31st International Symposium on Software Testing and Analysis

Finding Bugs in Gremlin-Based Graph Database Systems via Randomized Differential Testing

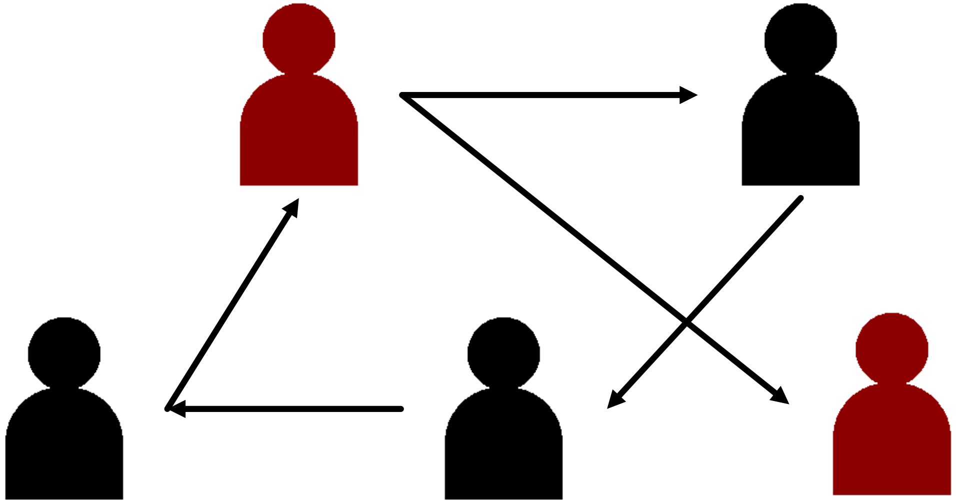
Yingying Zheng, Wensheng Dou, Yicheng Wang, Zheng Qin,
Lei Tang, Yu Gao, Dong Wang, Wei Wang, Jun Wei

Institute of Software, Chinese Academy of Sciences

University of Chinese Academy of Sciences

Graph Data

- Graph data consists of vertices and edges
  - A vertex represents an entity
  - An edge describes the relationship between two entities



Graph Database Systems (GDBs)

□ **GDBs support efficient storage and queries for graph data**

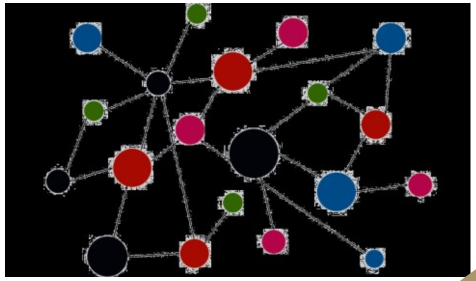
neo4j

Neo4j has been downloaded **2 million+** times[1].

[1] Neo4j. Retrieved May 23, 2022 from https://neo4j.com/product/neo4j-graph-database/.

Applications of GDBs

□ **GDBs play a significant role in numerous applications**



Knowledge graphs

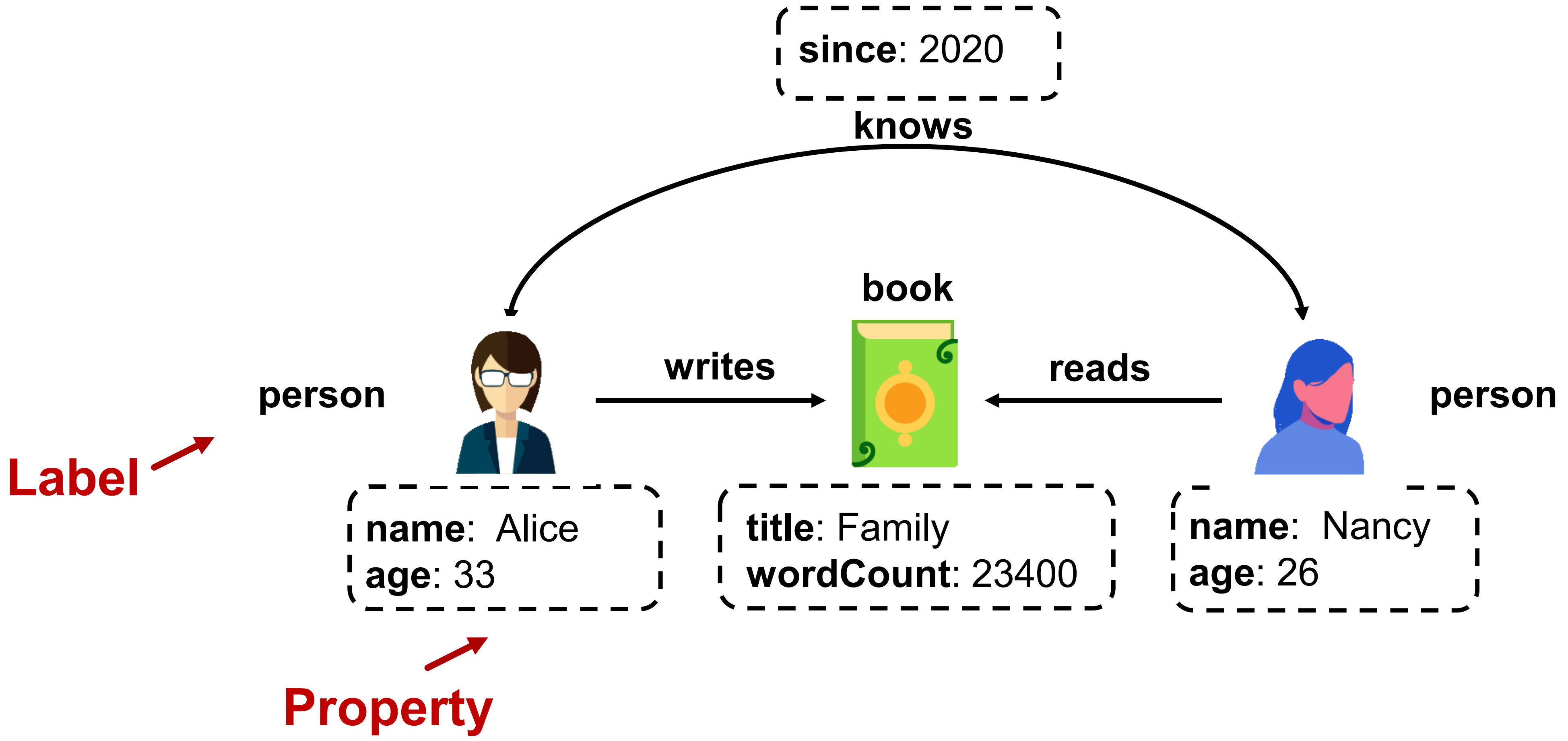
Social networks

Fraud detection

Medical

Labeled Property Graph Model

- Each vertex and edge has a label name and a set of properties



since: 2020

knows

book

person

writes

reads

person

Label

name: Alice
age: 33

title: Family
wordCount: 23400

name: Nancy
age: 26

Property

Graph Query Language

- □ **No standardized way in GDBs to query a graph**

Graph Database System	Graph Query Language
Neo4j	Cypher, Gremlin
Hugegraph	Gremlin
JanusGraph	Gremlin
TinkerGraph	Gremlin
NebulaGraph	nGQL
TigerGraph	GSQL
…	…

66% GDBs support Gremlin APIs[1]

[1] DB-Engines Ranking of Graph DBMS. Retrieved May 23, 2022 from https://db-engines.com/en/ranking/graph+dbms.

Gremlin Query Language

- Gremlin links a sequence of Gremlin API calls for traversing labeled property graphs

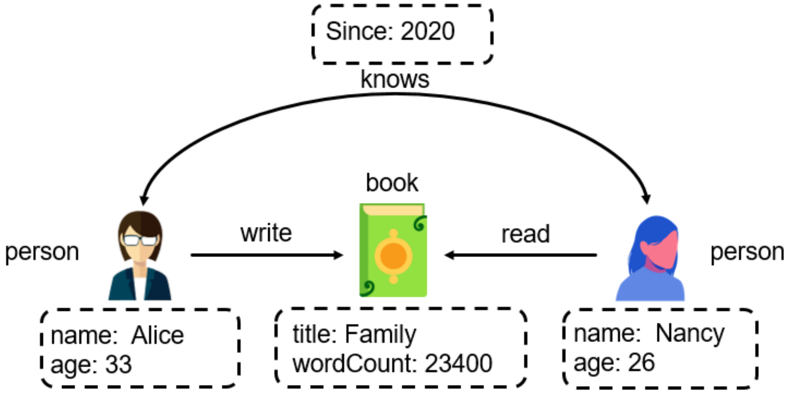
```
g.V().has(‘person’,  ‘age’,  between(20, 35)).count()
```

Get all vertices

Filter vertices with the condition that person’s age is between 20 and 35

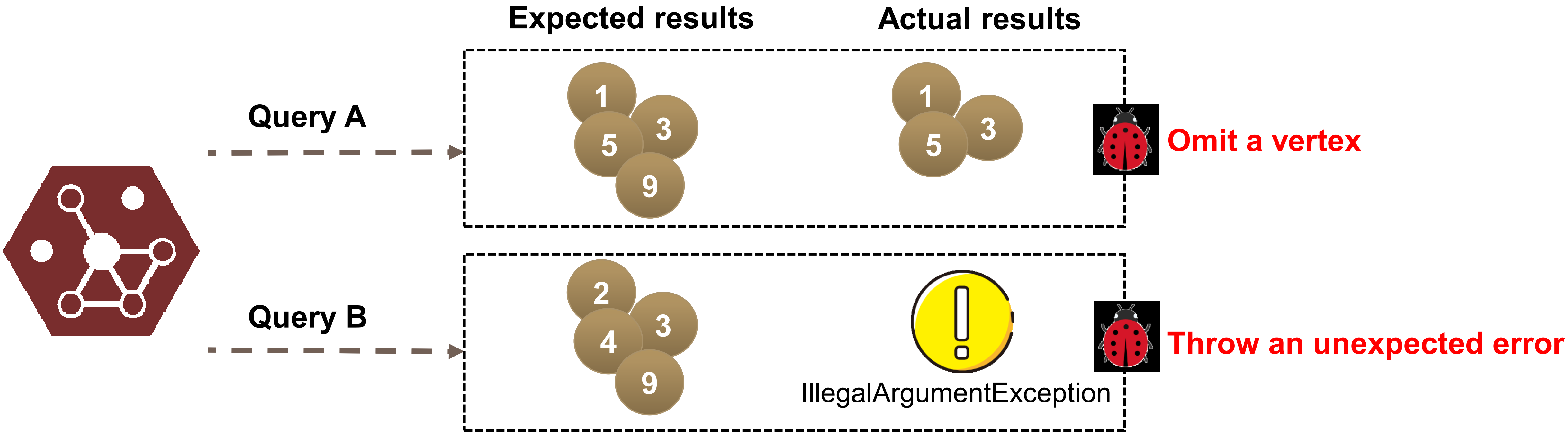
Count the number of persons

How many people are between 20 and 35 years old?

