IBM® DB2® Content Manager
Query Performance Guide

Document Version 1.1 (June 27, 2008)

Table of Contents

1 Introduction

IBM DB2 Content Manager Version 8 offers powerful capabilities to query (search) the data stored in the DB2 Content Manager system. This document provides insights into possible performance implications when using the DB2 Content Manager query component, ways to debug performance issues, suggestions on how to make performance improvements, instructions on how to gather logging data, and other performance-related information. This guide is a living document and can be updated periodically.

2 Disclaimer

Performance improvements recommended in this guide are not guaranteed to work in all circumstances. Performance troubleshooting is affected by many factors, including the workload scenario and the system environment. Performance issues have many different causes and aspects to them. Some performance issues might not have any suitable improvements, depending on the complexity of the queries, volume of data, data structure, and many other factors. Accordingly, IBM does not provide any representations, assurances, guarantees, or warranties regarding performance. Performance improvements might also need to be made in other related products used with IBM DB2 Content Manager as appropriate.

The information contained in this publication was derived under specific operating and environmental conditions. Furthermore, it has not been submitted to any formal IBM test and is distributed on an "as is" basis without any warranty either expressed or implied. The use of this information or the implementation of any of these techniques is the responsibility of the user and depends on the user's ability to evaluate and integrate them into the user's operational environment. While each item might have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Users attempting to adapt these techniques to their own environments do so at their own risk.

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM licensed program in this publication is not intended to state or imply that only IBM's program can be used. Any functionally equivalent program can be used instead. Equivalent product, program, or services that do not infringe any of the intellectual property rights of IBM can be used instead of the IBM product, program, or service.

The evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, are the responsibility of the user.

3 Target audience

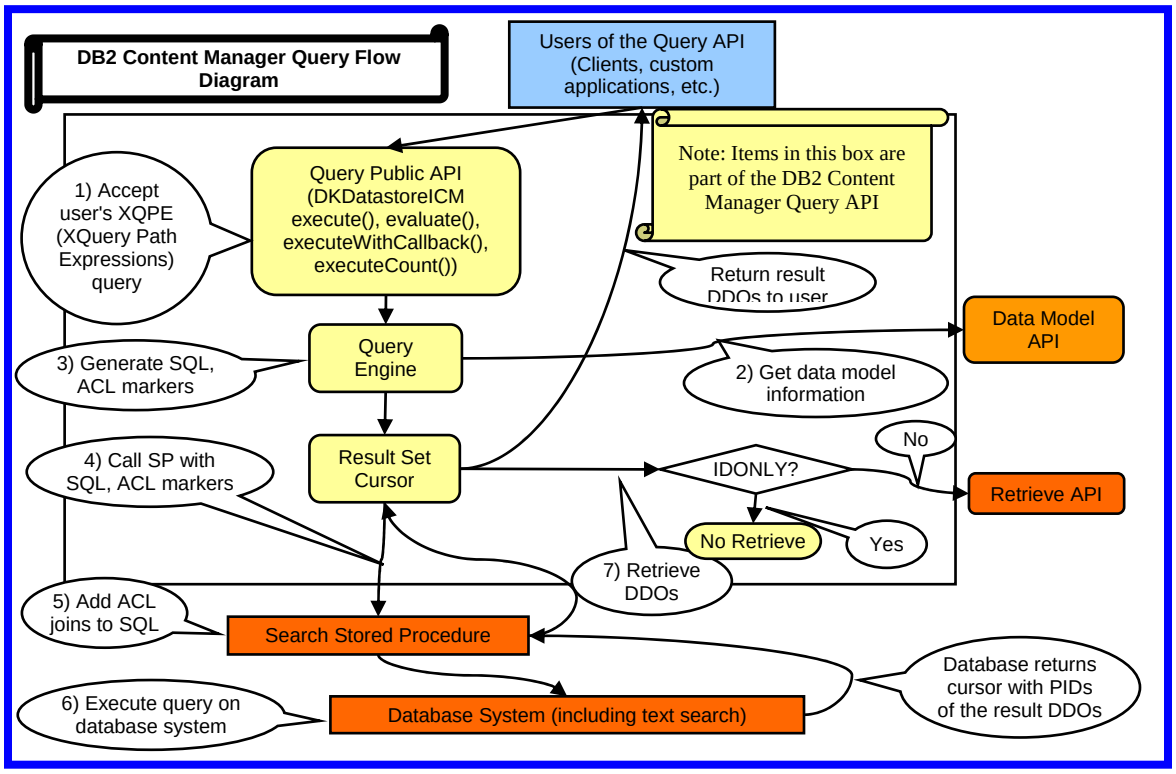
This document is primarily intended for DB2 Content Manager external and internal users who need to find ways to increase efficiency and resolve performance issues associated with DB2 Content Manager queries. Such users typically include the following:

