QCon

Please evaluate my talk via the mobile app!

How Shutl Delivers Even Faster Using Neo4j

Sam Phillips and Volker Pacher
@samsworldofno @vpacher

Sam Phillips

Volker Pacher

Graphs at Shutl

Graphs at Shutl

- Graph databases are awesome

Graphs at Shutl

- Graph databases are awesome
- We've seen lots of the talks about modelling

Graphs at Shutl

- Graph databases are awesome
- We've seen lots of the talks about modelling
- But querying is important too

Graphs at Shutl

- Graph databases are awesome
- We've seen lots of the talks about modelling
- But querying is important too
- So let's talk about querying too!

Show of hands

Show of hands

- Who has used graph databases before?

Show of hands

- Who has used graph databases before?
- Who has used Neo4j before?

Shutl

Shutl

ECOMMERCE IS QUICK & CONVENIENT

ECOMMERCE IS QUICK & CONVENIENT

PAYPAL FOR AWESOME DELIVERY

PAYPAL FOR AWESOME DELIVERY

PAYPAL FOR AWESOME DELIVERY

Branded, super quick delivery that people trust, embedded in merchant websites

HUB & SPOKE

HUB & SPOKE

HUB & SPOKE

Only cost effective means to deliver 10+ miles but slow and unpredictable

HUB & SPOKE

