Redis Graph

A graph database built on top of Redis

redisday
TLV 2017

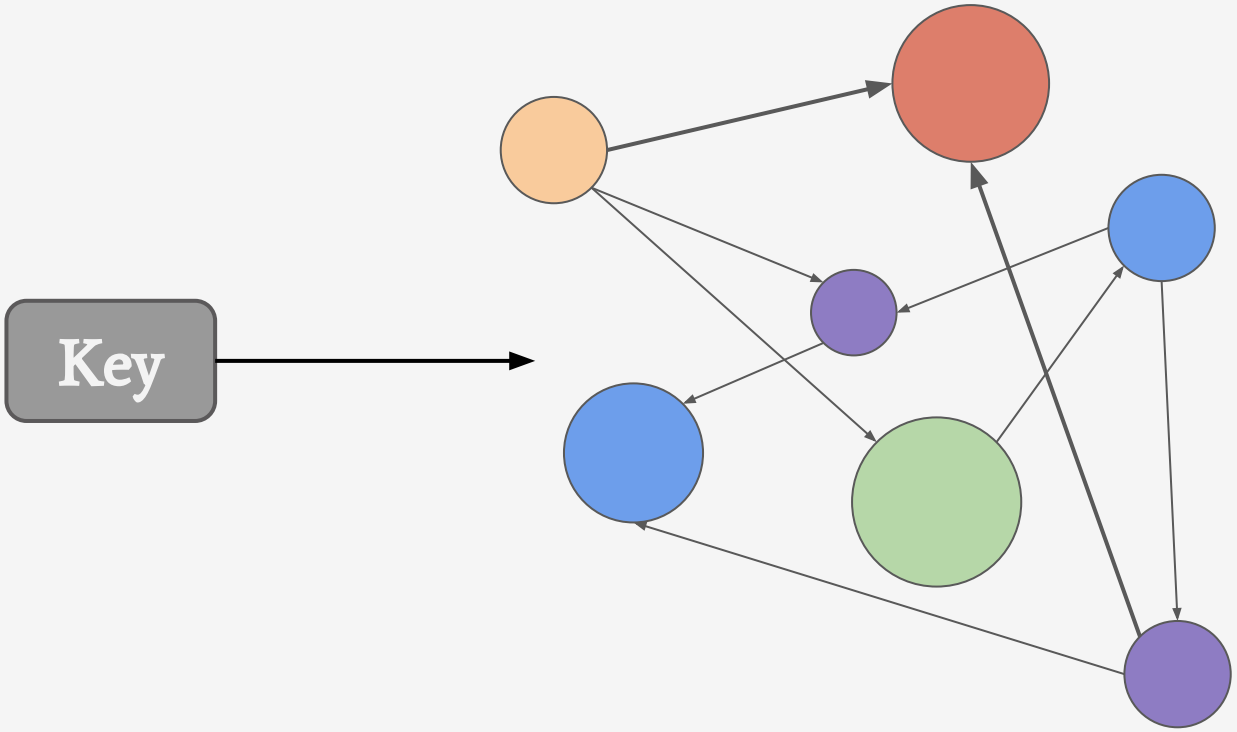
Why?

Load balancing

Quickest route to work

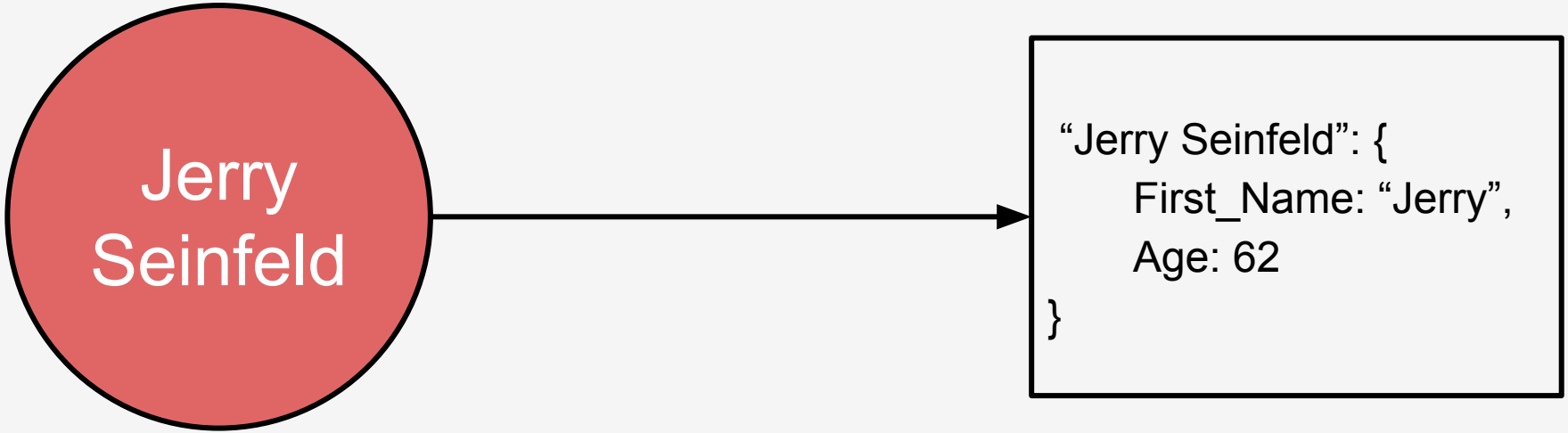
Friend of a friend

A Lego for your database



Graph !

Node



Relations

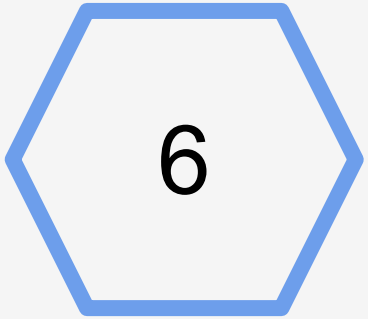
Hexastore

S
Subject

P
Predicate

O
Object

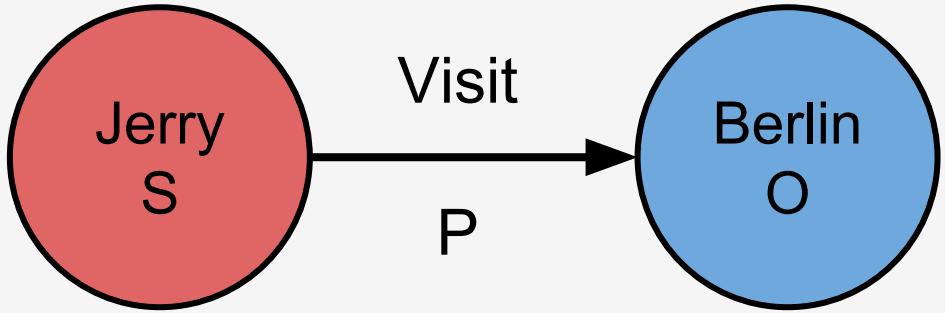
SPO OSP
SOP PSO
OPS POS



Hexastore

Triplets

SPO:Jerry:Visit:Berlin
SOP:Jerry:Berlin:Visit
OPS:Berlin:Visit:Jerry
OSP:Berlin:Jerry:Visit
PSO:Visit:Jerry:Berlin
POS:Visit:Berlin:Jerry



Hexastore

Places Jerry been to?

