Stardog Experience with LDBC

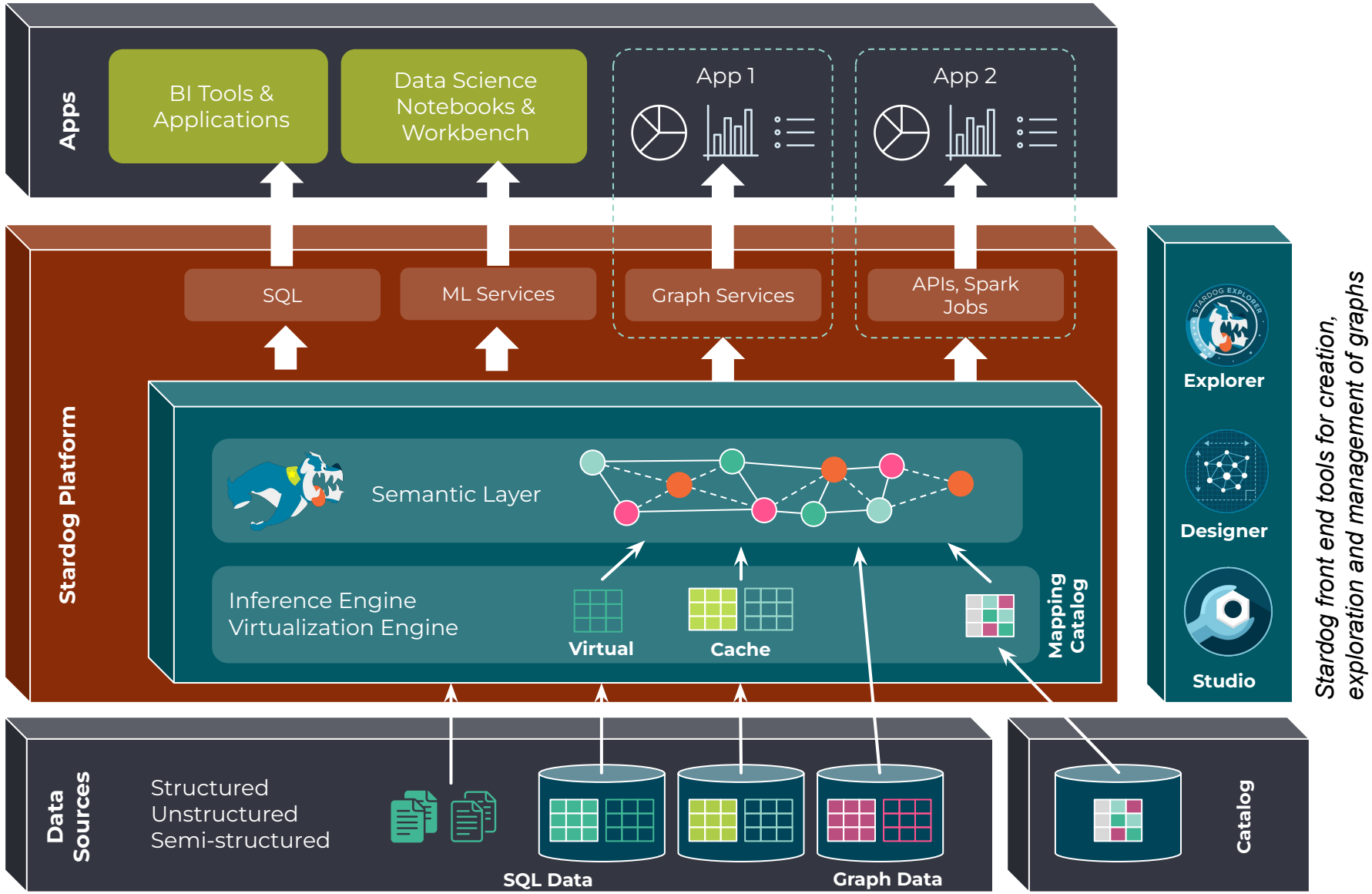
Evren Sirin
CTO & Co-Founder
Stardog

STARDOG

Stardog Platform Overview

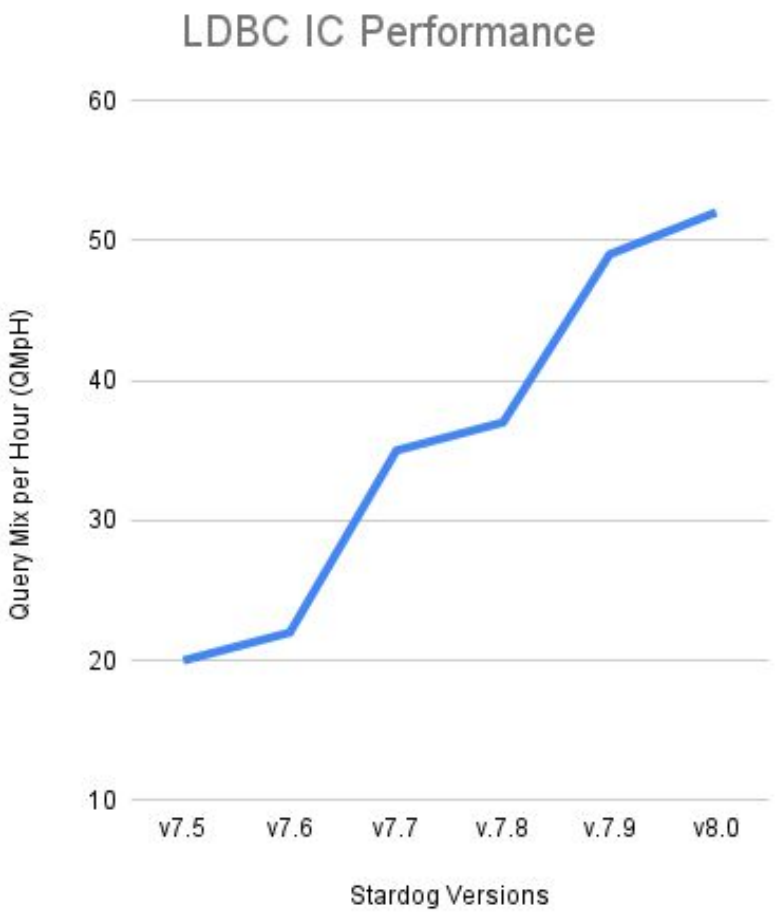
- Data model based on RDF
- Edge property extension (RDF-star)
- Load graph data into Stardog OR virtualize SQL & NoSQL databases as graphs
- Data storage in RocksDB, virtualization built on top of Calcite with MySQL protocol
- Query with SPARQL, GraphQL or SQL

STARDOG



Stardog and LDBC

- Following LDBC from a distance
- Started looking at SNB in more detail last year
  - Read-only workloads
  - Interactive Complex (IC) queries
  - Data materialized in Stardog
  - Only SPARQL query answering