Only cost effective means to deliver 10+ miles but slow and unpredictable

HUB & SPOKE

Only cost effective means to deliver 10+ miles but slow and unpredictable

POINT TO POINT

HUB & SPOKE

Only cost effective means to deliver 10+ miles but slow and unpredictable

POINT TO POINT

Fast and predictable but cost prohibitive over longer distances

HUB & SPOKE

97% Courier, Express & Parcel Market

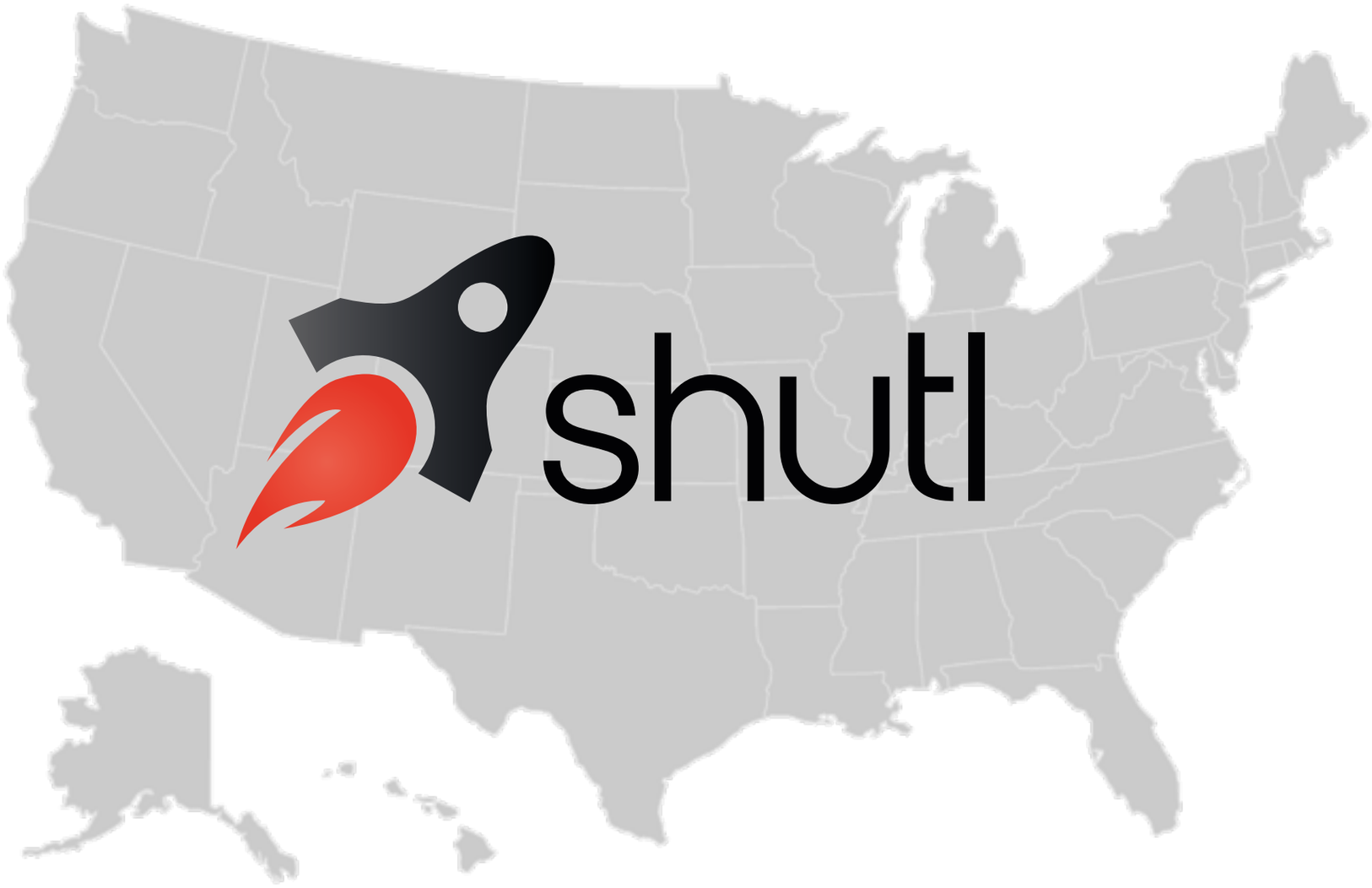
POINT TO POINT

3% Courier, Express & Parcel Market

POINT TO POINT

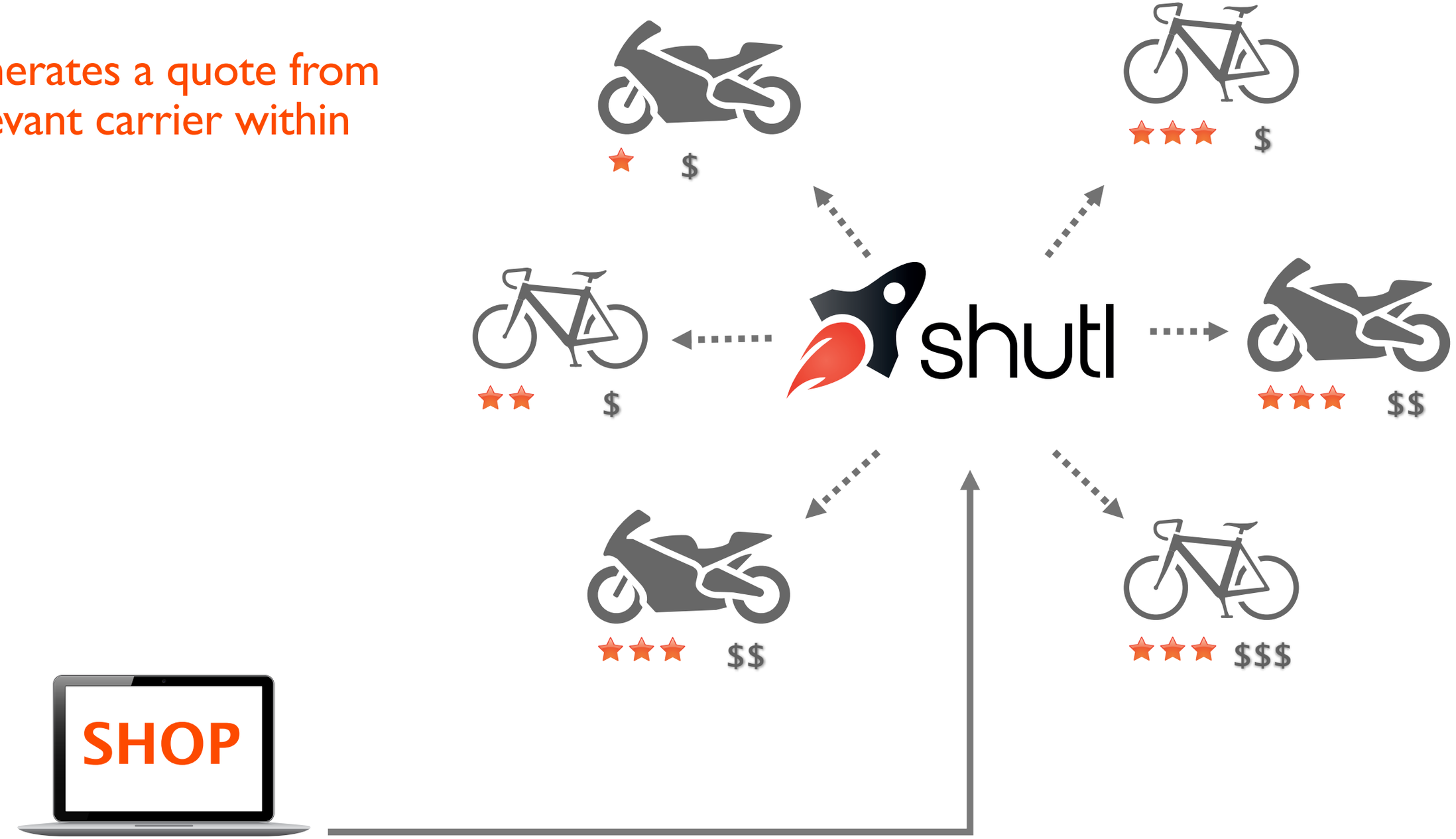
3% Courier, Express & Parcel Market

POINT TO POINT



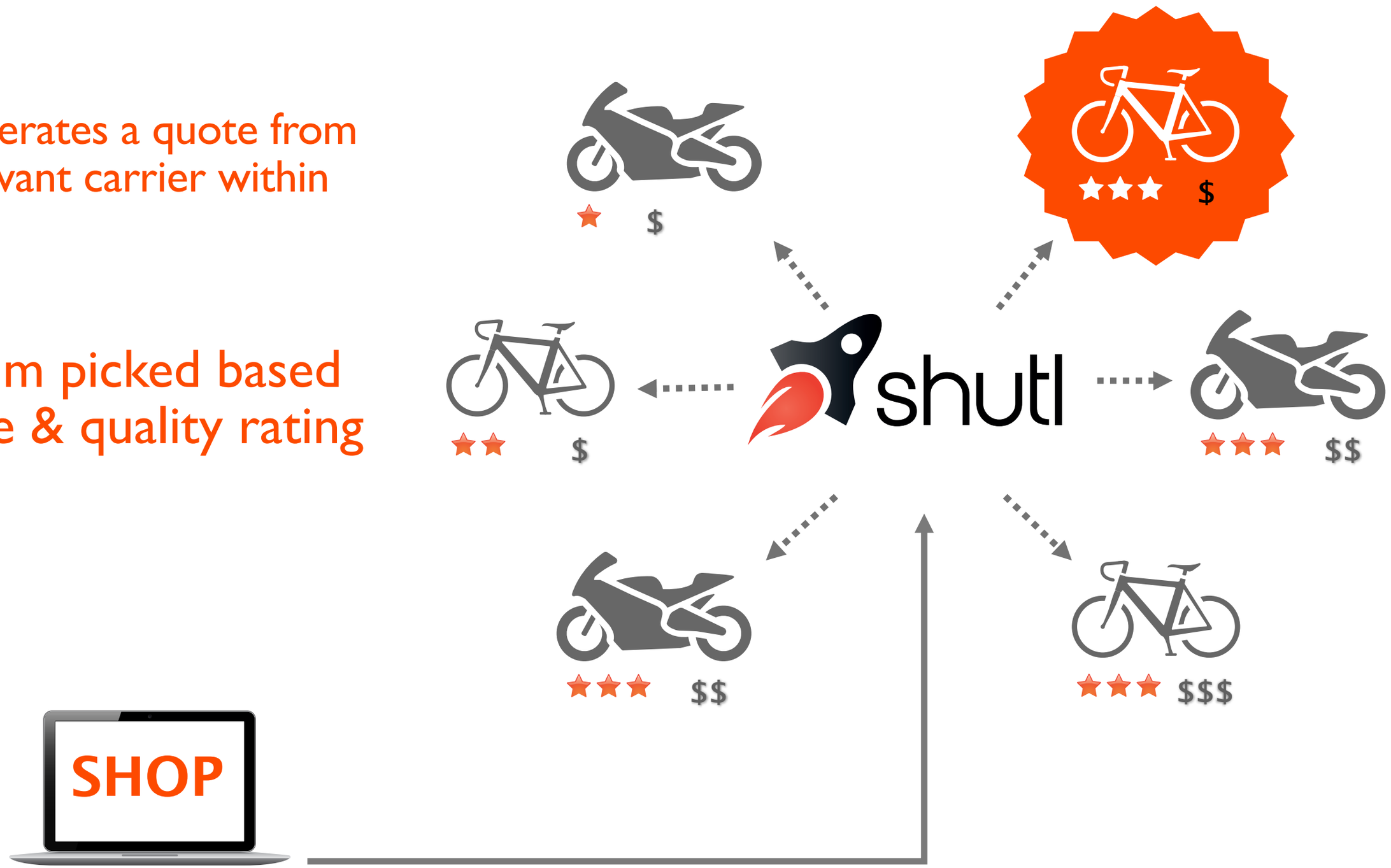

