通义

GraphScope

GOpt: A Graph-Native Query Optimization Framework

Speaker: Longbin Lai

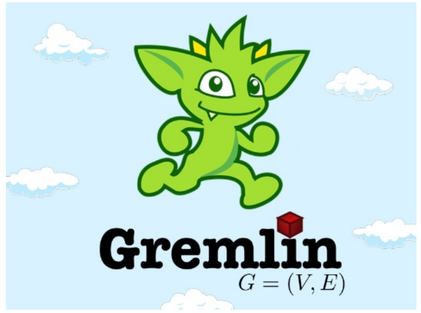
GraphScope • Tongyi Lab • Alibaba Group

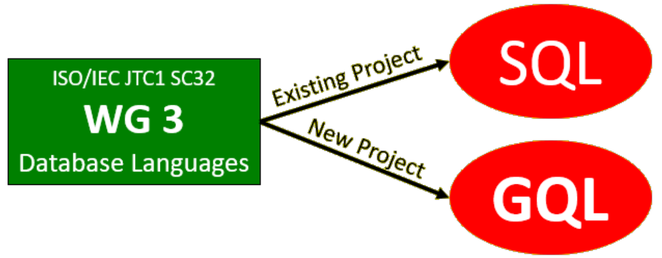
LDBC 18th TUC Meeting

01 / BACKGROUND

Motivation and Solution Overview

Background





SQL/PGQ

Declarative Graph Query Languages

Optimizer is a KEY component in (G)DBMS!

Graph Database

Relational Database

A Relational Optimization Pipeline

Relational Algebra

S(S#, SNAME, CITY, STATUS)

P(P#, PNAME, CITY, WEIGHT, COLOR)

SP(S#, P#, QUANTITY)

Example:

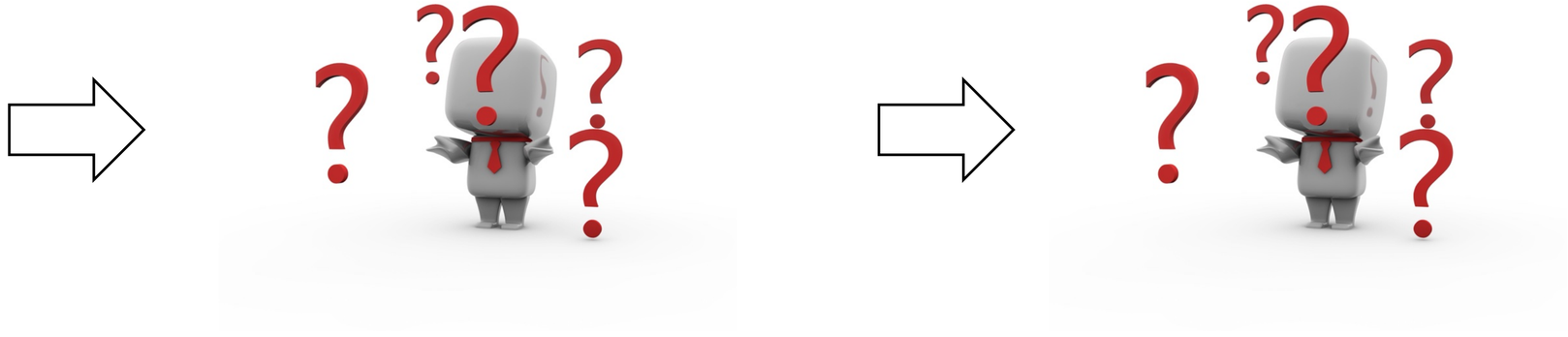
<Supplier names of suppliers who ship 'P3'>

$$\Pi_{SNAME}\left(S \bowtie \left(\Pi_{S\#}\left(\sigma_{P\#=3}\, SP\right)\right)\right)$$

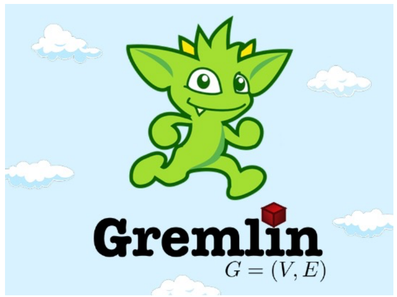
Relational Optimizer

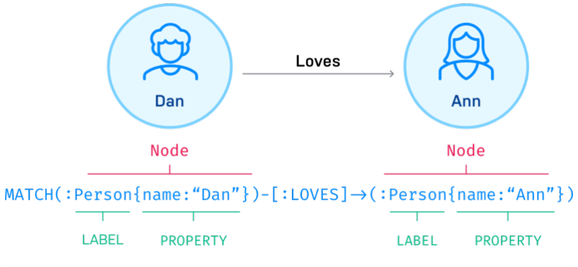
A Graph Optimization Pipeline?

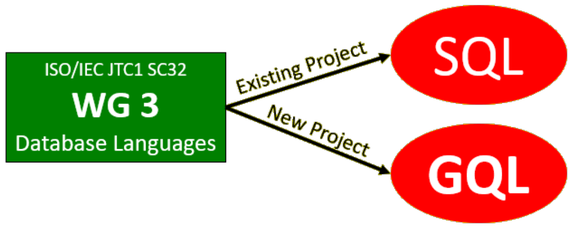
Graph Query



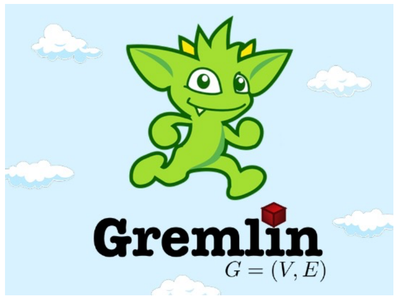
A Variety of Graph Query Languages

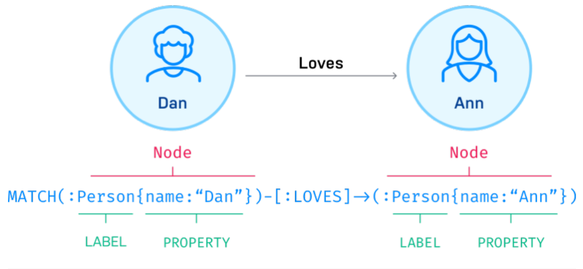


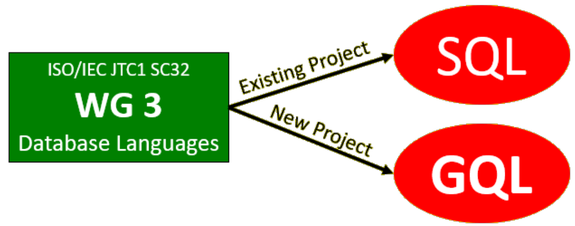




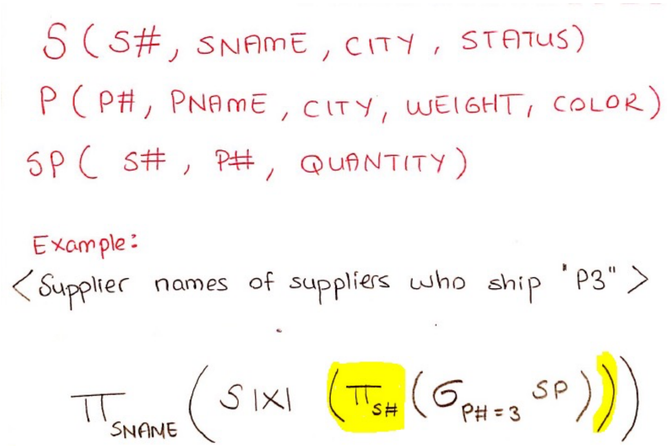
A **Non-graph-native** Pipeline for graph optimization







Relational Algebra

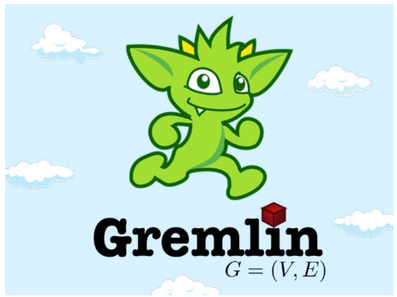


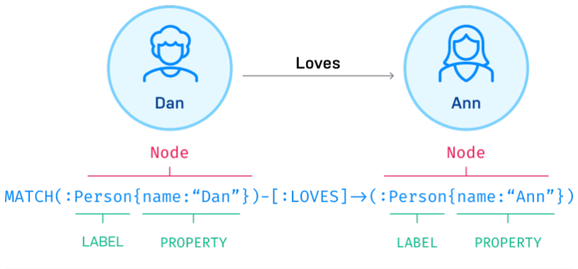
Relational Optimizer

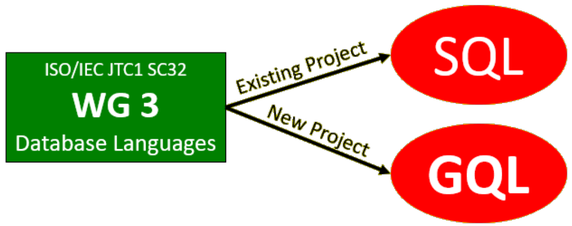
Modern Techniques for Querying Graph-Structured Relations: Foundations, System Implementations, and Open Challenges, Amine Mhedhbi, Semih Salihoğlu, VLDB 2022

Implementing Graph queries in a Relational Database, https://blog.whiteprompt.com/implementing-graph-queries-in-a-relational-database-7842b8075ca8?gi=fb6853dc262e

A **Non-graph-native** Pipeline for graph optimization (Cont.)