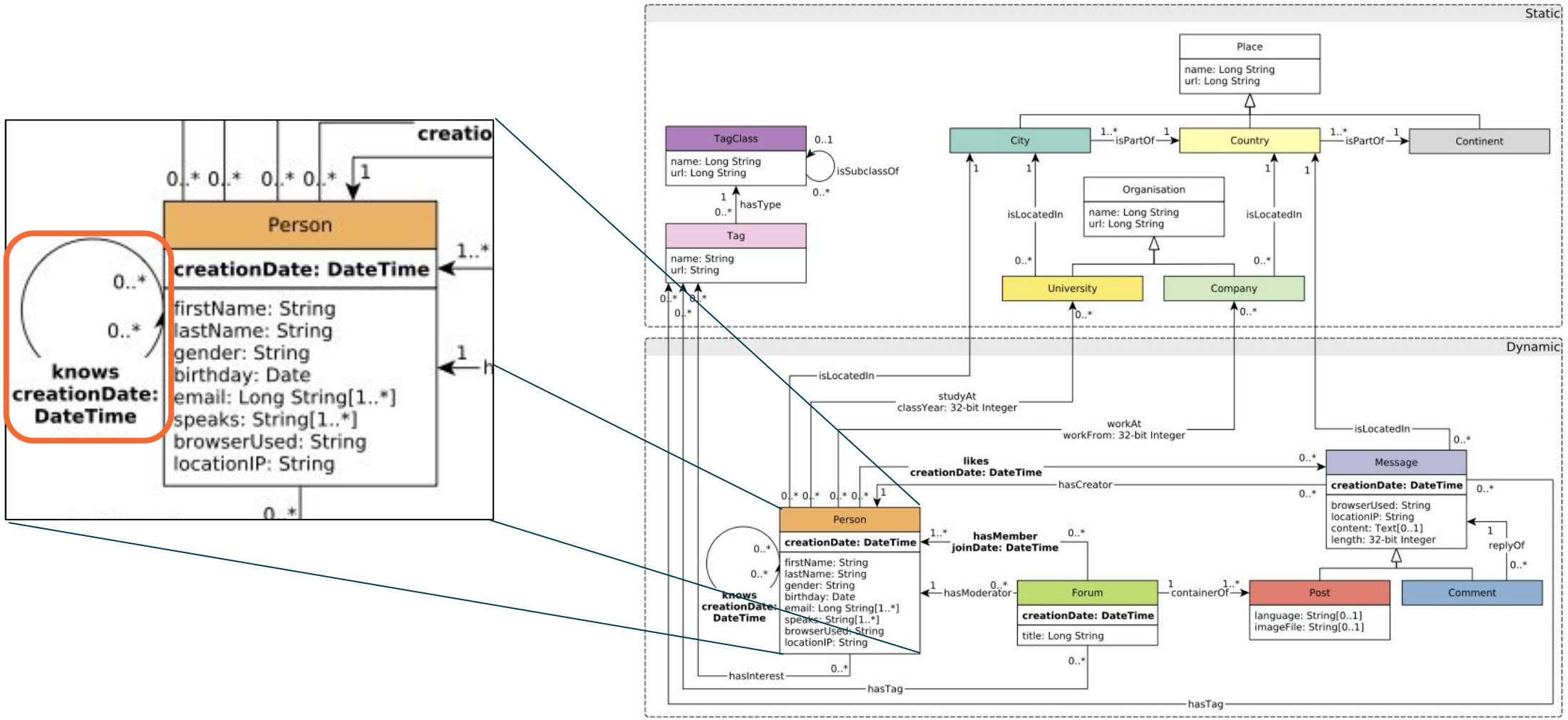


LDBC Challenges

1. Tooling Challenges
2. RDF Challenges
3. SPARQL Challenges
4. Query Optimizations Challenges

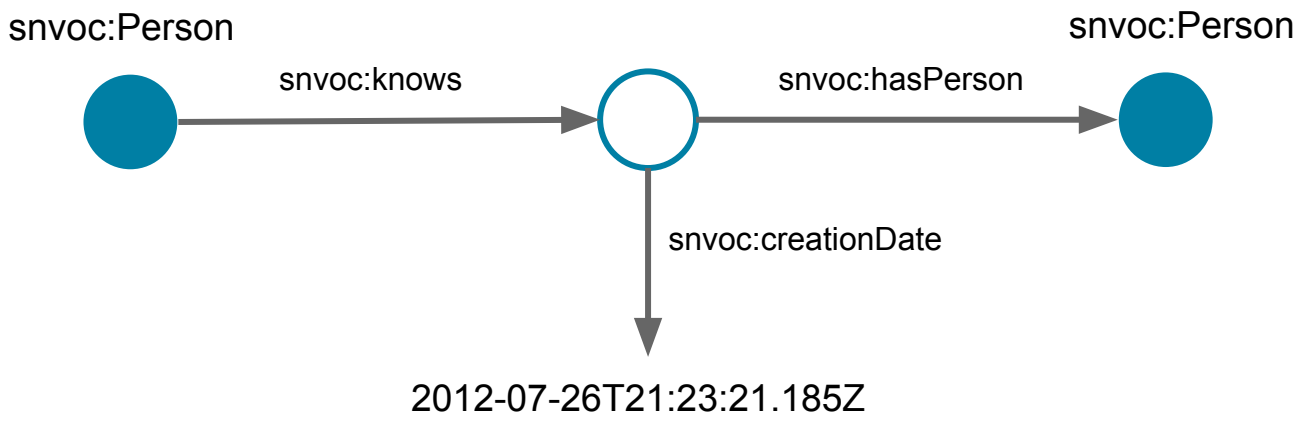
STARDOG

RDF Challenge - Edge Attributes



RDF/SPARQL Representation

An additional node is required in RDF to represent edge attributes

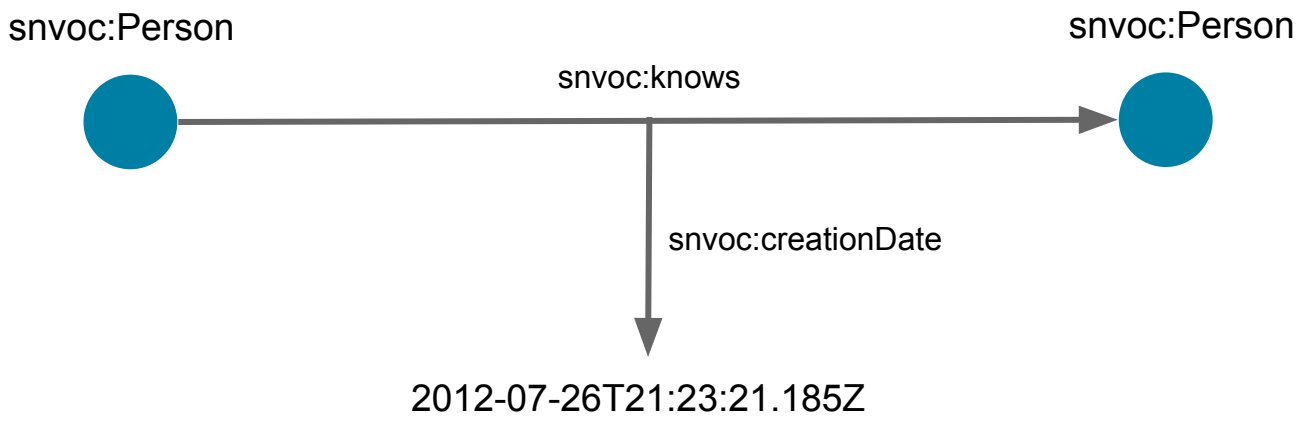


This results in additional joins for the SPARQL queries

```
?rootPerson snvoc:id ?rootId .
?fr a snvoc:Person .
{
    ?rootPerson (snvoc:knows/snvoc:hasPerson)|^(snvoc:knows/snvoc:hasPerson) ?fr .
    BIND( 1 AS ?distOneInner )
} UNION {
    ?rootPerson ((snvoc:knows/snvoc:hasPerson)|^(snvoc:knows/snvoc:hasPerson))/((snvo
    BIND( 2 AS ?distTwoInner )
```

RDF-star/SPARQL-star Representation

No additional node required in RDF-star



Results in simpler and more performant SPARQL queries (20-30% faster)

```
?rootPerson snvoc:id ?rootId .
?fr a snvoc:Person .
{
    ?rootPerson snvoc:knows|^snvoc:knows ?fr .
    BIND( 1 AS ?distOneInner )
} UNION {
    ?rootPerson (snvoc:knows|^snvoc:knows)/(snvoc:knows|^snvoc:knows) ?fr .
    BIND( 2 AS ?distTwoInner )
```

SPARQL Challenge - Property Paths

