Making proper use of transactional metadata queries

Axel Faust, Acosix GmbH

Quick Survey

- Know what transactional metadata queries (TMQs) are?
- Using TMQs for custom solutions?
- Confident in TMQ performance?

Basics

TMQ Support

- Lucene – Alfresco <= 4.2
  - “Semi-consistent” in cluster (async indexing)

- DB query – Alfresco >= 4.2
  - Alfresco FTS / CMIS query languages
  - Various enhancements in 5.x
  - Original requirements: https://issues.alfresco.com/jira/browse/ALF-19126
    - BTW: kudos to Andrew Hind for decent JIRA content

TMQ Conditions

1. Optional indices added
   - Still not the default ?!?
   - system.metadata-query-indexes.ignored=false
   - system.metadata-query-indexes-more.ignored=false (5.1+)
2. Compatible FTS / CMIS query
3. System / request set to allow transactional query

Compatible FTS Query

- Restricted selectors
  - PARENT, TYPE, ASPECT, EXACTTYPE, EXACTASPECT
  - Exact property terms, phrases or “prefix terms”
    - =cm:creator:afaust / =cm:author:”Axel Faust” / =cm:creator:afa*
  - Excluded: size, encoding, mimetype
  - d:boolean (5.1+)

- Operators
  - AND / NOT
  - OR (5.1+)

Compatible CMIS Query

- Comparisons
  - =, >, <, >=, <=, <>, (NOT) IN, LIKE, ANY, IS (NOT) NULL
  - contentStreamLength / contentStreamMimeType

- Operators and functions
  - AND / NOT
  - OR (5.1+)
  - IN_FOLDER

Query Consistency

- Global default
  - solr.query.fts.queryConsistency
  - solr.query.cmis.queryConsistency
- Request-based override (Java API only)
- Options
  - TRANSACTIONAL_IF_POSSIBLE, TRANSACTIONAL
  - EVENTUAL

Miscellaneous Limitations

- Multi-valued properties
  - Non equality – which row?
  - CMIS selector: (NOT) ANY … IN (…)
  - Unexpected sort order

node_íd	<value>	qname_id	list_index
123	abc	65	0
123	xyz	65	1
124	par	65	0

- Case sensitive comparison values

- Property existence check
  - CMIS: “IS NULL” => non-existence + actual NULL

Use Case Patterns

Use Case Fundamentals

- Stable, non-fragmented key metadata
  - Technical / business identifiers
  - List of pre-defined values
  - Node references

- Drill-down on hard data
  - Location irrelevant or coded in metadata
  - No room for fuzzy-ness

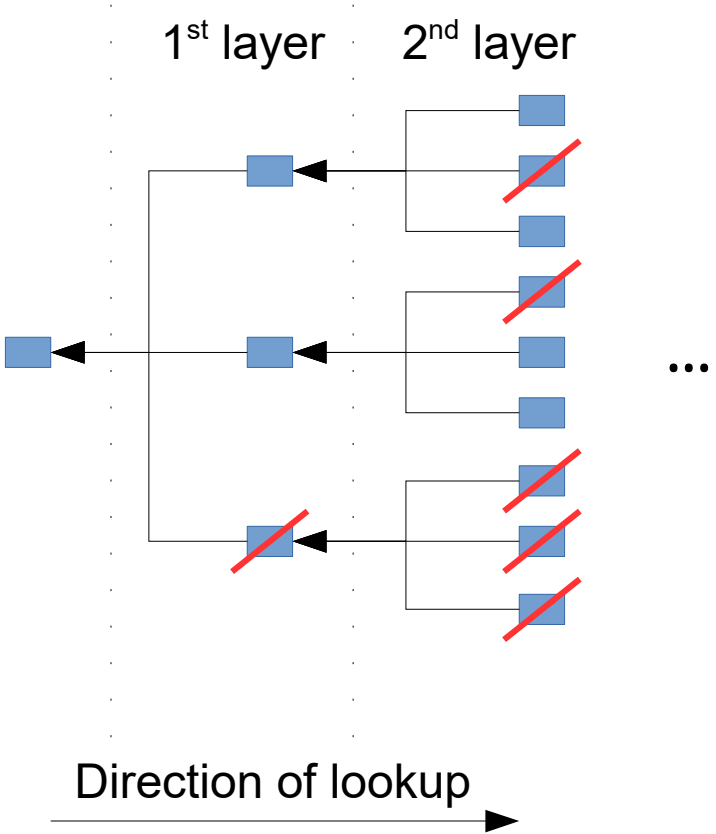
Scalable Technical Integration

- Example: SAP integration
  - Stateless with cluster failover + balancing
  - Many requests in short order
  - Multiple technical identifiers

- Document retrieval / listing
  - Pinpoint accuracy in millions of documents
  - Java API for explicit transactionality