Relational Algebra

S (S#, SNAME, CITY, STATUS)
P (P#, PNAME, CITY, WEIGHT, COLOR)
SP (S#, P#, QUANTITY)

Example:
<Supplier names of suppliers who ship 'P3">

$$\Pi_{SNAME}\left(S \bowtie \left(\Pi_{S\#}\left(\sigma_{P\#=3}\, SP\right)\right)\right)$$

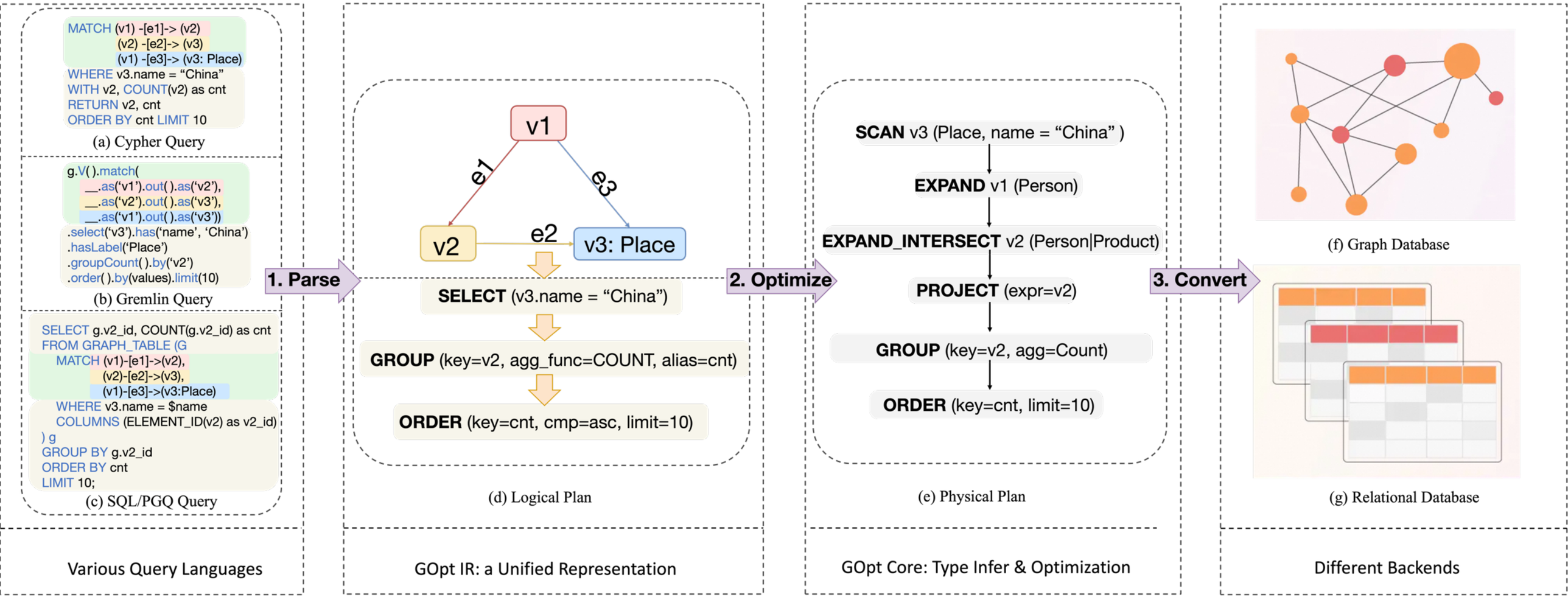
Relational Optimizer

Performance ↓

Bridging the Gap between Relational OLTP and Graph-based OLAP, Sijie Shen et al., ATC 2023 (The later GART talk)

GOpt: A Unified **Graph-native** Optimization Framework

A Graph-Native Query Optimization Framework, Lyu et al., arXiv:2401.17786

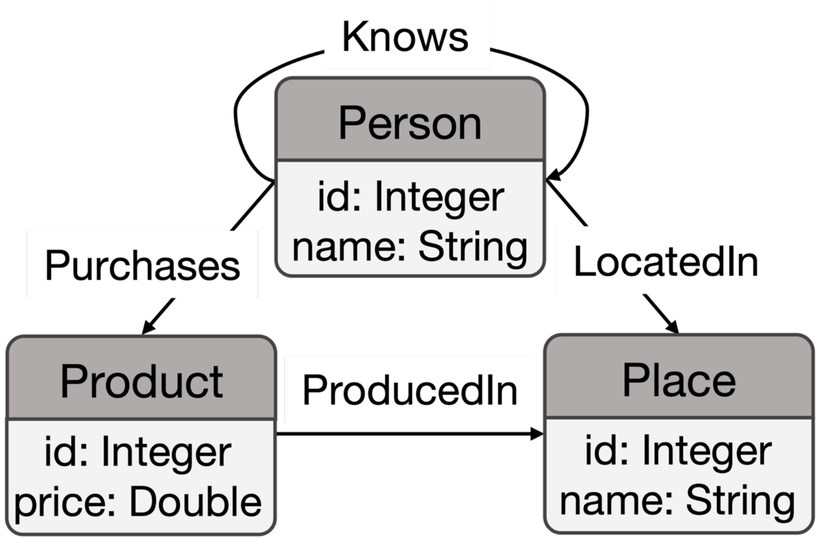


02 /
GOPT OVERVIEW

Key techniques in GOpt:

- Graph Intermediate Representation
- Type Inference
- RBO & CBO

A Simple Query Example



Schema

MATCH a triangle pattern containing a Place vertex, and then **FILTER** with a specified place name, **GROUP** the matchings by the name and the **TOP** 10 results are returned?

```
MATCH (v1) -[e1]->(v2),
      (v2) -[e2]->(v3),
      (v1) -[e3]->(v3: Place)
WHERE v3.name=$name
WITH v2, COUNT(v2) as cnt
RETURN v2
ORDER BY cnt LIMIT 10
```

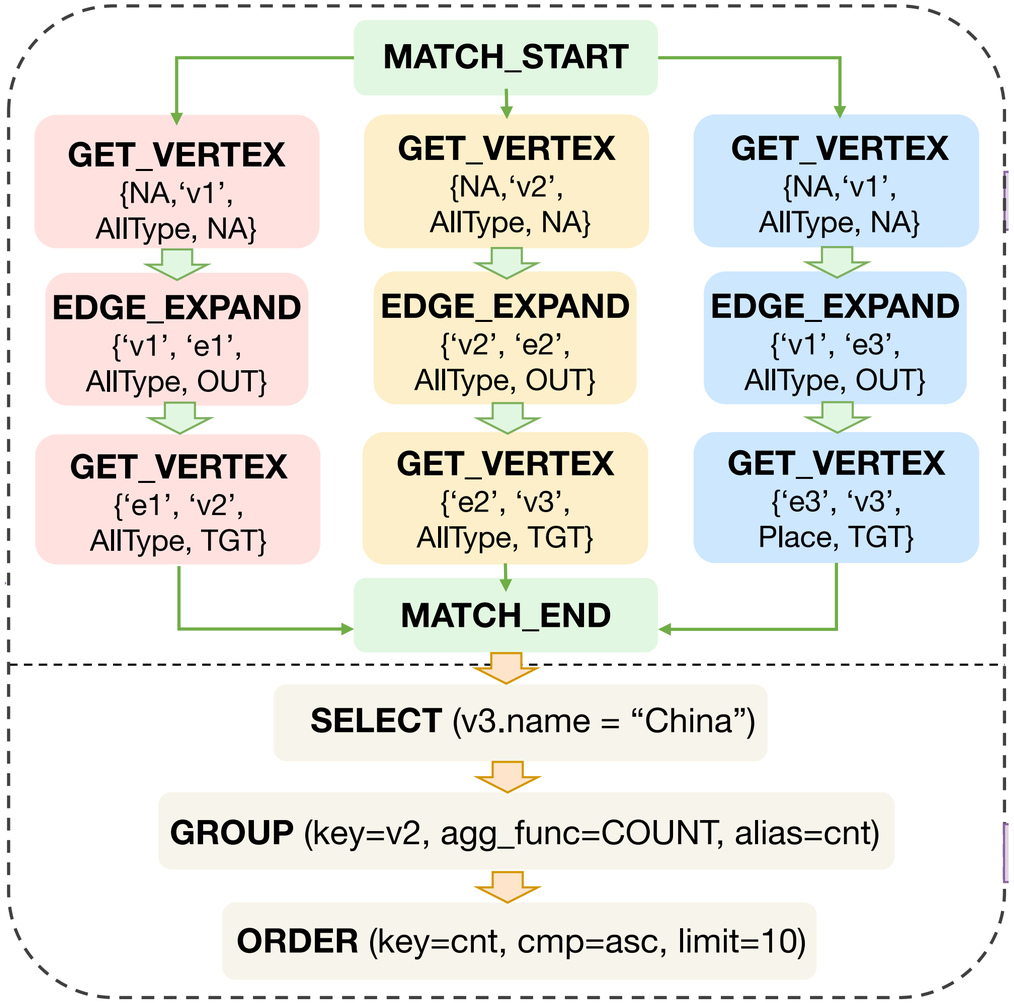
User Query

The Graph Intermediate Representation

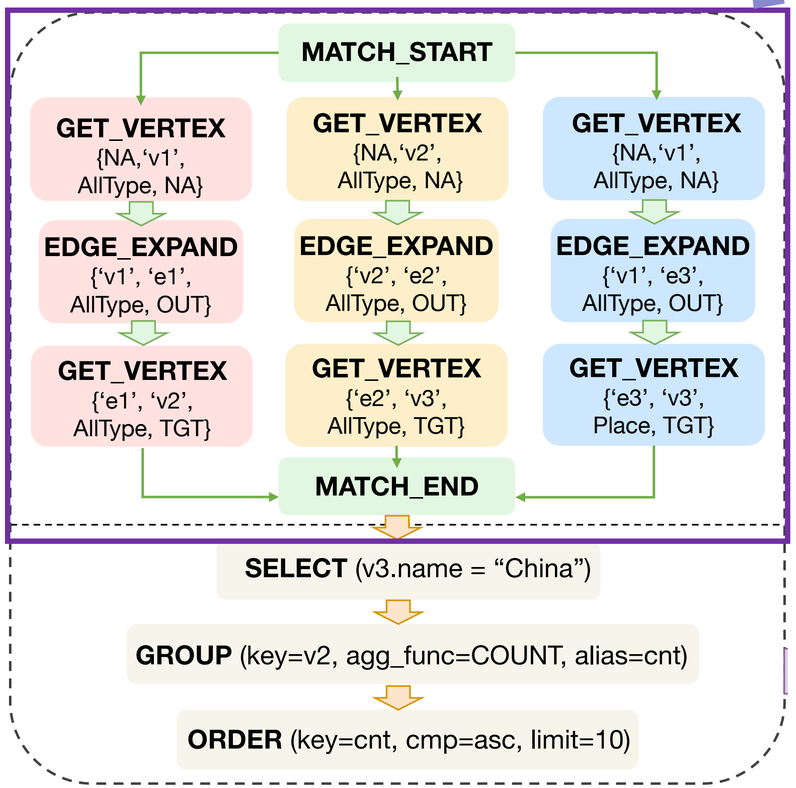
```
MATCH (v1) -[e1]->(v2),
      (v2) -[e2]->(v3),
      (v1) -[e3]->(v3: Place)
WHERE v3.name=$name
WITH v2, COUNT(v2) as cnt
RETURN v2
ORDER BY cnt LIMIT 10
```

