# DQL (Domino Query Language) Developer's Deep Dive

**March 26,** 2019

HCL

Copyright © 2019 HCL Products & Platforms | www.hcltech.com

# SETTING DEEP DIVE EXPECTATIONS

- Presumes NOTHING about your DQL knowledge except you want to know it all
- LOTS of detail, LOTS of content
  - Some detail subject to change
- Questions/comments encouraged though we will move fast


Copyright © 2018 HCL Technologies Limited | www.hcltech.com | HCL Confidential
PRODUCTS & PLATFORMS by HCL Technologies
HCL

# DOMINO GENERAL QUERY FACILITY DEEP DIVE AGENDA

- Introducing DQL
  - Syntax with processing details and DEMOs
- Design Harvesting/Design Catalog
- Programming DQL
  - DQLExplorer
  - DomQuery tool
  - (Notes)DominoQuery class
- Performance
- Security
- Remaining design decisions


Copyright © 2018 HCL Technologies Limited | www.hcltech.com | HCL Confidential
PRODUCTS & PLATFORMS
HCL

# INTRODUCING DQL (DOMINO QUERY LANGUAGE)

- First – the name
  - Query Facility – that which compiles, plans and executes queries
  - Query Language – the language which specifies the queries to run (current DQL is our 2nd one)
- Target developer – node.js Domino neophyte
  - A facility running in Domino core allowing for a terse shorthand syntax for finding documents according to a wide variety of complexity of terms. It leverages existing design elements without the need to write detailed code to access them.
  - Once concept was proven, inclusion in Domino backend (Notes)Database class was an easy fit
  - Command line (shell/DOS prompt) invocation
    - How it was developed
    - node.js from command line adds even more power


Copyright © 2018 HCL Technologies Limited | www.hcltech.com | HCL Confidential

# INTRODUCING DQL (DOMINO QUERY LANGUAGE) – PROBLEM 1

## Problem:

- **In a order-handling, workflow database containing 2M documents, you have 10 minutes to mark a specific set of documents for follow-up mail campaign.  The criteria are:**

Orders that originated in Detroit, Albuquerque, or San Diego only

Originated between 15 July 2014 and 14 July 2015

Contain each of 4 part numbers ordered (a multiply occurring field): 389, 27883, 388388, 587992

Are NOT in two special folders ‘Special processing’, ‘Special2’ and NOT in the view ‘Soon to be special’

Were NOT sold by Christen Summer or Isaac Hart

Find those documents and add a promo_code field of ‘FLLWUP_2014’

**AND … GO!  (how do you do it?)**


Copyright © 2018 HCL Technologies Limited | www.hcltech.com | HCL Confidential

PRODUCTS & PLATFORMS

HCL

# INTRODUCING DQL (DOMINO QUERY LANGUAGE) – PROBLEM 1 SOLUTION

## Solution (using Domino Query Language (DQL) in ND10):

**Order_origin in ('Detroit', 'Albuquerque', 'San Diego') and**
**Date_origin >= @dt('2014-07-15') and Date_origin <= @dt('2015-07-14') and**
**partno in all ( 389, 27883, 388388, 587992 ) and not**
**in ('Special Processing', 'Special2' , 'Soon to be special') and not**
**sales_person in ('Christen Summer', 'Isaac Hart')**

Documents are then easily updated to insert promo_code = 'FLLWUP_2014' using bulk document processing provided by the domino-db node.js API

The whole "job" runs from command line after a few trials testing the query


Copyright © 2018 HCL Technologies Limited | www.hcltech.com | HCL Confidential

# INTRODUCING DQL (DOMINO QUERY LANGUAGE) – PROBLEM 2

**Problem:**

- **A new application developer, skilled in node.js and MongoDB from university, is hired to write Domino applications using node.js, creating a database from scratch. After designing the database schema (fields, documents), s/he needs to read, update and delete sets of documents as part of the application. The fields and searches change constantly as the design changes to meet dynamic requirements. How does the developer find documents to process?**