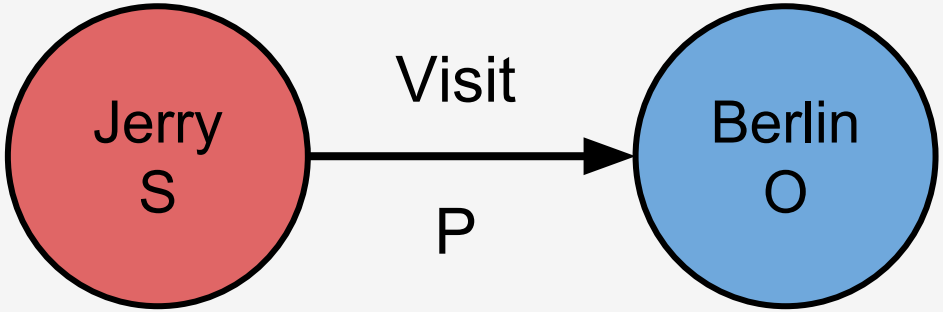
SPO:Jerry:Visit:*

Who visited Berlin?

OPS:Berlin:Visit:*

Who travels and to where?

PSO:Visit:*



Query language

```
MATCH (Jerry:'Jerry Seinfeld')-[friend]->(F)-[visit]->(country)

WHERE (F.age >= Jerry.age AND country.continent = 'Europe')

RETURN F.name, count(country.name) AS countriesVisited

ORDER BY countriesVisited, F.age DESC

LIMIT 2
```

Query language

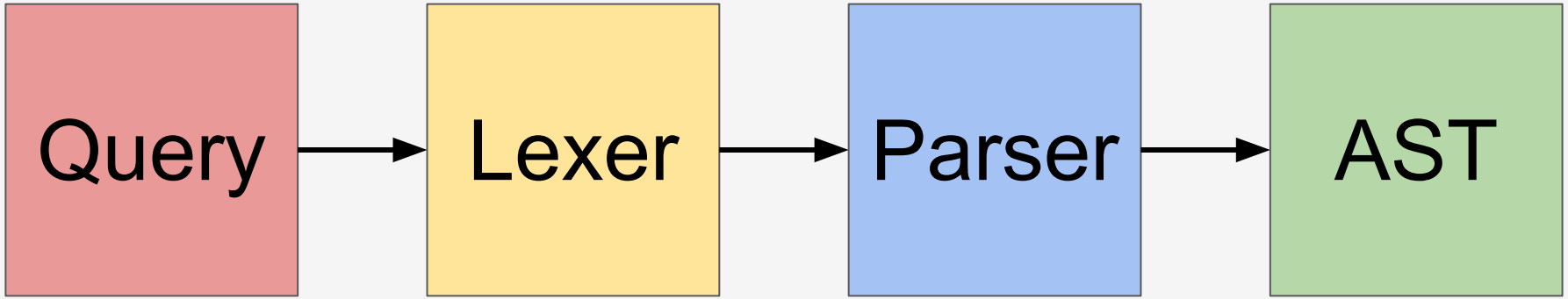
Tokenizer - Lex

Parser - Lemon, SQLite LALR(1) parser generator for C

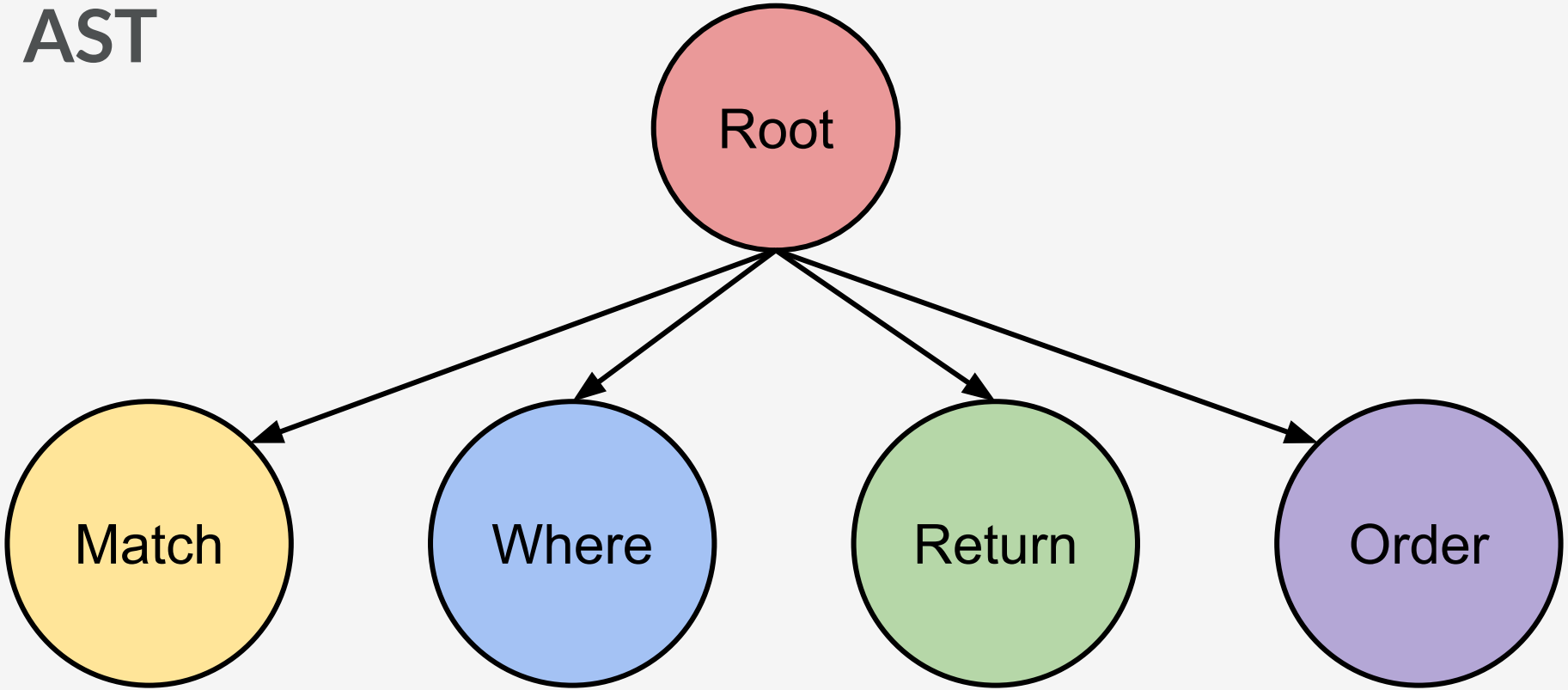
End to end

```
MATCH (Jerry:'Jerry Seinfeld')-[friend]->(F)-[visit]->(Country)

WHERE F.age >= 50 AND Country.continent = "Europe"

RETURN F.name, F.age, Country.name

ORDER BY F.age DESC

LIMIT 5
```