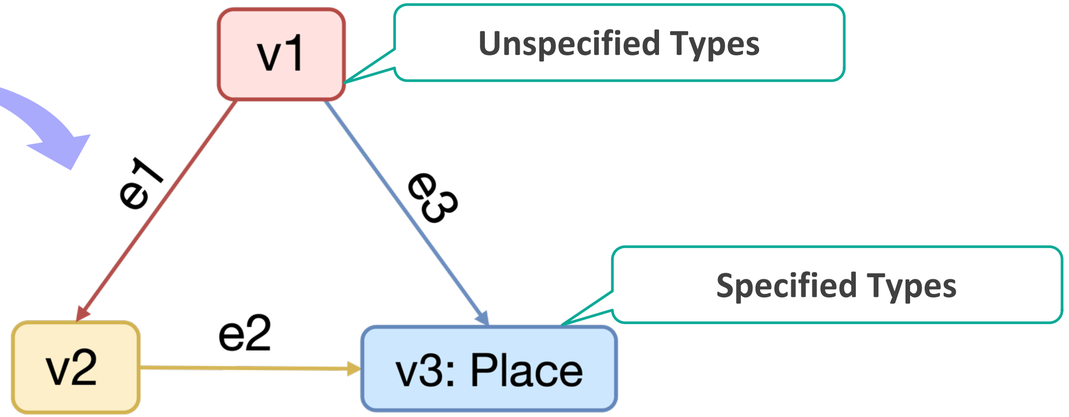
Cypher Query

1. Compile

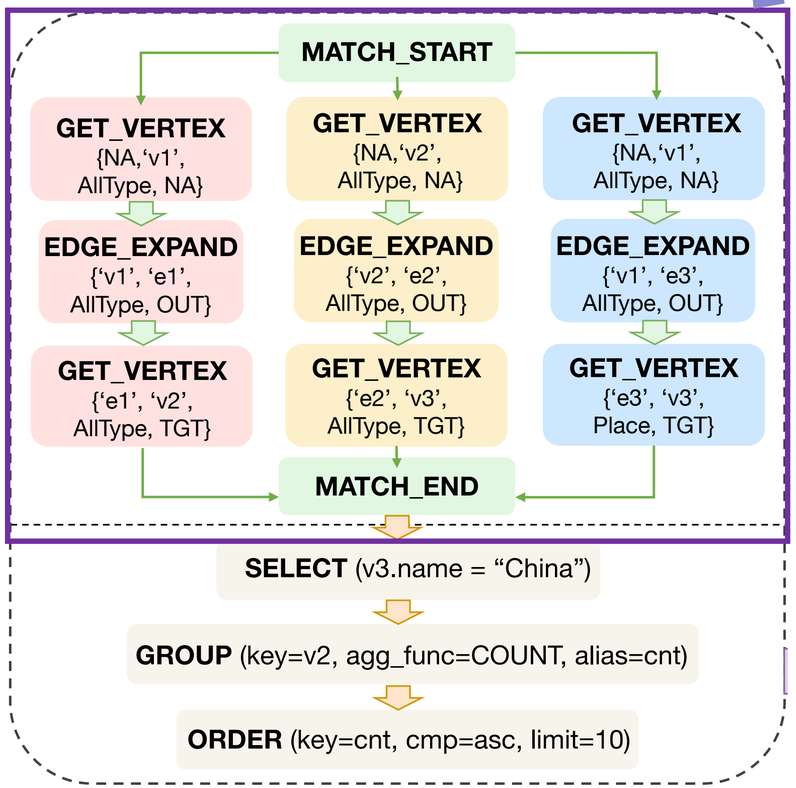


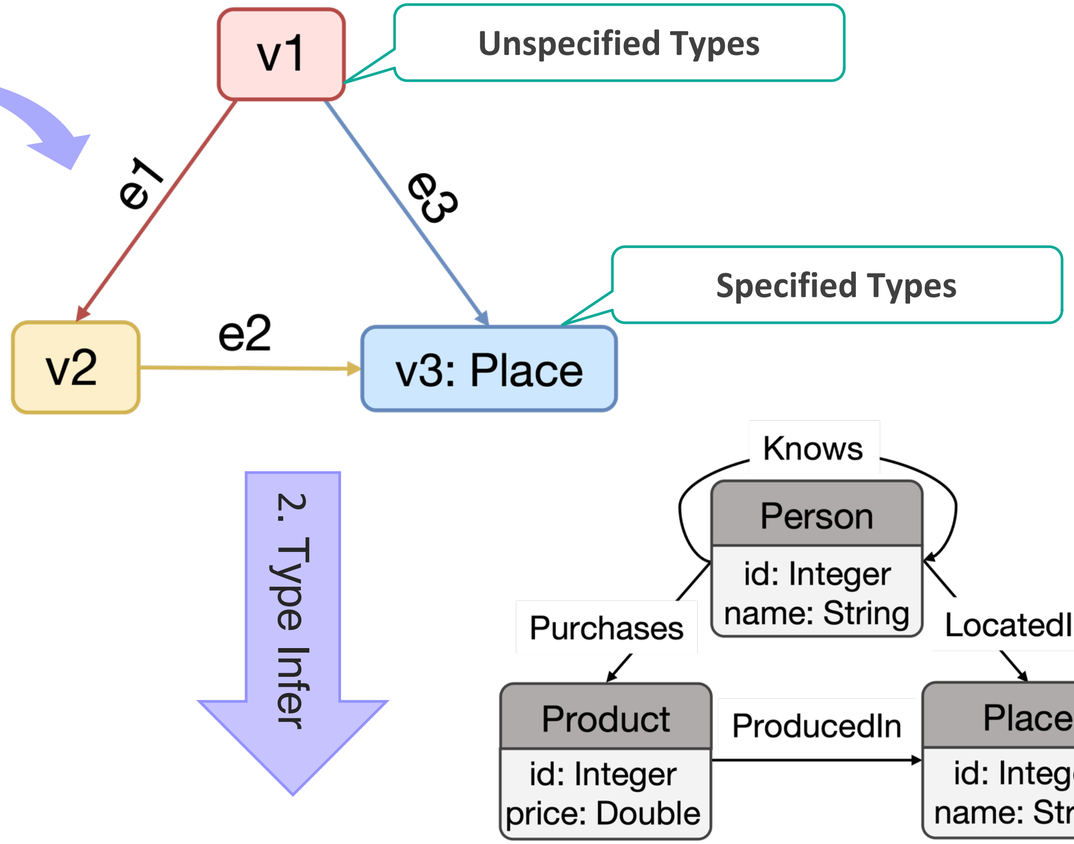
Type Inference



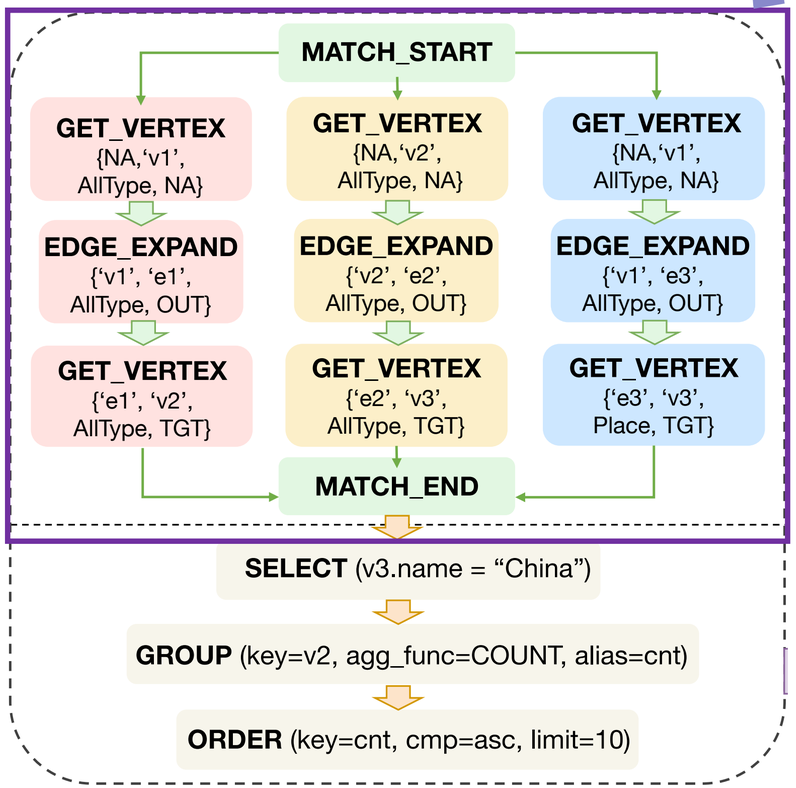


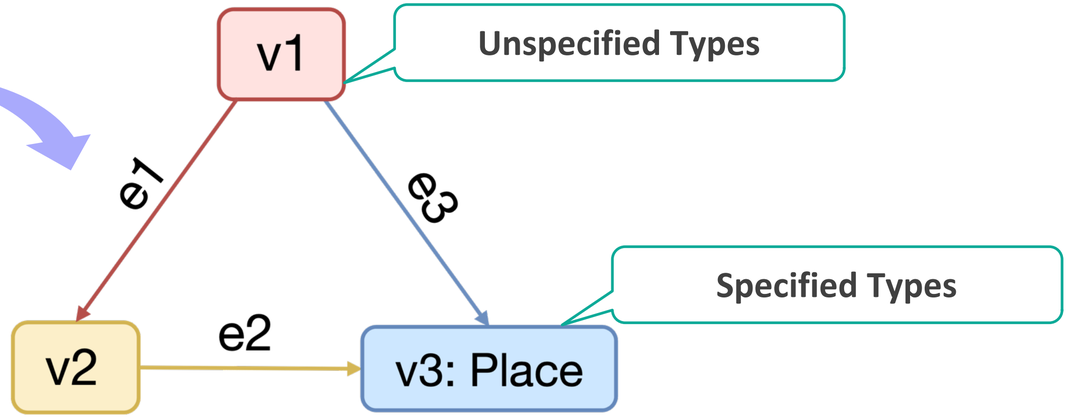
Type Inference (Cont.)



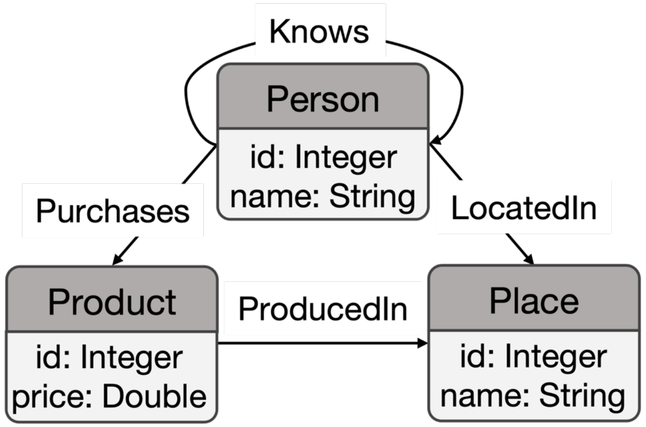


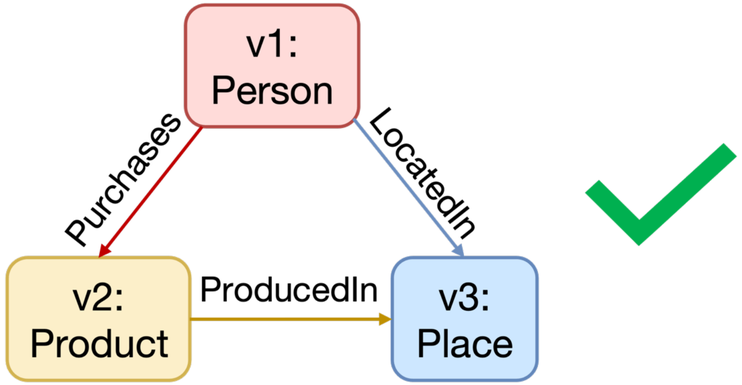
Type Inference (Cont.)