SHOP

Shutl generates a quote from each relevant carrier within platform



Shutl generates a quote from each relevant carrier within platform

Optimum picked based on price & quality rating



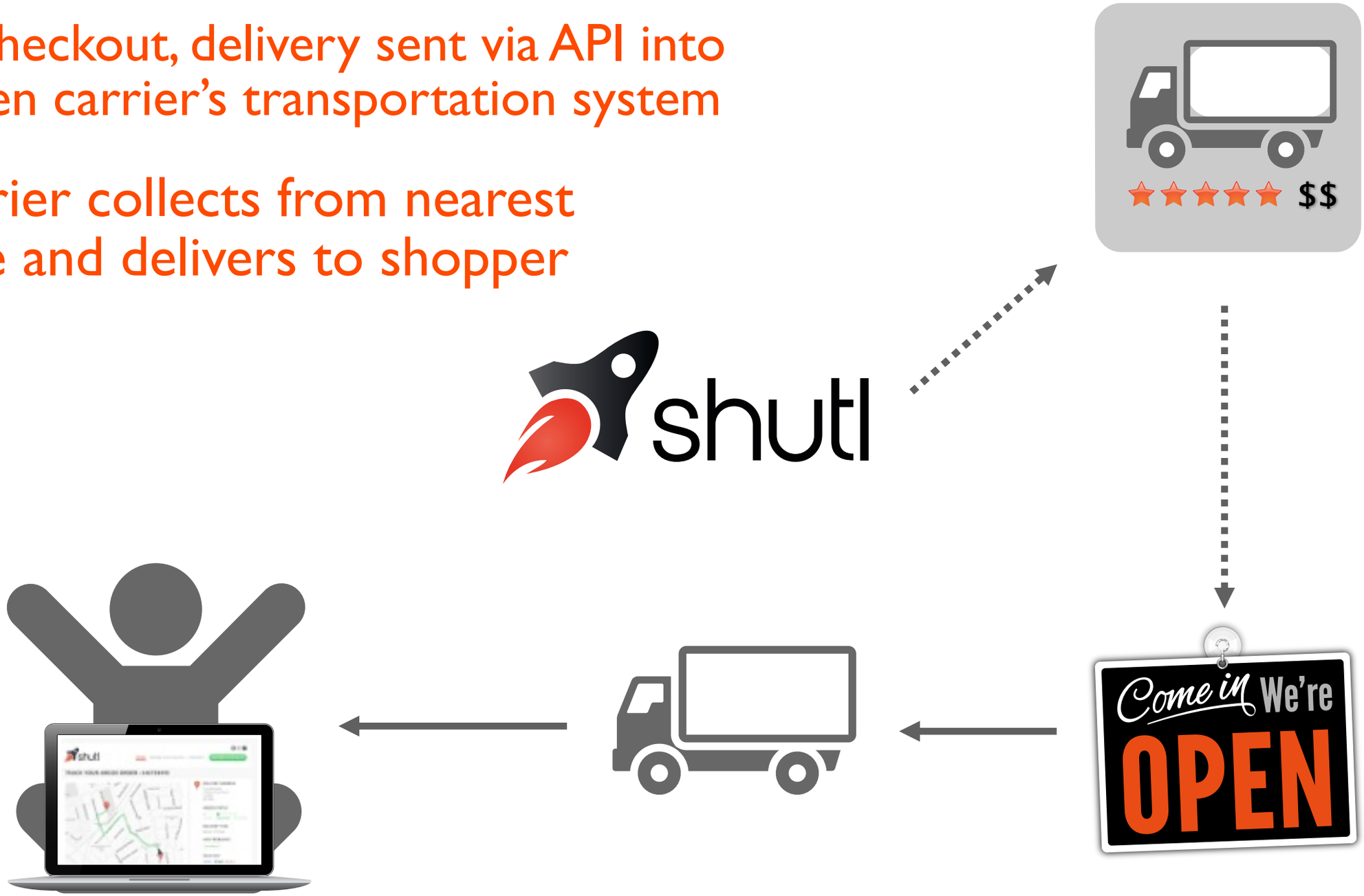

SHOP

On checkout, delivery sent via API into chosen carrier's transportation system

shutl

On checkout, delivery sent via API into chosen carrier's transportation system

Courier collects from nearest store and delivers to shopper



Delivery status updated in
real-time, performance
compared against SLA &
carrier quality rating updated

