Use the TABLE attribute to reference the inline view. This only works for simple inline views that have not been changed by automatic query transformations.

Table references in views

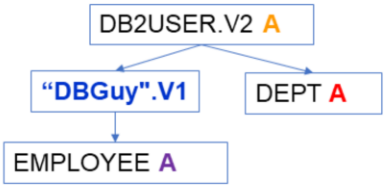
Tridex
A DB2 USER GROUP

- **Example**

```
CREATE VIEW "DBGuy".V1 as
  (SELECT * FROM EMPLOYEE A WHERE SALARY > 50000) ;

CREATE VIEW DB2USER.V2 AS (SELECT * FROM "DBGuy".V1, DEPT A
  WHERE A.MGR_ID IN ('52', '53','54') AND "DBGuy".V1.DEPTNO = A.DEPTNO );

SELECT * FROM DB2USER.V2 A WHERE A.HIRE_DATE > '01/01/2004'
/* <OPTGUIDELINES><IXSCAN TABLE='A/"DBGuy".V1/A'/></OPTGUIDELINES> */
```



- Extended syntax allows unambiguous table references in views
  - TABLE='A' is ambiguous and would return an error
- Extended name consists of exposed names in the path, from the statement reference to the nested reference, separated by slashes
- Same rules for exposed names apply to extended syntax

17

Optimization guidelines can use extended syntax to identify table references that are embedded in views. The extended syntax for identifying table references in views is a series of exposed names separated by a slash character. The value of the TABLE attribute A/"DBGuy".V1/A illustrates the extended syntax. The last exposed name in the sequence (A) identifies the table reference that is a target of the optimization guideline. The first exposed name in the sequence (A) identifies the view that is directly referenced in the original statement. The exposed name or names in the middle ("DBGuy".V1) pertain to the view references along the path from the direct view reference to the target table reference. The rules for referring to exposed names from optimization guidelines, described in the previous section, apply to each step of the extended syntax.

Had the exposed name of the EMPLOYEE table reference in the view been unique with respect to all tables that are referenced either directly or indirectly by the statement, the extended name syntax would not be necessary.

Extended syntax can be used to target any table reference in the original statement, SQL function, or trigger.

Table references in views

Tridex
A DB2 USER GROUP

- Extended syntax is not necessary if all exposed names for table references are unique
  - **Example**

```
CREATE VIEW "DBGuy".V1 as (SELECT * FROM EMPLOYEE E WHERE
    SALARY > 50,000) ;

CREATE VIEW DB2USER.V2 AS (SELECT * FROM "DBGuy".V1 WHERE
    DEPTNO IN ('52', '53','54') ;

SELECT * FROM DB2USER.V2 A WHERE V2.HIRE_DATE > '01/01/2004'
/* <OPTGUIDELINES><IXSCAN TABLE='E'/></OPTGUIDELINES> */ ;
```

18

Extended syntax is not necessary if all exposed names for table references are unique.

Ambiguous table references

Tridex
A DB2 USER GROUP

- **Example**

```
CREATE VIEW V1 AS
 (SELECT * FROM EMPLOYEE WHERE SALARY >
    (SELECT AVG(SALARY) FROM EMPLOYEE);

SELECT * FROM V1 WHERE DEPTNO IN ('M62', 'M63')
/* <OPTGUIDELINES><IXSCAN TABLE='V1/EMPLOYEE'/></OPTGUIDELINES> */ ;
```

- Which EMPLOYEE reference?
- The IXSCAN request is ignored
- Uniquely identify EMPLOYEE by adding correlation names in the view
- Use TABID
  - Correlation names in the optimized SQL are always unique

19

An optimization guideline is considered invalid and is not applied if it matches multiple exposed or extended names.

The optimizer considers the IXSCAN access request ambiguous, because the exposed name EMPLOYEE is not unique within the definition of view V1.

To eliminate the ambiguity, the view can be rewritten to use unique correlation names, or the TABID attribute can be used. Table references that are identified by the TABID attribute are never ambiguous, because all correlation names in the optimized statement are unique.

