FileNet P8 Content Engine
Query Performance Optimization Guidelines
Technical Notice

Release 3.5.x

June 2006

FileNet Corporation
3565 Harbor Boulevard
Costa Mesa, California 92626
1.800.FILENET (345.3638)
Outside the US, call:
1.714.327.3400
www.filenet.com

Contents

Revision Log

Date	Revision
6/20/2006	Initial document.

Introduction

This document is a guide for optimizing the performance of your Content Java API or Content Engine COM API client SQL queries made against a FileNet Content Engine server. This guide does not replace any application-specific performance studies or recommendations provided by FileNet.

Audience

This document is for writers and readers of Content Engine SQL queries. This primarily means application developers, and also database administrators who are working with and assisting developers.

References

For general database tuning information, see this location on the FileNet Customer Service and Support (CSS) web site: Product Tech Info > Content Manager (CM) > Product Documentation > 3.5.2x Documentation > FileNet P8 Platform 3.5.x Performance Tuning Technical Notice. Note that you need a valid CSS user ID and password to access the web site.

For information on Content Engine SQL syntax, see FileNet P8 Documentation **>** Developer Help > Developer Roadmap > Reference > SQL Syntax Descriptions.

For general information on queries using the Content Java API or Content COM API, see:

- FileNet P8 Documentation **>** Developer Help > Content Java API > Searching
- FileNet P8 Documentation **>** Developer Help > Content Engine COM API > COM API Guide > Using the Content Engine Database Engine > Querying Information About Document Objects

NOTE For guideline-specific references, see the appropriate topic under Guidelines.

Using Query Builder

Use the FileNet Enterprise Manager's Query Builder tool to visually construct your query or to quickly validate that your query works as intended. To run Query Builder, select an object store and then select Action > Search.

Be sure to include the object reference “this” in the SELECT clause when attempting to execute any query examples from this document in Query Builder > View > SQL View. For instance, enter the query

```
SELECT d.Id FROM Document d WHERE d.DocumentTitle = 'MyDoc'
```

into Query Builder in this form:

```
SELECT d.this, d.Id FROM Document d WHERE d.DocumentTitle = 'MyDoc'
```

For more information on using the Query Builder, see FileNet P8 Documentation > Developer Help > Content Engine COM API > COM API Guide > Using the Content Engine Database Engine > Querying Information About Document Objects > Using the Content Engine Query Builder.

Creating Indexes

As indicated by several of the guidelines, your query's performance might depend on database indexes existing for various Content Engine object properties. In most cases, use FileNet's Enterprise Manager as the most convenient means to create these indexes; see FileNet P8 Documentation > FileNet P8 Administration > Content Engine Administration > Classes > How to... > View/modify class property definitions > Set/remove indexing. In some cases you might need to use the tools provided by the database engine (Oracle, SQL Server, etc.) either to verify that the expected index exists or to create the index (when unable to do so using Enterprise Manager), and consequently need to know the table column that corresponds to the object property in question.

For class-to-table mappings, see FileNet P8 Documentation > Developer Help > Content Engine COM API > COM API Guide > Using the Content Engine Database Engine > Using Content Engine Database Tables > Classes and Database Tables. Note that the table column name, when not identical to the object property name, will have this format: xxx_<object property name>, where "xxx" is some generated prefix. For example, the object property Document.DocumentTitle might map to table column DocVersion.u2e_documenttitle.

Guidelines

Limit rows returned

Limit the number of rows returned from your query to a usable number, either by tailoring your query's row selection criteria to achieve that result, or by arbitrarily deciding on that number in advance by setting the Microsoft ActiveX Data Object (ADO) MaxRecords property.

When using the Content Java API, set the MaxRecords property indirectly via the passed XML. For an example, see FileNet P8 Documentation > Developer Help > Content Java API > Searching > Working with Search-related Objects > Performing an Ad hoc Search.

When using the Content COM API, set the property directly as shown in this example:

```
Set ADOCnn = ObjStore.GetADOConnection

Set RecSet = CreateObject("ADODB.Recordset")
RecSet.MaxRecords = 20

Call RecSet.Open(Query, ADOCnn, adOpenStatic, adLockReadOnly)
```

Avoid non-indexed ordering and searching

Avoid referencing any non-indexed column in a JOIN, WHERE, or ORDER BY clause. For example, optimize the query below by creating an index on Document.DocumentTitle (database column DocVersion.xxx_documenttitle):

```
SELECT d.Id FROM Document d WHERE d.DocumentTitle = 'MyDoc'
```

NOTE The Document class maps to the DocVersion table. See Creating Indexes.

For indexing discussions, see the *FileNet P8 Platform 3.5.x Performance Tuning Technical Notice*.

Avoid non-function-indexed case-insensitive comparisons (Oracle / DB2)

Avoid any column value comparison resulting from a JOIN, WHERE, or ORDER BY clause for any non-indexed column, or for any column belonging to an index that does not directly or indirectly use the LOWER function. Follow this guideline in these circumstances: you are working with an object store that uses Oracle or DB2 as the database engine, you have forced case-insensitive searches by setting the ForceCaseInsensitiveSearch property to true, and the Oracle or DB2 database has not been natively configured for case-insensitivity (and so setting ForceCaseInsensitiveSearch to true has a practical effect).

For DB2, generate a column by applying the LOWER function to a pre-existing column, and then create an index on the generated column. For Oracle, create a function-based index. For example, the index LOWER(DocVersion.xxx_documenttitle) makes the following queries more efficient:

```
SELECT d.Id FROM Document d WHERE d.DocumentTitle = 'MyDoc'

SELECT d.Id, d.DocumentTitle, d.Creator FROM Document d ORDER BY d.DocumentTitle
```

NOTE The Document class maps to the DocVersion table. See Creating Indexes.