**AND .. GO! (How do you instruct the developer to work with documents in Domino?)**


Copyright © 2018 HCL Technologies Limited | www.hcltech.com | HCL Confidential

# INTRODUCING DQL (DOMINO QUERY LANGUAGE) – PROBLEM 2 SOLUTION

**Solution:**

Use the ND10 Domino General Query Facility

- Familiar syntax construction, not unlike SQL or Cassandra
- Improvement over MongoDBs manual Boolean tree construction
- No prior knowledge of (though the DQL will use) design elements of forms, views or folders
- No need to learn Formula Language but existing formulas can be leveraged
- Where views and folders provide value for queries, they can be specified in the syntax


Copyright © 2018 HCL Technologies Limited | www.hcltech.com | HCL Confidential

# INTRODUCING DQL - TERMINOLOGY

- **Syntax –** the components of the language with rules, order, spacing, etc.
- **Term** – smallest component of a query - <fieldname (sometimes optional)> <operator> <value>
- **Boolean** – conditional operation to be performed on multiple terms – AND, OR, NOT
- **Identifier** – the left side of a term – field or view column (or not needed)
- **Operator** – specifies comparison between left and right sides of a term - =, >=, >, <=, <, IN
- **Value** – the right side of a term (may be multiple)
- **Precedence** – the order in which a query’s terms are executed
- **Foundset/Result** – the group of documents returned by a term, Boolean, or entire query


Copyright © 2018 HCL Technologies Limited | www.hcltech.com | HCL Confidential
PRODUCTS & PLATFORMS by HCL Technologies HCL

# INTRODUCING DQL

- Language constructs, syntax - identifiers and operators

  {<identifier>} <operator> <value(s)>

  - Some operators need no identifier
  - Operators are >, >=, =, <, <=, in {all} , contains {all}
  - Identifiers are
    - Sometimes not required (IN and CONTAINS)
    - Summary field names in the database – (though CONTAINS can reference ANY field)
      - Will not compile if there are no documents with that field in the database
      - Will find no documents if you use non-summary fields (except CONTAINS)
      - @functions (so far) @All, @ModifiedInThisFile, @DocumentUniqueID, @Created
- Boolean support – AND/OR/NOT
- "Natural" precedence of terms – ANDs before ORs except after NOTs + parentheses overrides

  Order_no = 3342 and sales_person = 'Trudy Ayton' or order_no = 23334 and sales_person = 'Norton Jaden'


Copyright © 2018 HCL Technologies Limited | www.hcltech.com | HCL Confidential

# INTRODUCING DQL

- The IN operator
  - **With** field or view column name - finds any field value in the parentheses (same as ORed terms)

    Part_no in (388388, 724, 90022) // part_no = 388388 or part_no = 724 or part_no = 90022
    Order_date in (@dt('2018-09-01'),@dt('2019-03-26T07:23:01.0000'), @dt('2019-03-27'))
    Sales_person in ('Trudi Ayton', 'Joshua Robinson', 'Erika Weber')

  - **Without** field or view column name – finds documents in any of the view or folder names in the parentheses

    in ('Sales_2018', 'Trudi_orders', 'Special Handling')
    in ('New folder 3', 'Returns')

  - With **all** operator for field values or view/folder names – does exclusive find (only values or documents in ALL

    Part_no in **all** (388388, 724, 90022) // documents having multiple part_no field values with all values matching
    in **all** ('Sales_2018', 'Trudi_orders', 'Special Handling') // documents in every one of the folders and views


Copyright © 2018 HCL Technologies Limited | www.hcltech.com | HCL Confidential
PRODUCTS & PLATFORMS
HCL

# INTRODUCING DQL - IN VIEW TERMS

- DQL allows you to leverage EXISTING views and folders as document sets
  - Views
    - Pre-selected sets of documents, kept current according to arbitrarily complex selection criteria
    - If your data is dynamic, use views in your IN clauses
  - Folders
    - Can be populated and repopulated on the fly with random query results
    - If you want to reuse query results in other queries OR if you have static document sets you want to keep using, use folders in your IN clauses
    - New concept to some – folders in applications
  - BOTH views and folders can propagate – need to be managed


Copyright © 2018 HCL Technologies Limited | www.hcltech.com | HCL Confidential

# INTRODUCING DQL

- The CONTAINS operator (ND11)
  - **With** field name - finds any word or phrase in the parentheses

