High performance model queries and their novel applications

Benedek Izsó
Zoltán Szatmári
István Ráth
The speaker’s background

- **Research @ academia**
  - Metamodeling, DSMLs, model transformations, ...
    - The VIATRA2 Model Transformation Framework (2004-)
  - Model queries
    - The EMF-IncQuery project (2010-)

- **Consulting & Development @ industry**
  - Eclipse consultant and trainer (2006-)
  - Architect, model-driven tools and projects (2008-)
The problem

- Scalable modeling tools
  - Issues encountered by several industrial partners using EMF-based tools
  - Modeling scenarios can get really complex really quickly
    - Instance models of size 1-10M and beyond
    - Performance issues with model transformations and code generators have an adverse effect on everyday development tasks

- Scalability
  - Complex (meta)models
  - Large instance models
  - High performance model I/O
Focus: model queries

- Model queries
  - “a piece of code that retrieves a given set of the model”

- Ubiquitous
  - Every model access/read is a (simple) query
  - More complex: Views, content providers
  - Most complex: Model transformations, code generators

- In interactive (user-driven) use cases, queries are far more common than model manipulation
  - → query performance is crucial

- Query performance
  - The speed of content retrieval
    - First vs consecutive vs throughput
    - Query result vs query contents

- Essence of performance issues
  - slow queries
  - Considerable programming effort necessary to get complex queries right
Technology landscape – what to build a modeling tool on?

- **Eclipse Modeling**
  - EMF (&variants, CDO, ...)
  - Proven tools: Plain Java queries, OCL (&variants), EMF Query (1-2)
  - New/experimental technologies: EMF-IncQuery, JBoss Drools, ...

- **Triple stores / graph databases**
  - OWL/RDF, SPARQL
  - RacerPro, Pellet, Sesame, Jena, OpenVirtuoso, Allegro, StarDog
  - Modeling tools: not (yet) a traditional use case
  - But...
    - Huge, distributed knowledge bases
    - Highly optimized SPARQL engines
    - Expensive commercial tools

- **Relational DB**
  - SQL, PL/SQL, In-memory
  - Overhead: ORM layers

Further options (not covered yet)
- Modeling frameworks other than EMF
- Plain XML/XPath
- OO DB
- K-V (NoSQL)
...
Assessment through benchmarking

- Railway system design DSML
- Benchmark: well-formedness checking in 4 phases, using **in-memory models**
  - Phase 1: model load into memory (READ)
  - Phase 2: rule validation (retrieve all invalid elements) (CHECK1)
  - Phase 3: scattered, programmatic, in-memory model modification (introducing new errors) (EDIT)
  - Phase 4: revalidation (retrieve all invalid elements) (CHECK2)
- Several test queries with different characteristics
  - Varying “difficulty”
  - Approx. same percentage of invalid elements present in the model for all queries
  - Tool expected to return ALL invalid model elements with references/IDs
- Measured
  - Execution times for all phases
  - Memory usage over the entire sequence
Focus

- Model sizes: up to 2.5M elements
  - Industrial partner: we would like to see 10M+ elements!
    - Note: 10M+ model elements ~ 50M+ triples
  - This is not yet possible with in-memory tech 😞

- Query complexity
  - Typical well-formedness constraints similar to actual tools
  - Ranging from simple attribute checking to complex structural checks (circle detection)

- Interesting use cases
  - READ+CHECK1: “batch validator” (models are read into memory, rules are evaluated)
  - EDIT+CHECK2: “on-the-fly revalidation in an editor” (list of invalid elements updated according to changes in the model)
RESULTS
Disclaimer

- This is still work in progress
- External audits (not complete yet)
  - OCL: Ed Willink
  - Drools: Mark Proctor
  - Java: our industrial partner (Eclipse-based tool development)
  - OWL/SPARQL: technology experts at an academic partner
- Detailed results, analysis and the corpus will be available online
  - This is just an appetizer 😊
Question 1

- Q: How does model size affect query performance?
Batch validation

SPARQL tools are generally 1-2 Orders-of-Magnitude (OM) slower than top EMF tools.

Read time is dominating.