Conflicting optimization guidelines

Tridex
A DB2 USER GROUP

- Example

```
<OPTGUIDELINES>
    <IXSCAN TABLE='"Tpcd".PARTS' INDEX='I_PTYPE'/>
    <IXSCAN TABLE='"Tpcd".PARTS' INDEX='I_SIZE'/>
</OPTGUIDELINES>
```

- Multiple optimization guidelines can't reference the same table
- The first reference is applied and the others are ignored
- If I_PTYPE doesn't exist but I_SIZE does, the guideline is still ignored !

20

Each of the IXSCAN elements references the "Tpcd".PARTS table in the main subselect.

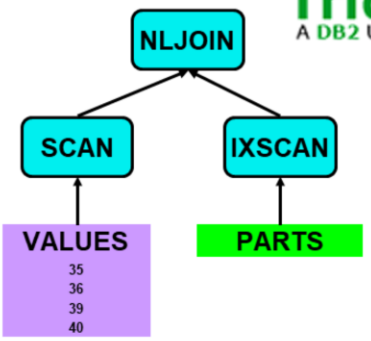
When two or more guidelines refer to the same table, only the first is applied; all other guidelines are ignored, and an error is returned.

Query Rewrite Guidelines

Tridex
A DB2 USER GROUP

- **Example:**

```
SELECT S.S_NAME ...
FROM
  "Tpcd".PARTS P, "Tpcd".SUPPLIERS S, "Tpcd".PARTSUPP PS
WHERE
  P_PARTKEY = PS.PS_PARTKEY AND
  S.S_SUPPKEY = PS.PS_SUPPKEY AND
  P_SIZE IN (35, 36, 39, 40) AND
  S.S_NATION IN ('INDIA', 'SPAIN')
ORDER BY S.S_NAME
/* <OPTGUIDELINES><INLIST2JOIN TABLE='P' /></OPTGUIDELINES> */
```



- INLIST2JOIN specifies that list of constants in IN list predicate should be transformed to an in-memory table (similar to a VALUES clause)
- In-memory table can then be joined to "Tpcd".PARTS using an indexed NLJN
  - Or a HSJN too. The join method can also be specified!
- Target IN-list identified by specifying table to which predicate is applied
- If there are multiple IN-lists for the same table, guideline can be further qualified with COLUMN attribute

21

This particular query rewrite optimization guideline specifies that the list of constants in the predicate P_SIZE IN (35, 36, 39, 40) should be transformed into a table expression. This table expression would then be eligible to drive an indexed nested-loop join access to the PARTS table in the main subselect. The TABLE attribute is used to identify the target IN-LIST predicate by indicating the table reference to which this predicate applies. If there are multiple IN-LIST predicates for the identified table reference, the INLIST2JOIN rewrite request element is considered ambiguous and is ignored.

In such cases, a COLUMN attribute can be added to further qualify the target IN-LIST predicate.

General Optimization Guidelines

Tridex
A DB2 USER GROUP

- **Example**

```
SELECT S.S_NAME, S.S_ADDRESS, S.S_PHONE
FROM "Tpcd".SUPPLIERS S
WHERE S.S_NATION IN (?, ?) AND S.S_SUPPKEY = ?
ORDER BY S.S_NAME
/* <OPTGUIDELINES> <REOPT VALUE='ONCE'/> </OPTGUIDELINES> */
```

- 'ONCE' indicates that optimization should be deferred until the first set of variable values is provided.
- This allows the optimizer to compare the input values to the statistics to get a better selectivity estimate and a better query execution plan

22

https://www.ibm.com/docs/en/db2/11.5?topic=wtqop-using-reopt-bind-option-input-variables-in-complex-queries

REOPT

Specifies whether to have Db2 optimize an access path using values for host variables, parameter markers, global variables, and special registers. Valid values are:

NONE

The access path for a given SQL statement containing host variables, parameter markers, global variables, or special registers will not be optimized using real values for these variables. The default estimates for the these variables will be used instead, and this plan is cached and used subsequently. This is the default behavior.

ONCE

The access path for a given SQL statement will be optimized using the real values of the host variables, parameter markers, global variables, or special registers when the query is first executed. This plan is cached and used subsequently.