- Customers and partners
  - Application designers and developers
  - Systems and database administrators
- IBM internal teams
  - Level 2 (L2) and Level 3 (L3) IBM technical support representatives
  - Other DB2 Content Manager technical teams

4 Brief Overview of the DB2 Content Manager Query Processing Flow

DB2 Content Manager Query provides a simple and efficient way to find data stored in DB2 Content Manager's datastore. Figure 1 shows a high-level diagram that depicts the steps involved in processing a DB2 Content Manager Query (including the interactions with different DB2 Content Manager modules). A query received by the CM Query API is translated into SQL by using the information contained in the DB2 Content Manager Data Model API. The SQL is then sent to the Search Stored Procedure, where security conditions (ACLs) are added and the query is executed. The result of the query is a list of IDs (called PIDs) that are then used to create DDOs to return to the caller. The contents of the DDOs can be retrieved by using the Retrieve API. Further details about the DB2 Content Manager Query are available in the DB2 Content Manager Application Programming Guide.

Figure 1: DB2 Content Manager Query Flow Diagram
Disclaimer: Internal stages of processing are subject to change at any time without notice



5 Common Ways to Analyze and Address Query Performance Issues

Here are some typical analysis methods for analyzing DB2 Content Manager query performance problems:

a. Verify performance expectations by checking how many items are present in the system, how many are in the tables that are being queried, and how many are supposed to be returned.
b. Perform basic database tuning (such as reorg/runstats/rebind) to optimize performance of the database system operations.
c. Narrow down the time spent in each component by using the TSearchICM tool or other analysis methods (see recommendations for analysis in this document).
d. Check the simplicity and effectiveness of your XQPE (XQuery Path Expressions) query and whether there are any available rewrites to your XQPE query. Some common rewrites are listed in the later section.
e. Determine whether further database tuning improves execution time of the SQL on the Library Server database:
   - Create database indexes on attributes and system columns
   - Examine database execution plan to evaluate costs and access paths
   - ACL conditions are added by the Library Server, so you may need to use the full SQL from the Library Server logs, not just the core SQL statement available from the API log

f. If performance impact comes from long processing in retrieving DDOs, consider using new optimized set of retrieve options described in the DKRetrieveOptionsICM class.
g. Evaluate possibilities for better data model design. For example, splitting the item type into several item types (e.g. by date or some other criteria) can help to reduce the number of items in the user item type table.

6 Database Tuning

Here are several steps to try at the beginning of analysis of query performance problems:

a. Updating database statistics.
   Before beginning the full performance troubleshooting process, ensure that the database statistics for the Library Server and Resource Manager databases are up to date. DB2 Content Manager is a significant, large-scale database application, and depends on accurate database statistics for optimizing all operations. Updating database statistics on the Library Server and Resource Manager databases relatively straightforward to do, and often turns out to be the only fix needed. There are several other publications (see the Reference section at the end of this document) that show how to accomplish these tasks. For example, the CM v8.3 Performance Monitoring Guide provides sample scripts which should be routinely executed periodically as part of maintaining performance.
b. Creating necessary database indices (indexes).
   The Library Server and the Resource Manager databases are installed with the appropriate indexes to optimize create, retrieve, update, and delete operations. When you define a new item type, the appropriate indexes are created automatically. Indexes on user-defined item attributes are not generated automatically. For the attributes that are frequently used for regular searches, creating indexes on these attributes often improves response time and reduces resource usage for queries using these attributes. Please note that usefulness of database indexes depends on type of queries used. For example, using a wildcard at the beginning of the search term (e.g. @Title LIKE "%Java%") may prevent the database system from effectively using the index. More information on this is available in the sections below and in the database documentation.

   The best way to create a new database index is by using the DB2 Content Manager System Administration Client. This way, the index will not only contain the column representing the user attribute, but also the other system columns that are necessary for optimal performance of database joins typically performance in the Content Manager environment. After creating a new index, run the `reorg`, `runstats`, and `rebind` commands to make sure that the DB2 UDB optimizer can effectively use the new index.
c. Optimizing an Oracle database.

A database table can become fragmented after many updates, causing the query performance to deteriorate. Queries might take longer because index entries in the library server and resource manager are no longer synchronized with the actual data in the database tables.

For an Oracle database, you can use the tools provided by Oracle to update the indexes and re-order as necessary. Below are the SQL commands that can improve the performance of your system. To use the commands, you must log in as sysdba. You should work with your Oracle DBA to add these SQL commands to a batch process to be run periodically.

In the following commands, replace ICMADMIN with the library server database administration ID, and replace RMADMIN with your resource manager database administration ID.