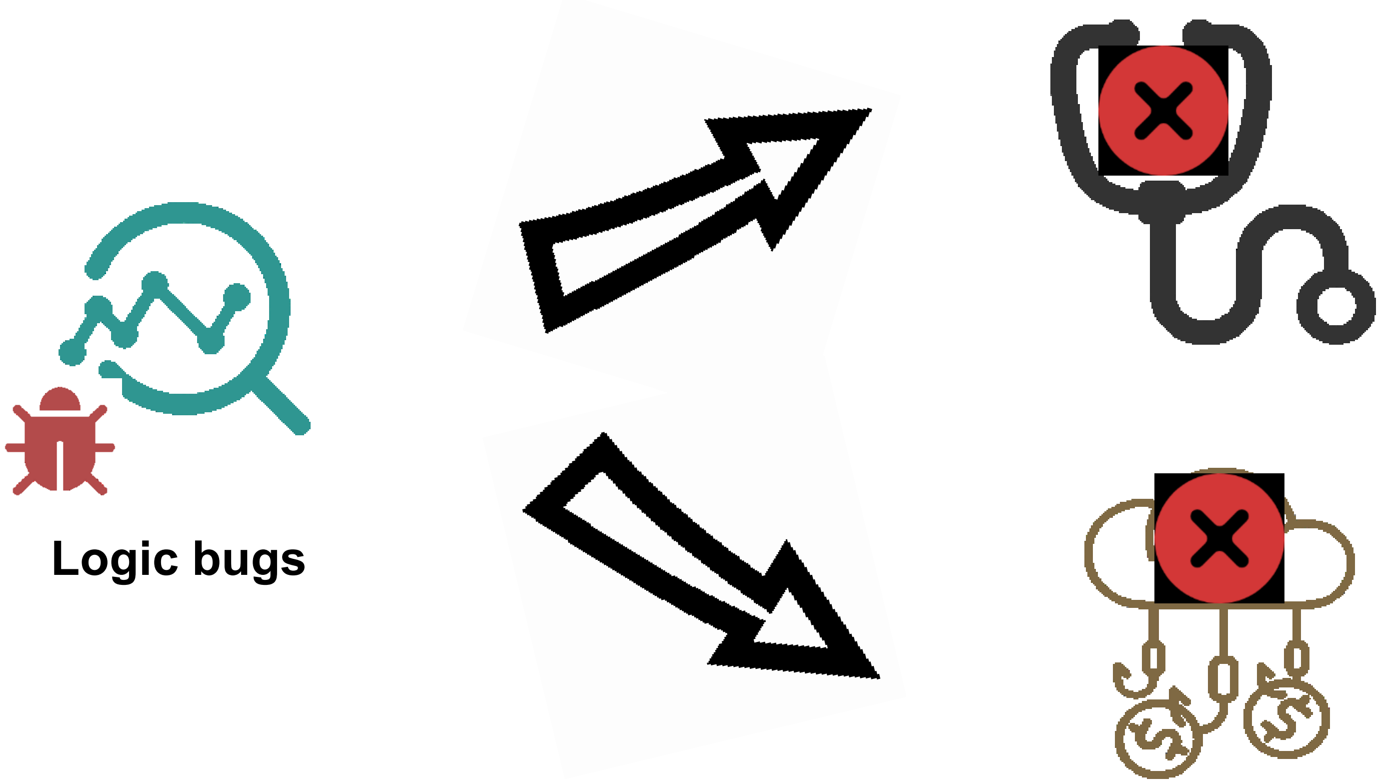


Logic Bug in GDBs

- GDBs suffer from logic bugs, in which a query returns an **incorrect query result** without crashing the GDBs



Logic Bugs Cause Severe Consequences



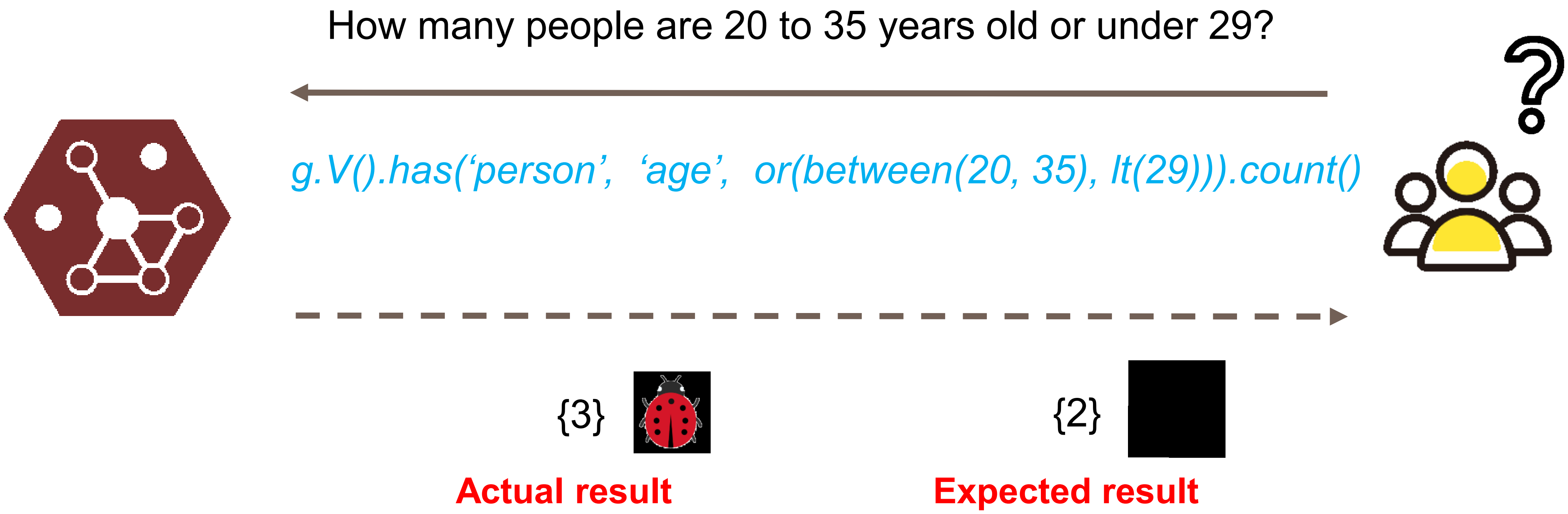
Logic bugs

Error diagnosis in medical application

Error detection in fraud detection application

A Real Logic Bug

- □ **HugeGraph forgets to deduplicate overlapping values for or() operation**



How many people are 20 to 35 years old or under 29?

g.V().has(‘person’, ‘age’, or(between(20, 35), lt(29))).count()

{3}

{2}

Actual result

Expected result

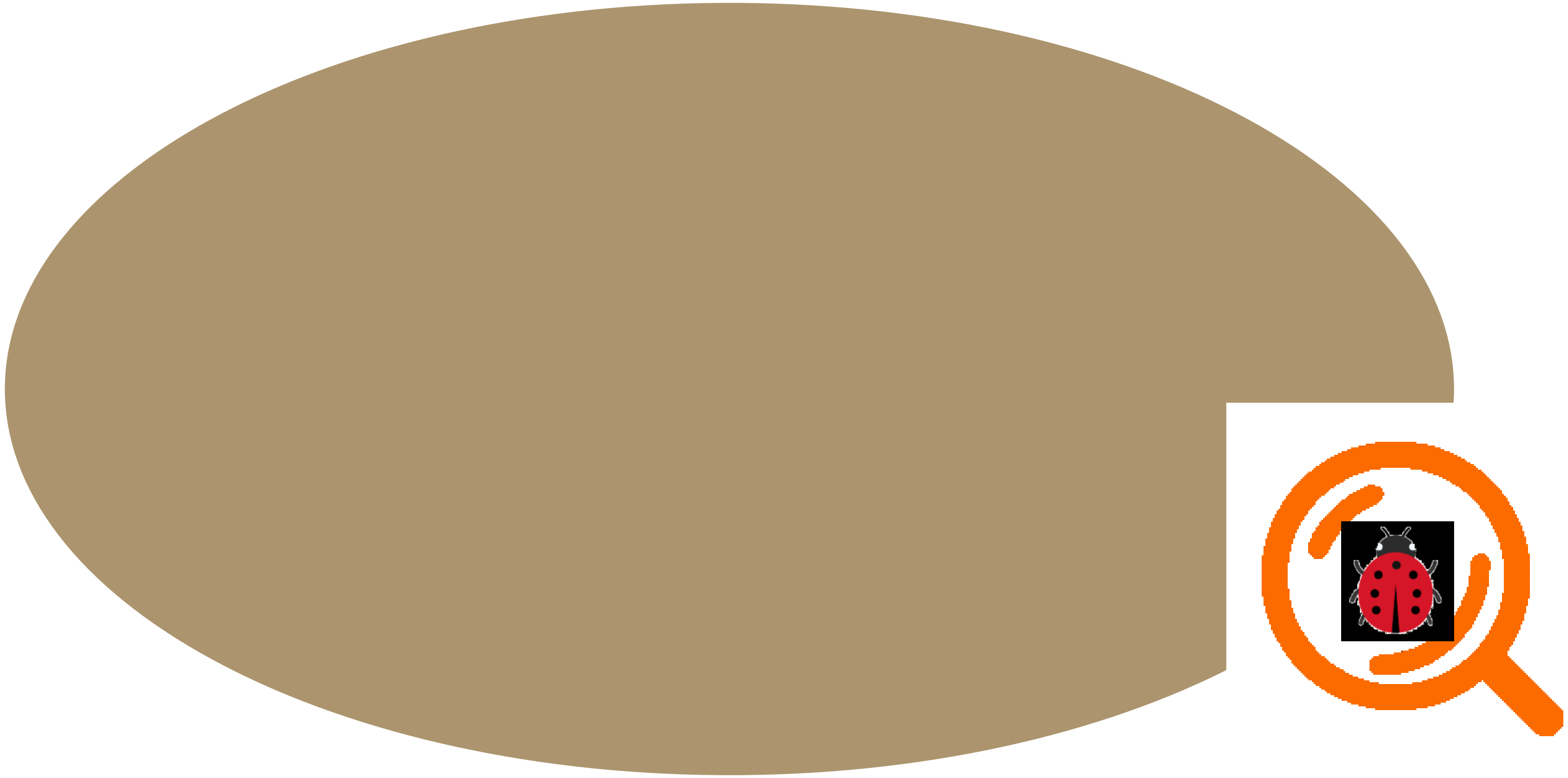
Existing Bug Detection Tools and Approaches

- **Relational database management systems (RDBMSs)**
  - **Differential testing:** RAGS[1], APOLLO[2]
  - **Fuzzing:** SQLSmith[3], AFL[4]
  - **Metamorphic testing:** Non-optimizing reference engine construction[5], Query partition[6]
  - **Testing oracle:** Pivoted query synthesis[7]

Cannot be directly applied to GDBs!