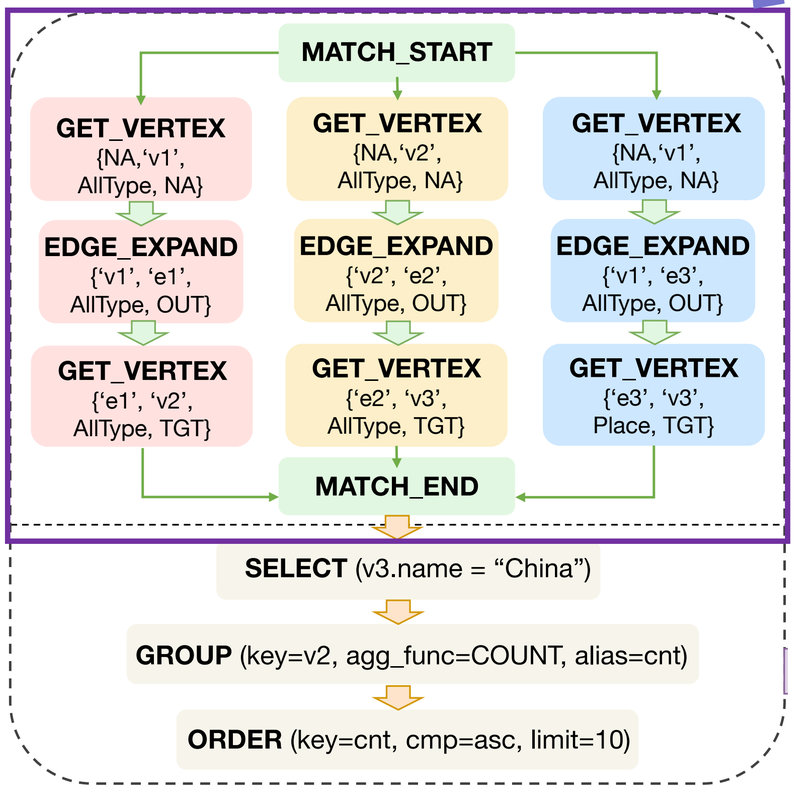


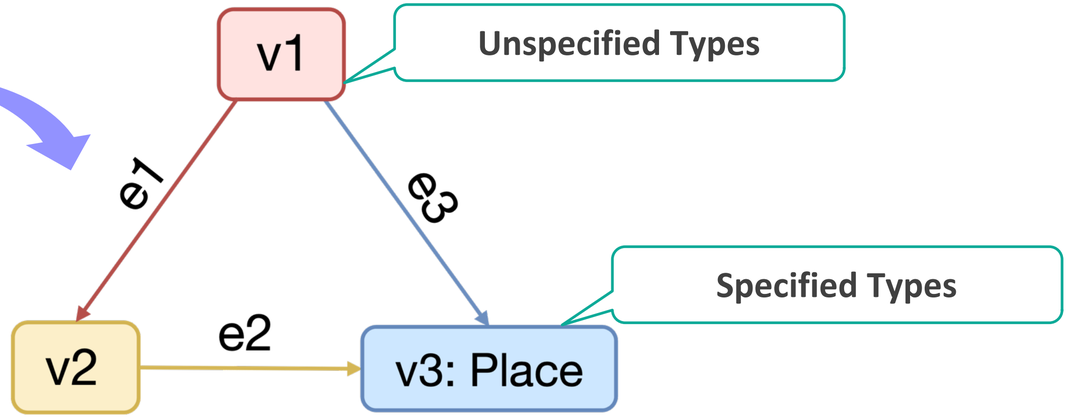
2. Type Infer



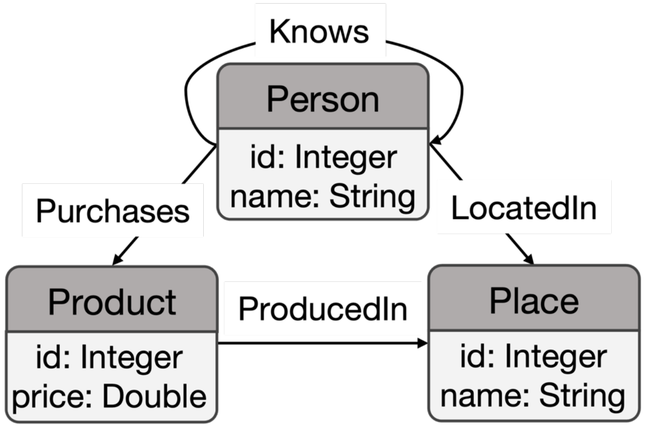


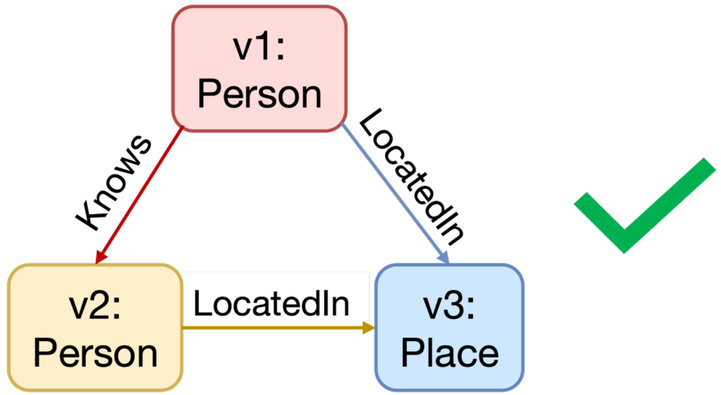
Type Inference (Cont.)



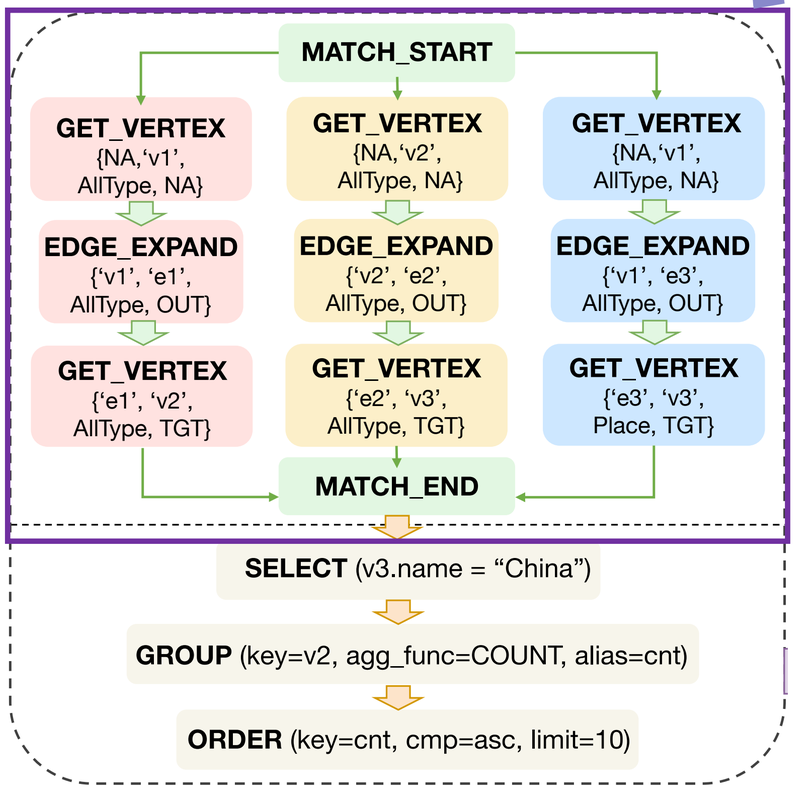


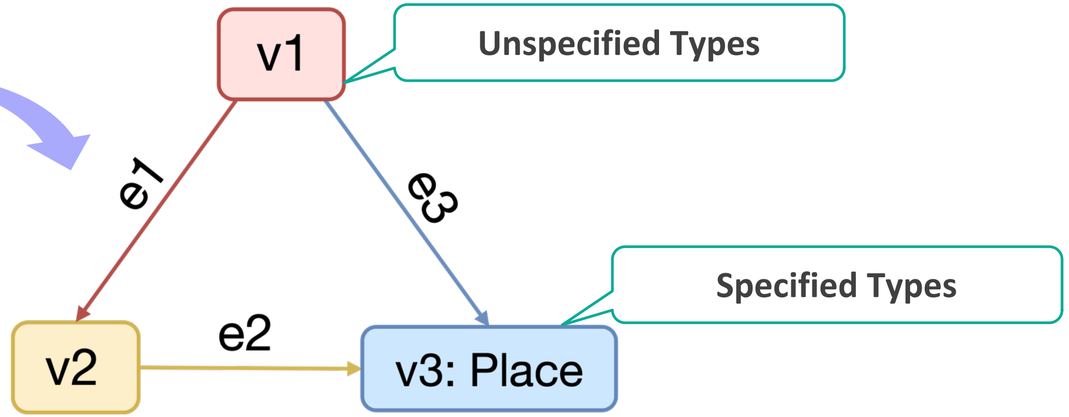
2. Type Infer



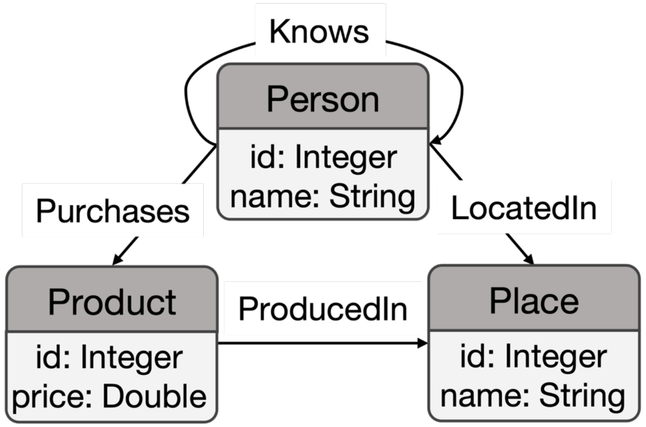


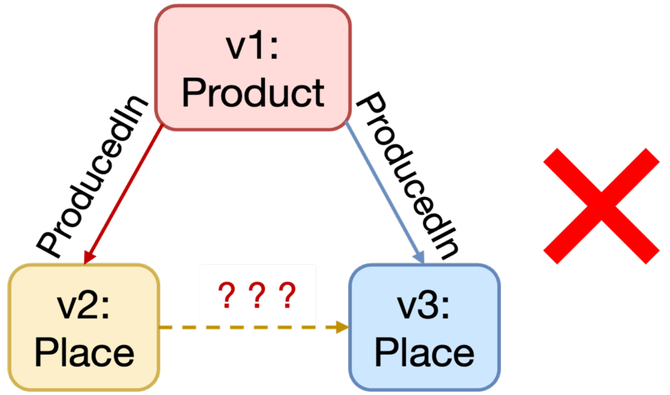
Type Inference (Cont.)