Eclipse OCL does really well here, followed closely by IncQuery.
Revalidation on-the-fly

Incremental EMF engines typically 5-10x faster, sub-100ms response times for up to 1.5M elements
A closer look to the top

Performance advantage of OCL-IA over OCL is not significant with complex queries.
Memory usage

AllTestCaseAvg Memory Usage

Model Size [#elem]

EMF-IncQuery  Eclipse OCL  OCL Impact Analysis

Memory [kByte]

1e+07

1e+06

1e+05

1e+04

1e+03

1e+02

1e+01
Answer 1

- **Q:** How does model size affect query performance?
- **A:** Polynomial growth in response times.
  - Batch validation: execution times grow according to a low-order polynomial (exponent depends on the technology)
  - On-the-fly revalidation: similarly, but incremental engines achieve much lower exponents (can be even close to constant)
Question 2

- Q: How does the complexity of the query affect performance?
Re-validation on-the-fly

• 1-2 OM differences with OCL
• Max 0.5 OM difference with IQ
Q: How does the complexity of the query affect performance?

A: Significantly!

- Metamodel and contents of the query together determine the overall performance
  - E.g. inverse relations and model navigation
  - Metamodel and queries should be optimized together
- RETE-based tools (Drools, IncQuery) can get around this
  - High performance regardless of metamodel & query structure
  - At the cost of increased memory overhead
    - But (much) less than OCL-IA!
Results

- Fastest batch validator: **Eclipse OCL**
- Fastest incremental validator: **EMF-IncQuery**
- Fast OWL tools (RDF reasoners): **Sesame, Jena**
  - Significantly better than commercial SPARQL engines
  - But at least 1-2 OM slower than top EMF technologies
- (In-memory) RDBMS are slow
  - Due to ORM overhead
  - Omitted for simplicity, full report will be online
How to build a scalable modeling tool?

- If everything fits into your workstations’ RAM
  - 8GB RAM → 1-2M model elements
  - You can (& should) (continue to) use EMF
    - EMF tweaks (e.g. diet) have no significant impact
    - Instead, optimize queries and metamodels
  - Use Eclipse OCL for batch validation
    EMF Query 1-2 has much less expressive power, not focused on in-memory models
  - Use incremental engines for on-the-fly queries
    - EMF-IncQuery supports on-the-fly validation, view maintenance, queries for frequently executed and performance sensitive model transformations, ...
    - With RETE-based tools such as Drools or IncQuery, no manual metamodel/query etc. optimization necessary
    - Cost: additional memory requirements

- Else
  - To be investigated
Overview

- A model query engine
  - Supports batch queries
  - Optimized for incremental queries!
- Incremental evaluation
  - Based on the RETE algorithm
  - *Compute once, update afterwards*
  - Gain: Instant re-evaluation
  - Price: Uses some more memory
    - Manageable with proper life cycles
Benefits

- Makes on-the-fly well-formedness validation, view maintenance, ... feasible over really large instance models
- Simplifies writing really complex queries
  - Graph pattern language
    - Capture local & global constraints
    - Compositions, reusable constructs
    - Arbitrary recursion, negation
  - Highly reusable, you can even build query libraries
- Easy-to-integrate into existing apps
  - works with any EMF DSL
  - Integration through standard EMF APIs
New features: IncQuery v0.7

- **Tooling**
  - Xtext2-based tooling
  - New language features (w.i.p.)
    - Unlimited recursion and transitive closure
    - Short attribute notation
    - Aggregate functions
    - Match/exceed the expressive power of OCL, while providing more flexible re-use

- **Runtime**
  - Generic queries
    - Build and execute queries on-the-fly, using Java and IQ PL
  - RETE construction optimizations
    - Memory footprint significantly reduced
  - SCC-based incremental transitive closure

- **Extensions / add-ons**
  - Derived features
  - Efficient complex event processing over models
  - Design-space exploration
Final points

- **EMF-IncQuery: proposal submission process underway**
  - New sub-project under EMFT

- **Pointers**
  - [http://viatra.inf.mit.bme.hu/incquery](http://viatra.inf.mit.bme.hu/incquery)

- Thank you very much!