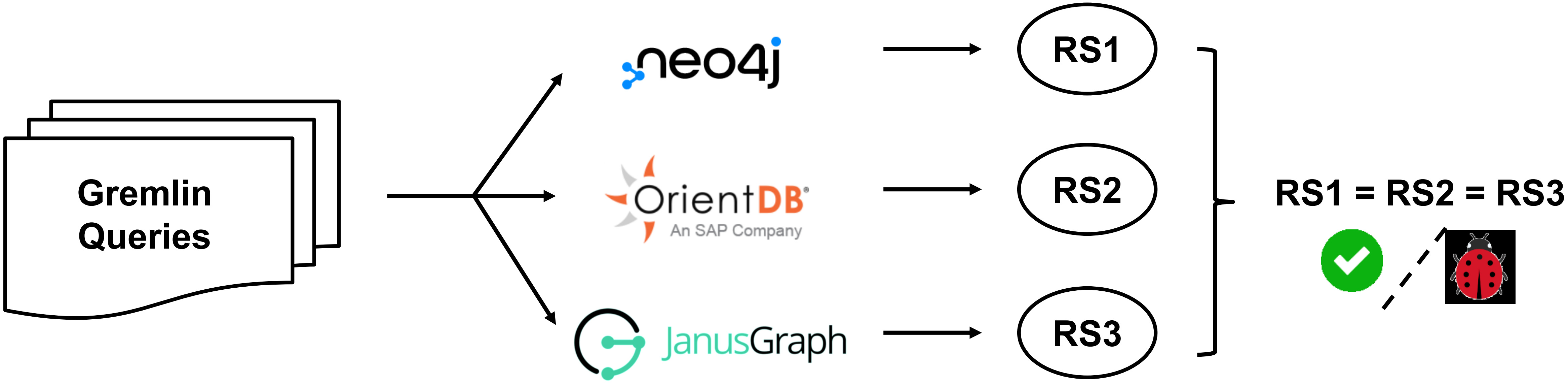
[1] Donald S. Slutz. Massive Stochastic Testing of SQL. VLDB 1998.
[2] Jinho Jung, et. al., APOLLO: Automatic Detection and Diagnosis of Performance Regressions in Database Systems. PVLDB 2019.
[3] SQLsmith. Retrieved August 5, 2021 from https://github.com/anse1/sqlsmith.
[4] AFL. Retrieved September 13, 2021 from https://github.com/google/AFL.
[5] Manuel Rigger and Zhendong Su. Detecting Optimization Bugs in Database Engines via Non-Optimizing Reference Engine Construction. FSE 2020.
[6] Manuel Rigger and Zhendong Su. Finding Bugs in Database Systems via Query Partitioning. OOPSLA 2020.
[7] Manuel Rigger and Zhendong Su. Testing Database Engines via Pivoted Query Synthesis. OSDI 2020.

Goal: Finding Bugs in Gremlin-Based GDBs

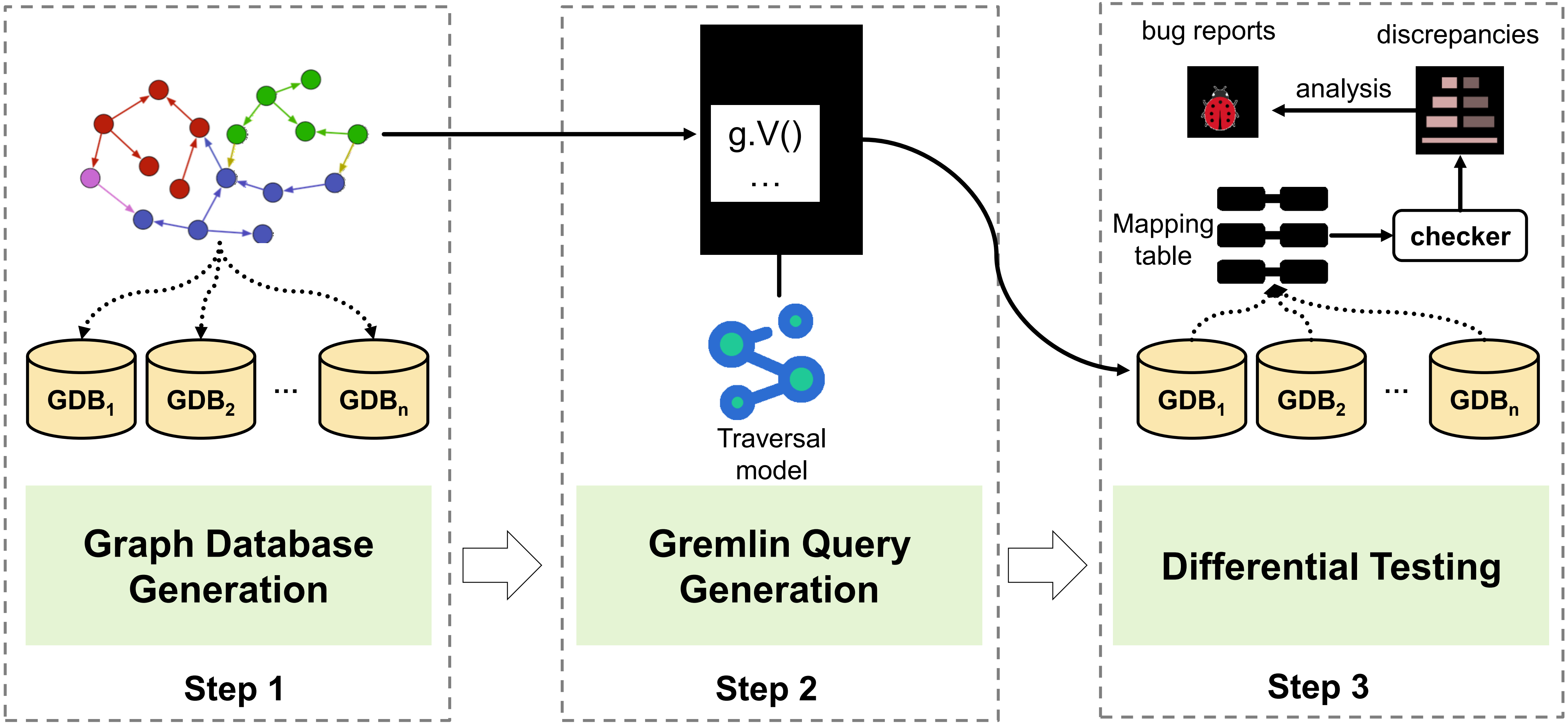


Grand: Randomized Differential Testing

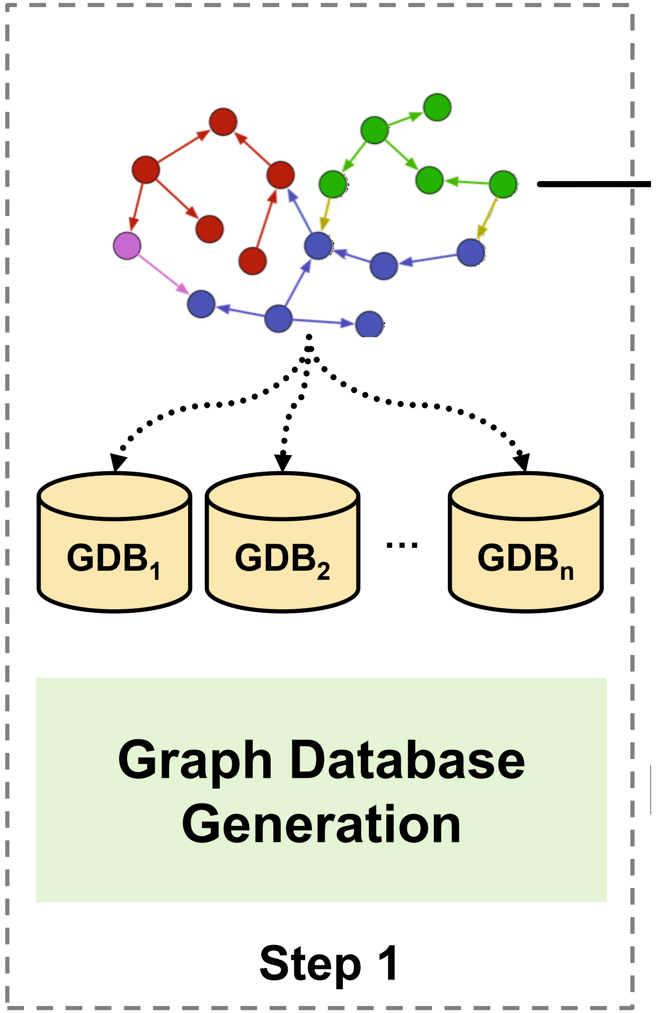
- Construct semantically equivalent databases for multiple GDBs, and then compare the results of a Gremlin query on these databases