End to end

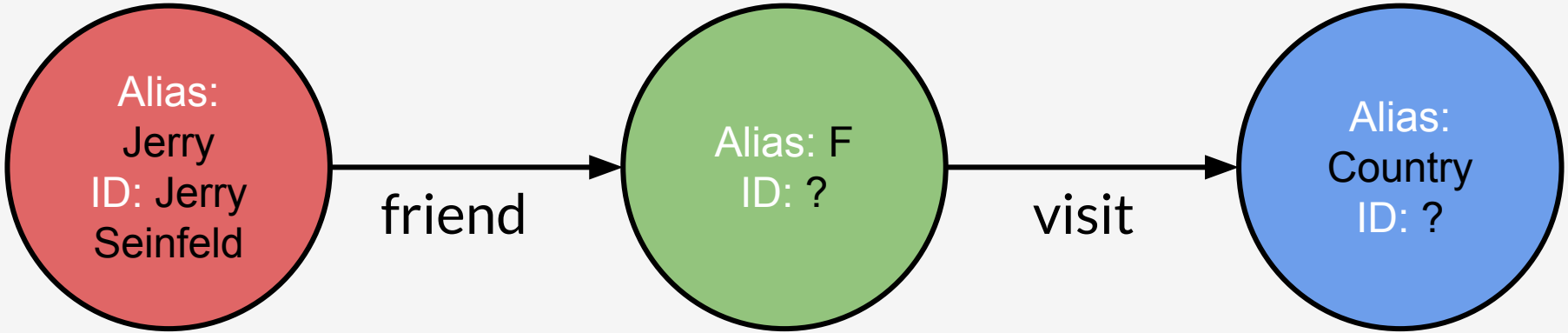


End to end AST

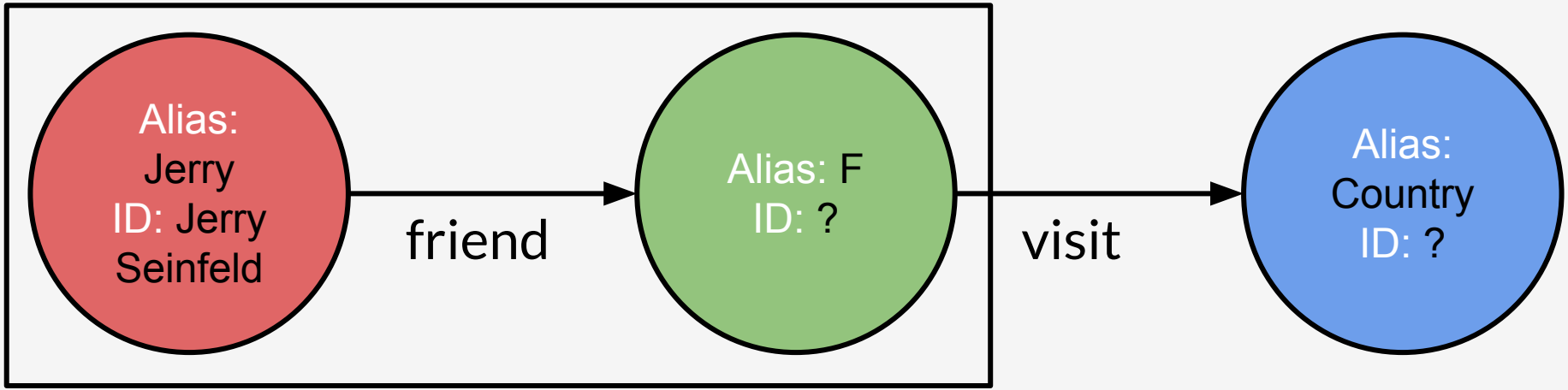


End to end

MATCH (Jerry:"Jerry Seinfeld")-[friend]->(F)-[visit]->(Country)

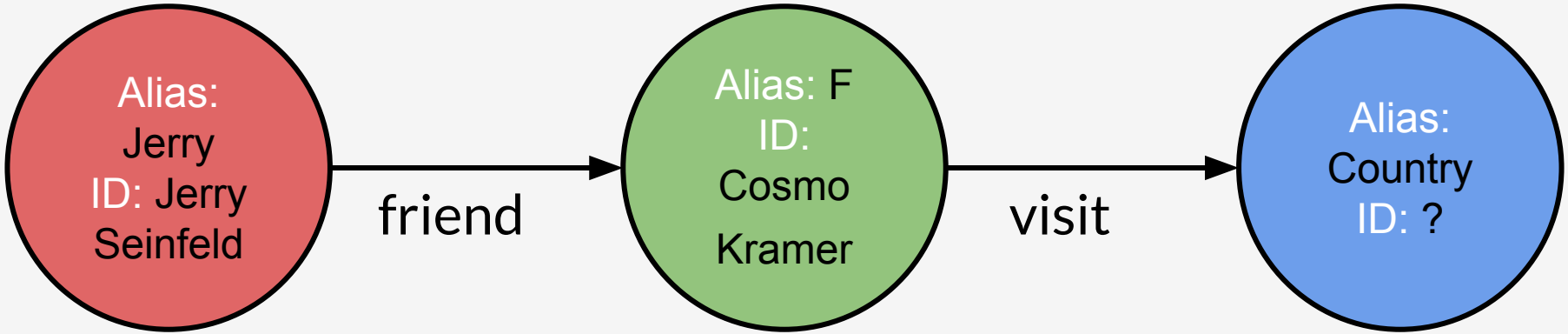


End to end



SPO:Jerry Seinfeld:friend:*
SPO:Jerry Seinfeld:friend:Cosmo Kramer
SPO:Jerry Seinfeld:friend:George Costanza