- All patterns in SPARQL are directional
  - Need to use union property paths (|) with inverse paths (^)
- Property paths in SPARQL do not have {min, max} limits
  - Need to use explicit UNION clauses

```
?rootPerson snvoc:id ?rootId .
?fr a snvoc:Person .
{
    ?rootPerson (snvoc:knows/snvoc:hasPerson)|^(snvoc:knows/snvoc:hasPerson) ?fr .
    BIND( 1 AS ?distOneInner )
} UNION {
    ?rootPerson ((snvoc:knows/snvoc:hasPerson)|^(snvoc:knows/snvoc:hasPerson))/((snvoc:knows/snvoc:hasPerson)|^(snvoc:knows/
    BIND( 2 AS ?distTwoInner )
} UNION {
    ?rootPerson ((snvoc:knows/snvoc:hasPerson)|^(snvoc:knows/snvoc:hasPerson))/((snvoc:knows/snvoc:hasPerson)|^(snvoc:knows/
}
```

SPARQL Challenge - Shortest Paths

- No shortest path feature in SPARQL
- Stardog provides a SPARQL extension for shortest paths
  - Next step: Try embedded path queries to solve the previous problem

```
PATHS
START ?person1 {
   ?person1 a snvoc:Person .
   ?person1 snvoc:id "28587302322515"^^xsd:long .
}
END ?person2 {
   ?person2 a snvoc:Person .
   ?person2 snvoc:id "4398046518685"^^xsd:long .
}
VIA {
   ?person1 ((snvoc:knows/snvoc:hasPerson)|^(snvoc:knows/snvoc:hasPerson)) ?person2
}
```