Overview of Grand

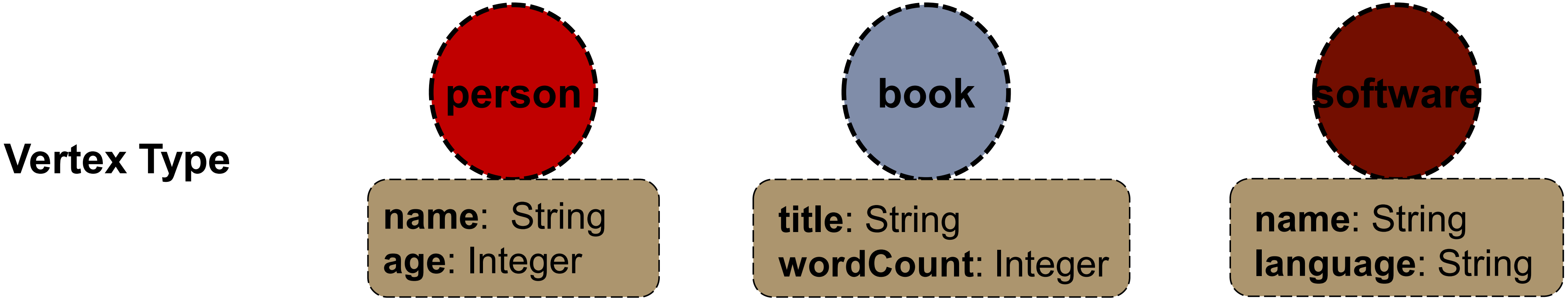


Step 1: Graph Database Generation



Graph Schema Generation

□ **Randomly generate vertex types and edge types**



Vertex Type

person

name: String
age: Integer

book

title: String
wordCount: Integer

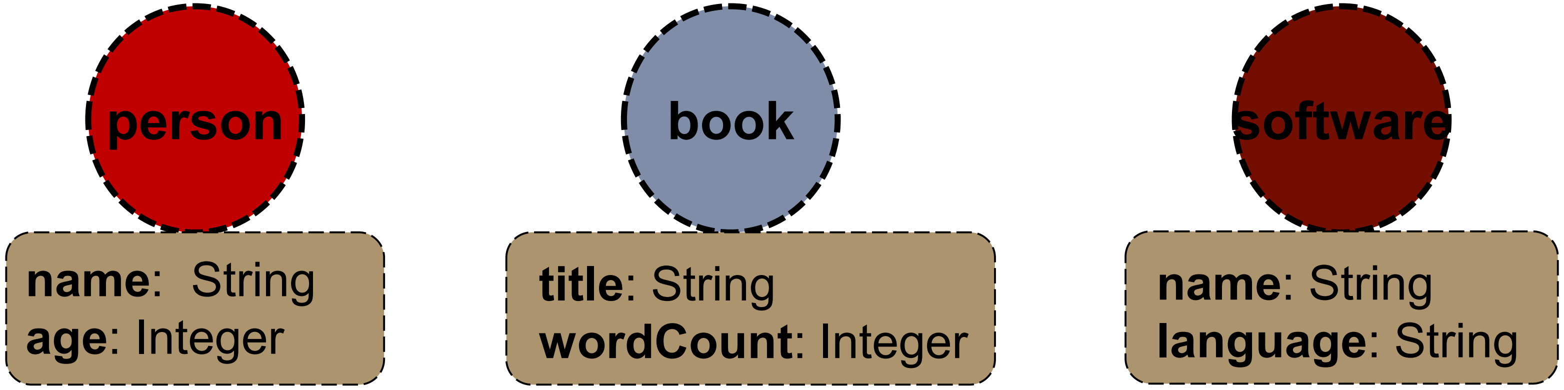
software

name: String
language: String

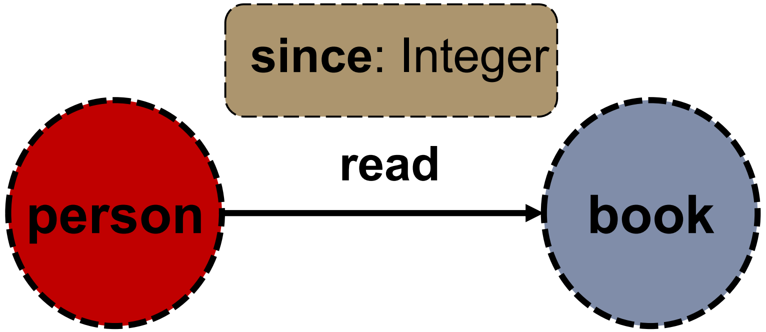
Graph Schema Generation

- Randomly generate vertex types and edge types

Vertex Type

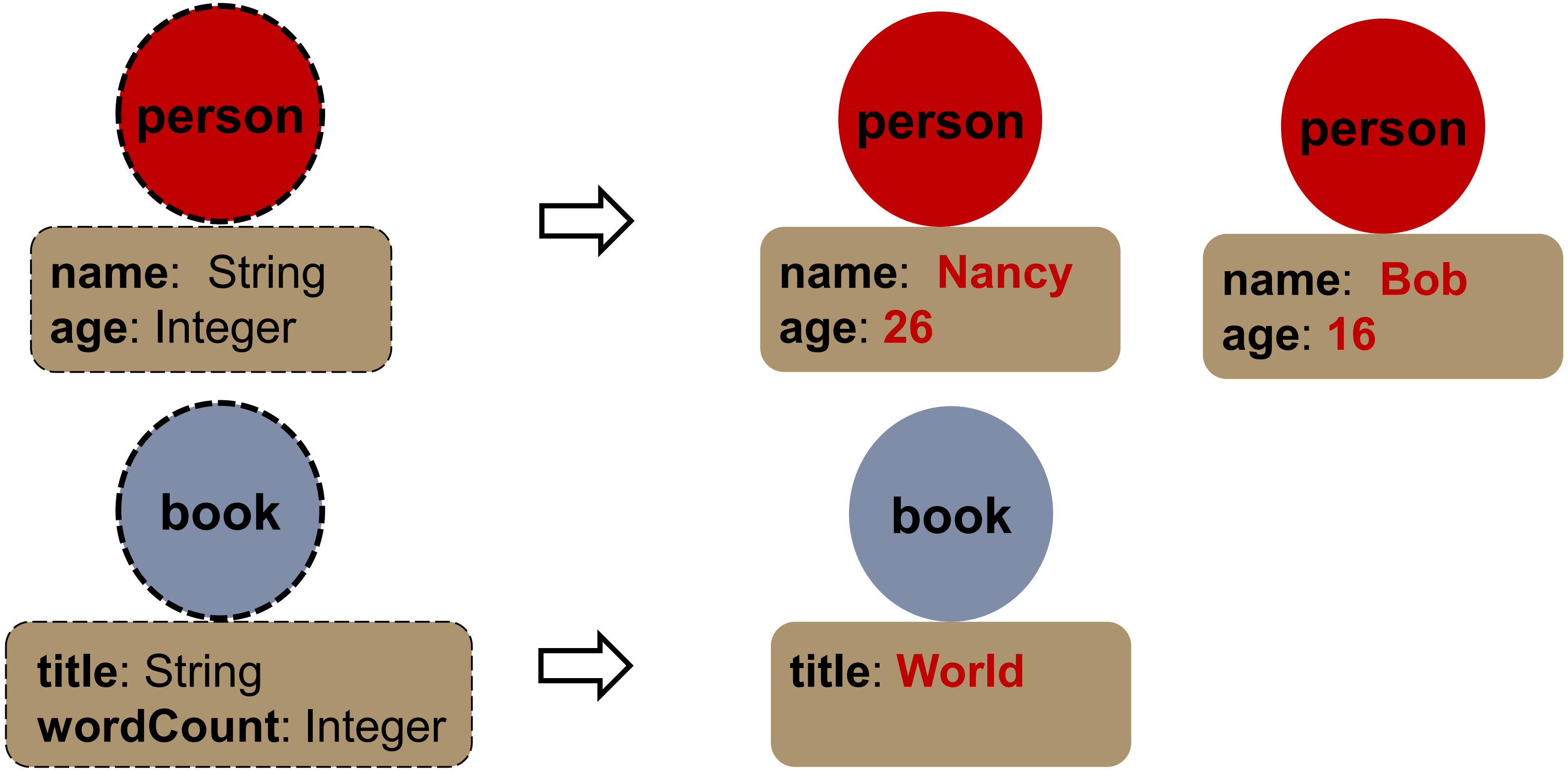


Edge Type



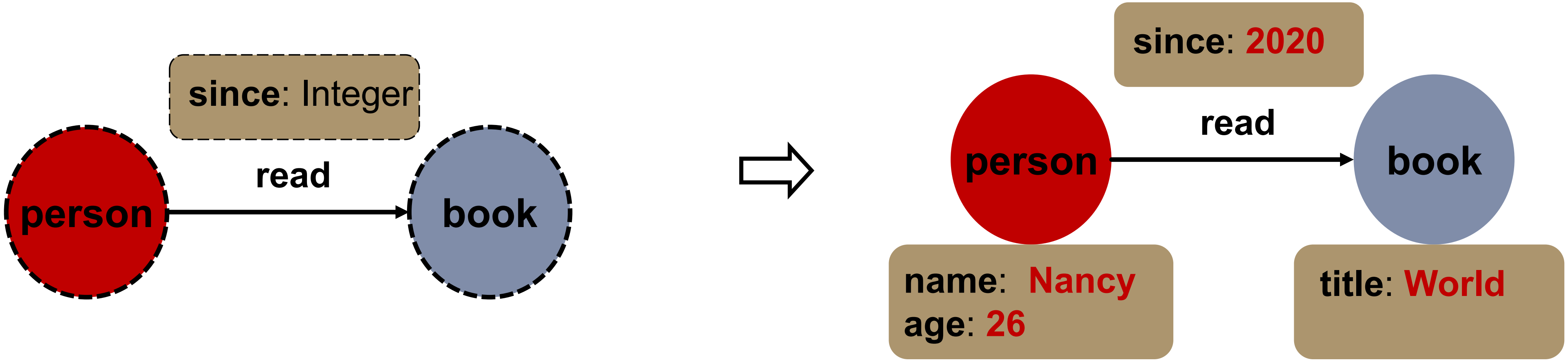
Graph Data Generation

- Based on the generated graph schema, Grand randomly generates a set of vertices and edges

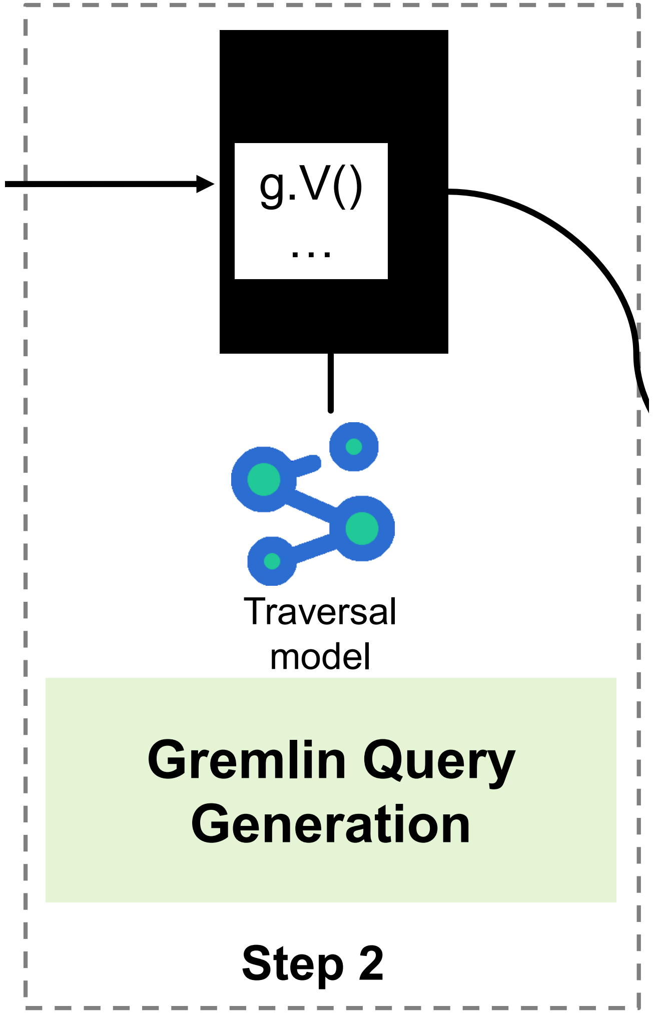


Graph Data Generation

- Based on the generated graph schema, Grand generates vertices and edges



Step 2: Gremlin Query Generation

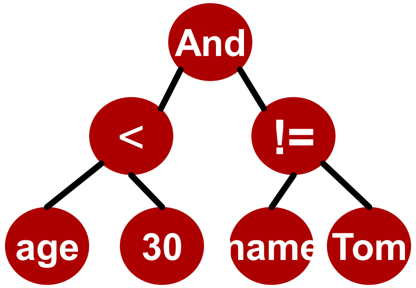


Existing Generation Tools are Unusable

- Gremlin has different syntax and query patterns from SQL

```
SELECT name
FROM t0
WHERE (t0.age < 30) AND
         (t0.name != 'Tom')
```

```
g.V()
.has('person', 'age', lt(30))
.has('person', 'name', neq('Tom'))
.values('name')
```



Abstract Syntax Tree

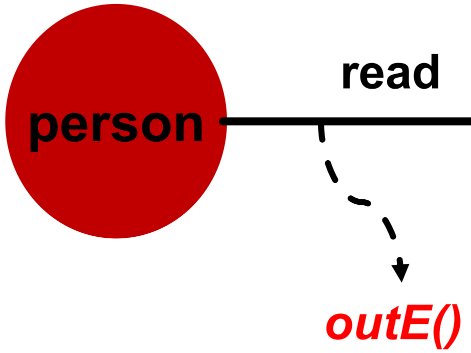
Random Gremlin Query Generation

- □ Generate some grammatically correct but meaningless queries and ignore the semantics of Gremlin APIs

We construct a model to guide us in generating syntactically correct and valid Gremlin queries.

Insight

The input type of a Gremlin API in a query should match the output type of its previous Gremlin API.