ALWAYS

The access path for a given SQL statement will always be compiled and reoptimized using the values of the host variables, parameter markers, global variables, or special registers known at each execution time.

Putting an optimization profile into effect

Tridex
A DB2 USER GROUP

- Create the OPT_PROFILE table in the SYSTOOLS schema:

```
CALL SYSPROC.SYSINSTALLOBJECTS('OPT_PROFILES', 'C',
   CAST (NULL AS VARCHAR(128)), CAST (NULL AS VARCHAR(128)))
```

- Alternatively, can issue this DDL directly:

```
CREATE TABLE SYSTOOLS.OPT_PROFILE (
    SCHEMA VARCHAR(128) NOT NULL,
    NAME VARCHAR(128) NOT NULL,
    PROFILE BLOB (2M) NOT NULL,
PRIMARY KEY ( SCHEMA, NAME ));
```

- Compose document, validate, insert into table with qualified name

Inserts inventory_db.xml from current directory into the SYSTOOLS.OPT_PROFILE table with qualified name "DBA"."PROFILE1"

```
File profiledata:
    "DBA","PROFILE1","inventory_db.xml"

IMPORT FROM profiledata OF DEL MODIFIED BY LOBSINFILE
INSERT INTO SYSTOOLS.OPT_PROFILE;
```

23

https://www.ibm.com/docs/en/db2/11.5?topic=profiles-configuring-data-server-use-optimization-profile

After an optimization profile is created and its contents are validated against the current optimization profile schema (COPS), the contents must be associated with a unique schema-qualified name and stored in the SYSTOOLS.OPT_PROFILE table.

Putting an optimization profile into effect

Tridex
A DB2 USER GROUP

- Alternative way to insert optimization profile:

```
insert into systools.opt_profile values('DBA', 'PROFILE1',
blob('<?xml version="1.0" encoding="UTF-8"?>
<OPTPROFILE>
  <STMTPROFILE ID="Test 1">
    <STMTKEY>
      <![CDATA[select * from ta where a1 > 10 and a1 < 20]]>
    </STMTKEY>
    <OPTGUIDELINES><IXSCAN TABLE="TA" INDEX="TA_IX1"/></OPTGUIDELINES>
  </STMTPROFILE>
</OPTPROFILE>'));
```

- Cast the input string to BLOB
- But must deal with embedded quotes!

24

Another way to add an optimization profile is to just insert it in the SYSTOOLS.OPT_PROFILE table using an INSERT statement. However, embedded quotes need to be doubled and the length of the profile will be limited by the maximum size of a string literal.

Putting an optimization profile into effect

Tridex
A DB2 USER GROUP

- At the package level using the **OPTPROFILE** bind option
- For existing packages, use ALTER PACKAGE:

```
ALTER PACKAGE DB2USER.EMPADMIN OPTIMIZATION PROFILE DBA.PROFILE1

/* Bind optimization profile "DBA"."PROFILE1 to the package "inventapp" */
DB2 PREP INVENTAPP.SQC BINDFILE OPTPROFILE DBA.PROFILE1
DB2 BIND INVENTAPP.BND
```

- At the dynamic statement level: using **CURRENT OPTIMIZATION PROFILE** special register

```
/* Optimize statements using profile 'DBA.PROFILE1' */
SET CURRENT OPTIMIZATION PROFILE = 'DBA.PROFILE1';

/* Optimize statements using profile 'JON.SALES' */
SET CURRENT SCHEMA = 'JON';
SET CURRENT OPTIMIZATION PROFILE = 'SALES';
```

25

An optimization profile needs to be put into effect for the connection/session or package.

An optimization profile can be specified for a new package using the OPTPROFILE bind option. It can be specified for existing packages by using the ALTER PACKAGE statement.

An optimization profile can be put into effect for dynamic SQL for the current connection/session using the CURRENT OPTIMIZATION PROFILE special register.

An optimization profile has a 2-part name. The name can be specified with a literal, host variable, or special register. The name specified is the name entered into the CURRENT OPTIMIZATION PROFILE special register. If the specified optimization-profile-name is unqualified, the value of the CURRENT DEFAULT SCHEMA register is used as the implicit qualifier. The default value of the special register is null.