    Order_description contains (‘Nuts’, ‘Bolt*’, ‘Washer?’) // supports wildcard

  - **Without** field name – finds any word or phrase in the parentheses across the entire database (all fields)

    contains (‘Trudi’, ‘San Francisco’, ‘Backorder spec*’)

  - With **all** operator with or without field names – finds documents containing every word of phrase in the parentheses

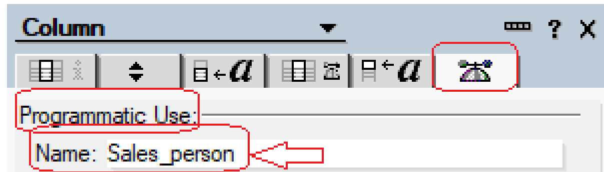
    Order_description contains **all** (‘Nuts’, ‘Bolt*’, ‘Washer?’)

  - NOT the same as “=“ – myfield contains (‘Habitat’) will find
    - ”Habitat for humanity”, “Perfect habitat for hares and squirrels” as well as just “Habitat”
    - **BUT** you know your data values of course


Copyright © 2018 HCL Technologies Limited | www.hcltech.com | HCL Confidential

# INTRODUCING DQL

- Language constructs, syntax - identifiers and operators (continued)
  - ‘viewname’.column name construct – will not compile if the view doesn’t have that name in a collated column:




- All identifiers and operators and @function names are case insensitive
- Value datatype determines type of search

  ‘string’ | @dt(‘<date>’) | <number>

  - where
    - ‘string’ is by default case and accent insensitive
    - Legal <date> values must in in RFC3339 format (Date only, Time only supported)
    - Legal <number> are all floating point notation, include scientific E-format

**DEMO 1**


Copyright © 2018 HCL Technologies Limited | www.hcltech.com | HCL Confidential

# INTRODUCING DQL

- Small print
  - @dt('2019-03-26') and @dt('09:03:37.0000') will **NOT** use view lookup – need full RFC3339 timedate – NSF scan
  - Maximum string value size is 256 bytes
  - Use of view columns (implicit and explicit) has rules


Copyright © 2018 HCL Technologies Limited | www.hcltech.com | HCL Confidential

PRODUCTS & PLATFORMS by HCL Technologies

HCL

# INTRODUCING DQL – HOW THE SYNTAX IS PROCESSED

| Operation | Views | NSF Scan | Full text (V11) |
| --- | --- | --- | --- |
| Field = ‘value’ (string) | X | X | X (know your data) |
| Field = @dt(‘date’) | X | X | |
| Field = number | X | X | |
| Field contains ‘value’ (V11) | | | X |
| Field IN (all) (value1, value2 …) (all types) | X | X | |
| Field >, >=, <, <= value (all types) | X | X | |
| ‘View’.column =, >, >=, <, <=, in (all) value | X | | |
| Document contains ‘value’ (V11) | | | X |
| Documents in (all) views or folders by name | X | | |


Copyright © 2018 HCL Technologies Limited | www.hcltech.com | HCL Confidential

# INTRODUCING DQL

- Language constructs, syntax - identifiers and operators (continued)
  - @function values are according to their expected type
  - @All – if specified, the only token that can be in a query
  - @Created, @ModifiedInThisFile - @dt(‘<RFC3339 compliant timedate>’)
  - @DocumentUniqueID – Unique Identifier string

    **@documentuniqeid = '9DCCEA37842AD0F38525828C0079C95A'**


Copyright © 2018 HCL Technologies Limited | www.hcltech.com | HCL Confidential
PRODUCTS & PLATFORMS
HCL

# INTRODUCING DQL - SAMPLE QUERY EXAMPLES

Order_origin in (‘London’, ‘LA’, ‘Tokyo’) AND date_origin > @dt(‘2016-05-11’) or partno = 388388

Finds documents with (any of the order_origin field values of ‘London’, ‘LA’ or ‘Tokyo’ AND sales date greater than 11 May 2016) OR having Part number of 388388

( In all (‘Soon to be special’, ‘Main View’) or order_no > 12751 and order_no < 14334 ) and sales_person = ‘Trudi Ayton’

Finds documents that are (in BOTH ‘Soon to be special’ and ‘Main View’ or have order numbers between 12751 and 14334) and were sold by Trudy Ayton