Library server:

```
    execute dbms_stats.gather_schema_stats (ownname=>'ICMADMIN',
method_opt=>'FOR ALL COLUMNS SIZE 1',granularity=>'ALL',
options=>'GATHER', cascade=>TRUE, degree=>16);
    execute dbms_stats.gather_schema_stats (ownname=>'SYS',
method_opt=>'FOR ALL COLUMNS SIZE 1',granularity=>'ALL',
options=>'GATHER', cascade=>TRUE, degree=>16);
    execute dbms_stats.gather_schema_stats (ownname=>'SYSTEM',
method_opt=>'FOR ALL COLUMNS SIZE 1',granularity=>'ALL',
options=>'GATHER', cascade=>TRUE, degree=>16);
```

Resource manager:

```
    execute dbms_stats.gather_schema_stats (ownname=>'RMADMIN',
method_opt=>'FOR ALL COLUMNS SIZE 1',granularity=>'ALL',
options=>'GATHER', cascade=>TRUE, degree=>16);
```

7 Narrowing Down Query Performance Problems

When investigating why "a query takes too long," it is important to figure out how much time is spent in which component. This way, the issue can be narrowed down to the component that has the most impact on the overall execution time. A query that returns several million results may be able to rather quickly get the list of PIDs for the results, but may actually take a rather long time to retrieve all of the DDO contents for those results, as expected. By narrowing down the times spent in each component, it is possible to get a better sense of whether the times are reasonable when compared to the amount of work to be performed by that component. The query flow diagram presented earlier should help to provide the background on what components (or modules) are involved in query processing. Recommendations that follow provide tips on how to actually determine how much time is spend in each stage of processing.

Preparation for performance analysis:

First make sure that there are no other processes taking up the CPU, memory, hard disks or other system resources on the machine used to measure performance. Utilization of the machine by other programs and processes could skew results and increase the analysis time.

Some methods for narrowing down time spent in different stages:

a. Use the TSearchICM sample program.
   - TSearchICM is a command-line tool available to simplify testing of DB2 Content Manager queries and problem determination of performance issues. With this tool, you can run queries with different query options and get basic information about query performance. TSearchICM is easy to use with good instructions and has been successfully used by IBM support teams and customers in multiple support engagements.
   - TSearchICM is available as a sample program as part of the II4C installation in both Java and C++ (either in the standard samples directories <IBMCMROOT>\samples\java\icm\TSearchICM.java and <IBMCMROOT>\samples\cpp\icm\TSearchICM.cpp or as a free download from the IBM DB2 Content Manager support web site).
   - TSearchICM standardizes the information gathering for DB2 Content Manager query customer requests (PMRs) and reproduces the problem in the customer's environment.
   - TSearchICM also produces performance information from the query and allows running the same query multiple times to get closer approximation of an average runtime.
   - In addition to query functionality, TSearchICM also allows the user to specify different retrieve options to compare the effect on the overall performance.

b. Perform query without retrieving results.
   - Using either the TSearchICM tool or the custom application, you can specify options such that the DB2 Content Manager query is performed without actually retrieving the result DDOs. This is often a quick way to determine whether the time is spent mostly in the query or in the retrieve stages of processing.
   - Using the TSearchICM tool, see the help provided with the tool for ways to specify the necessary retrieve options such that no results are retrieved.
   - Using a custom application, you can use the DK_CM_CONTENT_IDONLY retrieve option or no options with the DKRetrieveOptionsICM class when passing retrieve options to the query methods. With this approach, the query API performs no retrieval of result items. Then the application can explicitly do retrieval when it is necessary.
   - You can also use the DKDatastoreICM.execute method. With this method, results are not retrieved until the first result is requested by the application.

After the first result is requested, however, a block of results is retrieved (for optimization purposes). Unlike the DKDatastoreICM.execute method, other query methods, such as DKDatastoreICM.evaluate and DKDatastoreICM.executeWithCallback, retrieve results immediately. These results are retrieved either in a result collection in the case of DKDatastoreICM.evaluate method or in smaller collections that are sent to the dkCallback.dataCallback method by the DKDatastoreICM.executeWithCallback method. The DK_CM_CONTENT_IDONLY retrieve option or no options with the DKRetrieveOptionsICM class can also be used with the DKDatastoreICM.execute method to prevent the query API from performing internal retrieval, allowing the application to control explicitly when the retrieve function is called. Note that when you are using DKDatastore.execute method, a new database connection is created to run the query. For more details about the behavior of the DKDatastore.execute method, see the "Searching for data" section in the IBM DB2 Content Manager Application Programming Guide.

c. Analyze the logs.
    - Reviewing the DB2 Content Manager logs is often the most comprehensive way to determine the time spent in each component.
    - Log analysis also may be an initial step to using the TSearchICM tool if the DB2 Content Manager query string is generated by one of the CM client applications or if you are not sure which particular query takes a long time to complete.
    - Make sure to properly set log options to simplify your own analysis and to prepare the logs to be available for IBM support teams, if necessary. For details, see the section of this document about the information needed for support analysis.
    - Execute only a single query at a time. Remember to clear the logs before analyzing the next query.
    - Refer to the DB2 Content Manager Query Flow Diagram for reference of the different stages of processing. Note the times in the log that correspond to each stage based on log comments.
    - The SQL statement generated based on the customer query is available in both the API and the Library Server logs. The Library Server log will have the most complete SQL string that also includes the ACL join conditions. Often, it can be useful to take the SQL statement from the log and run it directly in the database system control window to evaluate the performance of the query outside of the Content Manager environment.