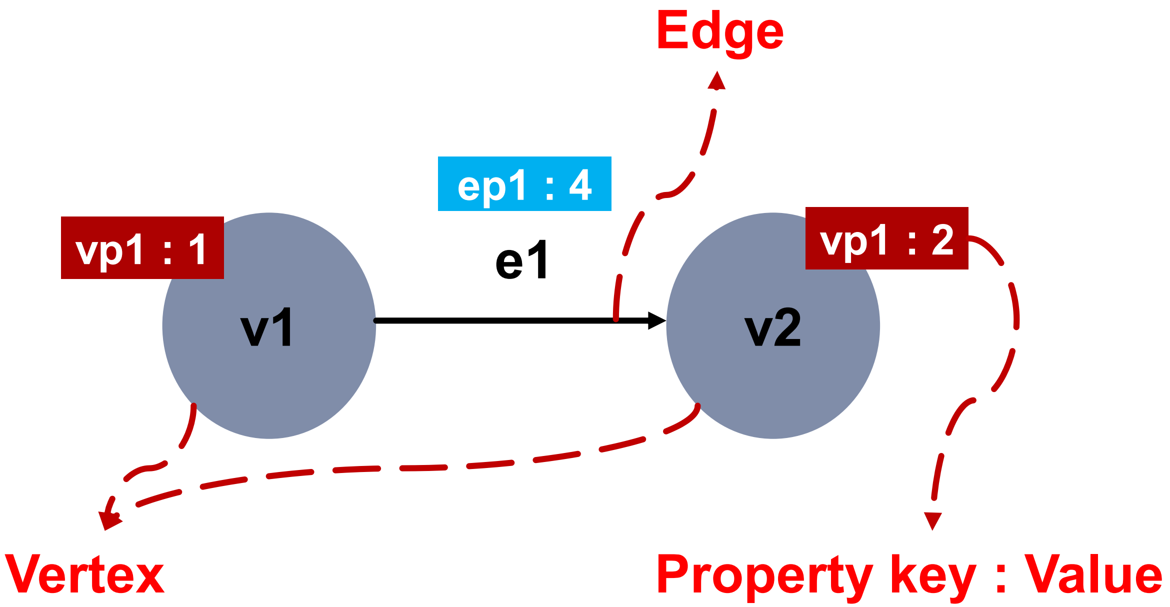


g.V().outE() ✓

g.E().outE() ✗

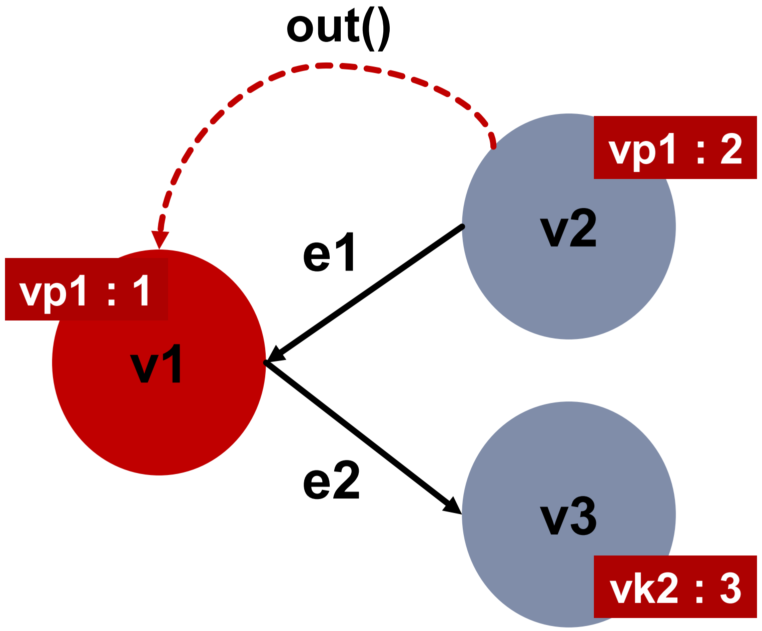
Traversal Model for Gremlin

- □ We abstract three entities, i.e., vertex, edge and value
- □ We construct a traversal model to describe the legal operations and semantics in these entities