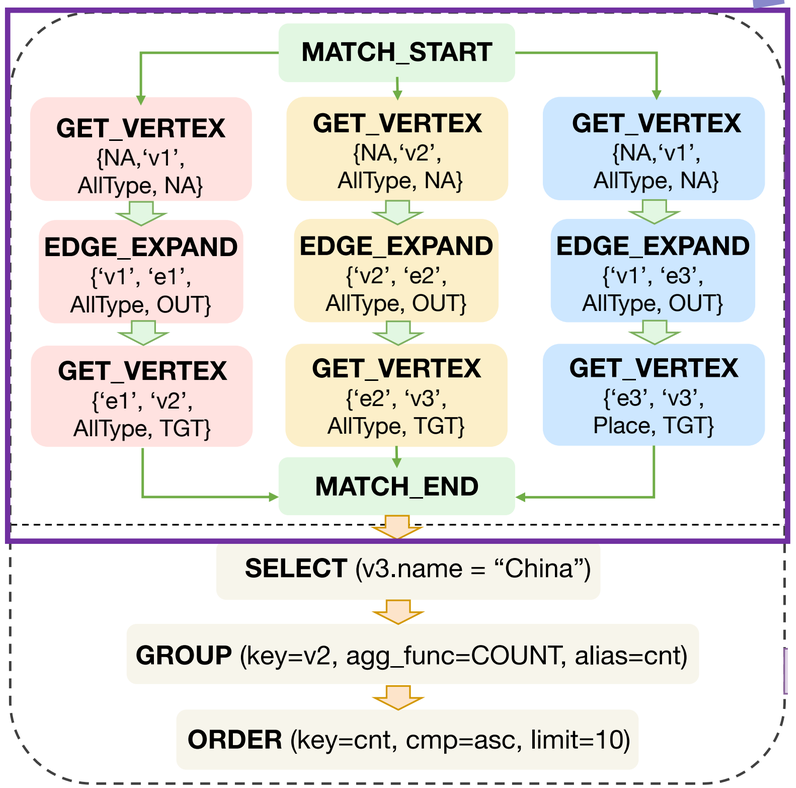


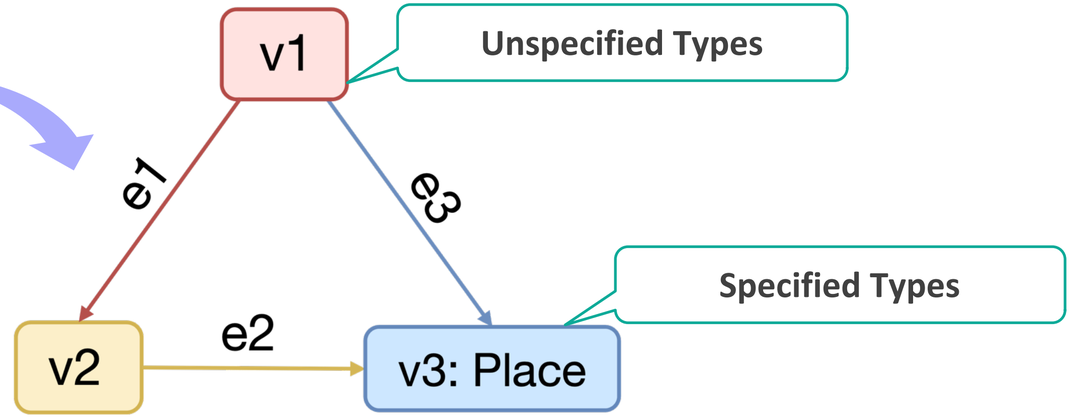
2. Type Infer



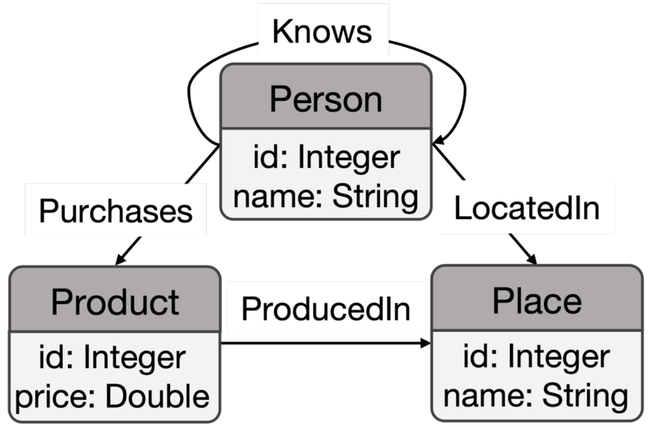


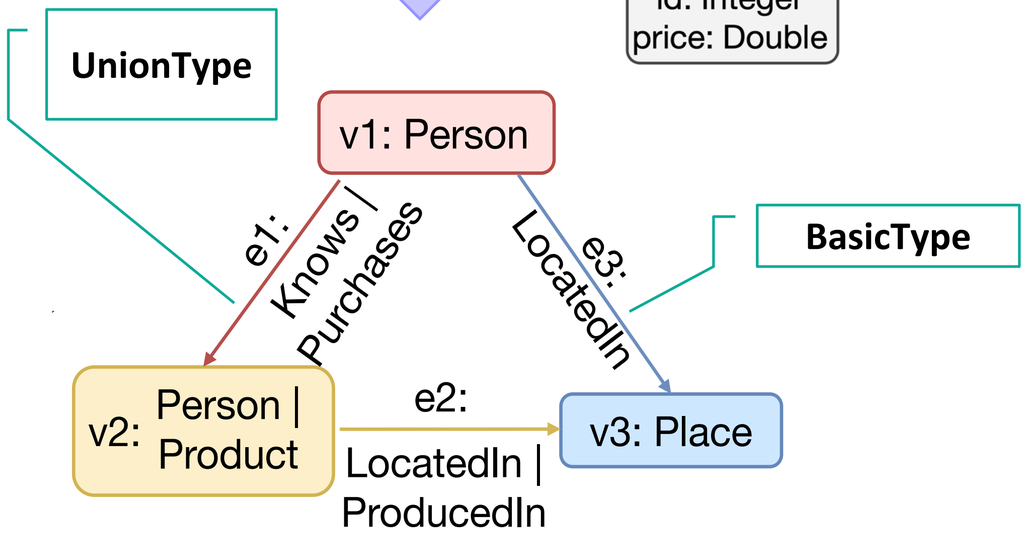
Type Inference (Cont.)



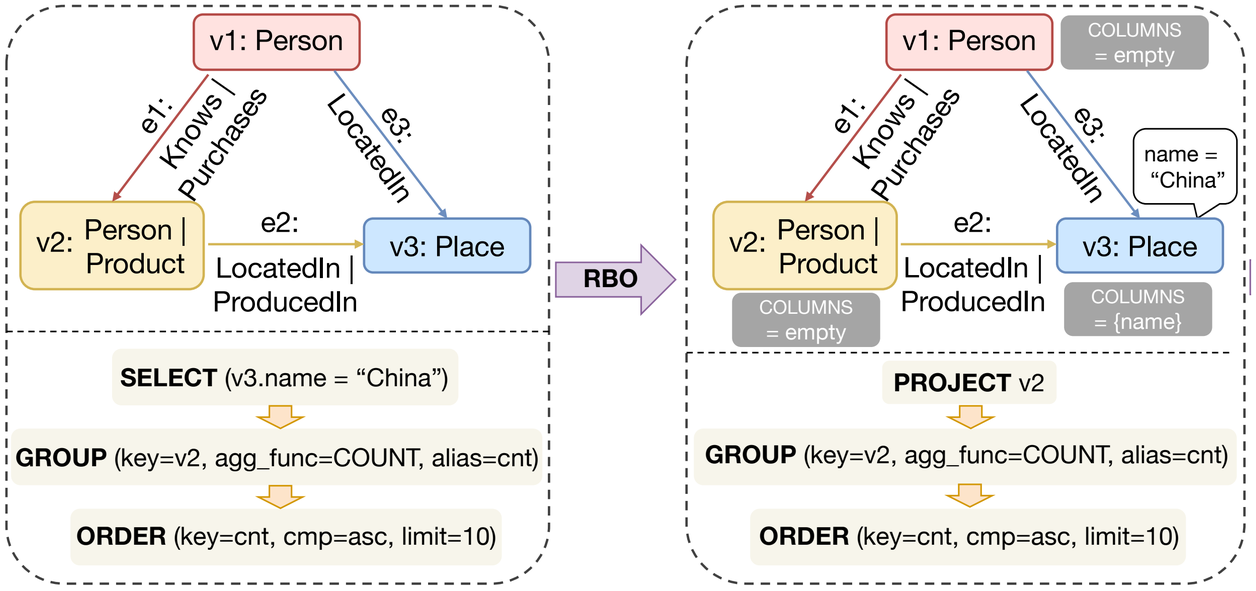


2. Type Infer





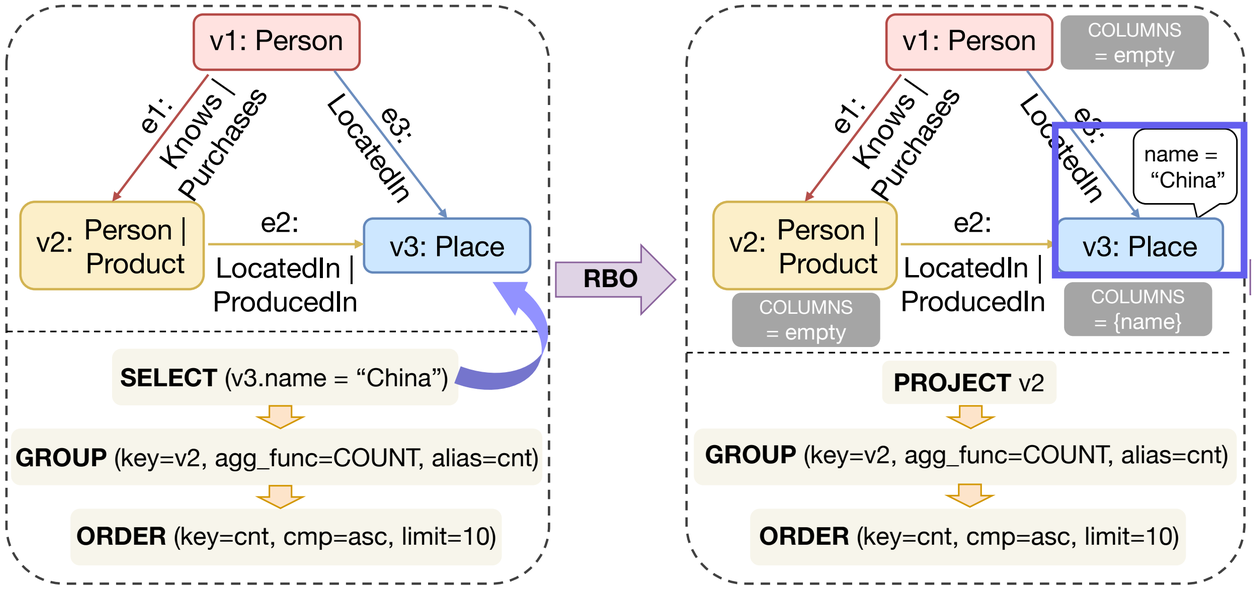
Rule-based Optimization (RBO)



(a) Logical Plan

(b) Optimized Logical Plan after RBO

Rule-based Optimization (RBO)