8 Common Rewrites to DB2 Content Manager Queries

This section contains some common ways to rewrite DB2 Content Manager queries to make them more efficient. These modifications can be made by users of the DB2 Content Manager system in the code of the DB2 Content Manager custom application.

The recommended way to use this section is to first scan through the titles and summaries of each rewrite to find the scenarios that best apply to your situation.

8.1 ICM Query: Improving performance of "child OR" queries

Question

How can I improve performance of IBM DB2 Content Manager queries that combine conditions on both root and child components with the OR operator?

Answer

Depending on the underlying database system, certain queries that use OR conditions might lead to poor performance. There are several possible rewrites to IBM DB2 Content Manager XPath-based queries available that could improve performance.

Before you begin the full performance troubleshooting process, first ensure that the database statistics for the library server and resource manager databases are up to date. If updating database statistics does not sufficiently improve performance, you can try the following rewrites:

Original query:

"/Journal [@Title = \"XML\" OR Journal_Article/@Title = \"XML\"]"

This query finds all Journals that either have a title of "XML" or that contain an article with that title.

Possible rewrite #1:

"/Journal [@Title = \"XML\" OR Journal_Article[@Title = \"XML\"]]"

Note that the meaning of the query is the same as before, however, in this case the condition on the child component view is written somewhat differently to mean "does there exist a Journal article child component, for which the Title is XML?"

Possible rewrite #2:

"/Journal [@Title = \"XML\"] UNION /Journal [Journal_Article/@Title = \"XML\"]"

Again, the meaning of the rewritten query is the same as that of the original query. However, in this case a union of two sets of Journals is used instead of multiple conditions on a single set of Journals.

8.2 ICM Query: Query optimization for queries which contain conditions on child component attributes

Problem

I see poor performance when I run a query to return items that include conditions on both the root component attributes and child component attribute. Is there a way to optimize these queries?

Cause

The backend database tries to pick the best access plan for any given query based on statistics and other intelligence. In some cases for this particular situation, some databases might choose the most optimal plan and could choose a plan that results in poor performance. Among reasons for the choice, a less-than optimal plan can result if the statistics are not current for the database.

Additionally, the same query can be written several ways. Your query is turned into SQL, the structure of which can vary depending on your exact original query string. Small differences in the original query can yield differences in the SQL that could cause the database to choose a different access plan.

Solution

For problems of this nature, you should first make sure that the database statistics are current for your database. The process for doing this is described in the IBM DB2 Content Manager V8.3 Enterprise Edition Performance Troubleshooting Guide.

If you still see this problem after updating the database statistics, then you can try rewriting the XPath query as follows. For example, start with this query:

/Journal[@Title LIKE "XML%" AND ./Journal_Article/@Author = "Richardt"]

If this query exhibits poor performance when there is a large amount of data in the system, then try rewriting this to the following:

/Journal[@Title LIKE "XML%" AND (@ITEMID = /Journal/Journal_Article[@Author = "Richardt"]/@ITEMID)]

Although the results of the query will be the same, the SQL generated for this query will differ from the SQL generated to perform the original query (assuming that versioning is turned off). This will cause the database to choose a different plan for execution.

You can do the same thing when versioning is used by adding conditions to the query. For versioning, use this query:

/Journal[@Title LIKE "XML%" AND (@ITEMID = /Journal/Journal_Article[@Author = "Richardt"]/@ITEMID) AND (@VERSIONID = /Journal/Journal_Article[@Author = "Richardt"]/@VERSIONID)]

This type of re-write helps in some cases. However, given the complexity in the database itself and variation in data, it might not perform better in all cases.

8.3 ICM query: Use IN operator to match one of several acceptable attribute values

Question

Use IN operator when performing a search to find results matching any of several different acceptable values. Do not use the set notation to list acceptable values for attribute conditions in your query string.

Cause

Using the set notation, ["myVarcharVal1","myVarcharVal2",...], does not perform as well as using the more efficient IN operator. The set notation results in a UNION, performing the search against the table for one value, unioned against that table for the next value, and so on. If you ever have duplicate values in your list, you unnecessarily search the same table for the same thing more than once. Duplicates will be filtered out during post-processing in the database, but at the cost of overall performance. Additionally, using the equals operator ("=") results in longer and potentially more complex internal queries and can result in DKUsageError "DGL7146A: The query string is too long or too complex.; ICM7074: The SQL to be generated is longer than the maximum length allowed. Change the application or simplify the query." Using the IN operator can shorten the internal query string length which can help alleviate this error in some cases.