Putting an optimization profile into effect

Tridex
A DB2 USER GROUP

```
/* Clear the special register and use the setting of the OPTPROFILE bind option */
SET CURRENT OPTIMIZATION PROFILE = NULL;

/* Don't use optimization profile  at all /
SET CURRENT OPTIMIZATION PROFILE = "";
```

- At the dynamic statement level: using db2_optprofile CLI option

```
-- After each successful connect to the SANFRAN database, the CLI client
   would issue the command:

   SET CURRENT OPTIMIZATION PROFILE=JON.SALES.

[SANFRAN]
DB2_OPTPROFILE JON.SALES
```

26

•If the value of the register specifies the name of an existing optimization profile, the specified optimization profile is used when preparing subsequent dynamic DML statements.

•If the value of the register is null, the optimization profile specified by the OPTPROFILE bind option, if any, is used when preparing subsequent dynamic DML statements.

•If the value of the register is null, and the OPTPROFILE bind option is not set, no optimization profile is used when preparing subsequent dynamic DML statements.

•If the value of the register is the empty string, then no optimization profile is used when preparing subsequent dynamic DML statements, regardless of whether the OPTPROFILE bind option is set.

•Subsequent changes to CURRENT DEFAULT SCHEMA do not have any effect on the optimization profile. The CURRENT OPTIMIZATION PROFILE register value is set with the two part name that is in effect at the time SET CURRENT OPTIMIZATION PROFILE statement is evaluated. Only another SET CURRENT OPTIMIZATION PROFILE statement can change the optimization profile that is used.

Putting an optimization profile into effect

Tridex
A DB2 USER GROUP

- Use a connect procedure to put an optimization profile in effect
  - Stored procedure that is executed when DB connection is established
- Avoids issuing an explicit SET CURRENT OPTIMIZATION PROFILE within each connection
- Useful for setting general optimization guidelines
- Can include conditions to set different profiles based on user or client information

```
-- Create the connection procedure
CREATE PROCEDURE DBA.CONNECTPROC ( )
READS SQL DATA
LANGUAGE SQL
 if (session_user like 'APP1%' then
   set current optimization profile 'OPTPROF_APP1'
 elseif
   set current optimization profile 'OPTPROF_APP2'
 end if @

-- Register the connection procedure in the DB config
db2 update db cfg for <dbname> using connect_proc "DBA.CONNECTPROC";
```

27

The connect procedure provides you a way to allow applications in your environment to implicitly execute a specific procedure upon connection. This procedure can allow you to customize an application environment to a database from a central point of control. For example, in the connect procedure you can set special registers such as CURRENT_PATH to non-default values by invoking the SET CURRENT PATH statement. This new CURRENT_PATH value will now be the effective default CURRENT_PATH for all applications.

Any procedure created in the database that conforms to the naming and parameter restrictions can be used as the connect procedure for that database. The customization logic is provided by you in the form of a procedure created in the same database and is allowed to do any of the usual actions of a procedure such as issue SQL statements.

https://www.ibm.com/docs/en/db2/11.5?topic=databases-customizing-application-environment-using-connect-procedure

Updating an existing optimization profile

Tridex
A DB2 USER GROUP

- Re-insert an updated XML document in SYSTOOLS.OPT_PROFILE
- Optimization profiles are parsed and cached to reduce overhead when used for different statements
- Must flush the cache in order to pick up updated version:
  - Flush the entire cache:
    FLUSH OPTIMIZATION PROFILE CACHE
  - Flush a cache entry for a specific optimization profile
    FLUSH OPTIMIZATION PROFILE CACHE SALES

28

https://www.ibm.com/docs/en/db2/11.5?topic=profiles-modifying-optimization-profile

When you make a change to an optimization profile there are certain steps that need to be taken in order for the changes in the optimization profiles to take effect.

When an optimization profile is referenced, it is compiled and cached in memory; therefore, these references must also be removed. Use the FLUSH OPTIMIZATION PROFILE CACHE statement to remove the old profile from the optimization profile cache. The statement also invalidates any statement in the dynamic plan cache that was prepared by using the old profile (logical invalidation).