(a) Logical Plan

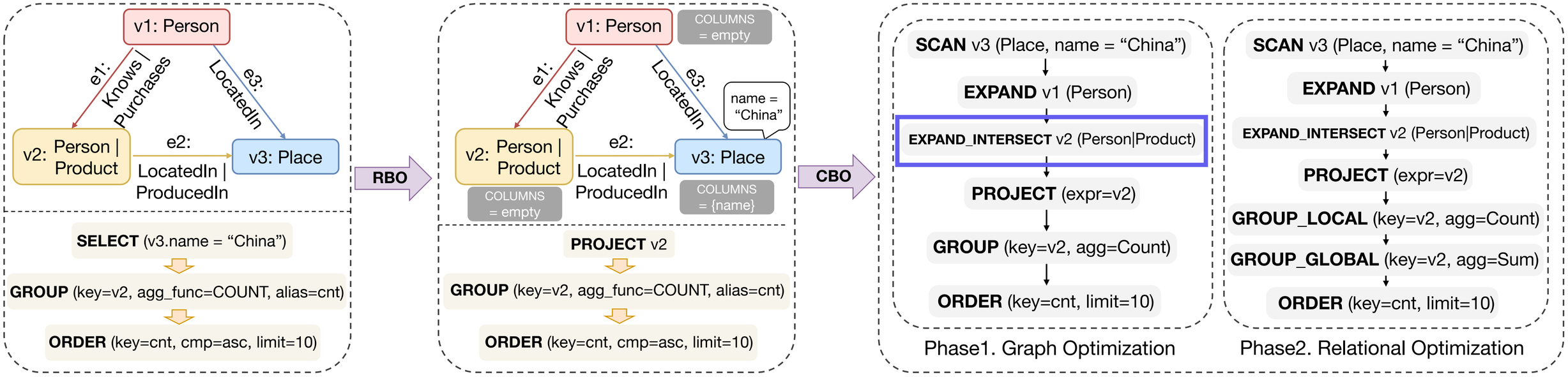
(b) Optimized Logical Plan after RBO

Cost-based Optimization (CBO)

High-order statistics: Frequencies of small patterns like 2-hop paths & triangles

Worst-case optimal join: Expand & Intersection

GLogS: Interactive Graph Pattern Matching Query At Large Scale, Lai et al, ATC 2023

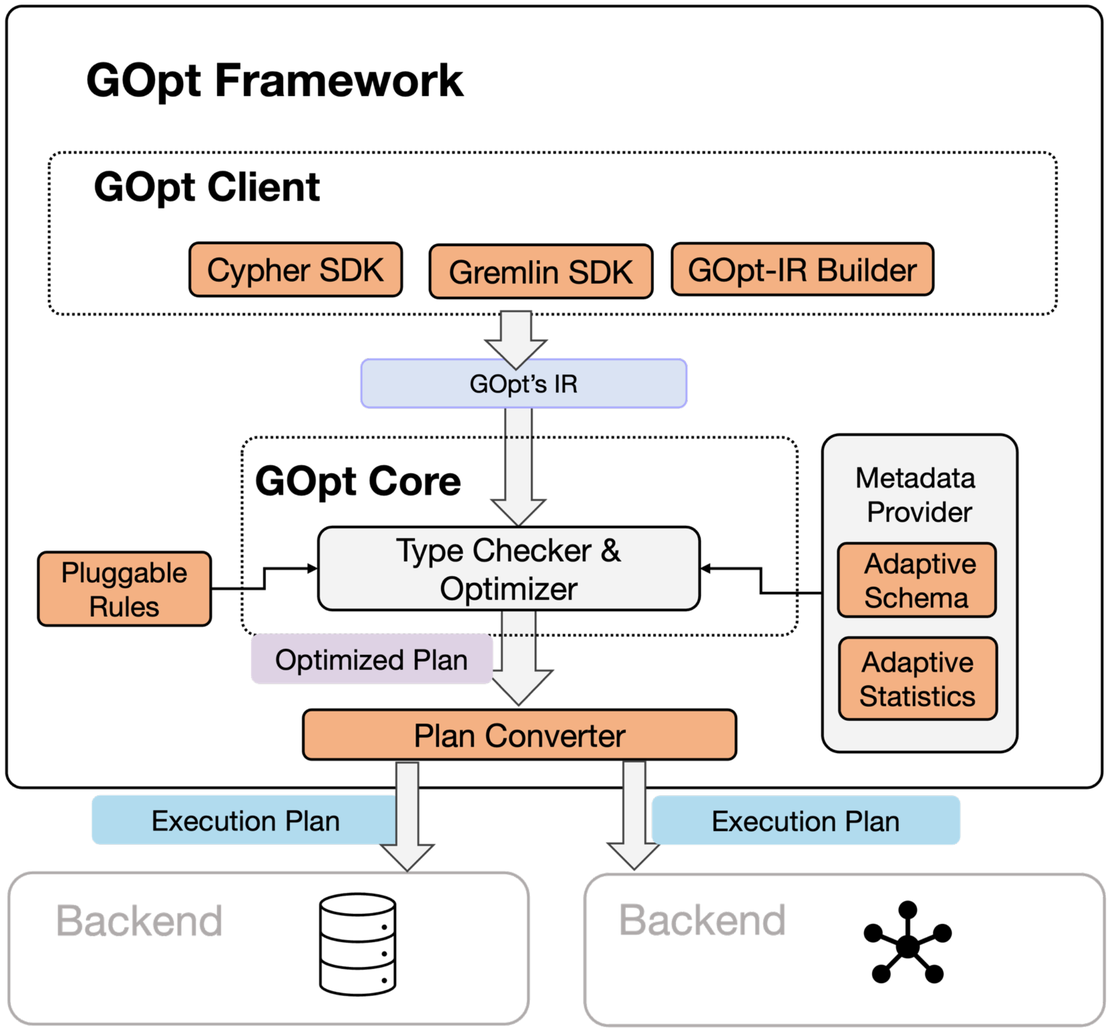


(a) Logical Plan (b) Optimized Logical Plan after RBO (c) Optimized Physical Plan after CBO

System Architecture of GOpt

User-friendly interfaces in GOpt

- Input: various clients to the unified IR
- Output: convert to alternative execution plans
- Metadata: adaptive schema and statistics
- Optimizer: pluggable optimization rules