Answer

Use the IN operator to add a condition to an attribute in your query to request only items or components that have an attribute value that matches any of those specified in a list that you specify.

Example:

use the query: /Employee [@status IN ("ACTIVE","RETIRED")]
instead of the query: /Employee [@status = ["ACTIVE","RETIRED"]]

8.4 ICM Query: Using text search instead of LIKE conditions

Problem

A query that uses LIKE operators with wildcards performs poorly, taking a long time to execute. This document provides a solution for improvement. For example, "/Journal [(@Title LIKE \"%Java%\") OR (@Title LIKE \"%java%\")]" can possibly be replaced with
"/Journal [contains-text-basic (@Title, \"Java\") = 1]" for performance improvement.

Cause

The usage of a LIKE operator with wildcards often leads to heavy processing on the database system. The database system might need to choose a costly execution plan for processing the results to account for all variations possible to support the query logic. The combination of multiple LIKE conditions, especially with an OR operator to connect them, can lead to further performance issues.

Solution

The first query provided in the Problem section is performing a case-insensitive search, trying to find all Journals with a Title about Java™, with the word "Java" starting with either an uppercase letter or a lowercase letter.

Text search provides a way to eliminate multiple performance issues with this query. Text search is by default case-insensitive, so a single condition would suffice instead of having two separate conditions. In addition, text search can help to eliminate the wildcards present in the query string, thus increasing performance. Here is one possible way to rewrite the original query string to use text search:

```
"/Journal [contains-text-basic (@Title, \"Java\") = 1]"
```

Note that with this rewrite, any Journals that have the word Java in any combination of case-sensitivity in the Title would be returned. If you would like to return Journals that have "Java" as part of a word, and not necessarily as a separate word, you can use wildcards in the text search condition itself as follows:

```
"/Journal [contains-text-basic (@Title, \"*Java*\") = 1]"
```

Also note that text search results are based on the data currently in the index and when the index was last updated. So the behavior of text search can be a little different depending on when the frequency of index updates. This is an important consideration on whether this is an effective solution in your particular system configuration.

8.5 ICM Query: Avoid wildcards at the beginning of the string with the LIKE operator

Problem

A query that uses the LIKE operator with wildcards performs poorly. For example, this query returns all Journals with a Title about Java™:

```
"/Journal [@Title LIKE \"%Java%\"]"
```

Cause

When a wildcard (the percent sign (%) for multiple characters or the underscore (_) for a single character) is used at the beginning of the string literal, the underlying database system creates a query that is inefficient. Even if a database index has been defined on the Title attribute in the Journal item type, the database system would likely not be able to use such an index and would instead resort to a generally more expensive table scan.

Solution

If possible, avoid placing the wildcard character at the beginning of the literal that is used with the LIKE operator. Whether this is possible would depend on the actual data

that is present in the system. However, if you know that the word you are searching for (for example, Java) is the first word of the title, then avoiding the initial wildcard would likely lead to higher performance. An example of a rewritten query would be:

"/Journal [@Title LIKE \"Java%\"]"

Note that the rewritten query might not return the same set of results as the original query, depending on the actual data present in the system.

Alternatively, you can define a text index on the Title attribute and use text search to perform the original query as follows:

"/Journal [contains-text-basic (@Title, \"Java\") = 1]"

8.6 Search that is not case-sensitive is possible with DB2 Content Manager Version 8

Problem

Is it possible to do a search that is not case-sensitive of string attributes in DB2 Content Manager Version 8? This is possible in the query language by using the database UPPER() function. To do this, the application has to rewrite its queries. In some cases, a change to the data model is required to support this feature efficiently. Using the database UPPER function may have performance issues, so a new attribute might be needed to hold the value of the string in its uppercase format.

Solution

The DB2 Content Manager Version 8 query language does case sensitive matching when comparing string attributes. For example, the query /Book[Book_Author/@LastName = "Richardt"] would only match the exact case specified in the query. In this example, any items with LastName equal to "Richardt" would be returned, but items with LastName equal to "richardt" or "RICHARDT" would not be returned.

One way to achieve a search that is not case-sensitive is to use the database UPPER function. The following query uses this database function to do case insensitive string matching:

/Book[UPPER(Book_Author/@LastName) = UPPER("Richardt")]

This query does a case insensitive search, but it may cause performance degradation over a case sensitive search. The reason for this in DB2 is that the UPPER() function always requires a table scan. For the above example, if there is an index of the LastName attribute, this index would be ignored when the UPPER() function is used. This is probably acceptable for item types that have a small number of items, but it is bad for item types that contain a large number of items.