Query Planning in Stardog

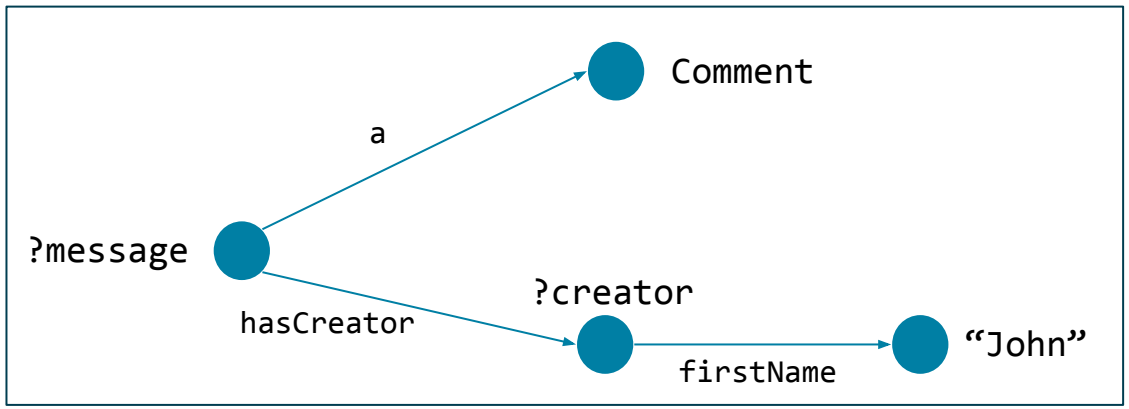
- Stardog implements the Volcano model where each algebraic expression corresponds to some executable operators (cf. Graefe work on Cascades framework)
  - triple patterns → index scans
  - BGPs → joins over scans
  - joins → merge, hash, loop (etc.) join algorithms

- Information (SPARQL solutions) flows bottom-up

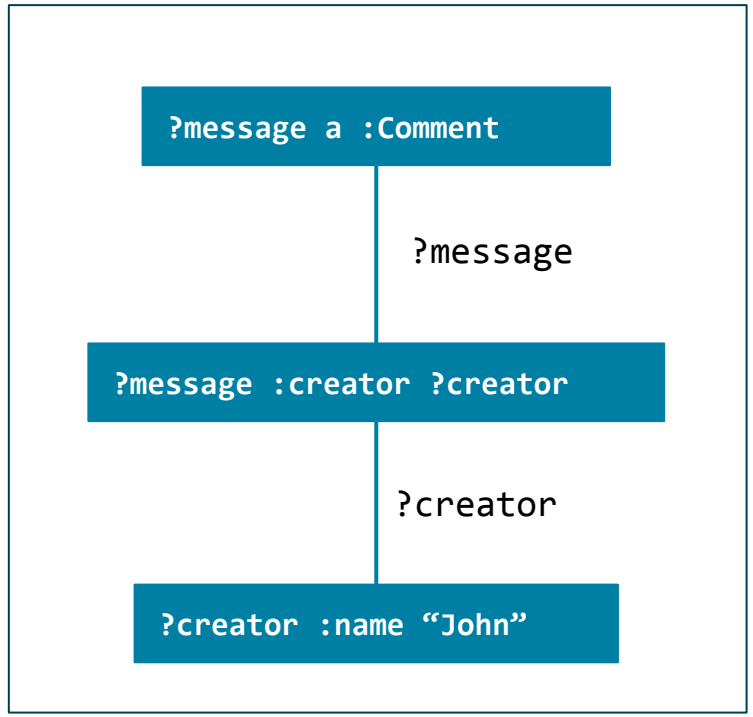
Query Planning Steps

SPARQL Query

```
?message a :Comment .
?message :hasCreator ?creator .
?creator :firstName "John"
```