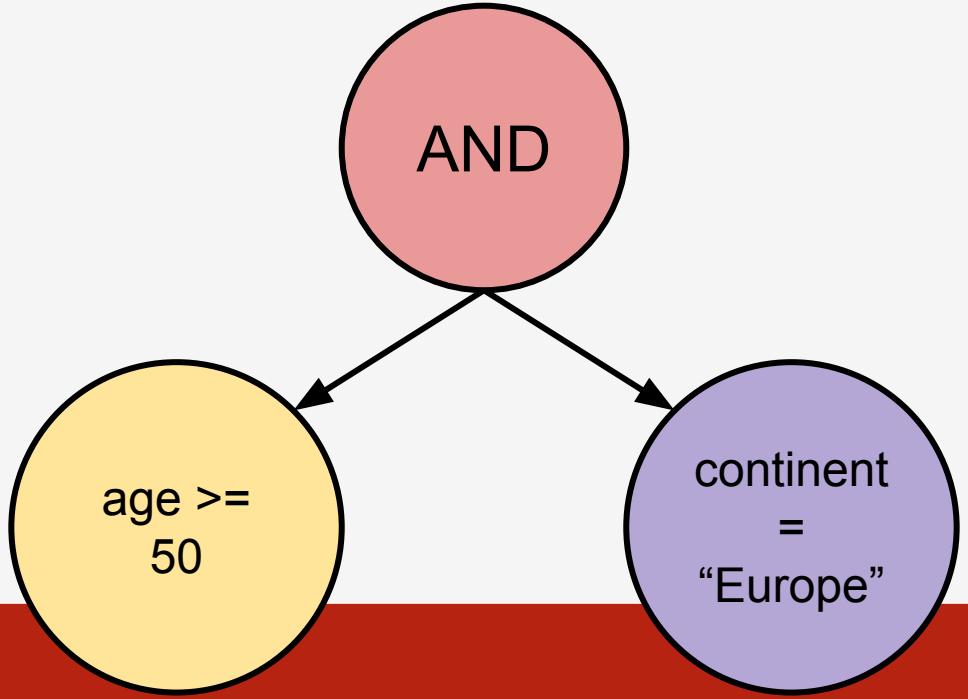
End to end


Alias:
Jerry
ID: Jerry
Seinfeld
friend
Alias: F
ID:
Cosmo
Kramer
visit
Alias:
Country
ID: ?

End to end

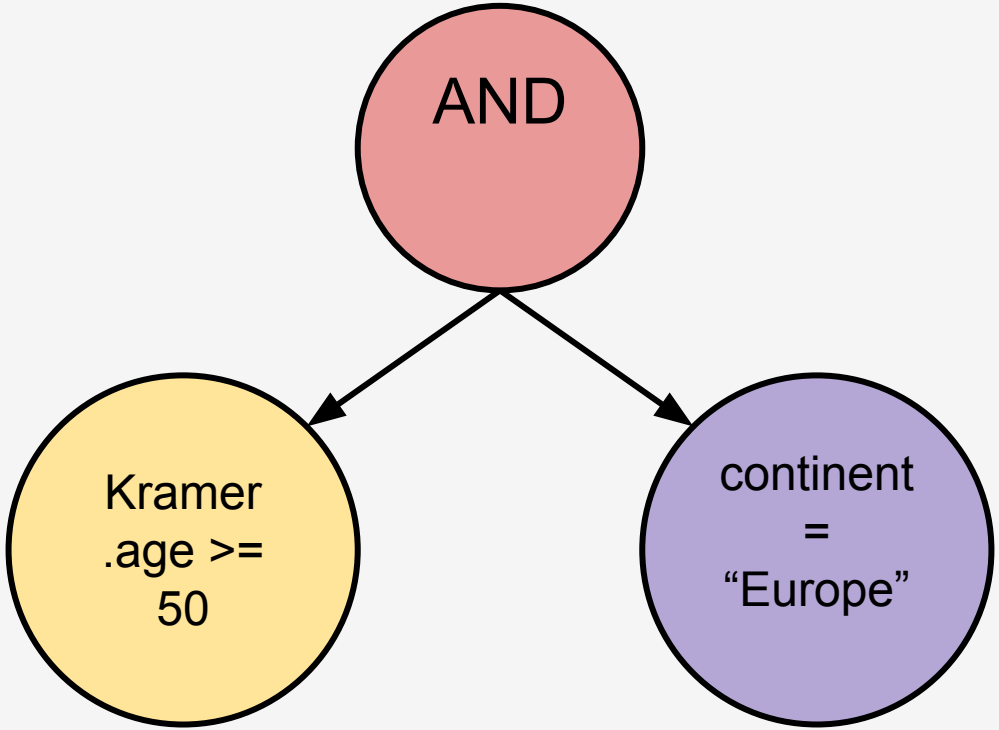
WHERE F.age >= 50 AND Country.continent = “Europe”