One solution to this problem is to change the application data model and store an uppercase copy of the string. In this way, when the application needs to do a case insensitive search, this new attribute is used. In the example above, you might add a new attribute called LastName_upper. Whenever a value is stored in the Author attribute, the application would store the uppercase copy in LastName_upper. To do case insensitive search, this new attribute is used:

```
/Book[Book_Author/@LastName_upper = "RICHARDT"]
```

To optimize the performance of these queries, an index may be created on the LastName_upper attribute. Because the UPPER function is not needed, the index will be used in resolving this query. In this way, case insensitive string matching may be done efficiently.

Another possible solution is to use text search. Text search is not case-sensitive by default. In the example above, a text index would be created on the attribute LastName. After this is done and the text index is updated, then the following query will work:

```
/Book[contains-text-basic(Book_Author/@LastName, "Richardt")=1]
```

In general, text search will be less efficient than the second solution above, but it does not require duplicate data to be stored. Another issue with using text search is that the text indexes are updated asynchronously. This means that after an update of the attribute, the text index will not match the attribute data until the next time the text index is updated. Another possible issue is that text search does not do exact matching, so if the word specified is anywhere in the attribute, it will match.

8.7 ICM query: use upper-case literals instead of UPPER function for case-insensitive search

Problem

When performing a case-insensitive search such as the search in the following example, the performance of the query is not optimal:

```
/Book[UPPER(Book_Author/@LastName) = UPPER("Bondar")]
```

Solution

When a literal to be compared is available to the application that is constructing the query string, it is more efficient for the application to convert the literal to upper-case instead of requesting that the database function UPPER perform the conversion. A more optimal query string, when compared to the string listed in the Problem section, would be similar to the following example:

```
/Book[UPPER(Book_Author/@LastName) = "BONDAR"]
```

For attribute values that are stored in the database and are not available to the application (such as Book_Author/@LastName), the use of the UPPER function is the proper and efficient way to use case-insensitive comparison.

For more information about case-insensitive search, see the technote number 1154595 titled "Search that is not case-sensitive is possible with DB2 Content Manager Version 8...."

8.8 ICM Query: Improving performance of upward traversal queries

Problem

A query string using the upward-traversal operator (..) takes more time than expected to run.

"/Journal/Journal_Article [(../@Title = \"XML\") OR (../@Organization = \"IBM\")]"

Cause

The query unnecessarily and inefficiently uses the upward-traversal operator. Each usage of the upward-traversal operator requires an extra join to be generated in the underlying SQL statement, which in turn results in longer processing time on the database system. The processing time is generally further increased by the usage of the OR operator with the attribute conditions.

Solution

To increase the performance of this query, rewrite the query string to put the conditions directly on the parent component type view (in this case, Journal) before traversing down to the child component type view (in this case, Journal_Article).

A more efficient rewritten query string would be similar to the following example:

"/Journal [(@Title = \"XML\") OR (@Organization = \"IBM\")]/Journal_Article"

Note that the results of the query would not change due to the rewrite, but the performance of the query should improve.

8.9 ICM Query: It is possible to query a character-based attribute for numeric comparison

Problem

Sometimes, numeric data might need to be stored in a character-based attribute such as VARCHAR. However, this practice is not generally recommended. With IBM DB2 Content Manager, you can query numeric data stored in VARCHAR as a number for comparison purposes. However, if you do this, there are performance and other implications.

Solution

To query a character-based attribute for numeric comparison through the DB2 Content Manager query function, you can use the IBM DB2 scalar function called INTEGER. To use the INTEGER function, you wrap the name of the character-based attribute inside this function within the DB2 Content Manager query string. For example:

/Journal [INTEGER(@Classification) <= 5000]

The function can be used with other comparison operators just as a normal numeric attribute is used. For example:

/Journal [INTEGER(@Classification) BETWEEN 1000 AND 5000]

Important:
Do not use double quotation marks around the numeric value that the attribute is being compared to. The quotation marks must be omitted because the value returned by the INTEGER function is numeric. Therefore, the comparison value should also be numeric (without quotation marks). Otherwise, a type mismatch exception is thrown.

Remember that it is not recommended to perform conversion between different data types because this might lead to performance issues and other issues. The following list contains examples of the types of issues you should consider:
Performance impact: Using this function can lead to some performance degradation caused by longer database processing time. This longer processing time is mostly due to type conversion. You should carefully consider this performance impact when you use this function on item types that contain a lot of entries.
Only data that can be converted to a number can be stored in this field. Otherwise, a database error will be thrown.

Other than the character-based data types (such as character and variable character), it might be possible to use the INTEGER scalar functions on attributes of other data types such as date, time, and other numeric types. For more information about what this INTEGER function supports and the other considerations for this function, see the description of the INTEGER scalar function in the reference section of the IBM DB2 user's guide.