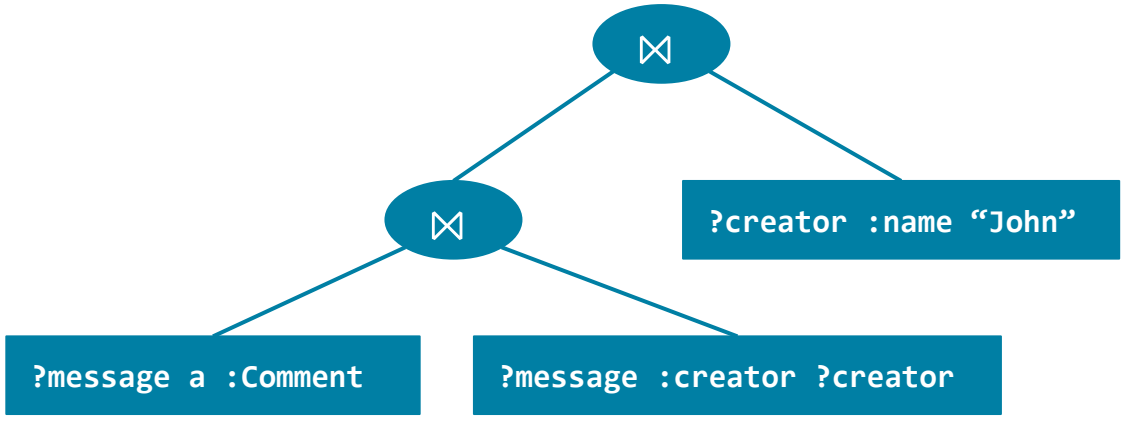
Graph representation of the query

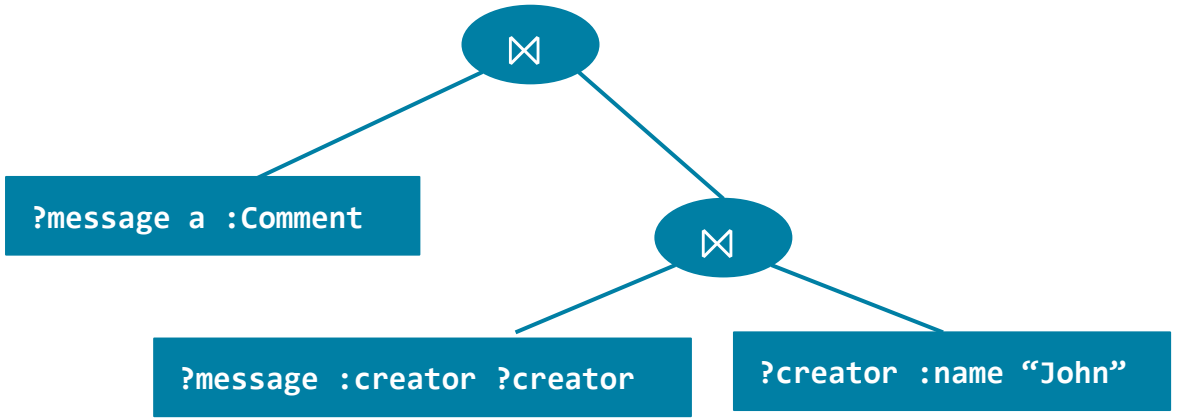


Join Graph

Join order optimization in Stardog

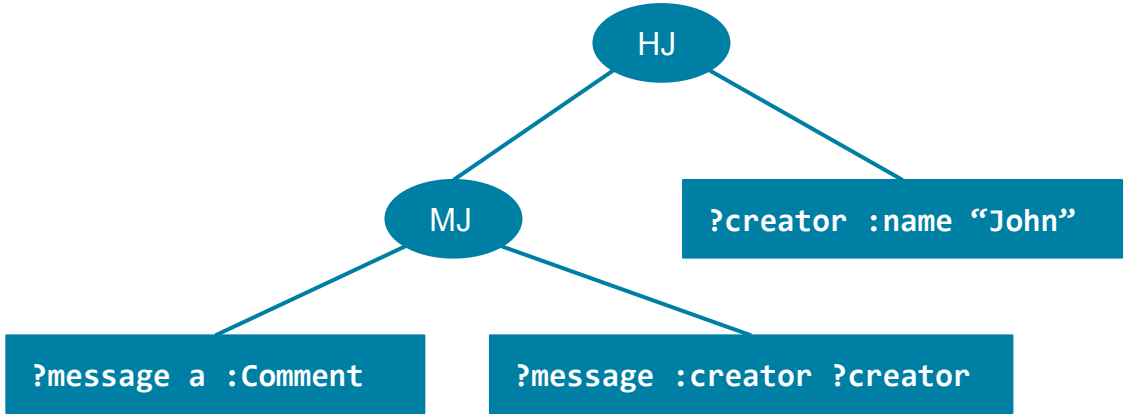
- Each join order (JO) corresponds to an algebraic expression (query plan)
- Each query plan has an associated cost
- The JO optimiser tries to find the plan with the least cost