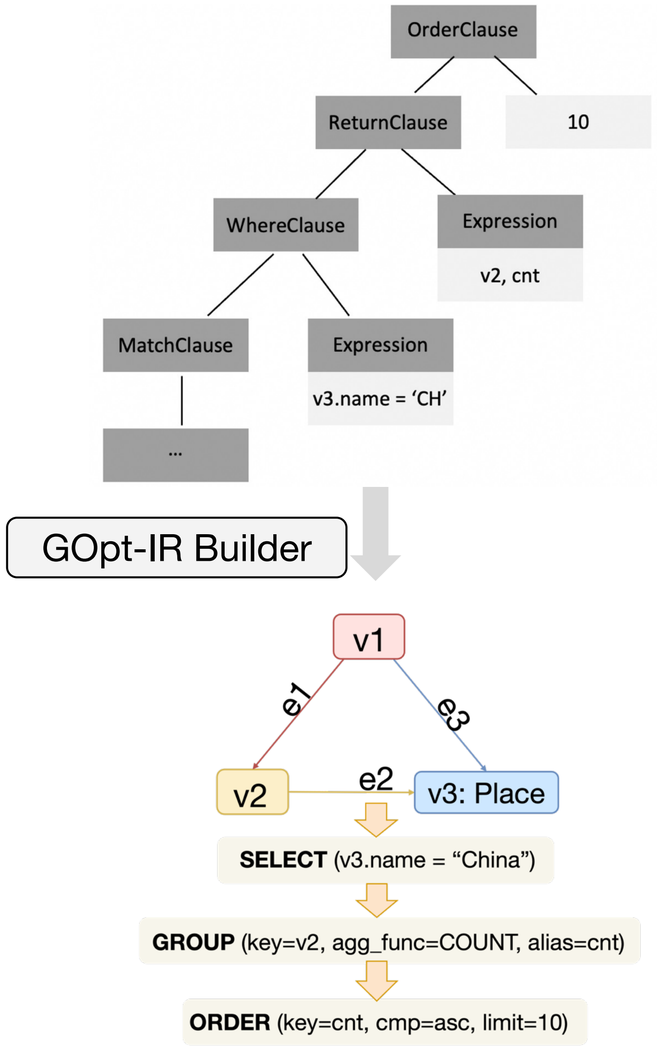


03 /
GOPT INTEGRATION

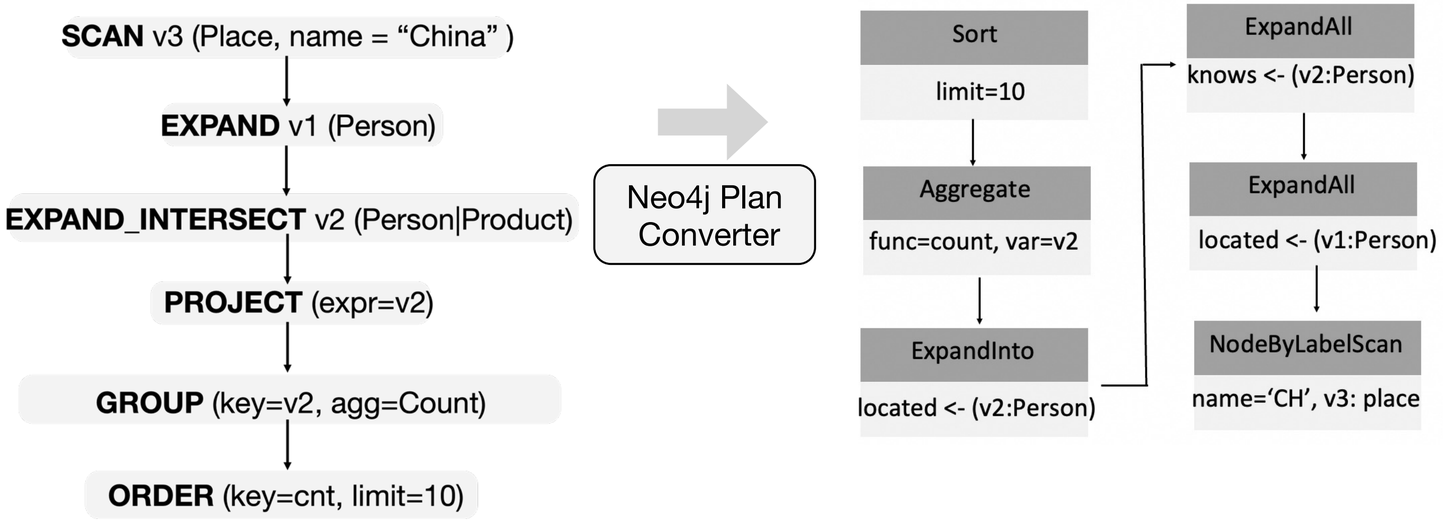
Neo4j - Cypher
DuckDB – SQL/PGQ

GOpt on Neo4j: Convert Input and Output

- Input
  - Neo4j AST-> GOpt-IR

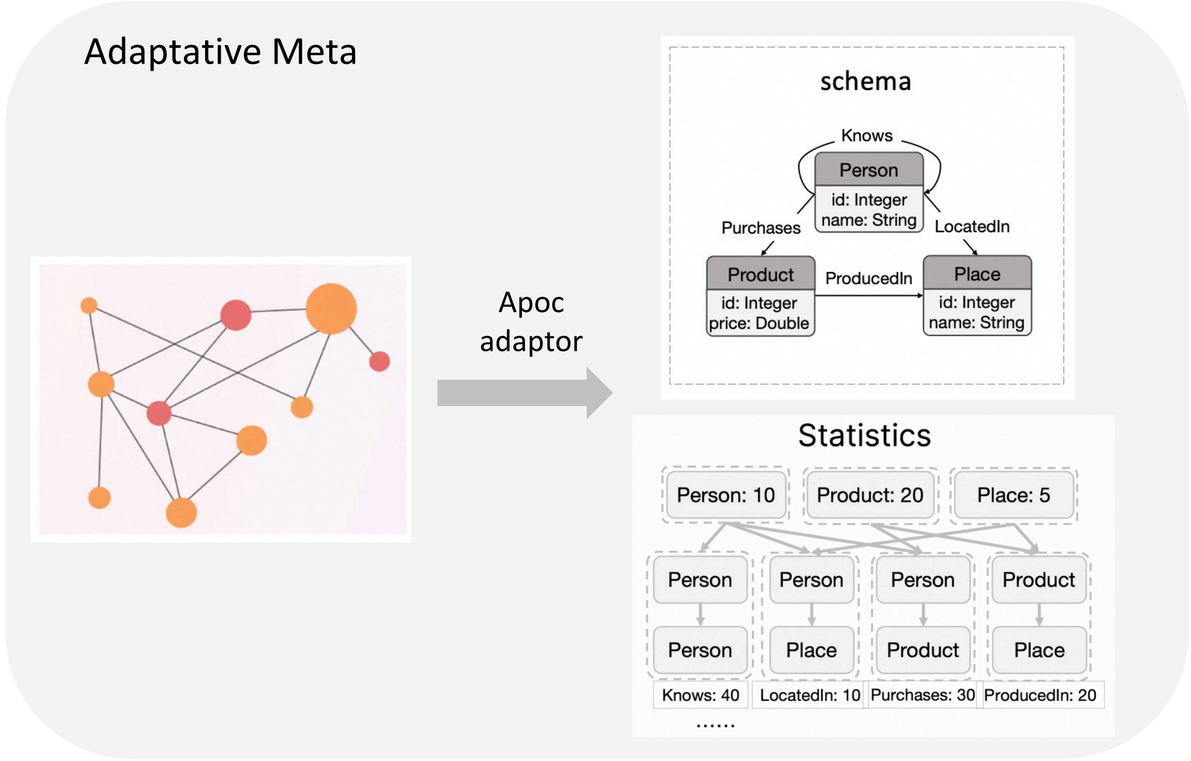


- Output
  - GOpt Physical Plan -> Neo4j Execution Plan
    - WCOJ is unsupported in Neo4j natively
    - Convert ExtendIntersect to ExpandAll + ExpandInto
    - GOpt optimized plan can reflect Expand orders in Neo4j

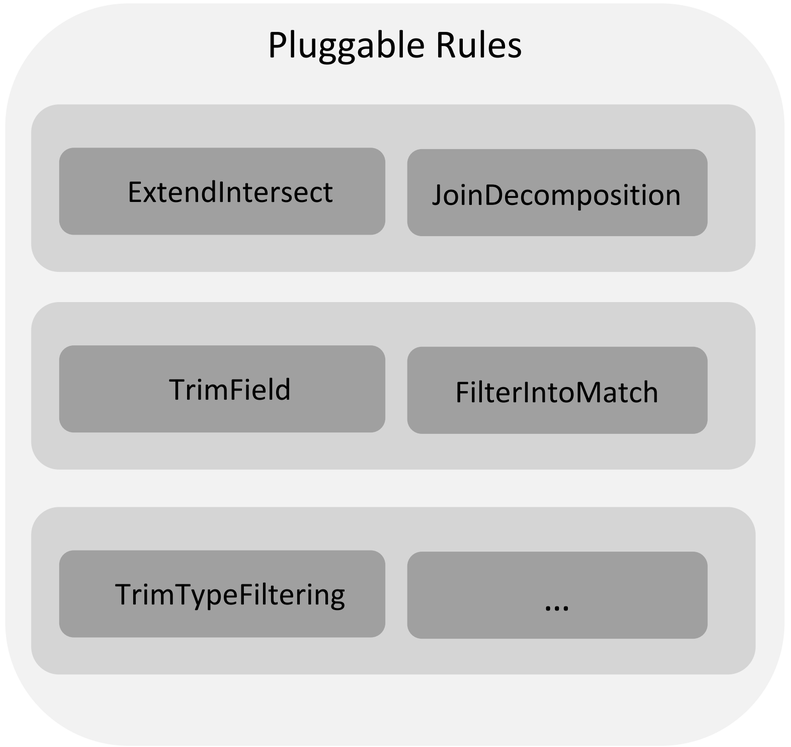


GOpt on Neo4j: Register Configuration

- Register Metadata
  - Sample neo4j graph data by APOC procedure
  - Adapt data model & statistics



- Register Optimize Rules
  - Leverage Build-in Heuristic Rules
  - Design TrimTypeFiltering Rule for Neo4j to remove unnecessary type filtering

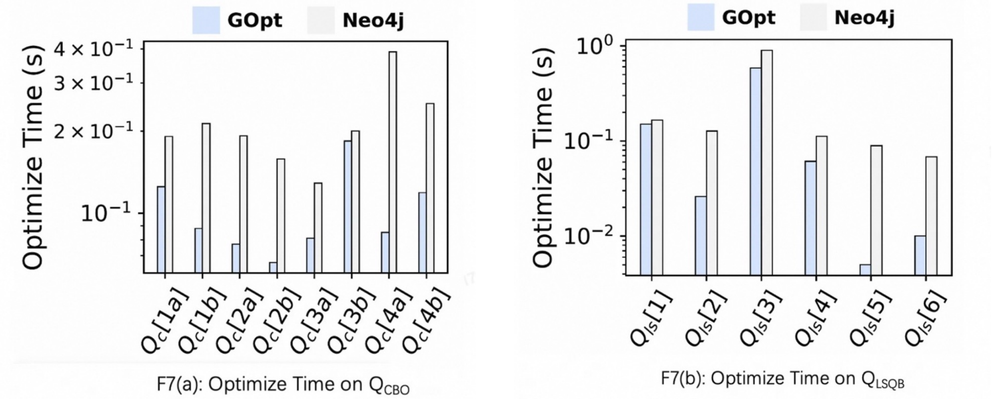


GOpt on Neo4j: Evaluation

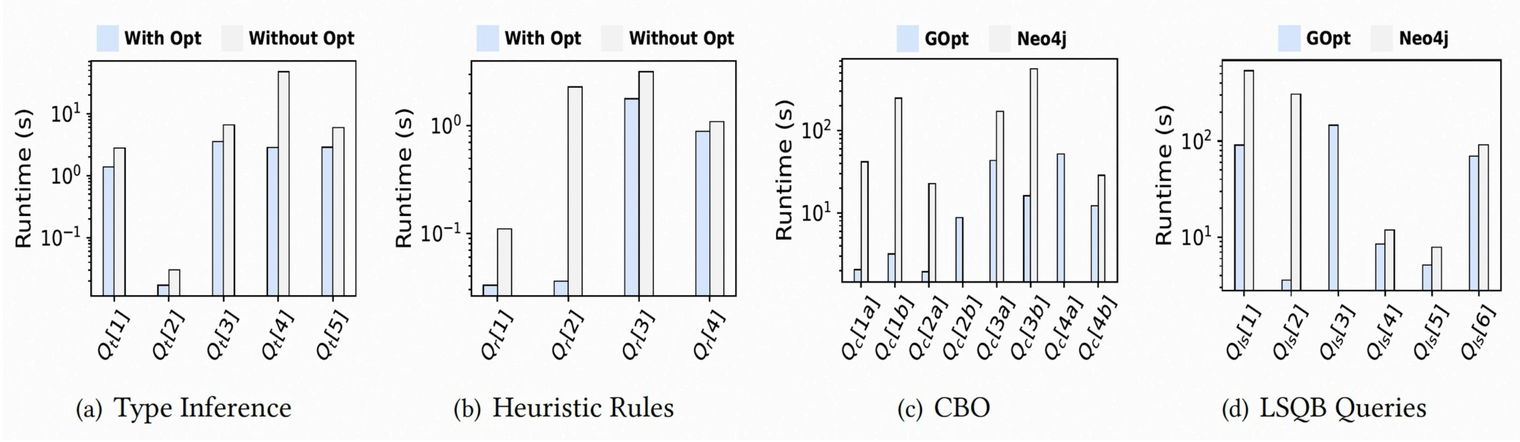
- Dataset:
  - LDBC Graph with 3 million vertices and 17 million edges
- Queries
  - Designed query sets for key techniques in GOpt and pattern matching queries LSQB

Enhancing Neo4j Query Efficiency with Seamless Integration of the GOpt Optimization Framework, Lyu et al, LSGDA Workshop 2024

Optimize time:



Execution time:

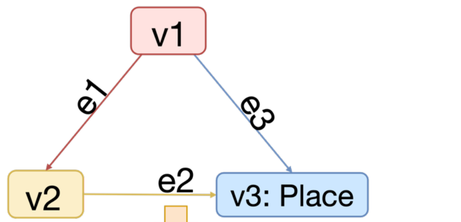


GOpt on DuckDB: Convert Input and Output

- Input
  - SQL/PGQ Query -> GOpt-IR

```
SELECT g.v2_id, COUNT(g.v2_id) as cnt
FROM GRAPH_TABLE (G
   MATCH (v1)-[e1]->(v2),
         (v2)-[e2]->(v3),
         (v1)-[e3]->(v3:Place)
   WHERE v3.name = $name
   COLUMNS (ELEMENT_ID(v2) as v2_id)
) g
GROUP BY g.v2_id
ORDER BY cnt
LIMIT 10;
```

DuckDB Parser + GOpt-IR Builder



SELECT (v3.name = “China”)

GROUP (key=v2, agg_func=COUNT, alias=cnt)

ORDER (key=cnt, cmp=asc, limit=10)

- Output
  - GOpt Physical Plan -> DuckDB Execution Plan

SCAN v3 (Place, name=“China”)

EXPAND v1(Person)

EXPAND_INTERSECT v2 (Person)

PROJECT (expr=v2)

DuckDB Plan Converter

SCAN v1 (Person)

PhysicalMergeSIPJoin ← **SCAN** v3 (Place, name=“China”)