Better performing carriers
get more deliveries & can
demand higher prices

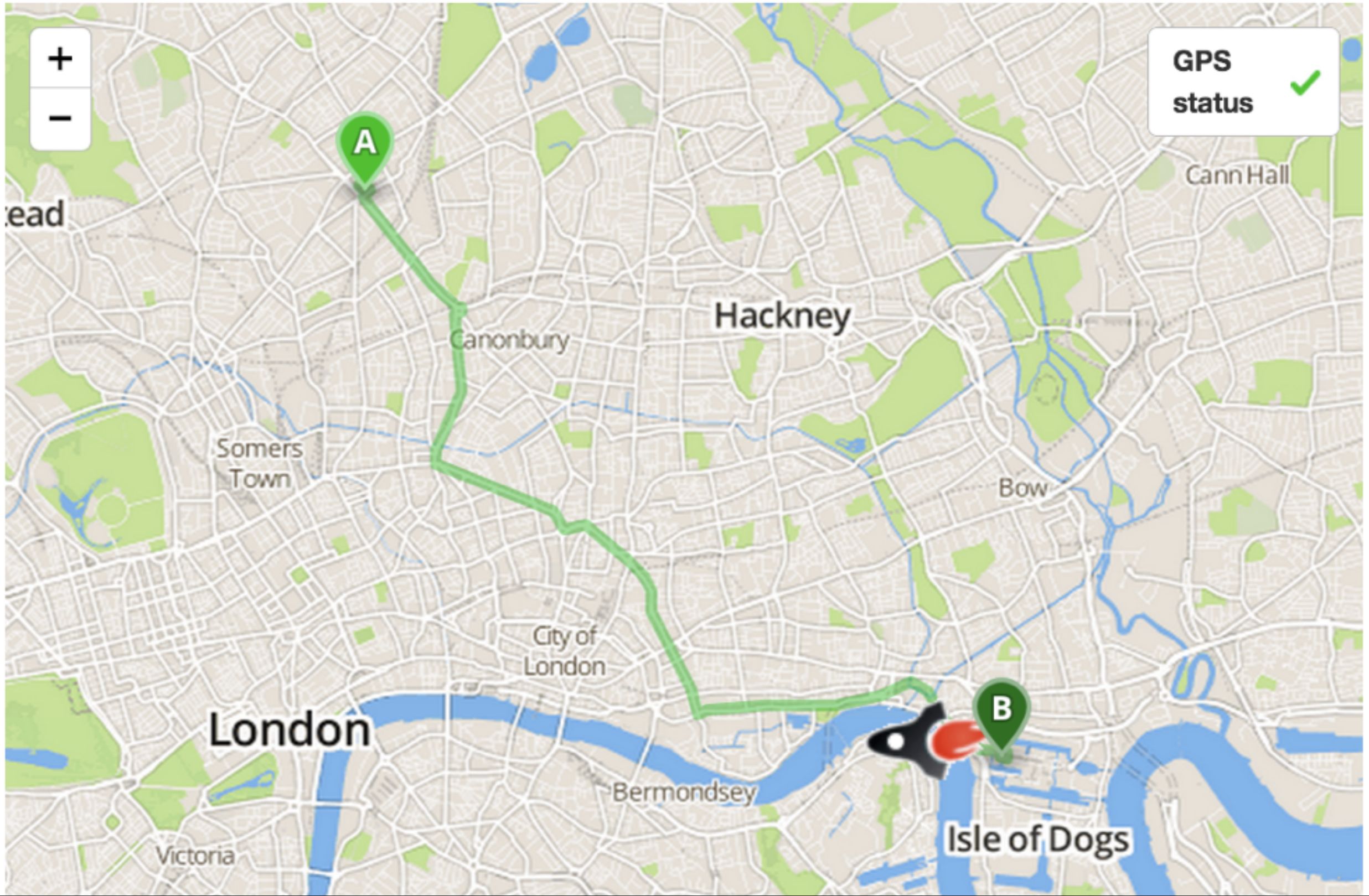
Delivery status updated in real-time, performance compared against SLA & carrier quality rating updated