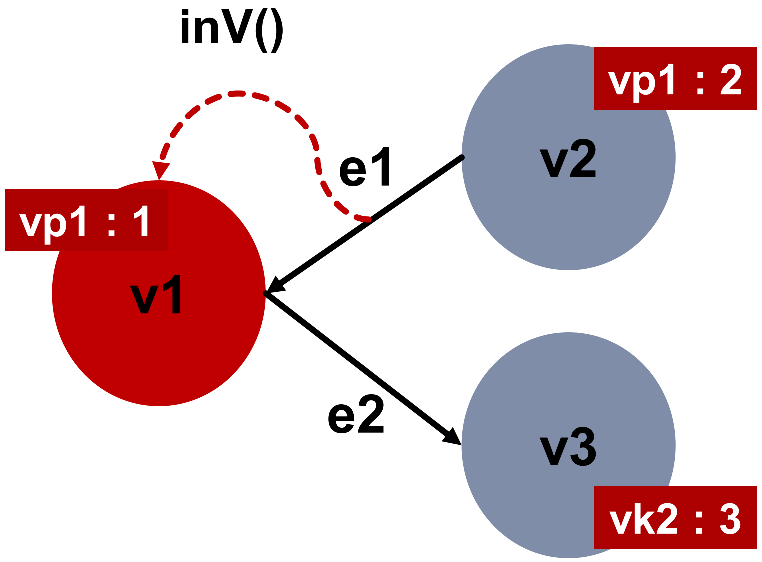


Traversal Model - Vertex

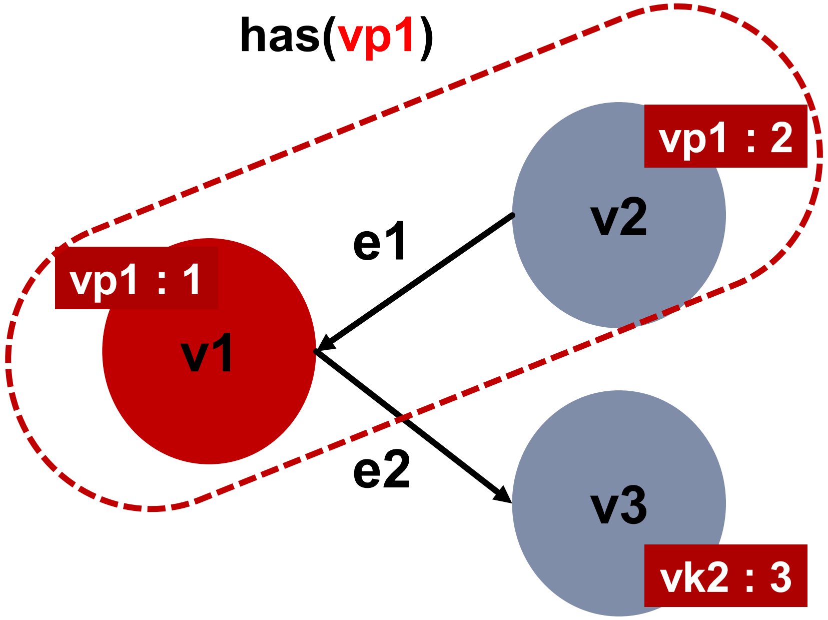
- Legal operations on a vertex



Traverse v1 from vertices



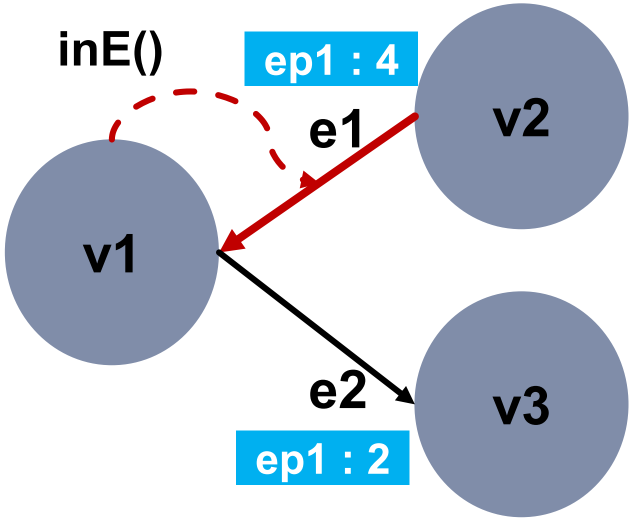
Traverse v1 from edges



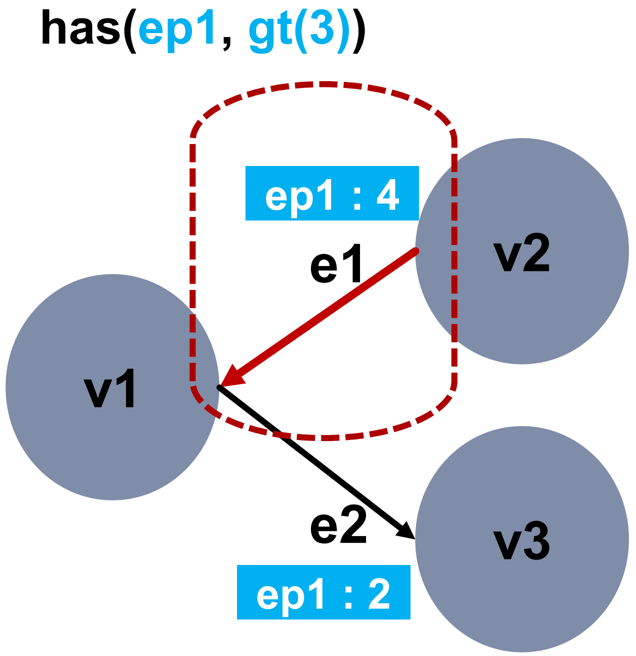
Filter v1 by the given conditions

Traversal Model - Edge

- Legal operations on an edge



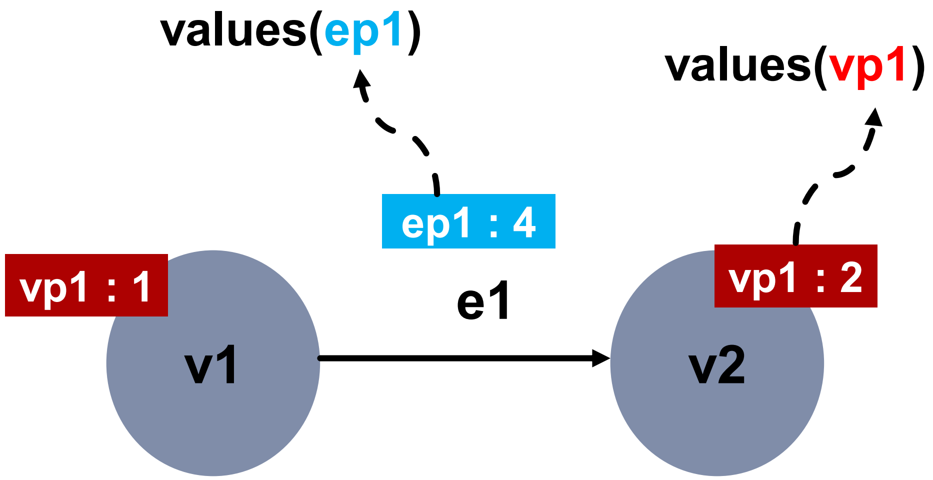
Traverse e1 from vertices



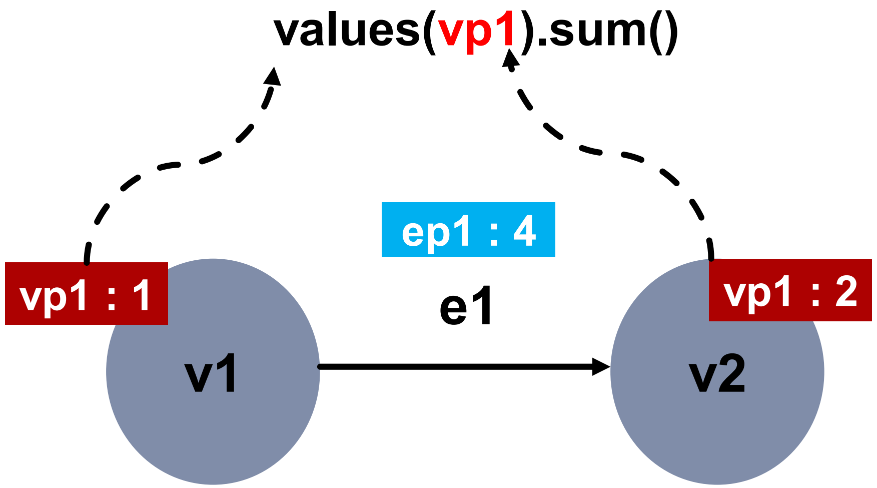
Filter e1 by the given conditions

Traversal Model - Value

- Legal operations on values



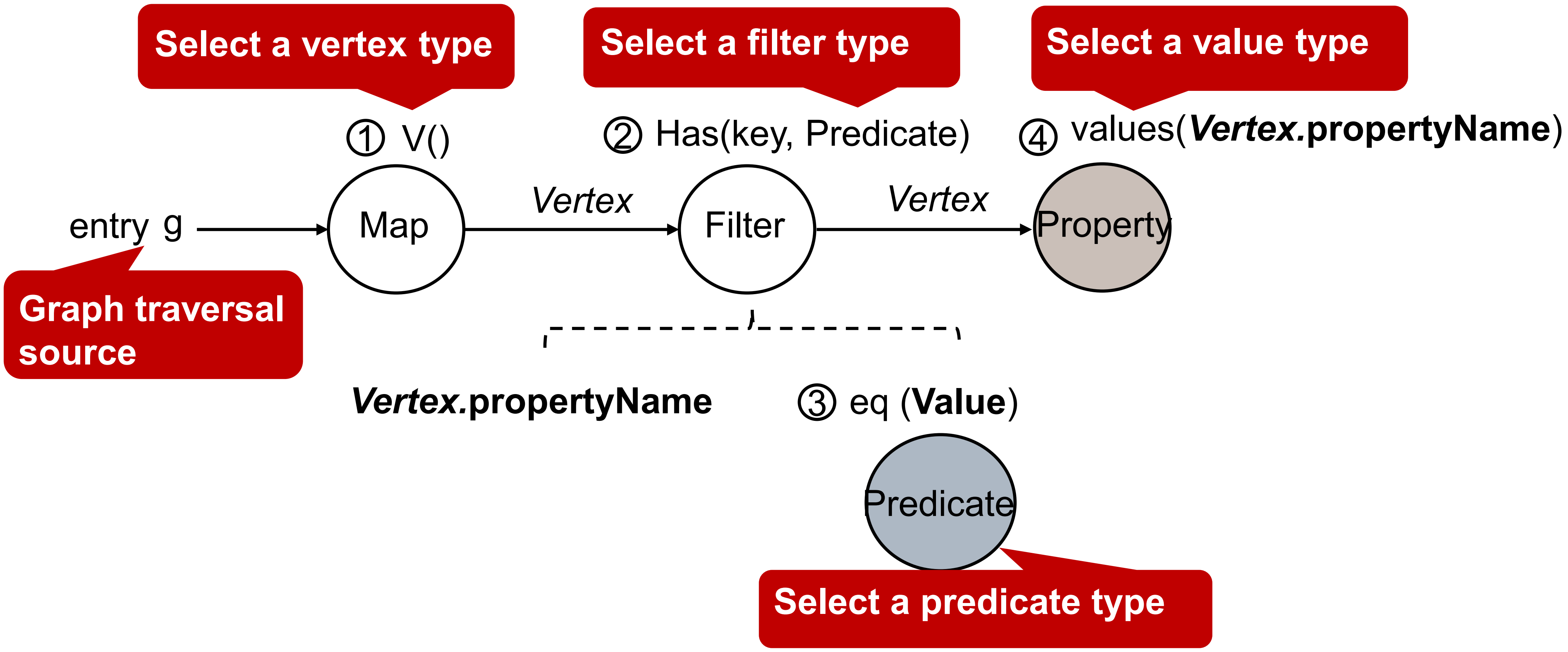
Retrieve property values from vertices or edges



Aggregate property values

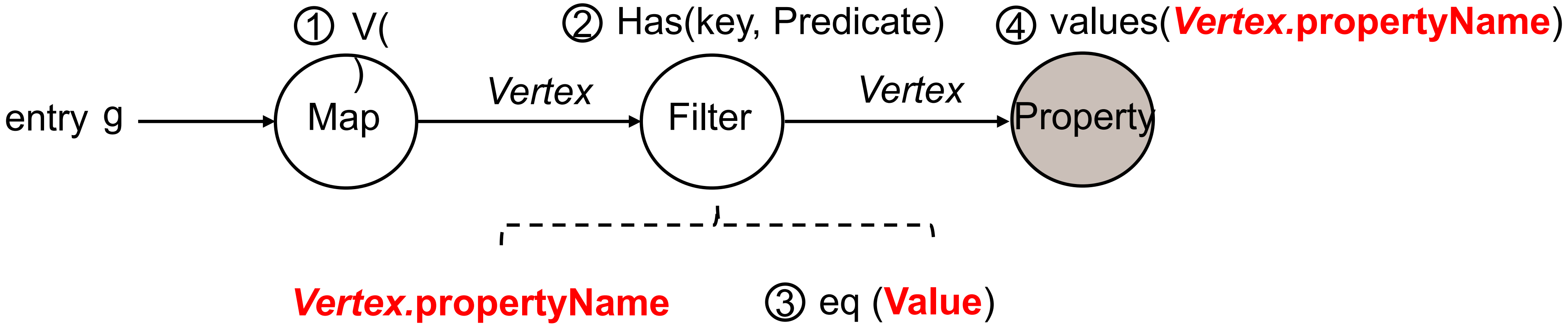
Model-based Query Generation

- Randomly select Gremlin APIs until the maximum query length is reached or exit condition is satisfied



Parameters in Gremlin Query

- **Completely random parameter generation returns lots of empty results, which can greatly affect the effectiveness of GDB testing**



① V( )

② Has(key, Predicate)

④ values(*Vertex*.propertyName)

entry g

Map

Vertex

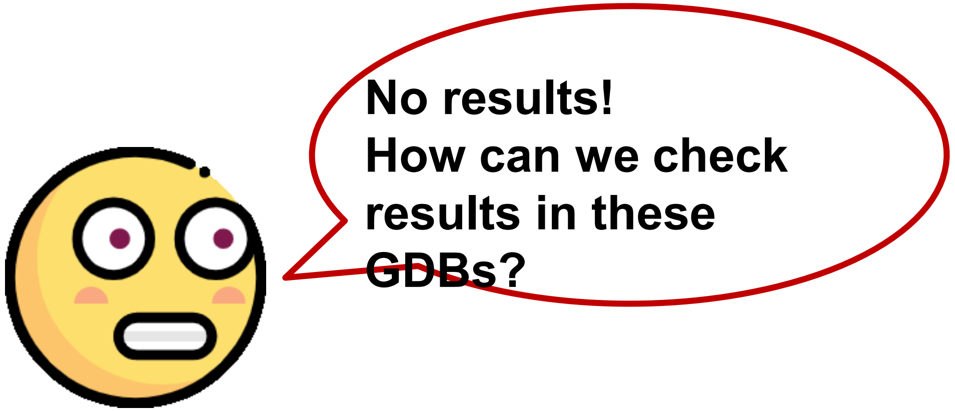
Filter

Vertex

Property

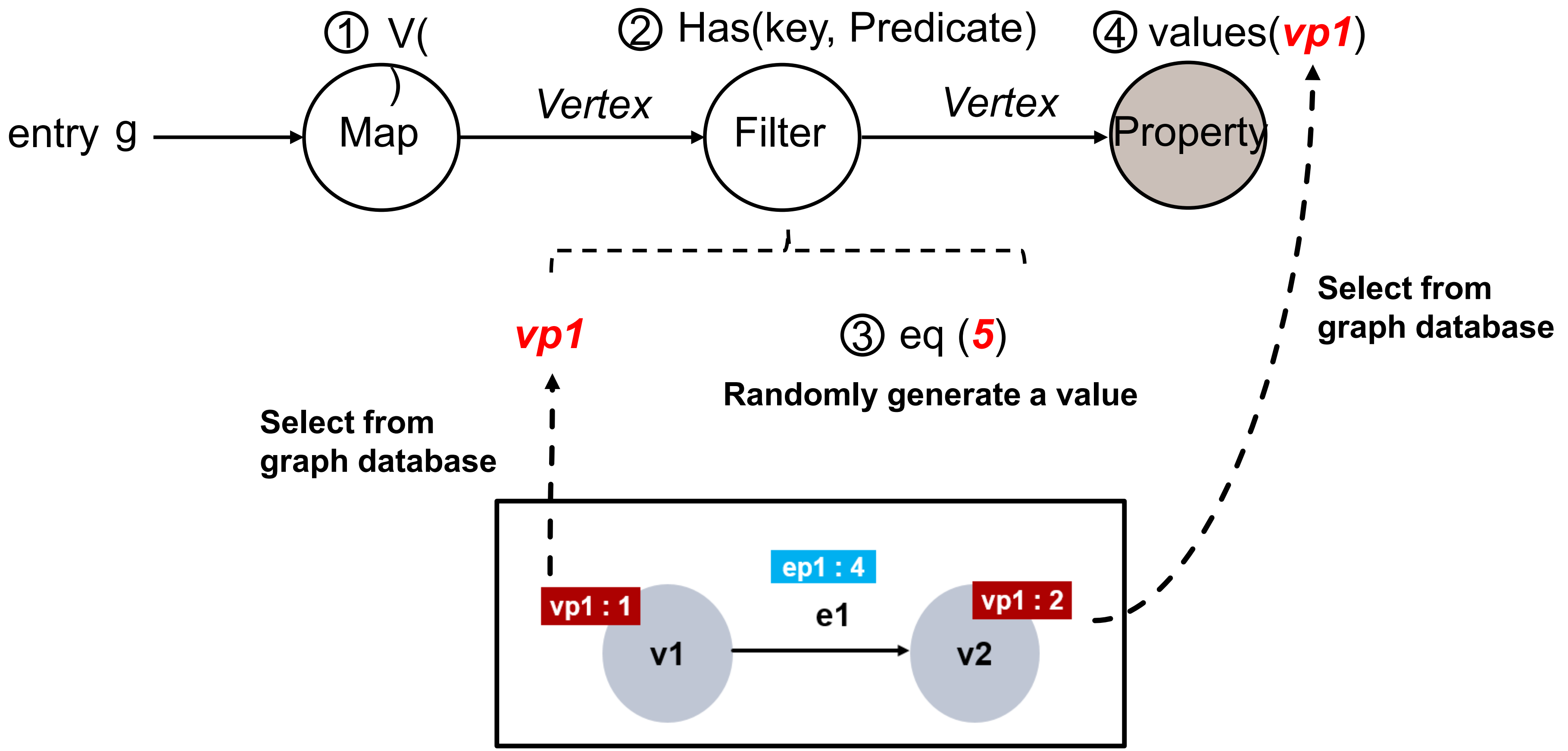
Vertex.propertyName

③ eq (Value)