8.10 ICM Query: Using query to efficiently find folders which are not contained in another folder

Problem

How can I efficiently find folders or documents that are not inside any folder? Can query be used to find these folders or documents?

Solution

You can efficiently find folders or documents that are not inside any folder using query. You can limit your search to the following criteria:

- Only items (documents or folders) that are in a folder.
- Only items (documents or folders) that are not in a folder.
- Only items (documents or folders) that do not participate in any link relationships.

To efficiently find all folders that are not contained inside any other folder, you can use the query below for the claimFolder item type:

/claimFolder [INBOUNDLINK[@LINKTYPE = "DKFolder"]]

The query checks for the existence of INBOUNDLINKs of the type DKFolder. The existence of these links indicates that the items are contained in a folder.

Variations on this query can be used to do related queries. The following query finds items that are not contained in any folder:

/claimFolder [NOT INBOUNDLINK[@LINKTYPE = "DKFolder"]]

If the claimFolder item type may contain folders or documents, then you can add a predicate to limit the result to only documents or only folders. This query limits the results to only folders:

/claimFolder [@SEMANTICTYPE = 2 AND INBOUNDLINK [@LINKTYPE = "DKFolder"]]

This same type of query can be used for other link types, for example:

/claimFolder [INBOUNDLINK[@LINKTYPE = "Contains"]]

It can also be used to query the existence of links of any type, for example:

/claimFolder [INBOUNDLINK]

These queries are extremely efficient since they do not have to traverse the links to look at the containing folder itself. This query just checks the existence of links. This has the minor side-effect that you may see the existence of links to items which security would prevent you from accessing. If this side effect is a concern, then you must traverse the link. However, doing this make the query much less efficient. For example:

/claimFolder [INBOUNDLINK [@LINKTYPE = "DKFolder"]/@SOURCETARGETREF => *]

This query removes the side-effect, but is much less efficient.

8.11 Can a 'sounds like' query be done for DB2 Content Manager V8?

Problem

Customers need to query in DB2 Content Manager V8 for words that sound like a specific word. This can be used to tolerate some misspellings and typos. This can be achieved by using the database SOUNDEX function.

Solution

The DB2 Content Manager V8 query language allows the use of database scalar functions in queries. DB2 Universal Database™ and Oracle support the SOUNDEX scalar function to do sounds like queries. This function can be used in DB2 Content Manager V8 queries.

For example, the following query looks for an exact match on a string attribute:

```
/Journal[@Keyword = "Java"]
```

To change this query into a sounds like query, it would look like this one:

```
/Journal[SOUNDEX(@Keyword) = SOUNDEX("Java")]
```

Before using this query in any application, you should evaluate its performance for your application. This sort of a query will probably cause a table scan because it can not be resolved in a database index. Therefore this function might have a negative impact on performance and scalability.

8.12 Performance Considerations Associated with Row-based View Filtering

If you plan to use row-based view filtering in your data model to automatically add query conditions to filter the data, you need to take special performance considerations into account. Refer to the documentation under IBM DB2 Content Management Information Center in section "Searching for data" and then "Understanding row-based view filtering in query" for more information.