‘Soon to be special’.Status = ‘Shipping’ and ( order_origin = ‘LA’ or sales_person in (‘Chad Keighley’, ‘Jeff Chantel’, ‘Louis Cawlfield’, ‘Mariel Nathanson’))

Finds documents with values of ‘Shipping’ using the Status column in the ‘Soon to be special’ view AND with either an order_origin of ‘LA’ or sold by any of Chad Keighley, Jeff Chantel, Louis Cawlfield or Mariel Nathanson


Copyright © 2018 HCL Technologies Limited | www.hcltech.com | HCL Confidential

# INTRODUCING DQL - VALUE

| | |
|---|---|
| **For Domino beginners** | Database context-free searching – no deep design knowledge required<br>Built with node.js as target environment and programming model |
| **Set-based processing** | Simple yet powerful bulk-operations support<br>No need to code loops or do document-at-a-time processing |
| **Programming power** | Full support for boolean processing - AND, OR, NOT<br>Concise shorthand accomplishing complex and detailed find operations<br>Simple to read EXPLAIN output to show how to optimize |
| **High performance** | Automatically finds view columns to use to satisfy query terms<br>Partial results injected into child and sibling processing for smart, optimal atomic search operations<br>Limit values in the API to control runaway queries |
| **Guards investment** | Seamless invocation from either node.js or Lotusscript/JAVA<br>Perfect fit into Domino backend (Notes)DocumentCollection object |
| **Domino centric** | All documents in existing or new Views or Folders can be ANDed, ORed<br>View/Folder column values (utilizing all existing design)<br>@ModifiedInThisFile, @DocumentUniqueID, @Created support |


Copyright © 2018 HCL Technologies Limited | www.hcltech.com | HCL Confidential

PRODUCTS & PLATFORMS

HCL

# DESIGN HARVESTING/DESIGN CATALOG

- Existing design elements and DQL
  - Since high-speed access to design element innards is not possible using design notes
    - They need to be extracted –
      - a process called design harvesting
      - fast-access data stored in the design catalog (current name – GQFdsgn.cat), housing metadata for views and view columns
        - V11 – GQFDsgn.cat goes away and design catalog is inboard – in the application database)
      - No, we won't be providing access to it – it will not be there in ND11

- Harvesting/exposing existing design for Domino neophytes, node.js developers will be a theme going forward with other node.js features


Copyright © 2018 HCL Technologies Limited | www.hcltech.com | HCL Confidential

PRODUCTS & PLATFORMS

HCL

# DESIGN HARVESTING/DESIGN CATALOG

- Updall and Design Harvesting
  - 2 new updall flags –

    **Updall <database path> -d** = Design catalog refresh

    **Updall <database path> -e** = Design catalog rebuild (for this database)
  - Unless design catalog populated for a database, DQL will not execute

# DEMO (2) of updall –e

- Caveats
  - Secure views (with reader lists) not supported (until ND11)
  - Bugs (FP2)
    - Hidden views (and their databases) fail to load (SPR JCUSBAFRN2)

 Copyright © 2018 HCL Technologies Limited | www.hcltech.com | HCL Confidential

PRODUCTS & PLATFORMS

HCL

# PROGRAMMING DQL - DQLEXPORER AND DOMQUERY

## DEMO 3 – DQLExplorer

- DomQuery <options>

  -f [DBName] data directory relative path, REQUIRED

  -q [double quoted string query] query string - either this or -z file required

  -z [QueryFile path] full path to a file containing query syntax queries
  delimited by #* at preceding line begin

  -e Explain the nodes

  -p Parse only (for testing)

  -v [MaxEntries] Maximum view entries to be scanned

  -c [MaxDocsScanned] Maximum number of documents to be scanned

  -m [Msecs] Maximum milliseconds to execute

  -x Exit on error (-z file case)

  -j No view processing performed (only NSF document scan and FT)

  -o [Output Report File path] full path to a file to which output will be written

- Finds documents and counts them – to get queries operational and optimized (-e recommended)


Copyright © 2018 HCL Technologies Limited | www.hcltech.com | HCL Confidential
PRODUCTS & PLATFORMS HCL