Filter tree



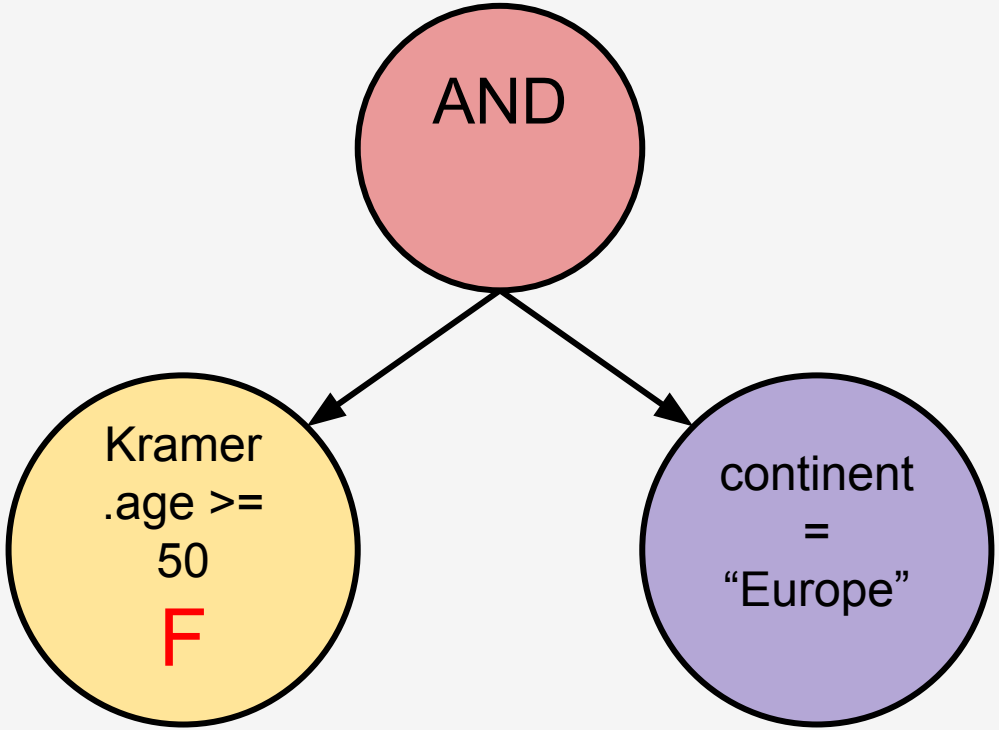
End to end

Cosmo Kramer: {
    Name: ‘Cosmo Kramer’,
    Age: 48
}



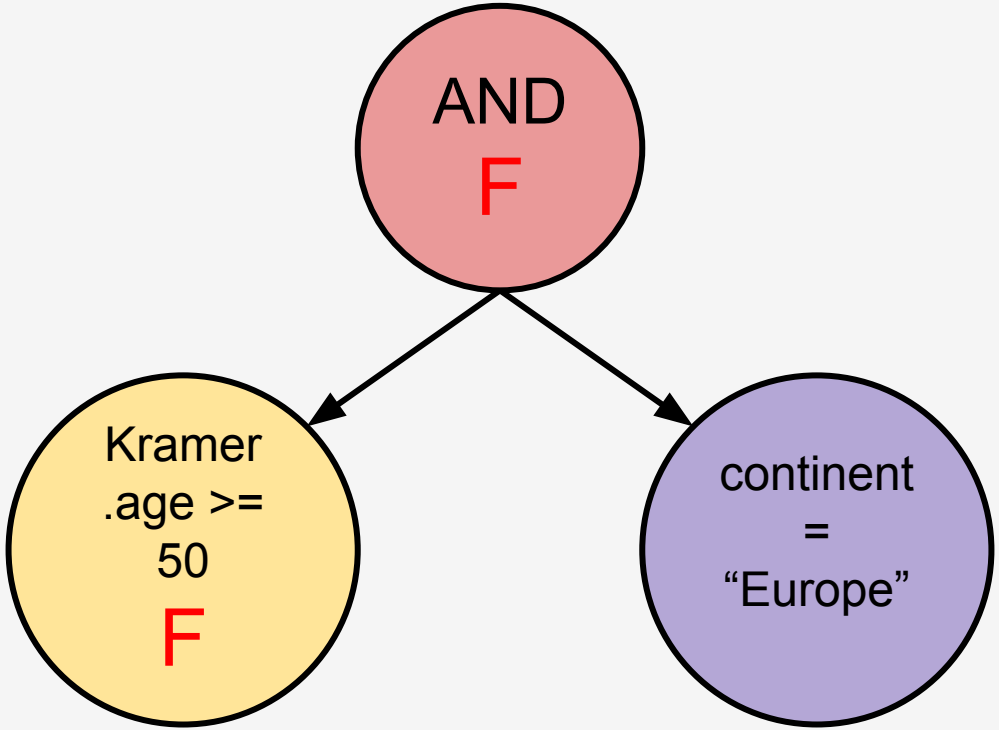
End to end

Cosmo Kramer: {
    Name: ‘Cosmo Kramer’,
    Age: 48
}

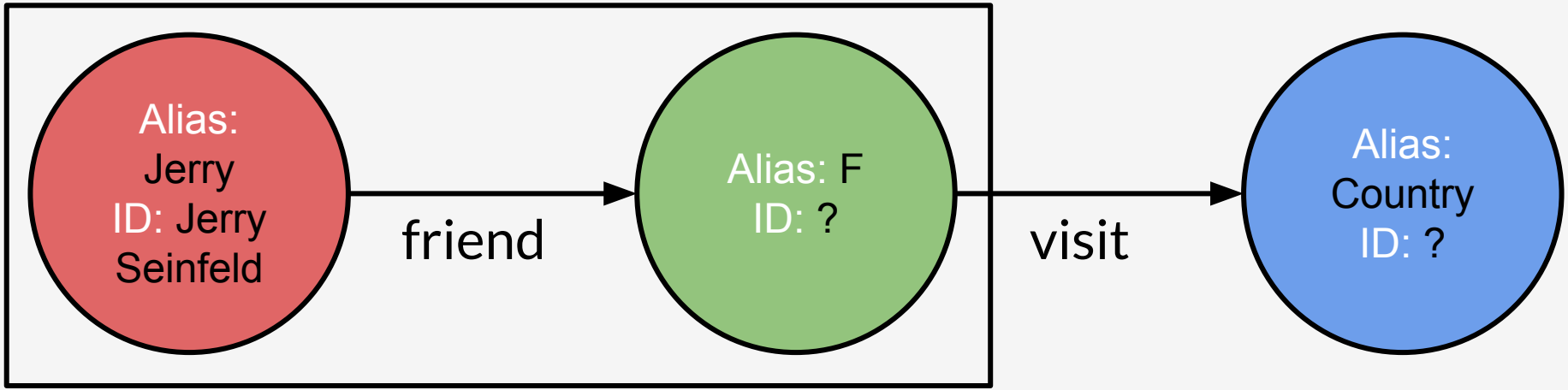


End to end

Cosmo Kramer: {
    Name: ‘Cosmo Kramer’,