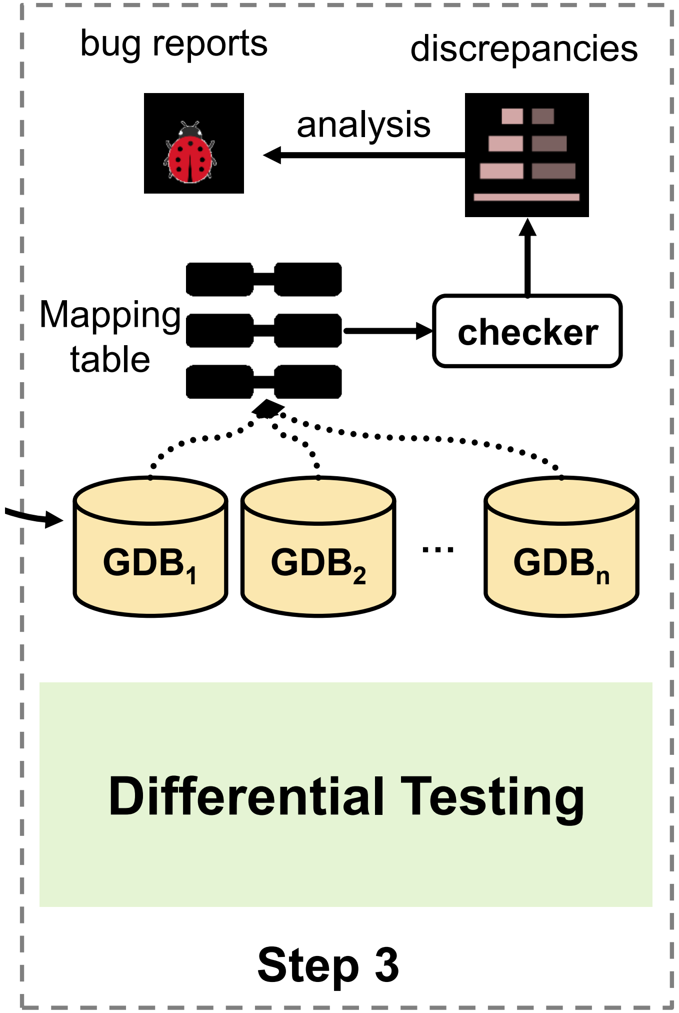


Parameter Values Generation

- □ **Randomly select a value from the generated graph database**
- □ **Randomly generate a value**



Step 3: Differential Testing in GDBs



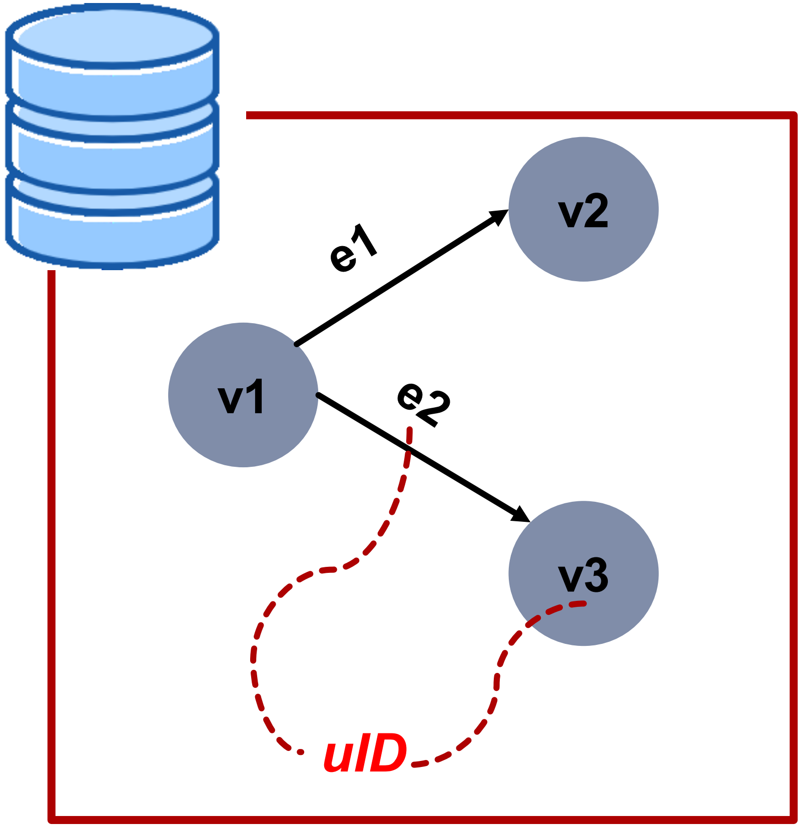
Differential Testing

- Execute Gremlin queries on target GDBs and compare the return results

We convert the query results from different GDBs into a unified format.

Query Result Mapping

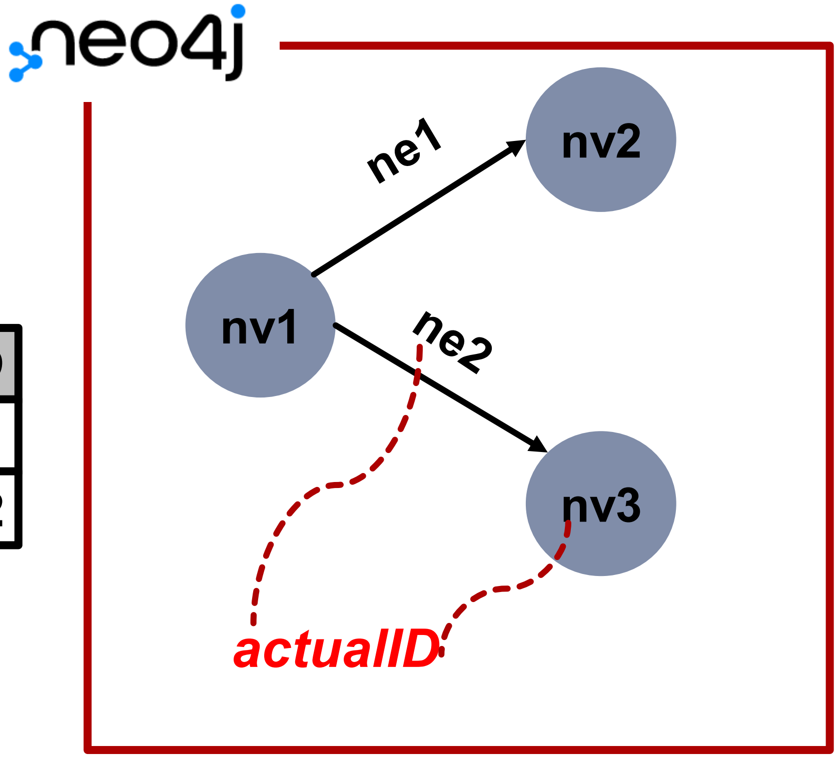
- Record the mapping relations between uID and $actualID$ in all GDBs



Mapping Table

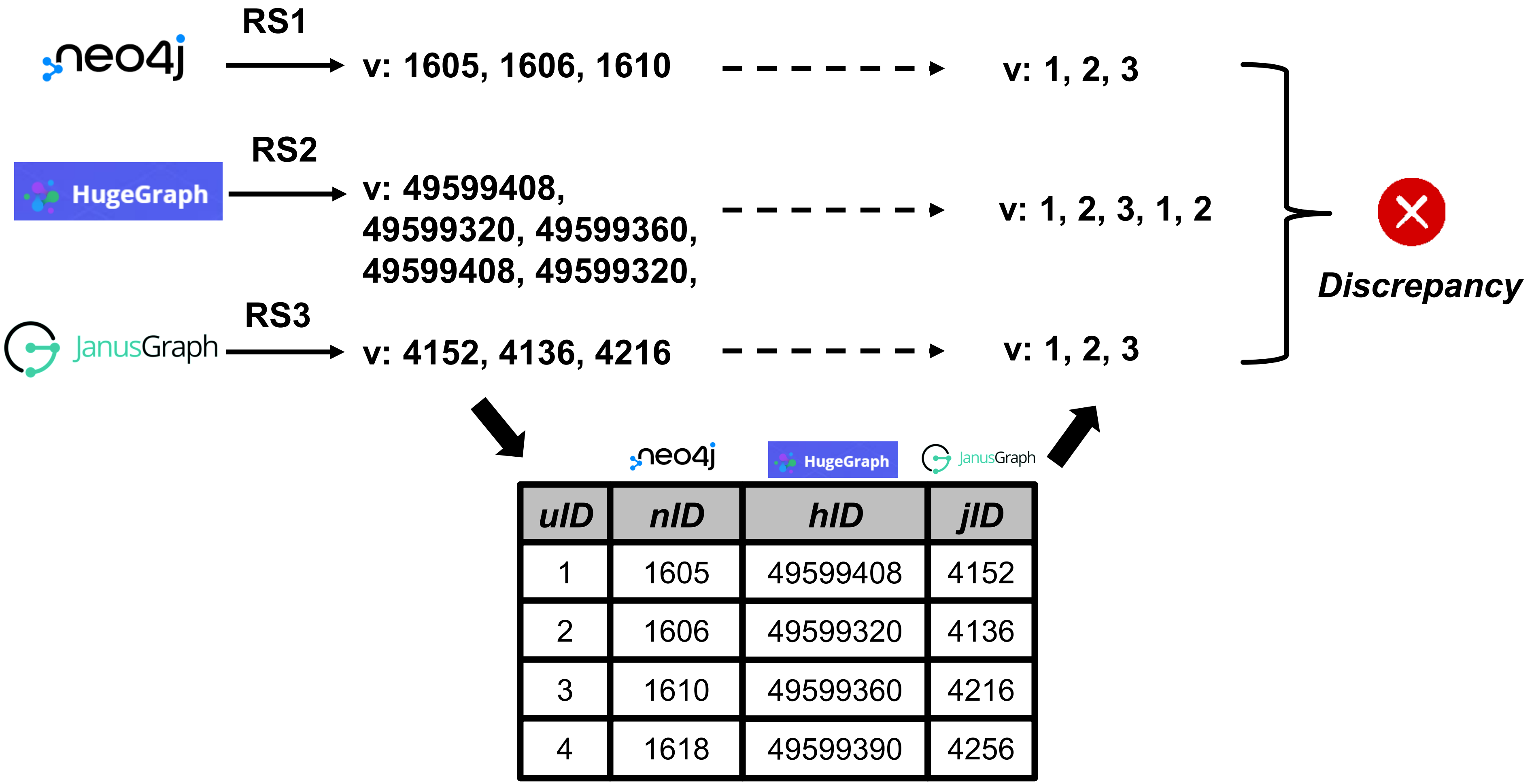
uID	*nID*
v1	nv1
v2	nv2
v3	nv3

uID	*nID*
e1	ne1
e2	ne2



Differential Results Verification

- Verify the unified query results of all target GDBs



neo4j RS1 → v: 1605, 1606, 1610 → v: 1, 2, 3

HugeGraph RS2 → v: 49599408, 49599320, 49599360, 49599408, 49599320, → v: 1, 2, 3, 1, 2

JanusGraph RS3 → v: 4152, 4136, 4216 → v: 1, 2, 3

Discrepancy

uID	*nID*	*hID*	*jID*
1	1605	49599408	4152
2	1606	49599320	4136
3	1610	49599360	4216
4	1618	49599390	4256

Evaluation

- Target GDBs
  - 6 widely-used graph database systems

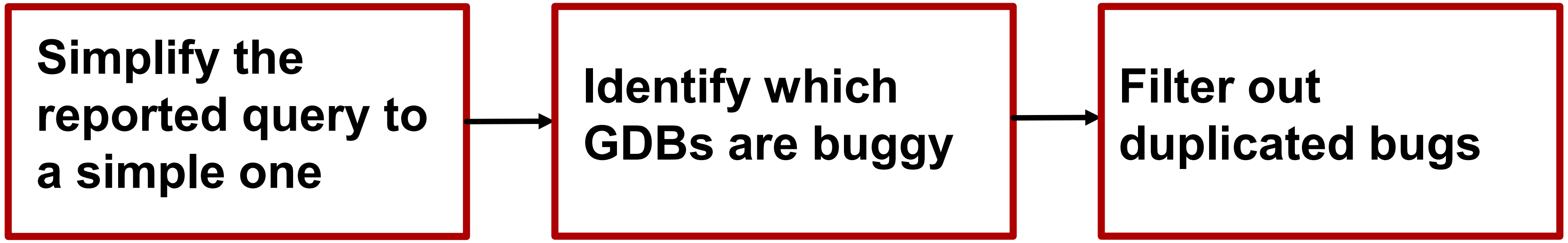
GDB	Rank*	GitHub Star	Initial Release
Neo4j	1	9.2k	2007
OrientDB	5	4.4k	2010
JanusGraph	8	4.1k	2017
HugeGraph	22	1.7k	2018
TinkerGraph	23	1.4k	2009
ArcadeDB	27	119	2021

* There are total 36 GDBs in DB-Engines Ranking of Graph DBMS.