9 Logging and Other Information Typically Needed for Analysis by DB2 Content Manager Level 2 and Level 3 Support Teams

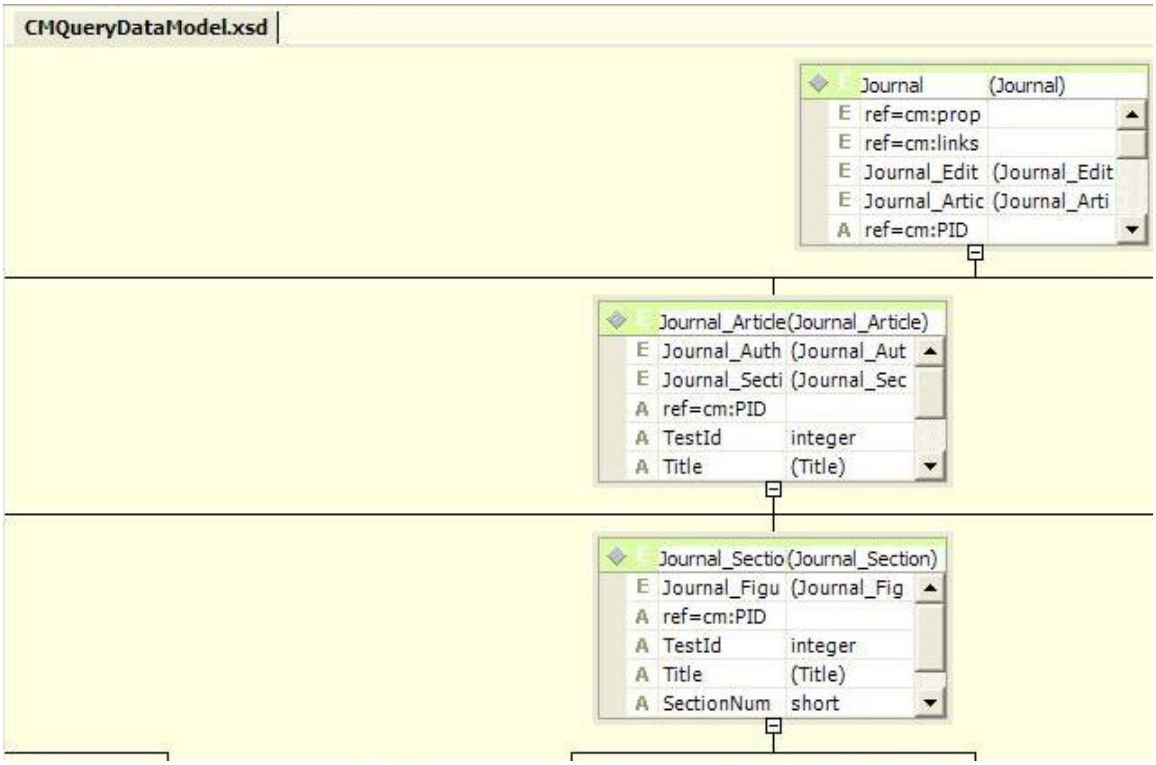
If after following the guidelines and recommendations described in this document, you still need to contact IBM support, make sure to provide as much appropriate information as possible in order to expedite the analysis of your issue and avoid additional requests for more information. Here are some recommendations for information to provide to the IBM support team related to DB2 Content Manager API query performance questions:

- ALWAYS provide the XQPE query string, not just the SQL string that was eventually generated based on user's XQPE query string.
- Clear all log files before initiating the test.
- Preferably have only a single query execution per log to avoid confusion and speed up analysis.
- Set the logging levels to DEBUG for the API (e.g. DKLogPriority = DEBUG in cmblogconfig.properties file) and for the Library Server (e.g. by selecting all trace levels in the DB2 Content Manager System Administration Client) to capture all of the processing information.
- Provide API log at DEBUG logging level.
- Provide a Library Server log at DEBUG logging level (make sure that the Library Server log matches the API log).

- If appropriate, provide logs from all other components involved in processing (e.g. eClient, Beans)
- Increase the maximum size of the log as necessary (for example, by setting DKLogOutputFileSize = 50 in the cmblogconfig.properties file to set the max to 50MB), so that all of the logging for given query processing is contained in a single file and it is easier for the support teams to look at and search through the log without juggling multiple files. You can decrease the size of the log file as desired after capturing the necessary information.
- Make sure that the logs are complete and not truncated. Look at the beginning of the file and make sure that it corresponds to the beginning of your sample program runtime (e.g. connecting to the datastore). Then look at the end of the log file and ensure that the file ends after all of the processing, including retrieval of results, has completed (such as after disconnecting from the datastore).
- Provide an XML export of your version of the DB2 Content Manager data model to help the support teams visualize and see the details of the item type views, attributes, and other data model constructs that you are querying on.

10 XML Data Model Export

An XML export of the data model is often very useful in analyzing ways to rewrite queries or resolve query problems. Please provide this model to DB2 Content Manager support team for analysis whenever possible. Since DB2 Content Manager queries are written to traverse the item type view hierarchy, the details of this hierarchy provided by the data model export are very important. An export of the DB2 Content Manager data model can be easily obtained from the DB2 Content Manager System Administration Client. The following picture provides a partial visual example of what a .xsd file containing an exported data model might look like.



11 Reference

a. IBM DB2 Content Manager Enterprise Edition V8.3 Performance Tuning Guide. http://www.ibm.com/support/docview.wss?uid=swg27006452
b. IBM DB2 Content Manager Enterprise Edition V8.3 Performance Monitoring and Maintenance Guide. http://www.ibm.com/support/docview.wss?uid=swg27006451
c. IBM® DB2® Content Manager V8.3 Enterprise Edition Performance Troubleshooting Guide. http://www-1.ibm.com/support/docview.wss?uid=swg27006450&aid=1
d. Redbook: Performance Tuning for Content Manager. http://www.redbooks.ibm.com/redbooks/pdfs/sg246949.pdf
e. IBM DB2 Content Management Information Center (DB2 Content Manager Application Programming Guide). http://publib.boulder.ibm.com/infocenter/cmgmt/v8r4m0/topic/com.ibm.programmingcm.doc/dcmcp011.htm
f. IBM Content Manager support web site to search for tech notes and other technical information. http://www.ibm.com/software/data/cm/cmgr/mp/support.html If you cannot find the documents that you need, try searching from the main support web site http://www.ibm.com/software/support/ by choosing "Information Management" from the "Select a brand and/or product:" drop-down menu.