    Age: 48
}



End to end

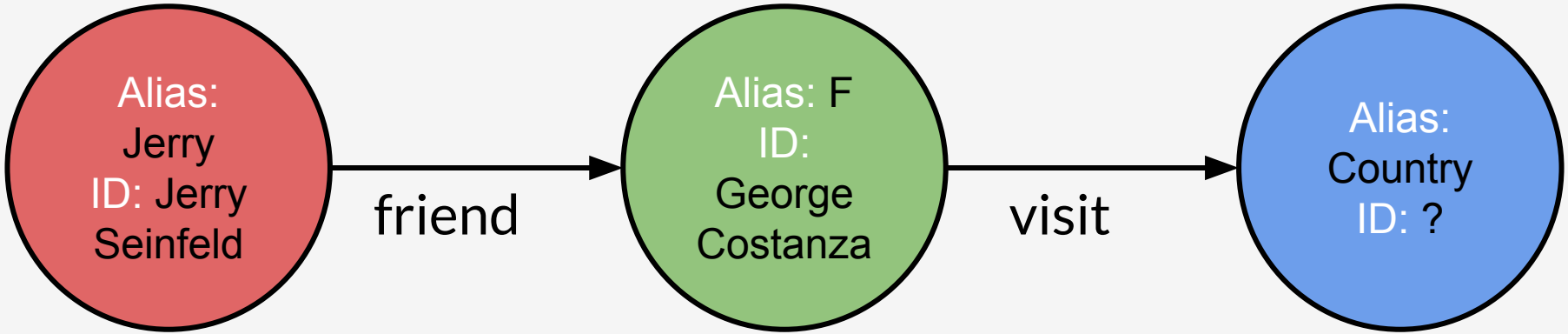


SPO:Jerry Seinfeld:friend:*

~~SPO:Jerry Seinfeld:friend:Cosmo Kramer~~

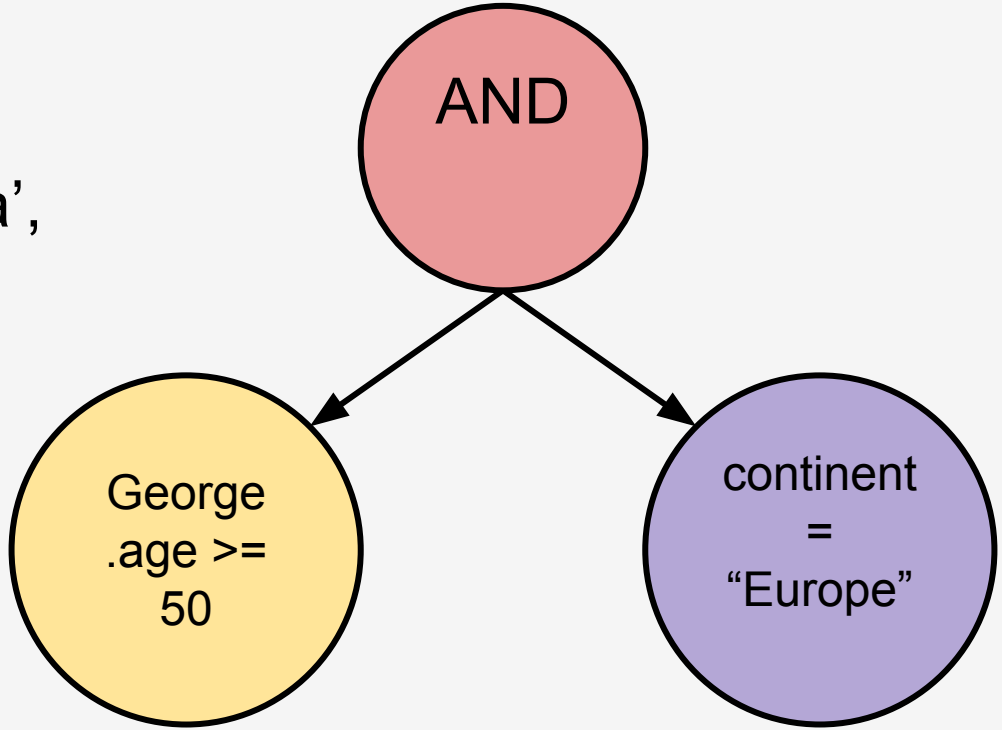
SPO:Jerry Seinfeld:friend:George Costanza

End to end


Alias:
Jerry
ID: Jerry
Seinfeld
friend
Alias: F
ID:
George
Costanza
visit
Alias:
Country
ID: ?