Handling SQL in procedures

Tridex
A DB2 USER GROUP

- Use **SET_ROUTINE_OPTS** to specify the optimization profile

```
CALL SET_ROUTINE_OPTS('OPTPROFILE DBA.INVENTDB ') %

CREATE PROCEDURE MY_PROC
BEGIN
      DECLARE CUR1 CURSOR FOR SELECT ...
END %
```

- SQL might be modified during CREATE PROCEDURE processing
- Use explain facility or query system catalogs to get the modified SQL statements to include in optimization profile STMTKEY element for profile statement matching
  - See speaker notes for an example

29

STMTNO should be the line number in the source code of the CREATE PROCEDURE, relative to the beginning of the procedure statement (line number 1)

```
SELECT STMTNO, SEQNO, SECTNO, TEXT
  FROM SYSCAT.STATEMENTS AS S,
       SYSCAT.ROUTINEDEP AS D,
       SYSCAT.ROUTINES   AS R
 WHERE PKGSCHEMA = BSCHEMA
   AND PKGNAME = BNAME;
   AND BTYPE = 'K'
   AND R.SPECIFICNAME = D.SPECIFICNAME
   AND R.ROUTINESCHAME = D.ROUTINESCHEMA
   AND ROUTINENAME = ?
   AND ROUTINESCHEMA = ?
   AND PARM_COUNT = ?
 ORDER BY STMTNO
```

Embedded optimization guidelines – precedence order (1|2)

Tridex
A DB2 USER GROUP

- Embedded overrides identical optimization guidelines specified in the global section of an optimization profile e.g.

```
SELECT COUNT(*) FROM TAB1
/* <OPTGUIDELINES> <REOPT VALUE='ONCE'/> </OPTGUIDELINES> */;
```

Takes precedence over this active optimization profile:

```
<?xml version="1.0" encoding="UTF-8"?>
<OPTPROFILE VERSION="11.5.0">
 <OPTGUIDELINES> <REOPT VALUE='ALWAYS'/> </OPTGUIDELINES>
</OPTPROFILE>
```

30

Embedded optimization guidelines override identical optimization guidelines specified in the global section of an optimization profile.

Embedded optimization guidelines – precedence order (2|2)

Tridex
A DB2 USER GROUP

- Statement profiles override embedded guidelines e.g.

```
SELECT COUNT(*) FROM TAB1
/* <OPTGUIDELINES> <REOPT VALUE='ONCE'/> </OPTGUIDELINES> */;
```

Is ignored with this active optimization profile:

```
<?xml version="1.0" encoding="UTF-8"?>
<OPTPROFILE VERSION="11.5.0">
<STMTPROFILE ID="STMTPROF1">
    <STMTKEY> <![CDATA[SELECT COUNT(*) FROM TAB1]]> </STMTKEY>
    <OPTGUIDELINES> <IXSCAN TABLE="TAB1"/> </OPTGUIDELINES>
</STMTPROFILE>
</OPTPROFILE>
```

31

Optimization guidelines provided by way of a statement profile section of an optimization profile take precedence over embedded optimization guidelines. That is, if the CURRENT OPTIMIZATION PROFILE register contains the name of an optimization profile, and the specified optimization profile contains a matching statement profile for a statement with embedded optimization guidelines, then the embedded optimization guidelines are ignored by the optimizer.

Inexact SQL Statement Matching (1|4)

Tridex
A DB2 USER GROUP

- Default: SQL statement text must be an exact match, other than white space
- It is difficult to create optimization profiles for complex query workloads
  - Queries might have same 'shape' but only differ by literals in predicates
- Match predicates on same columns with same relational operators, but different literal values:
  NAME = 'Joe' ←→ NAME = 'Bob'
  IN (1,2,3) ←→ IN (4,5,6)
- Match IN-list predicates with different numbers of items:
  IN (1,2) ←→ IN (1,2,3)
  IN (?,?) ←→ IN (?,?,?)
- Match predicates with references to different host variables:
  A = :hv1 ←→ A = :hv2

32

The statement key identifies the application statement to which statement-level optimization guidelines apply. The matching method can be specified using the STMTMATCH element in the optimization profile.