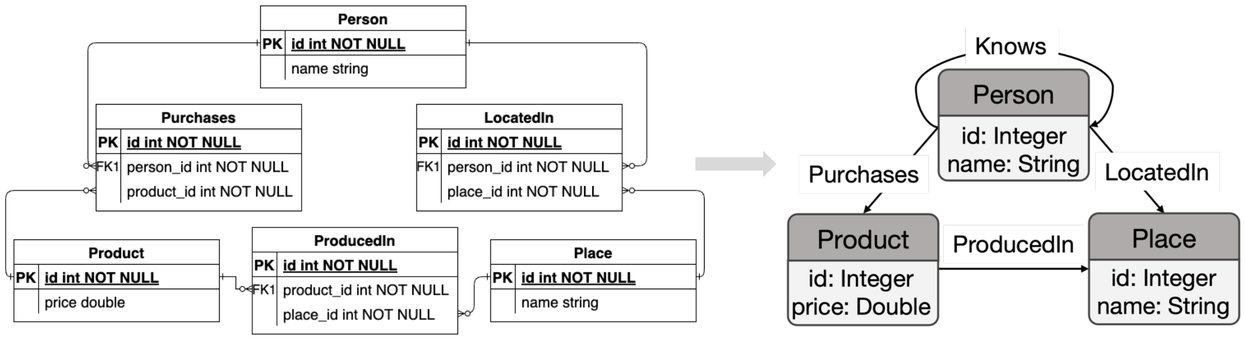
PhysicalExtendIntersect ← **SCAN** v2 (Product)

PhysicalProject v2.id

GOpt on DuckDB: Register Configuration

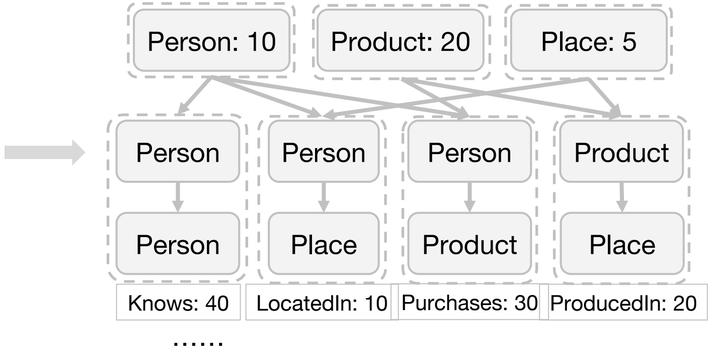
Register Metadata

- Relational Schema -> Graph Schema



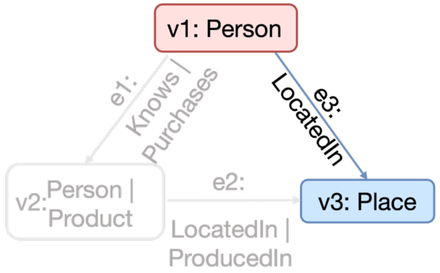
- Relational Tables -> Graph Statistics

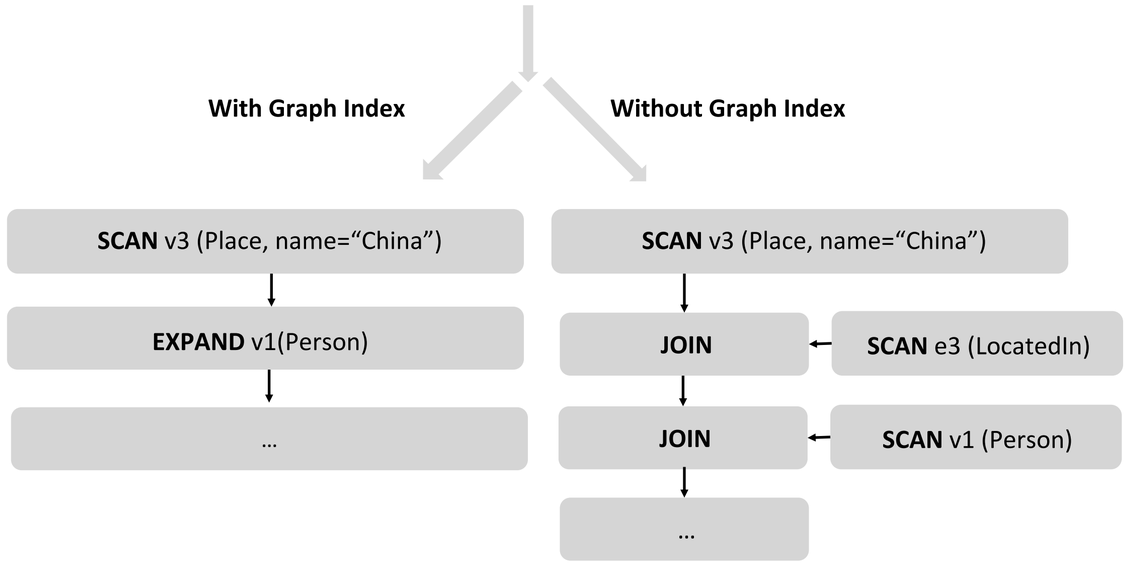
Relations	Row Number
Person	10
Product	20
Place	5
Knows	40
Purchases	30
LocatedIn	10
ProducedIn	20



Register Optimize Rules

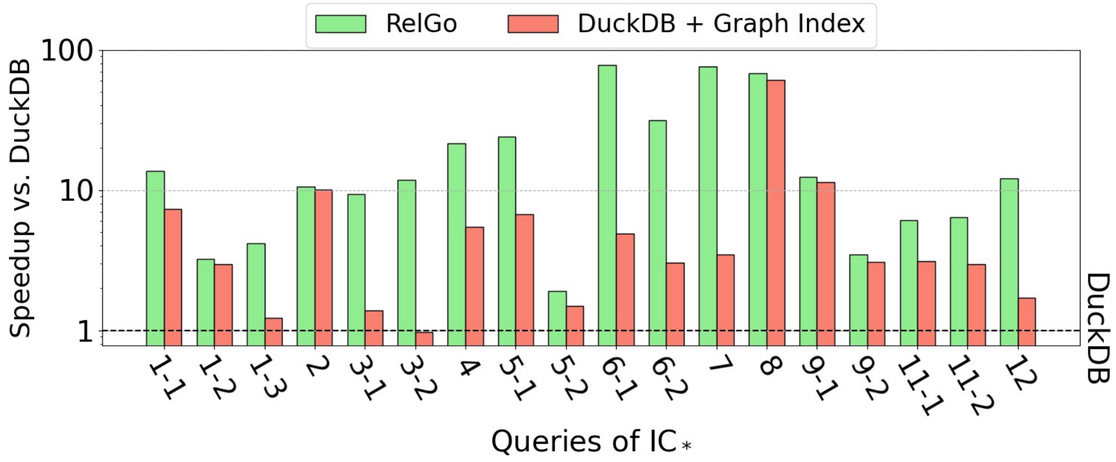
- Design GraphIndexAwareRule for DuckDB

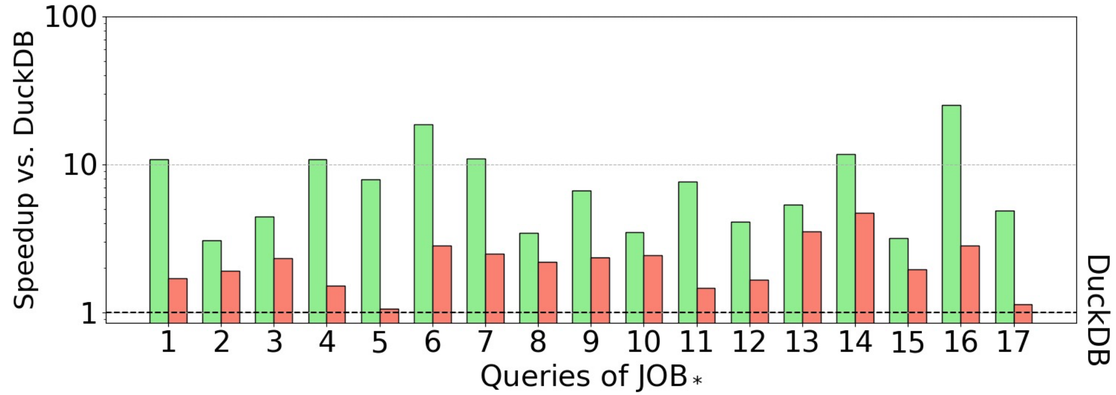


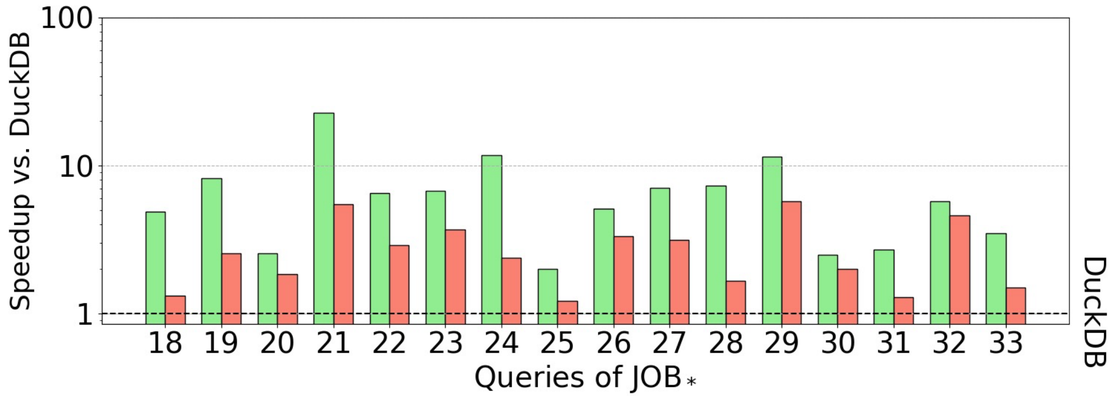


GOpt on DuckDB: Evaluation

- Experimental Settings
  - LDBC Graph with 282 million vertices and 938 million edges
  - IMDB Graph with 50 million vertices and 162 million edges
- Experimental Results
  - IC Queries on LDBC
    - GOpt on DuckDB is about **21.9×**, **5.4×** faster than DuckDB, DuckDB + Graph Index (GrainDB)
  - JOB Queries on IMDB
    - GOpt on DuckDB is about **8.2×**, **4.0×** faster than DuckDB, DuckDB + Graph Index (GrainDB)







Yunkai Lou, Longbin Lai, Bingqing Lyu, Yufan Yang, XiaoLi Zhou, Wenyuan Yu, Ying Zhang, and Jingren Zhou. 2024. Towards a Converged Relational-Graph Optimization Framework. Accepted by SIGMOD 2025.

Acknowledgements

GraphScope

Bingqing Lyu

Xiaoli Zhou

Yunkai Lou

Yufan Yang

References

Modern Techniques for Querying Graph-Structured Relations: Foundations, System Implementations, and Open Challenges, Amine Mhedhbi, Semih Salihoğlu, VLDB 2022

A Graph-Native Query Optimization Framework, Lyu et al., arXiv:2401.17786

GLogS: Interactive Graph Pattern Matching Query At Large Scale, Lai et al, ATC 2023

Towards a Converged Relational-Graph Optimization Framework, Yunkai Lou, Longbin Lai, Bingqing Lyu, Yufan Yang, XiaoLi Zhou, Wenyuan Yu, Ying Zhang, and Jingren Zhou. Accepted by SIGMOD 2025.

Bridging the Gap between Relational OLTP and Graph-based OLAP, Sijie Shen et al., ATC 2023

THANKS

Q&A

Try GOpt

Contact Us