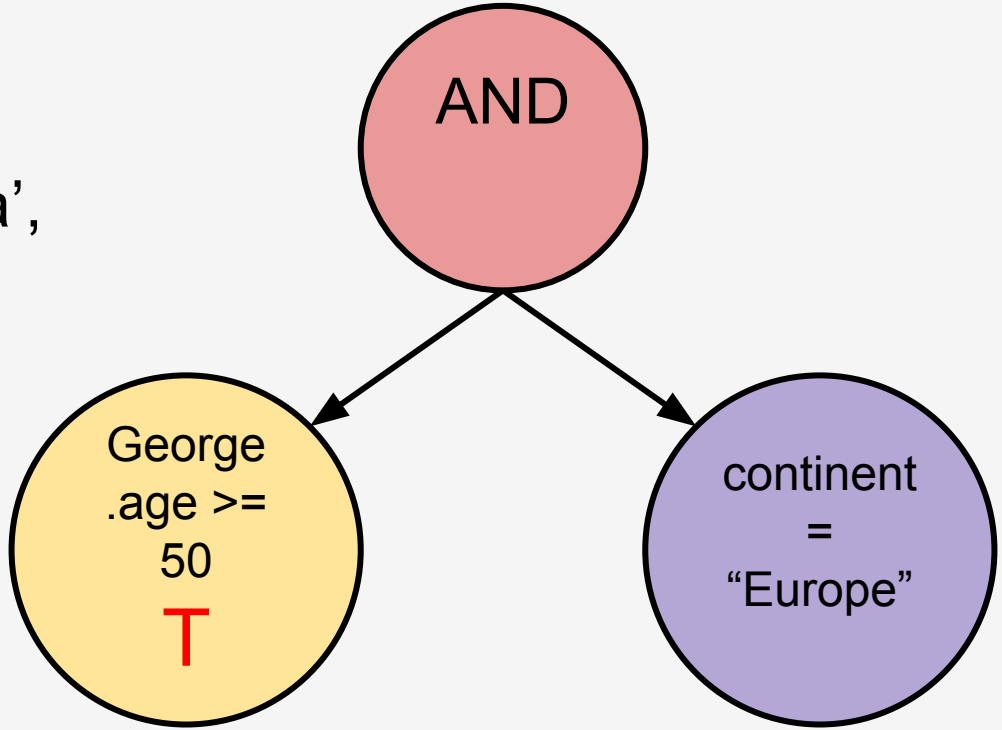
End to end

George Costanza:{
    Name: ‘George Costanza’,
    Age: 52
}



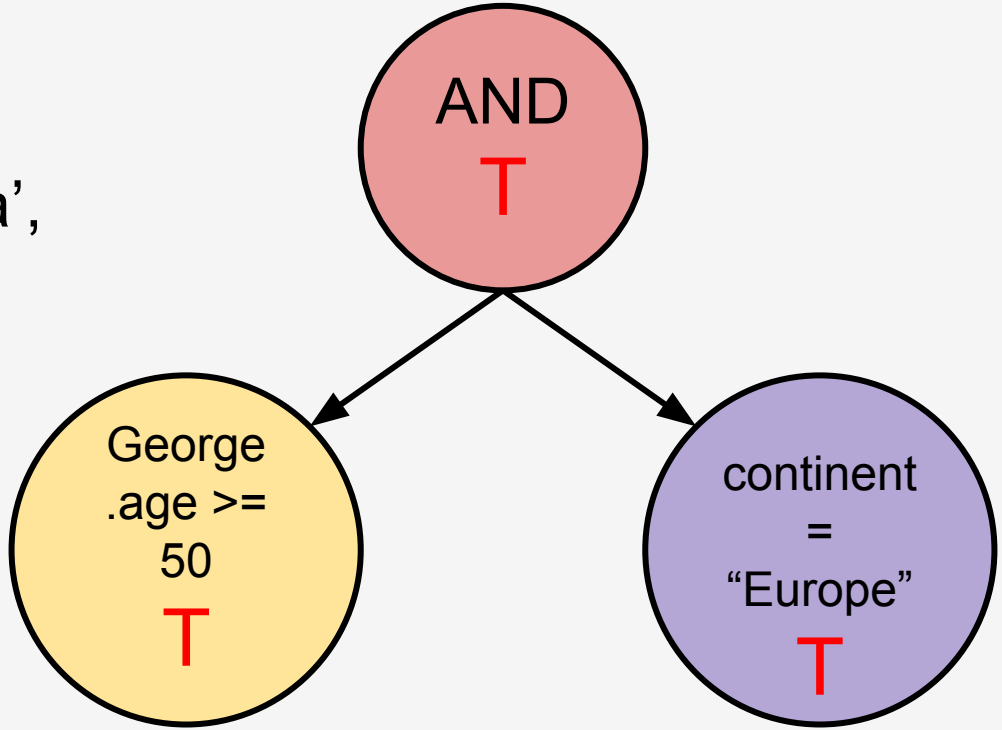
End to end

George Costanza:{
    Name: ‘George Costanza’,
    Age: 52
}

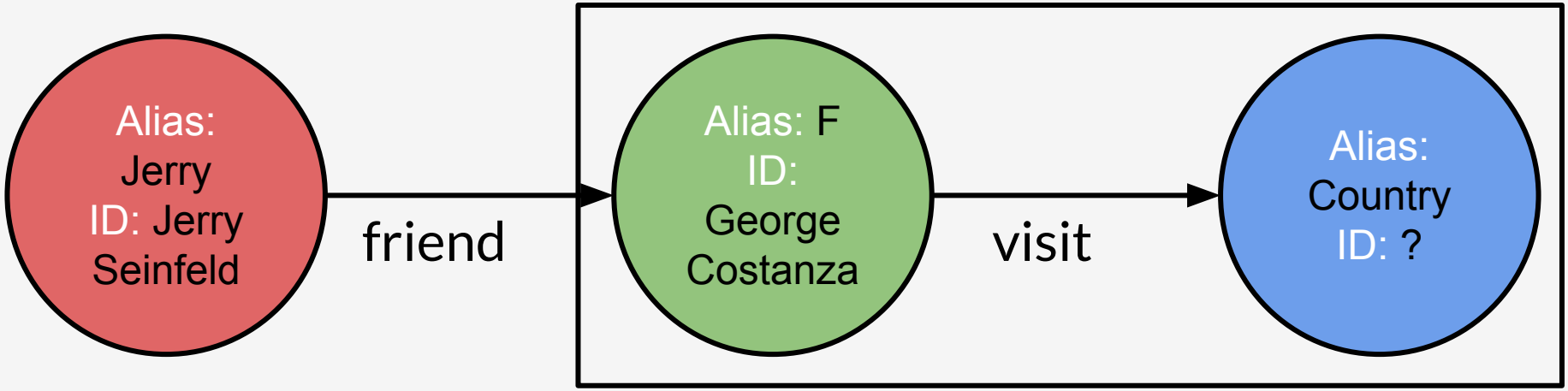


End to end

George Costanza:{
    Name: ‘George Costanza’,
    Age: 52
}

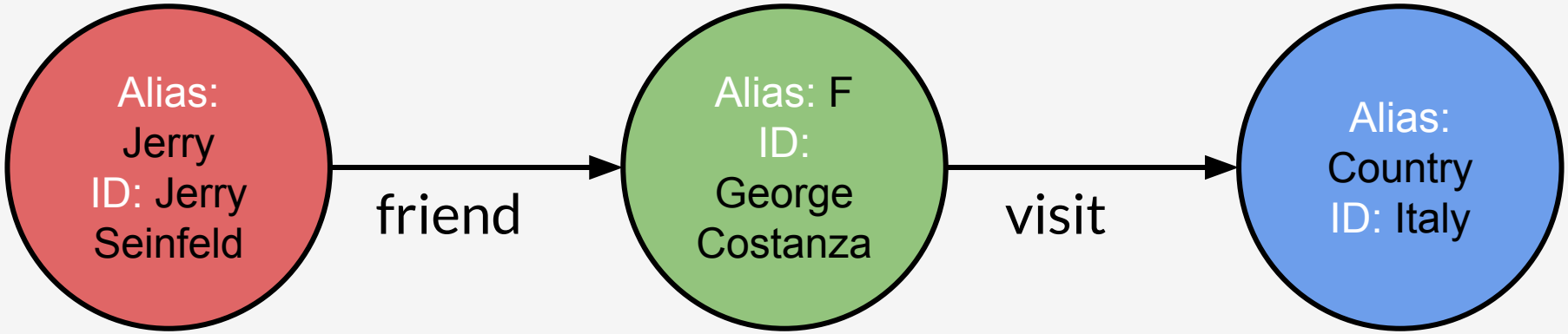


End to end



SPO:George Costanza:visit:*
SPO:George Costanza:visit:Italy
SPO:George Costanza:visit:Cuba