When the data server compiles an SQL statement and finds an active optimization profile, it attempts to match each statement key in the optimization profile with the current compilation key. The type of matching depends if exact or inexact matching is specified in the optimization profile. You can specify which type of matching to use by specifying the STMTMATCH element in the optimization profile. By setting the EXACT attribute to TRUE or FALSE, you can enable either exact or inexact matching. If you do not specify the STMTMATCH element, exact matching is automatically enabled.

Inexact SQL Statement Matching (2|4)

Tridex
A DB2 USER GROUP

- Inexact matching applies to all literals in the statement
- Different special registers will not match
- Statements that won't match:

```
C1 BETWEEN 5 AND :HV
5 BETWEEN C1 AND C2

C1 IN (SELECT C1 FROM T1)
C1 IN (1,2,3)

C1 IN (C1, 1, 2)
C1 IN (C2, 1, 2)
```

```
A = 5
A = 5 + :HV

WITH RR
WITH RS

C2 < CURRENT TIME
C2 < '11:12:40'

C3 > CURRENT TIMESTAMP
C3 > '07/29/2010'
```

33

With inexact matching, literals, host variables, and parameter markers are ignored when the statement text from the statement key and compilation key is being matched.

Exact and inexact matching operates as follows:

Matching is case insensitive for keywords. For example, select can match SELECT.

Matching is case insensitive for nondelimited identifiers. For example, T1 can match t1.

Delimited and nondelimited identifiers can match except for one case. For example, T1 and "T1" will match, and so will t1 and "T1". However, t1 will not match with "t1".

Inexact SQL Statement Matching (3|4)

Tridex
A DB2 USER GROUP

- Applies to XQuery too
  - Except for literals passed as function parameters representing SQL statement text or column names
  - XMLQUERY, XMLEXISTS and XMLTABLE are always exactly matched
- Inexact matching is specified using **STMTMATCH** attribute
- Can be specified at the global or statement level
  - Global: applies to all statements executed when profile is active
- Statement level specification takes precedence over global level

34

Inexact matching is applied to both SQL and XQuery statements. However, string literals that are passed as function parameters representing SQL or XQuery statements or statement fragments, including individual column names are not inexactly matched. XML functions such as XMLQUERY, XMLTABLE, and XMLEXISTS that are used in an SQL statement are exactly matched. String literals could contain the following items:

- A whole statement with SQL embedded inside XQuery, or XQuery embedded inside an SQL statement
- An identifier, such as a column name
- An XML expression that contains a search path

For XQuery, inexact matching ignores only the literals. The following literals are ignored in inexact matching with some restrictions on the string literals:

- decimal literals
- double literals
- integer literals
- string literals that are not input parameters for functions: db2-fn:sqlquery, db2-fn:xmlcolumn, db2-fn:xmlcolumn-contains

Inexact SQL Statement Matching (4|4)

Tridex
A DB2 USER GROUP

```
<?xml version="1.0" encoding="UTF-8"?>
<OPTPROFILE>
<!--Global section -->
  <STMTMATCH EXACT='FALSE'/>
<!-- Statement level profile -->
  <STMTPROFILE ID='S1'>
    <STMTMATCH EXACT='TRUE'/>
    <STMTKEY>
      <![CDATA[select t1.c1, count(*) from t1,t2 where t1.c1 =
  t2.c1 and t1.c1 > 0]]>
    </STMTKEY>
    <OPTGUIDELINES>
      <NLJOIN>
        <TBSCAN TABLE='T1'/>
        <TBSCAN TABLE='T2'/>
      </NLJOIN>
    </OPTGUIDELINES>
  </STMTPROFILE>
  <STMTPROFILE ID='S2'>
    <STMTKEY>
      <![CDATA[select * from t1 where c1 in (10,20,30)]]>
    </STMTKEY>
    (etc)
</OPTPROFILE>
```

- Statement S1 will use exact matching
- Statement S2 will use inexact matching

35

This example shows how inexact matching can be specified globally for all statements executed when the profile is in effect, or locally for a specific SQL statement. The global setting is EXACT='FALSE', so exact matching is not done for statement S1. Statement S2 has its own STMTMATCH element with EXACT='TRUE' so its text will be matched exactly.