Evaluation

□ Testing methodology

- ➢ Run 15 times and 1000 random queries in each time
- ➢ Manually reproduce and analyze the reported discrepancies



Simplify the reported query to a simple one

Identify which GDBs are buggy

Filter out duplicated bugs

Analyzing 709 discrepancies and obtaining 21 logic bugs

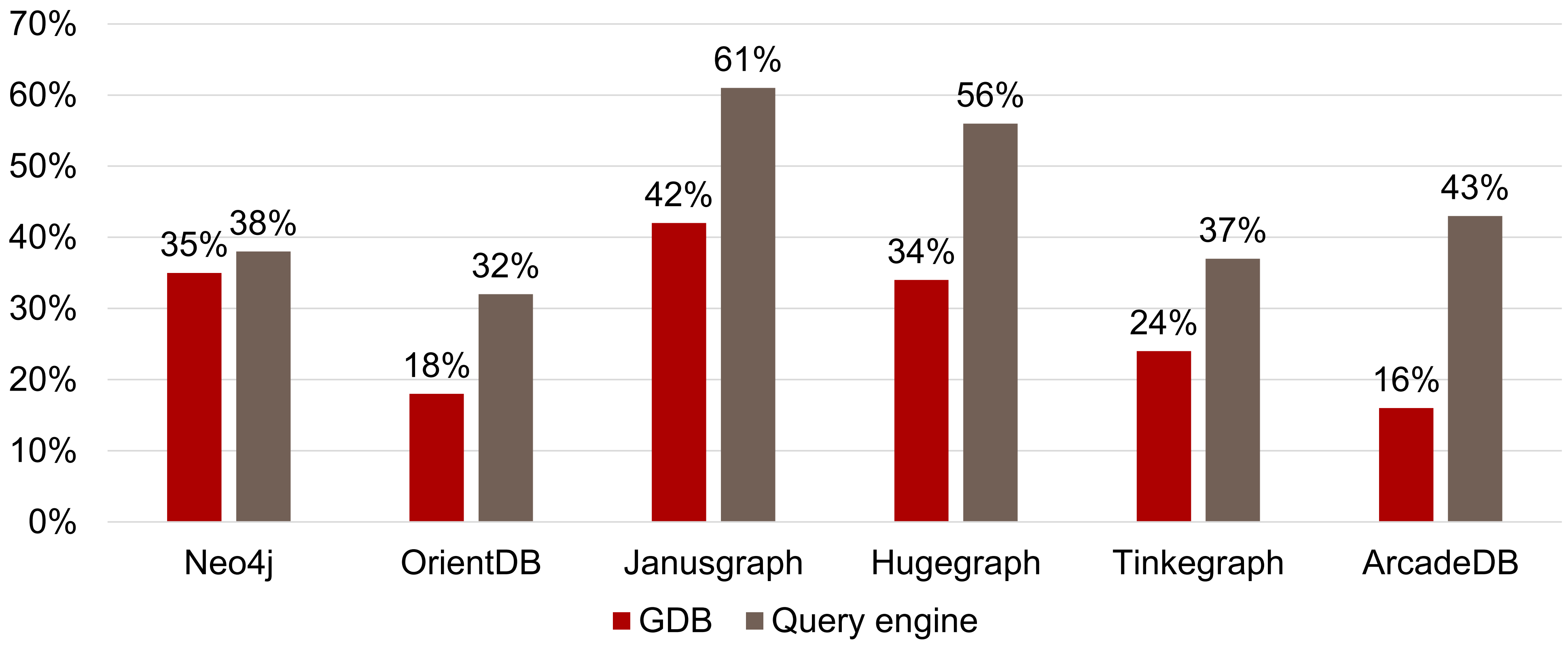
Bug Overview

- □ 21 bugs have been found in six widely-used GDBs

GDB	Detected	Confirmed	Fixed
Neo4j	3	2	1
OrientDB	1	0	0
JanusGraph	3	3	2
HugeGraph	9	9	3
TinkerGraph	3	3	1
ArcadeDB	2	1	0
Total	**21**	**18**	**7**

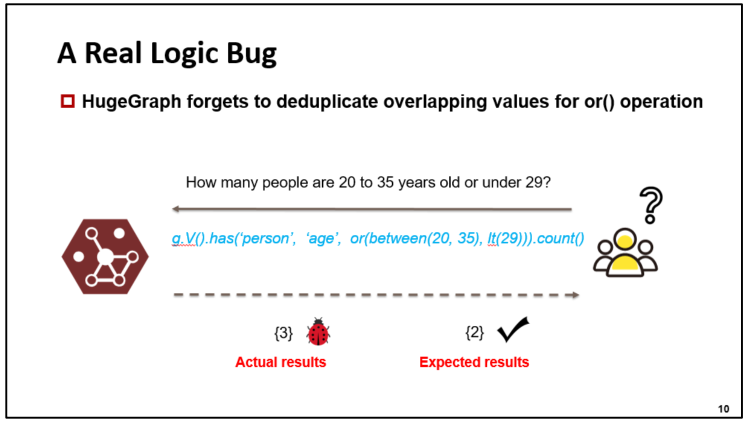
Instruction Coverage

- □ **Achieve coverage from 32% to 61% for query engines and 16% to 42% for target GDBs**



Conclusion

Goal: Find logic bugs in GDBs



A Real Logic Bug

- HugeGraph forgets to deduplicate overlapping values for or() operation

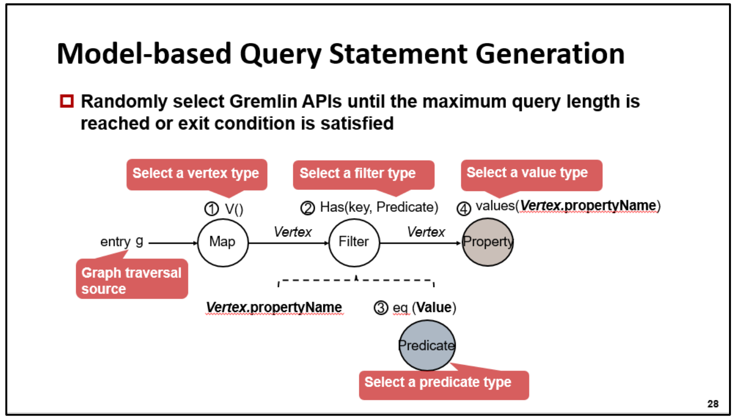
How many people are 20 to 35 years old or under 29?

g.V().has('person', 'age', or(between(20, 35), lt(29))).count()

{3} Actual results

{2} Expected results

Model-based query generation



Model-based Query Statement Generation

- Randomly select Gremlin APIs until the maximum query length is reached or exit condition is satisfied

Select a vertex type

Select a filter type

Select a value type

① V()

② Has(key, Predicate)

④ values(*Vertex*.propertyName)

entry g

Map

Vertex

Filter

Vertex

Property

Graph traversal source

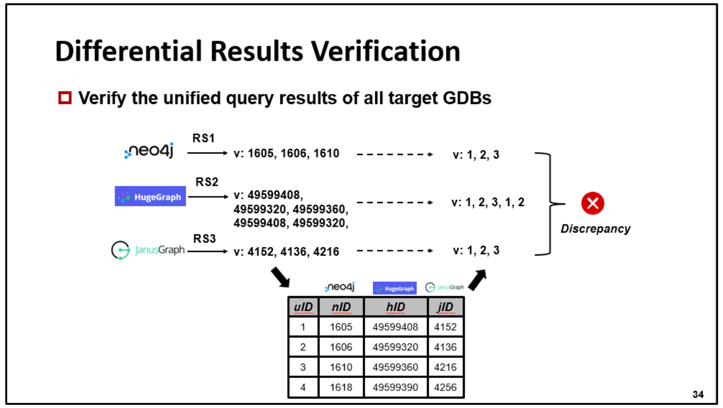
Vertex.propertyName

③ eq (**Value**)

Predicate

Select a predicate type

Differential results verification



Differential Results Verification

- Verify the unified query results of all target GDBs

neo4j RS1 v: 1605, 1606, 1610 → v: 1, 2, 3

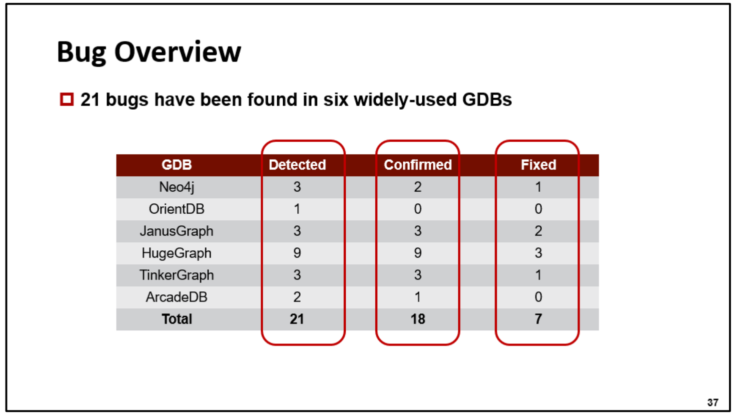
HugeGraph RS2 v: 49599408, 49599320, 49599360, 49599408, 49599320, → v: 1, 2, 3, 1, 2

JanusGraph RS3 v: 4152, 4136, 4216 → v: 1, 2, 3

Discrepancy

uID	*nID*	*hID*	*jID*
1	1605	49599408	4152
2	1606	49599320	4136
3	1610	49599360	4216
4	1618	49599390	4256

Evaluation: 21 bugs in six GDBs



Bug Overview

- 21 bugs have been found in six widely-used GDBs

GDB	Detected	Confirmed	Fixed
Neo4j	3	2	1
OrientDB	1	0	0
JanusGraph	3	3	2
HugeGraph	9	9	3
TinkerGraph	3	3	1
ArcadeDB	2	1	0
Total	**21**	**18**	**7**

https://github.com/tcse-iscas/Grand.

Q&A

THANK YOU!