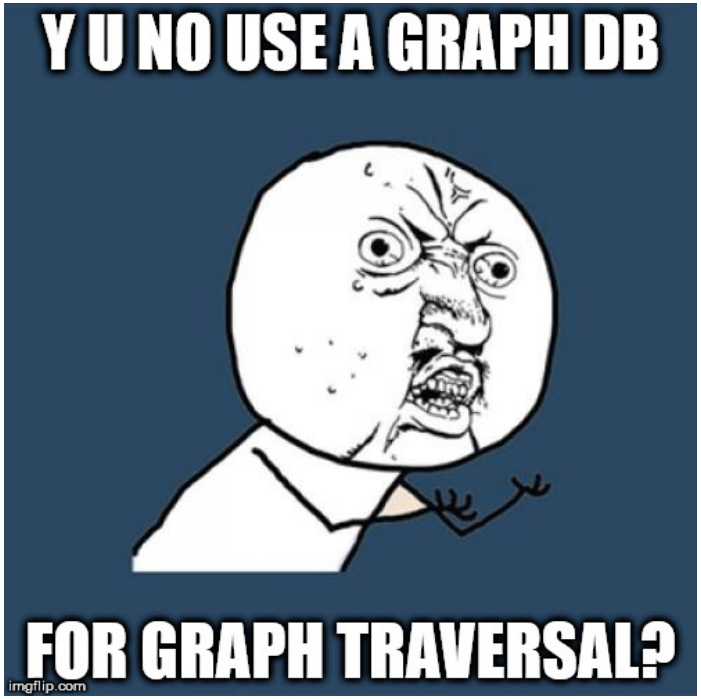
connexas

Reverse Reference Chain Traversal



- Processing dependent objects
  - Example: (virtual) property inheritance
  - Filtered by state
- d:noderef property
  - Copy/clone of association
- Each layer in single query
  - CMIS: IN comparison

Reverse Reference Chain Traversal



- Graph use case = anti-pattern?
  - Re: “Alfresco Anti-Patterns”
    (Jeff Potts, ecmarchitect.com)
- Graph DB without ECM features
- Traversal only a simplification
  - Conserve CPU cycles / memory

Multi-Parent Child Lookup

- Similar to reference chain traversal
  - Two stage look-up
  - Alternative to expensive PATH

- Example query
  - PATH:”/app:company_home/st:sites/*/cm:documentLibrary/* AND ASPECT:”my:acme”
  - 1st stage: ASPECT:”st:siteContainer” AND =cm:name:”documentLibrary”
  - 2nd stage: PARENT:(“<nodeRef1>” … OR “<nodeRefN”>) AND ASPECT:”my:acme”

- Useful e.g. for data list aggregation

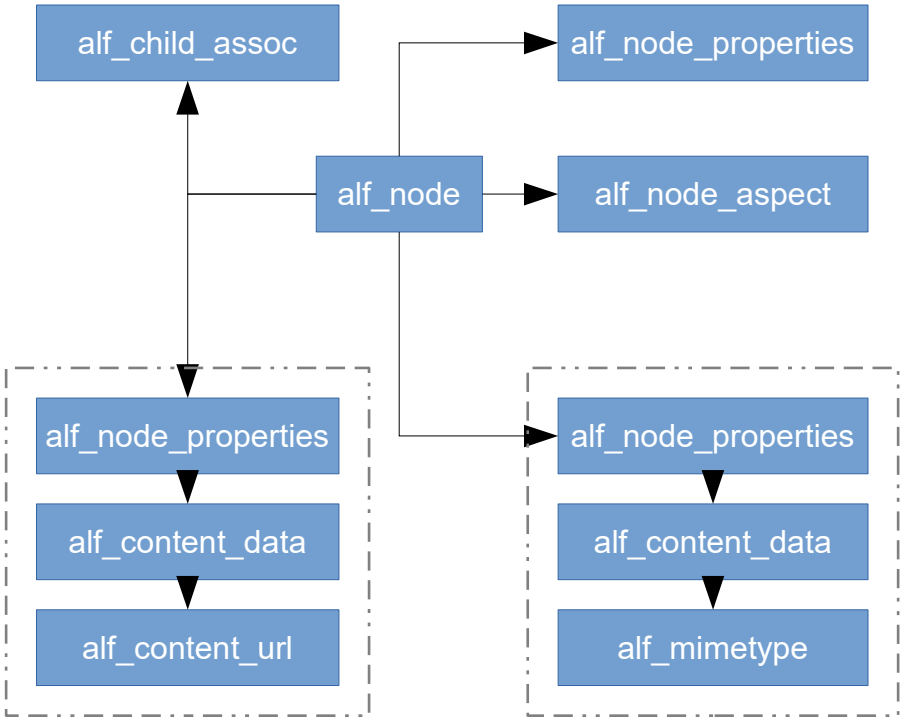
End-User Queries

- Limited Usability?
  - No full-text search capabilities
  - No stemming / fuzzy metadata searches
  - No range queries (FTS)

- Ideal for custom UIs
  - Case management or archive
  - Interchangeable FTS / CMIS use

Performance aspects

“[TMQ] is never going to be fast”



- JOINs + sub-SELECTs
- Extremely normalized data
  - Relative table sizes
- ACL check via post-processing

Corner Stones of Performance

- Query selectivity
- Sensible permission model
  - Beware DynamicAuthority for “Read”
- Base DB configuration + maintenance
- Cache configuration / state

Query Selectivity

- Unlimited select in DBQueryEngine
  - Query plan + memory utilisation
  - Worst case: table scan
- Target: <= 2-5000 raw results
- Alternative: patch for incremental query
  - Focus on “first pgae(s)” performance

Query Planner – The “Saboteur”

- Basis for all SQL: Prepared Statements
  - Dynamically generated, yet likely identical
  - How are query plans created for these?