Optimization Profile SQL Statement Matching

Tridex USER GROUP

- Other flexible matching done by default:
  - Match any SQL keywords regardless of case e.g.
    SELECT ←→ SeLecT
  - Match delimited and undelimited identifiers e.g.
    "T1" ←→ T1

36

Matching is case insensitive for keywords. For example, select can match SELECT.

Matching is case insensitive for nondelimited identifiers. For example, T1 can match t1.

Delimited and nondelimited identifiers can match except for one case. For example, T1 and "T1" will match, and so will t1 and "T1". However, t1 will not match with "t1".

Specifying SQL Compiler Registry Variables

Tridex
A DB2 USER GROUP

- SQL compiler registry variables only affect SQL compilation and are dynamic
  - i.e. no instance restart is necessary (db2set –im)

```
<OPTGUIDELINES>
      <REGISTRY>
         <OPTION NAME='DB2_ANTIJOIN' VALUE='EXTEND'/>
         <OPTION NAME='DB2_REDUCED_OPTIMIZATION' VALUE='YES'/>
      </REGISTRY>
</OPTGUIDELINES>
```

- Can be specified at the global or statement level
- Allows control at the package or statement level
- TIP: Use with a connect proc to limit SQL compiler registry variables based on query environment
  - E.g. user, various client attributes, etc.
- Usage is identified in the explain output
  - ENVVAR argument of the RETURN plan operator:

```
ENVVAR  : (Environment Variable)
          DB2_ANTIJOIN=NO[Global Optimization Guideline]
```

37

Optimization profiles can have different registry variable values applied to a specific query statement or to many query statements used in an application.

Setting registry variables in an optimization profile can increase the flexibility you have in using different query statements for different applications. When you use the db2set command to set registry variables, the registry variable values are applied to the entire instance. In optimization profiles, the registry variable values apply only to the statements specified in the optimization profile. By setting registry variables in an optimization profile, you can tailor specific statements for applications without worrying about the registry variable settings of other query statements.

Only a subset of registry variables can be set in an optimization profile.

See here for the full list:

https://www.ibm.com/docs/en/db2/11.5?topic=profiles-sql-compiler-registry-variables-in-optimization-profile

Optimization guideline construction – access requests

- Access requests – specify desired method for satisfying table reference
- Correspond to Db2 data access methods
  - ANY (let the optimizer choose the base access)
  - TBSCAN (table scan)
  - IXSCAN (index scan)
  - LPREFETCH (list prefetch)
  - IXAND (index ANDing)
  - IXOR (index Oring)
  - XISCAN (XML index scan)
  - XANDOR (XML index ANDing and ORing)
  - ACCESS (any access type)

38

•TBSCAN, IXSCAN, LPREFETCH, IXAND, IXOR, XISCAN, and XANDOR

These elements correspond to Db2® data access methods, and can only be applied to local tables that are referenced in a statement. They cannot refer to nicknames (remote tables) or derived tables (the result of a subselect).

•ACCESS

This element, which causes the optimizer to choose the access method, can be used when the join order (not the access method) is of primary concern. The ACCESS element must be used when the target table reference is a derived table. For XML queries, this element can also be used with attribute TYPE = XMLINDEX to specify that the optimizer is to choose XML index access plans.

Optimization profile construction – join requests

Tridex
A DB2 USER GROUP

- Join requests – specify desired method and order for joining tables
- Contain access or other join requests
- Correspond to Db2 join methods
  - NLJOIN
  - HSJOIN
  - MSJOIN
  - JOIN (any join type)

39

•NLJOIN, MSJOIN, and HSJOIN

These elements correspond to the nested-loop, merge, and hash join methods, respectively.

•JOIN

This element, which causes the optimizer to choose the join method, can be used when the join order is not of primary concern.

All join request elements contain two sub-elements that represent the input tables of the join operation. Join requests can also specify an optional FIRST attribute.