For additional information on performance issues associated with case-insensitive queries, see the *FileNet P8 Platform 3.5.x Performance Tuning Technical Notice*.

For additional information on case-insensitive queries or the ForceCaseInsensitiveSearch property, see:

- FileNet P8 Documentation **>** FileNet P8 Administration > Content Engine Administration > Search and bulk operations > How to > Force case insensitive searches
- FileNet P8 Documentation > Developer Help > Content Java API > Searching > Working with Search-related Objects > Performing an Ad hoc Search
- FileNet P8 Documentation **>** Developer Help > Content Engine COM API > COM API Reference > Properties > ForceCaseInsensitiveSearch Property

Avoid unnecessary object type searches

Avoid unnecessary searching for subclass types by adding the EXCLUDESUBCLASSES operator to your query. You need to explicitly add this operator because a query has an implicit INCLUDESUBCLASSES operator by default.

For example, as a result of the implicit INCLUDESUBCLASSES, the following query might return objects belonging to a Document subclass, in addition to those belonging to the Document class:

```
SELECT d.Id FROM Document d WHERE DocumentTitle = 'MyDoc'
```

Presuming only objects from the Document class are needed, instead of writing the query as

```
SELECT d.Id FROM Document d WHERE DocumentTitle = 'MyDoc' AND ISCLASS(d, Document)
```

use the EXCLUDESUBCLASSES operator in this manner:

```
SELECT d.Id FROM Document d WITH EXCLUDESUBCLASSES WHERE DocumentTitle = 'MyDoc'
```

For a discussion of the EXCLUDESUBCLASSES operator, see FileNet P8 Documentation > Developer Help > Developer Roadmap > Reference > SQL Syntax Descriptions > Include/Exclude Subclasses Function Description.

Avoid unnecessary column returns

Avoid returning all of the columns in a table when only some are needed. For example, instead of

```
SELECT d.* FROM Document d WHERE d.DocumentTitle = 'MyDoc'
```

specify the needed columns, as in this example:

```
SELECT d.Id FROM Document d WHERE d.DocumentTitle = 'MyDoc'
```

This minimizes the amount of data transmitted over the network. Also, when all of the columns in the SELECT clause are indexed, the data for the query might be retrieved directly from the index, thereby making the physical row lookup unnecessary.

Use the free-threading model

Use the free-threading model for Microsoft ActiveX Data Objects (ADO) on the Content Engine server. Using this model can improve the performance of your queries, including those executed via the Content Java API.

For details on changing the ADO threading model, see FileNet P8 Documentation > Developer Help > Content Engine COM API > COM API Guide > Using the Content Engine Database Engine > Querying Information About Document Objects.

For a discussion of the Content Java API Listener, and its implementation using COM objects, see FileNet P8 Documentation > Developer Help > Content Java API > Overview.

Tune query batch size parameters

Optimize query response time by tuning the Default Query Batch Size and Enumeration Batch Size parameters. These parameters control the size of the data chunks returned by the Content Engine server. For more information, see the *FileNet P8 Platform 3.5.x Performance Tuning Technical Notice*.

Avoid complex table linkages

Avoid referencing three or more tables in a query. More tables might degrade query performance, and complicate performance tuning efforts.

Avoid unnecessary result row ordering

Avoid an unnecessary ORDER BY clause. Explicit row ordering might be unnecessary when you can rely on your query returning its results in a particular order as a side effect of an indexed search. For example, the ORDER BY clause in the query below might be unnecessary since a composite index exists for the Container table on (parent_container_id, name).

```
SELECT f.FolderName FROM Folder f WHERE f.Parent = Object('/sub1/sub1a') AND
f.IsHiddenContainer = false ORDER BY f.FolderName
```

NOTE The Folder class maps to the Container table.

Avoid subqueries (Oracle)

Avoid using subqueries and operators that indirectly generate subqueries whenever you are querying an object store that uses Oracle as the database engine. Potential subquery-related issues exist with the Oracle optimizer.

Specifically, use an INNER JOIN instead of a subquery (although see the Avoid complex table linkages guideline). For example, re-write the potentially slow query

```
SELECT d.Id FROM Document d WHERE d.Id IN (SELECT b.Bp8ObjectGuid FROM Bp8Attachment b)
```

in this functionally equivalent form:

```
SELECT d.Id FROM Document d INNER JOIN Bp8Attachment b ON d.Id = b.Bp8ObjectGuid
```

NOTE FileNet's Business Process Framework product uses the Bp8Attachment table.

Also, because the Content Engine uses a subquery to implement the INFOLDER operator, avoid using that operator. For example, re-write the query

```
SELECT f.FolderName FROM Folder f WHERE f.this INFOLDER '/sub1/sub1a' AND
f.IsHiddenContainer = false
```

in this manner:

```
SELECT f.FolderName FROM Folder f WHERE f.Parent = OBJECT('/sub1/sub1a') AND
f.IsHiddenContainer = false
```

Also, re-write

```
SELECT d.id FROM Document d WHERE d.This INFOLDER '/sub1/sub1a'
```

in this more efficient form:

```
SELECT d.Id FROM Document d INNER JOIN ReferentialContainmentRelationship r ON d.This =
r.Head WHERE r.Tail = OBJECT('/sub1/sub1a')
```

For a discussion of the INFOLDER operator, see FileNet P8 Documentation > Developer Help > Developer Roadmap > Reference > SQL Syntax Description > Folder Operators Description.