# PROGRAMMING DQL – JAVA AND LOTUSSCRIPT

- New 10.0.1 backend class – (Notes)DominoQuery
  - Created via (Notes)Database.createDominoQuery()
  - Methods
    - Parse – flags any syntax errors, does not run queries
    - Explain – executes the query, returns the way it was done – for tuning
    - Execute – executes the query, returns a (Notes)DocumentCollection
      - (Notes)DocumentCollections can be walked, intersected, etc.
    - SetNamedVariable, ResetNamedVariables
  - Attributes (Methods in Java)
    - NoViews – Run without view access (some syntax will fail)
    - MaxScanDocs, MaxScanEntries, TimeoutSec – limit settings (described later)
    - RefreshViews – all views opened and refreshed before used

Query String Input


Copyright © 2018 HCL Technologies Limited | www.hcltech.com | HCL Confidential

# PROGRAMMING DQL – JAVA AND LOTUSSCRIPT – IN VIEW TERMS

- Example – ONLY with small results (< 5000) where the query runs long
  - London orders originating in 2018 will not grow or change (it's 2019)
  - Code (Java)

```
DominoQuery dq = db.createDominoQuery();
String query = ("order_origin = 'London' and order_origin >= @dt('2018-01-01') and
        date_origin < @dt('2019-01-01')"
DocumentCollection doccol = dq.execute(query);
View ordfold = db.EnableFolder("Orders_2018");
doccol.putAllInFolder("London_Orders_2018");
```

  - At this point, the "London_Orders_2018" folder is available for all query inclusion/exclusion
  - Also, Part Numbers between 200000 and 205000 require special handling, and you ALWAYS want to find them as a document set, up to date, in queries
  - Create (or use the existing) Parts_200000 view with selection criteria of

```
SELECT Part_no >= 200000 & Part_no <= 205000
```


Copyright © 2018 HCL Technologies Limited | www.hcltech.com | HCL Confidential

PRODUCTS & PLATFORMS
HCL

# PROGRAMMING DQL – JAVA AND LOTUSSCRIPT – IN VIEW TERMS

- Example
  - Sample queries you can run with this folder and view (in view terms are fast!):

```
In ('Orders_2018', 'Parts_200000') - documents in either

In all ('Orders_2018', 'Parts_200000') – documents in both

In ('Orders_2018') and not in ('Parts_2000000') – documents in 2018 that are
                                                         NOT in Parts_200000

Order_origin = 'LA' and in ('Parts_200000') and not
       in ('Orders_2018') – Los Angeles orders in Parts_200000 and NOT in 2018

Part_no in all (198013, 111900, 304566) and
       in all ('Orders_2018', 'Parts_200000') – orders originating in 2018 and
                                             in Parts_200000 which also contained
                                             each of the parts 198013, 111900 and 304566
```


Copyright © 2018 HCL Technologies Limited | www.hcltech.com | HCL Confidential

PRODUCTS & PLATFORMS

HCL

# PROGRAMMING DQL – JAVA AND LOTUSSCRIPT

- “SQL Injection” attack exposure and remedy
  - Problem – queries are built on the fly, exposing syntax to users

```
String queryString = “part_no = “ +
                part_no_input +  // string supplied by user input
                “ or in (‘orders_2018’)”;
```

  - … if user  entered “299333 or sales_person < ‘’ ” the query returns every document

  - Solution – named substitution variables

```
String queryString = “part_no = ?partno or in (‘orders_2018’);
```

    - Use setNamedVariable prior to Execute call:

```
dq.setNamedVariable(“partno", 299333);   // where “partno” MUST match the query token following ?
```

      - Supports all (3) data types

 Copyright © 2018 HCL Technologies Limited | www.hcltech.com | HCL Confidential

PRODUCTS & PLATFORMS by HCL Technologies
HCL

# PROGRAMMING DQL – JAVA AND LOTUSSCRIPT

- "SQL Injection" attack exposure and remedy (continued)
  - You MUST call resetNamedVariables to clear memory to change values or to use different query syntax with different names

```
dq.resetNamedVariables();
```

  - Substitution variables can appear anywhere in DQL syntax a value can appear
    - Including view names within in terms (NOT 'viewname'.columnname)

```
dq.setNamedVariable("partno", 299333);
dq.setNamedVariable("view1", "Orders_2018");
dq.setNamedVariable("view2", "Parts_200000");
dq.setNamedVariable("sp", "Trudi Ayton");
dq.setNamedVariable("dtorigin", Trudi Ayton");
String queryString = "part_no = ?partno or in all (?view1, ?view2) or sales_person = ?sp");
Doccol = dq.execute(queryString);

...
```