Optimization profile construction – rewrite requests

Tridex
A DB2 USER GROUP

- Specify query rewrite transformations
- Correspond to Db2 query rewrite transformation rules
  - INLIST2JOIN (IN-list predicate to join)
  - NOTEX2AJ (NOT EXISTS subquery to anti-join)
  - NOTIN2AJ (NOT IN subquery to anti-join)
  - SUBQ2JOIN (Subquery to join)

40

•**IN-LIST-to-join query rewrite requests**
A INLIST2JOIN query rewrite request element can be used to enable or disable the IN-LIST predicate-to-join rewrite transformation. It can be specified as a statement-level optimization guideline or a predicate-level optimization guideline. In the latter case, only one guideline per query can be enabled. The INLIST2JOIN request element is defined by the complex type inListToJoinType.

•**NOT-EXISTS-to-anti-join query rewrite requests**
The NOTEX2AJ query rewrite request element can be used to enable or disable the NOT-EXISTS predicate-to-anti-join rewrite transformation. It can be specified as a statement-level optimization guideline only. The NOTEX2AJ request element is defined by the complex type notExistsToAntiJoinType.

•**NOT-IN-to-anti-join query rewrite requests**
The NOTIN2AJ query rewrite request element can be used to enable or disable the NOT-IN predicate-to-anti-join rewrite transformation. It can be specified as a statement-level optimization guideline only. The NOTIN2AJ request element is defined by the complex type notInToAntiJoinType.

•**Subquery-to-join query rewrite requests**
The SUBQ2JOIN query rewrite request element can be used to enable or disable the subquery-to-join rewrite transformation. It can be specified as a statement-level optimization guideline only. The SUBQ2JOIN request element is defined by the complex type subqueryToJoinType.

Problem determination

Tridex
A DB2 USER GROUP

- Invalid optimization guidelines are ignored
- SQL code +437 reason code 13 is returned
- Doesn't prevent the SQL statement from running
- Use the explain facility to understand why guideline wasn't applied
- 2 Explain tables for diagnostics
  - EXPLAIN_DIAGNOSTICS
  - EXPLAIN_DIAGNOSTICS_DATA
- db2exfmt will provide the formatted messages
- A table function is also available
  - EXPLAIN_GET_MSGS
- Use explain facility to verify if optimization profiles have been applied:

```
Profile Information:
--------------------
OPT_PROF: (Optimization Profile Name)
        DBA.PROFILE1
STMTPROF: (Statement Profile Name)
        SQL1
```

41

If OPT_PROF is missing, that means that an optimization profile wasn't in effect when the statement was explained. Check the CURRENT OPTIMIZATION PROFILE special register or OPTPROFILE bind option.

If OPT_PROF appears but STMTPROF doesn't, and you expected one to be in effect, then the SQL statement text probably didn't match.

Problem determination

Tridex
A DB2 USER GROUP

- Example diagnostic messages:
  - EXP0004W The optimization profile or embedded optimization guideline is either not well-formed or is invalid. Line number ”554”, character number ”20”.
  - EXP0012W Invalid access request. The index "index2" could not be found. Line number "573", character number "20".
  - EXP0009W Invalid access request. The table reference identified by the TABLE attribute could not be found. Line number "573", character number "20"
- Line number
  - Relative to position within the XML document
- Character number
  - Relative to beginning of line specified by Line number

42

Some explain diagnostic messages for optimization profiles provide the line number relative to the start of the XML document as well as a character number relative to the beginning on the specified line.

Summary

Tridex
A DB2 USER GROUP

- Help the optimizer do its job for all queries by providing it good information:
  - Statistics
  - System configuration information
  - Input variable values
  - Schema information via DB constraints
- Tune specific queries, where the above approaches fail
- Use more direct approaches like optimization profiles as a last resort
- Optimization profiles provide a high degree of control over query transformations and the access plan

Thank You

Speaker: John Hornibrook
Company: IBM Canada
Email Address: jhornibr@ca.ibm.com

John is a Senior Technical Staff Member responsible for relational database query optimization on IBM's distributed platforms. This technology is part of Db2 for Linux, UNIX and Windows, Db2 Warehouse, Db2 on Cloud, IBM Integrated Analytics System (IIAS), IBM Db2 Analytics Accelerator (IDAA) and Db2 Big SQL. John also works closely with customers to help them fully realize the benefits of IBM's relational DB technology products.