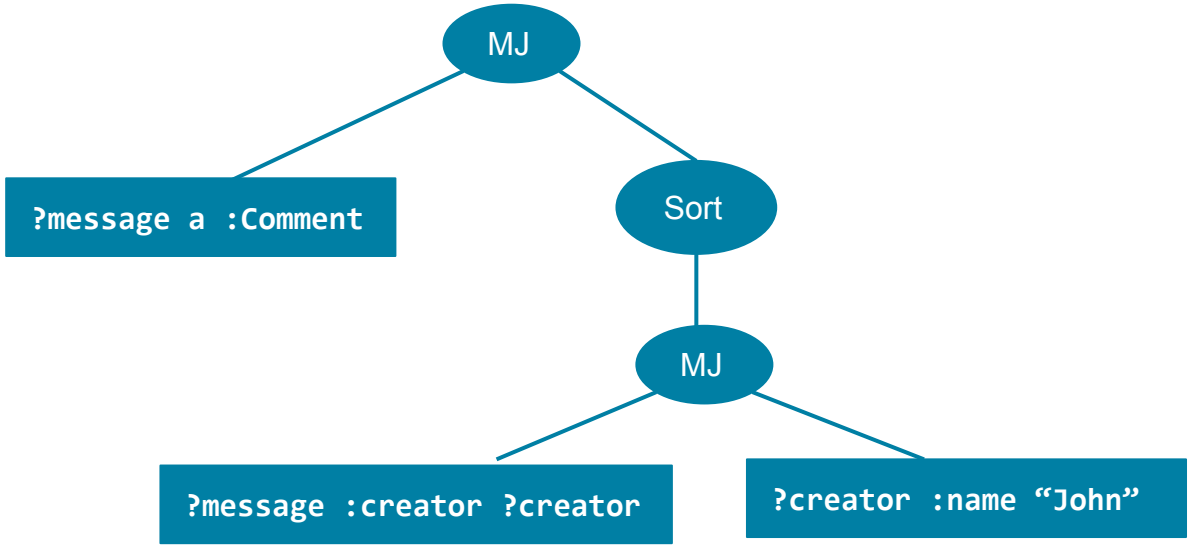




Join order optimization in Stardog

- A bit more complex than this because:
  - need to pick join algorithms too (merge, hash, bind, nested loops, …)
  - choice of join algorithm depends on order of solutions from children
  - huge search space (> factorial)





Query Optimization Challenges

- Complex queries have a lot of joins
  - SPARQL query does not provide any execution hints
  - JO optimization has to deal with a large search space
- Accurate cardinality estimations needed
  - Need to avoid snowball effect for misestimations
  - Deal with renamings - `FILTER(?x = ?y) BIND(123 AS ?id)`
  - Estimations for patterns/chains with and without constants
    - Auto compute characteristic sets for star-shaped graphs
    - Combine it with probabilistic count-min sketches to track frequent nodes
    - Detect functional relationships, collect statistics about 2-hop chains, ...
- Eliminate non-determinism during planning

Some Observations / Suggestions

- Access patterns are very similar in all queries
  - Every query takes person ID as an input parameter
  - Why not look up person by email (multi-valued attribute)?
- Schema flexibility is an important differentiator for graphs
  - Why not have updates that modify graph schema/structure?
- Queries differ very widely based on implementations
  - Is any query change really ok as long as you get the same results?

```
?rootPerson a snvoc:Person .
?rootPerson snvoc:id ?rootId .
?fr a snvoc:Person .
?fr snvoc:id ?frId .
FILTER(?frId != ?rootId) .
```

```
?rootPerson snvoc:id ?rootId .
?fr snvoc:id ?frId .
FILTER(?frId != ?rootId) .
```

STARDOG