Copyright © 2018 HCL Technologies Limited | www.hcltech.com | HCL Confidential

PRODUCTS & PLATFORMS

HCL

# PROGRAMMING DQL – JAVA AND LOTUSSCRIPT

- Related (and very useful) feature to (Notes)ViewEntryCollection object
  - Problem – when (Notes)DocumentCollections are intersect'ed or subtract'ed from a ViewEntryCollection, it loses whatever document order was in forced when the ViewEntryCollection was created
  - Solution – add new argument (maintainOrder) to preserve that order, allowing for automatic sorting of DQL results (Java)

```
DominoQuery dq = db.createDominoQuery();
View myview = db.getView("Myview");
myview.resortView("column7"); // establishes specific order for the view
ViewEntryCollection vec = myview.getAllEntries();
DocumentCollection doccol =
       dq.execute("sales_person = 'Trudi Ayton' and part_no in all (399322, 389923, 378883)");
vec.intersect(doccol, true);  // second parameter is NEW to 10.0.1 – maintain order
ViewEntry ve = vec.getFirstEntry();
```

DEMO 4 – Sorted results using intersect parameter maintainOrder

 Copyright © 2018 HCL Technologies Limited | www.hcltech.com | HCL Confidential

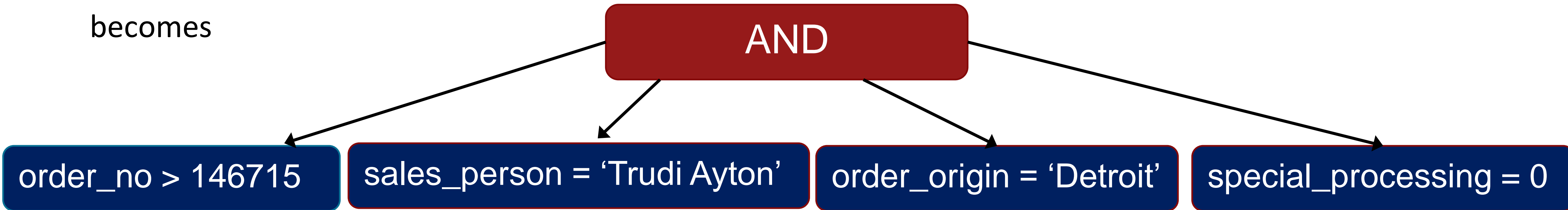
PRODUCTS & PLATFORMS

HCL

# DQL PERFORMANCE – BOOLEAN TREES

**order_no > 146751 and sales_person = 'Trudi Ayton’ and order_origin = ‘Detroit’ and special_processing = 0**

becomes



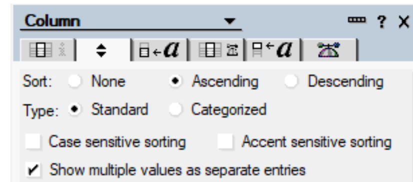

Where

- each “leaf” node can be executed in order according to cost (speed) – **NOT** in query term order
- the results of each leaf node can be injected other siblings as a pre-filter
- Boolean nodes control performance


Copyright © 2018 HCL Technologies Limited | www.hcltech.com | HCL Confidential
PRODUCTS & PLATFORMS HCL

# DQL PERFORMANCE

- Views and view columns
  - In order for DQL to use a view column for solving a <field> <operation> <value> query term
    1. View **must** have only Select @All as its selection criteria
    2. There **must** be a collated column with ONLY the field name as its formula
       - A collated column is either the leftmost column in the view with Sort order of Ascending checked in the view or
       - A column with ✔ Click on column header to sort Ascending checked
    3. Column **must** be non-categorized and show multiple values as separate entries (for multiply occurring fields)




  - If you want to use a view column that DOESN'T conform, use the 'Viewname'.column syntax
    - It **must** be sorted ascending
    - Allows for dual selection – by view selection criteria and query term


Copyright © 2018 HCL Technologies Limited | www.hcltech.com | HCL Confidential

# DQL PERFORMANCE

- View searching is faster than NSF scans, particularly for small ranges or equality searches
  - If you want to optimize a document scan, create a compliant collated index column
  - NSF scans are summary only and use injected prior results

- ANDing and ORing and sibling optimization
  - Sibling terms – any set of terms ANDed or ORed at the same precedence level
  - For this query:

**order_no > 146751 and sales_person = 'Trudi Ayton’**

where both order_no and sales_person can use qualifying view indexed columns ..