- Unsuitable plan re-use (Oracle, DB2, MS SQL)
  - “Locked in” assumptions (or slow adjustment over time)
  - Ignored custom indices

Query Planner – Explain Yourself

- PostgreSQL / MySQL / MariaDB
  - PREPARE mystmt (varType1,…) AS SELECT…WHERE colA = $1…AND colB = $n;
  - EXPLAIN ANALYZE EXECUTE mystmt (val1,…);
  - DEALLOCATE PREPARE mystmt;

- Challenges
  - Obtaining statements (p6spy / datasource-proxy)
  - Identical statements (case / whitespace)
  - DB memory state

Query Planner – General Re-Optimisation

- Different between DBs
  - Oracle: Undocumented BIND_AWARE query hint
  - MS SQL: OPTION(RECOMPILE) query option
  - DB2: REOPT ALWAYS bind option

- Limited help from support

Statistics Limitations

- Data in key columns skewed
  - E.g. alf_node.transaction_id + batch loads
  - E.g. alf_node_properties.string_value + unique NodeRefs
- Recommendation: “sensible” granularity increase
  - Sampling rate / explicit target
    - Per column if supported
  - Planning vs execution trade-off

Index Limitations

- Generic indices not ideal
  - Mixed data
  - Size / IO influences planner

node_id	..id	..id	boolean_value	long_value	float_value	double_value	string_value	serializable_value	qname_id	list_index	locale_id
4035970	6	6	FALSE	0	0	0	acosix-utility	<binary data>	26	-1	1
4035969	7	6	FALSE	0	0	0	2011-08-25T22:00:00.000Z	<binary data>	270	-1	1
4035969	6	6	FALSE	0	0	0	SL4683098611	<binary data>	269	-1	1
4035969	6	6	FALSE	0	0	0	Hasselt	<binary data>	254	-1	1
4035969	6	6	FALSE	0	0	0	High	<binary data>	253	-1	1
4035969	6	6	FALSE	0	0	0	In	<binary data>	252	-1	1
4035969	6	6	FALSE	0	0	0	AUTOMA	<binary data>	251	-1	1
4035969	6	6	FALSE	0	0	0	Unknown	<binary data>	250	-1	1
4035969	6	6	FALSE	0	0	0	partyHolder/96722	<binary data>	249	0	1
4035969	6	6	FALSE	0	0	0	Unknown	<binary data>	248	-1	1
4035969	6	6	FALSE	0	0	0	Lifetime	<binary data>	247	-1	1
4035969	6	6	FALSE	0	0	0	EmailUP	<binary data>	246	-1	1
4035969	6	6	FALSE	0	0	0	Secret	<binary data>	245	-1	1
4035969	6	6	FALSE	0	0	0	Unknown	<binary data>	244	-1	1
							(26-08-11) BULTOT Virginie 200287297-				

Index Limitations

- Generic indices not ideal
  - Mixed data
  - Size / IO influences planner
- Convenient option: custom partial / filtered index
  - PostgreSQL / MS SQL only
- Advanced option: table partitioning
  - Secondary benefit: statistics

Using Partial Indices

```
-- may help reference lookups
create index idx_np_sv_noderef
        on alf_node_properties (qname_id, string_value, node_id)
        where string_value LIKE 'workspace://SpacesStore/%';


-- may support core use case – common value and well known property (id via alf_qname)
create index idx_np_string_s_party
        on alf_node_properties (qname_id, string_value, node_id)
        where qname_id = 249;
```

- Use dependent on query + statistics + planner

Partitioning alf_node_properties

- Identify + use property clusters
  - “Dead weight”: version, thumbnail, archive…
  - References: tags, categories…
  - Key identifiers
  - Fixed vs. free form text, numerics, booleans...
- Transparent view + insert trigger
- Drawback: potential upgrade effort

Cache Impact

- No “bulk fetch” – limit DB round-trips
  - node.nodesSharedCache
  - node.aspectsSharedCache / node.propertiesSharedCache
- Read access checks
  - Shortcut to cache, unless custom “Read” DynamicAuthority
  - readersSharedCache / readersDeniedCache

Final Remarks

Don’t Panic + KISS

- Just as fast as SOLR
  - Even faster on “unseen” queries
- Scales well into 10^6 nodes
  - Example: SAP integration (connexas)
- Adopt TMQ gradually
  - Focus on metadata, not PATH
  - Restrictive query constructs whenever possible

Don’t Panic + KISS

- Stick to appropriate use cases
  - Small result sets, simple CRUD patterns
  - Convenience features? - Use SOLR
- Sort by objective data
  - Numeric / temporal values
  - Design list constraint for locale-independent sort
- Personal plea: stop using “lucene” / lucene-related API

@ReluctantBird83

axel.faust@acosix.de