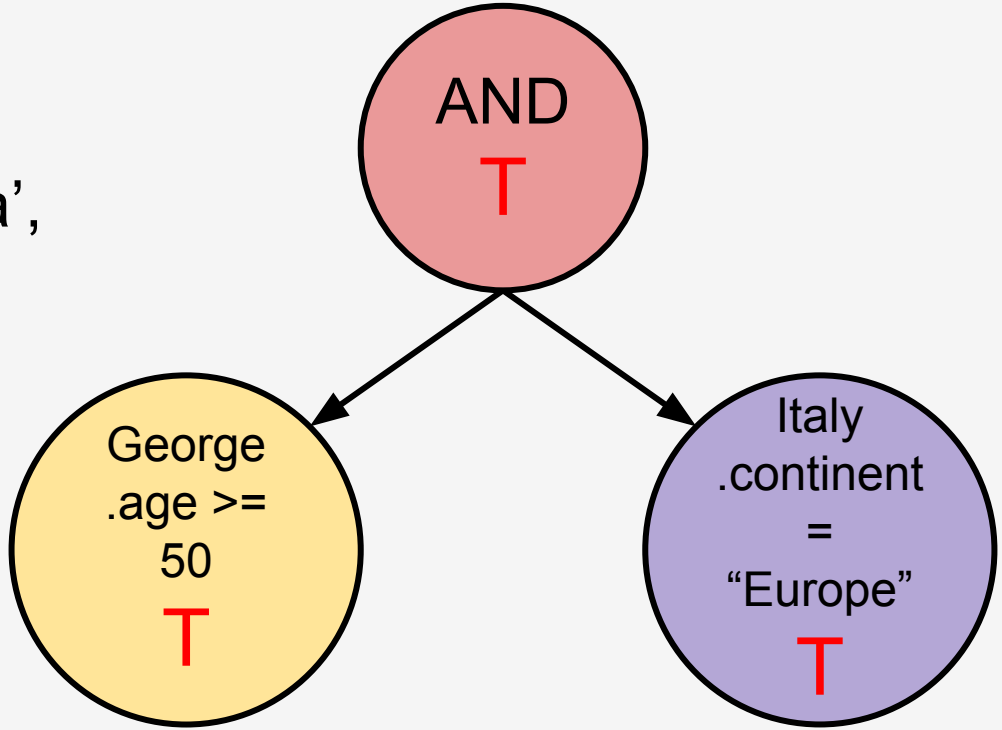
End to end


Alias:
Jerry
ID: Jerry
Seinfeld
friend
Alias: F
ID:
George
Costanza
visit
Alias:
Country
ID: Italy

End to end

George Costanza:{
Name: ‘George Costanza’,
Age: 52 }

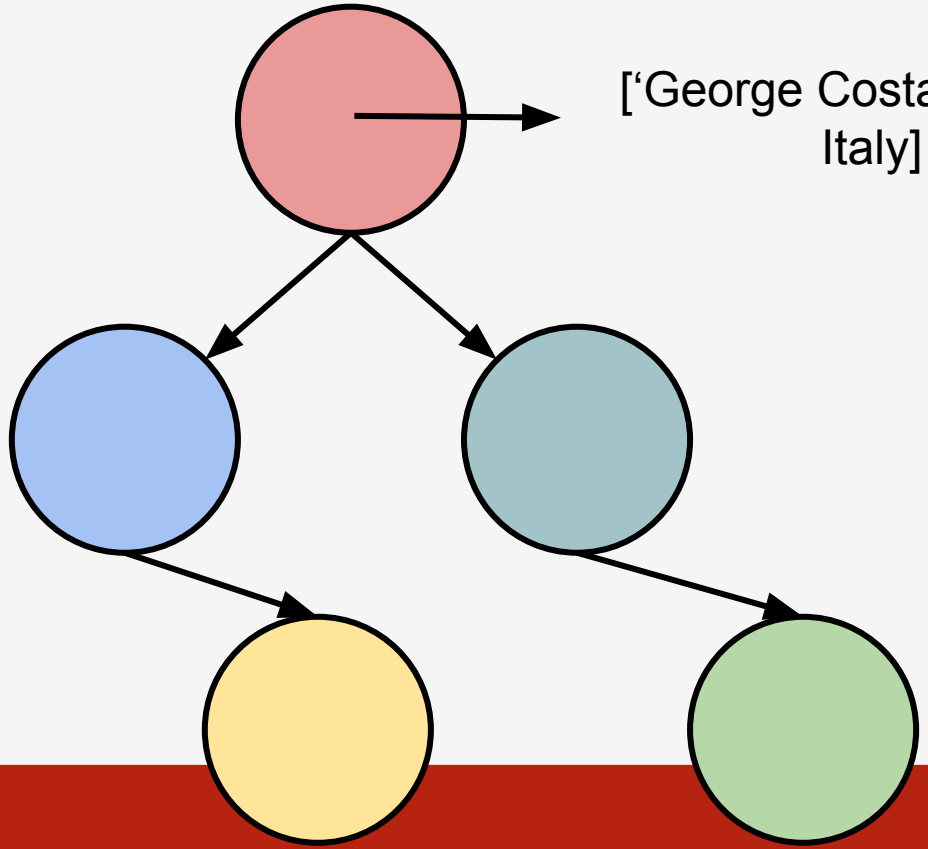
Italy: {
Continent: ‘Europe’,
Population: 1000,
Name: ‘Italy’ }



End to end

TOP K heap

RETURN F.name,
F.age, Country.name
ORDER BY F.age
DESC
LIMIT 5



Features

Multi hop, multi entry point
(A)-[R1]->(C)<-[R2]-(B)
(Nicolas:'Nicolas Cage')-[act]->(Movie)<-[act]-(Actor)

Aggregations, Group bys
RETURN F.gender, AVG(F.age) AS average_age

Order bys, Distinct

Benchmark

150K inserts per second

15K simple queries per second

There’s still work to be done

- Single node query: MATCH (A) RETURN A
- OPTIONAL MATCH
- WITH, SKIP, UNION
- Curly brackets filters (john {name: ‘John’})
- In place evaluations WHERE Me.age > F.age + X
- Shortest path between nodes (A)-[*]-(B)
- A number of aggregation functions: stdev, precentileCount

Roadmap

- Hexastore sorted-set -> Trie
- Indexed entities
- Single node query
- Python lib

Contribute/Contact

https://github.com/swilly22/redis-module-graph

@roilipman