DQL the results of the cheaper term (sales_person = ‘Trudi Ayton’) are injected into the more expensive one

Terms must be ANDed together for this optimization


Copyright © 2018 HCL Technologies Limited | www.hcltech.com | HCL Confidential

PRODUCTS & PLATFORMS

HCL

# DQL PERFORMANCE

DEMO 5 – optimizing order_no range query by adding an index


Copyright © 2018 HCL Technologies Limited | www.hcltech.com | HCL Confidential
PRODUCTS & PLATFORMS by HCL Technologies
HCL

# DQL PERFORMANCE

- ANDing and ORing and sibling optimization (continued)
  - ANDed, Index-satisfied range siblings:

**order_no > 146751 and order_no < 150111**

will use a single, bounded view scan to satisfy both terms:

```
1.order_no > 146751 View-based range search estimated cost = 10
        Prep 0.164 msecs, Exec 21.178 msecs, ScannedDocs 0, Entries 558, FoundDocs 558
1.order_no < 150111 View-based range search estimated cost = 10
        Prep 0.93 msecs, Exec 0.0 msecs, ScannedDocs 0, Entries 0, FoundDocs 0 (0 cost - coupled with earlier sibling)
```

  - All NSF scan siblings (ORed or ANDed) are satisfied with one pass of the document summaries

```
1. OR           (childct 2) (totals when complete:) Prep 0.0 msecs, Exec 157.701 msecs, ScannedDocs 10000, Entries 0, FoundDocs 8485
        2.partno = 389 NSF document search estimated cost = 100
                Prep 0.67 msecs, Exec 157.695 msecs, ScannedDocs 10000, Entries 0, FoundDocs 5400
        2.partno = 587992 NSF document search estimated cost = 100
                Prep 0.67 msecs, Exec 0.0 msecs, ScannedDocs 4600, Entries 0, FoundDocs 3085
```

  - NSF scans are smart and pre-filtered when possible


Copyright © 2018 HCL Technologies Limited | www.hcltech.com | HCL Confidential

PRODUCTS & PLATFORMS

HCL

# DQL PERFORMANCE

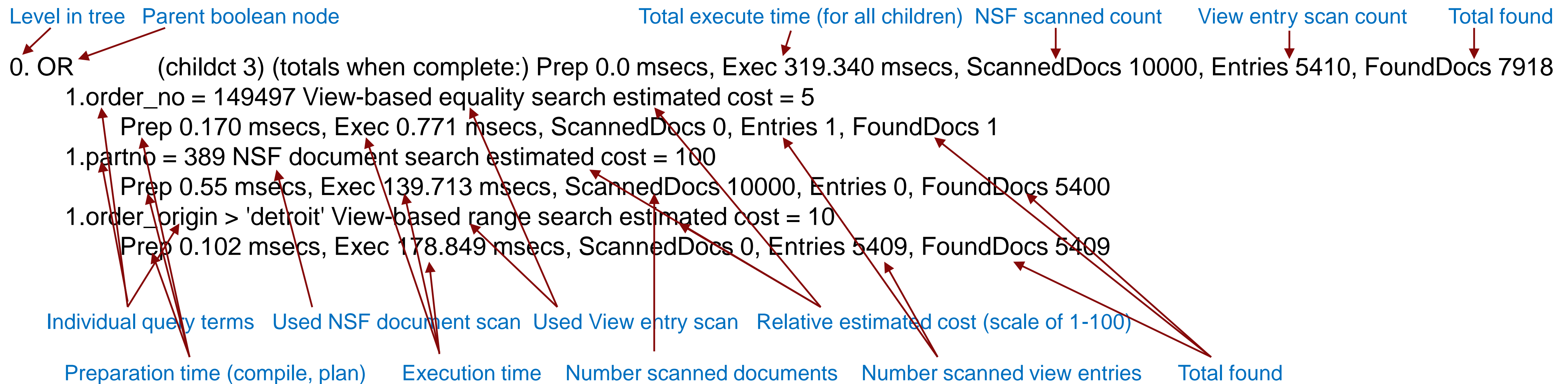
- Max limit settings
  - To protect from runaway queries
    - All settings are cumulative for processing for the entire query (defaults subject to change)
    - MaxDocsScanned – maximum allowable NSF document scanned (500000)
    - MaxEntriesScanned – maximum allowable index entries scanned (200000)
    - MaxMsecs – maximum time consumed in milliseconds (300000) (5 minutes)
    - Limits checked periodically across all operations
    - Can be overridden via notes.ini for the server or per call to the DGQF
  - If limits are exceeded DQL will return an error and no results


Copyright © 2018 HCL Technologies Limited | www.hcltech.com | HCL Confidential

# DQL PERFORMANCE

- EXPLAIN
- Tool to show the Boolean tree used to process your query:

order_no = 149497 or partno = 389 or order_origin > 'detroit’





Copyright © 2018 HCL Technologies Limited | www.hcltech.com | HCL Confidential

PRODUCTS & PLATFORMS HCL

# DQL PERFORMANCE – HOW DQL CAN BECOME SLOW

- Cardinality –
  - All Domino (and for that matter all database) operations go slower with more data to find and process
  - Processing > 10K documents may create too much lag for people waiting for data
  - High cardinality queries (most useful for reporting and batch processing) should be run off-hours or in batch mode
- Volatility/Contention
  - Due to the number of views and indexes within them, Domino indexing is periodic, NOT at time of updates
  - When lots of threads try to “refresh” (really “update”) a view, they will contend and wait for each other
  - Some remediation with V10 inline indexing

DEMO 6 - Cardinality

 Copyright © 2018 HCL Technologies Limited | www.hcltech.com | HCL Confidential

PRODUCTS & PLATFORMS

HCL

# DQL AND SECURITY

## Honoring reader lists

- Reader list processing slows down view processing but no effect on NSF document scanning
- View and folder document extraction (IN clause with view names) can be VERY expensive when processing Reader lists
  - If multiple views/folders are specified with IN ALL (ANDing the sets), reader list processing only necessary on the smallest view or folder (counts are kept in the design catalog)
  - For ORing, when a view has lots of documents, it will exceed the view entry maximum and fail the query

## Design ACLs

- V10 - to avoid breaching security of ACL-protected views, design catalog CANNOT be exposed
  - Server ID only has read/write privileges
  - Fixed in V11
- Disclaimer – we may need to exclude ACL-protected views from the catalog entirely


Copyright © 2018 HCL Technologies Limited | www.hcltech.com | HCL Confidential
PRODUCTS & PLATFORMS
HCL

# DQL REMAINING DESIGN DECISIONS

Processing Reader lists is **NOT** free!

Poll – show of hands for Option A or B

| Option A | Option B |
| --- | --- |
| Results returned filtered by reader lists | Results and include documents and, application code |
| Counts are filtered – do not include documents the user cannot see | Counts include documents that user can't see |
| Application code requires no logic to catch security filtering | Application code has - if doc.IsValid() = TRUE |
| Speed: 10 seconds/1 minute/10 minutes | Speed: 100 msecs/600 msecs/6 seconds |

Of course there is Option C – HCL drastically speeds up reader list processing, but it will be a V12 effort


Copyright © 2018 HCL Technologies Limited | www.hcltech.com | HCL Confidential

# DQL REMAINING DESIGN DECISIONS

## Multi-database queries

- Current plan is to deliver
  - Federated queries – same query, same fields across multiple databases (NOT joins)
  - Sorted results across all database

## Formula Language

- Avoid direct @functions in the DQL syntax
- Instead, supply named reference to already-existing Formula Language entities – e.g. (sub)form display-only fields
- But .. what are the critical @functions that **should** be in raw in DQL?


Copyright © 2018 HCL Technologies Limited | www.hcltech.com | HCL Confidential

# Questions/discussion


Copyright © 2018 HCL Technologies Limited | www.hcltech.com | HCL Confidential

PRODUCTS & PLATFORMS
by HCL Technologies
HCL