Better performing carriers get more deliveries & can demand higher prices

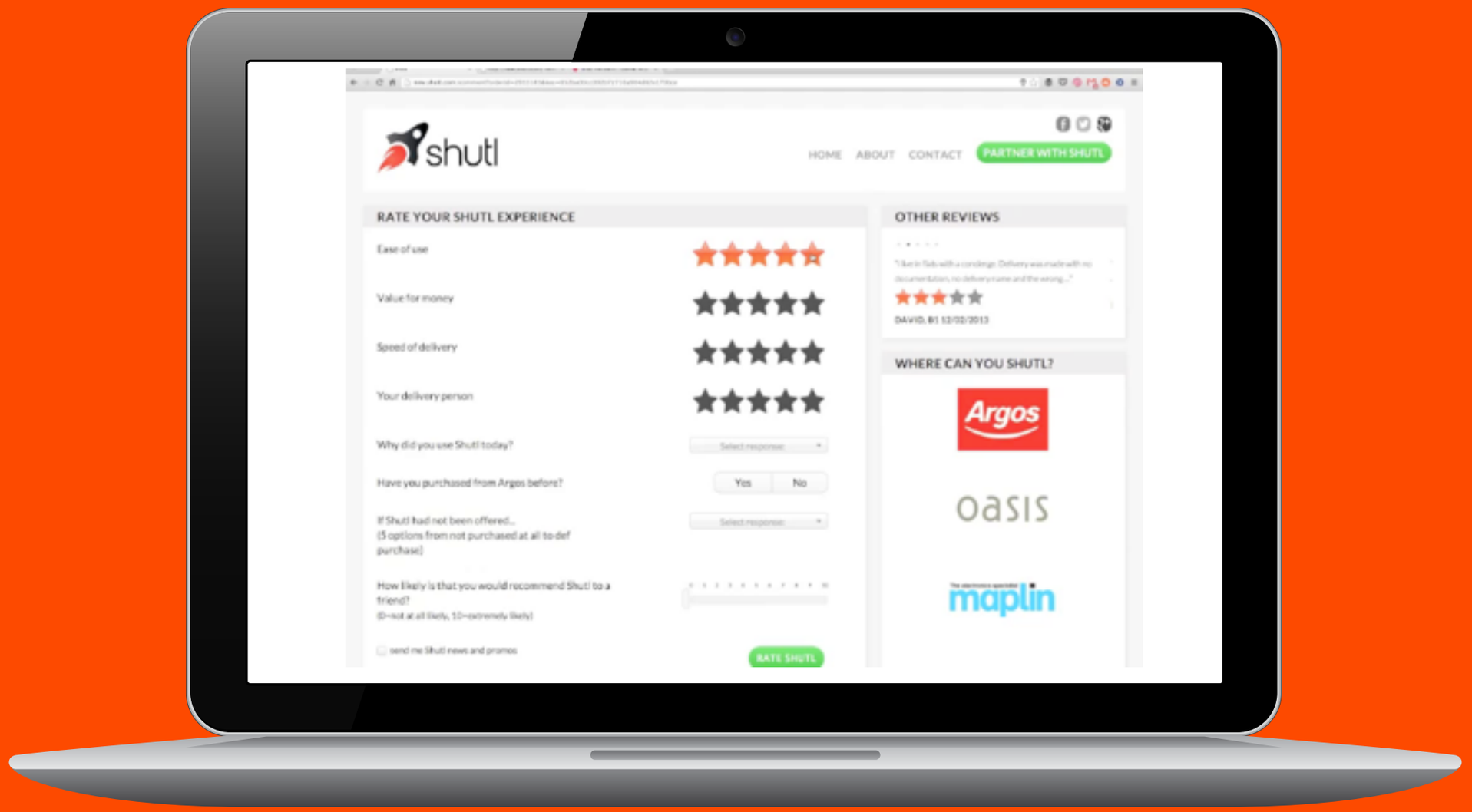
Delivery status updated in real-time, performance compared against SLA & carrier quality rating updated

Better performing carriers get more deliveries & can demand higher prices

Track your order online…

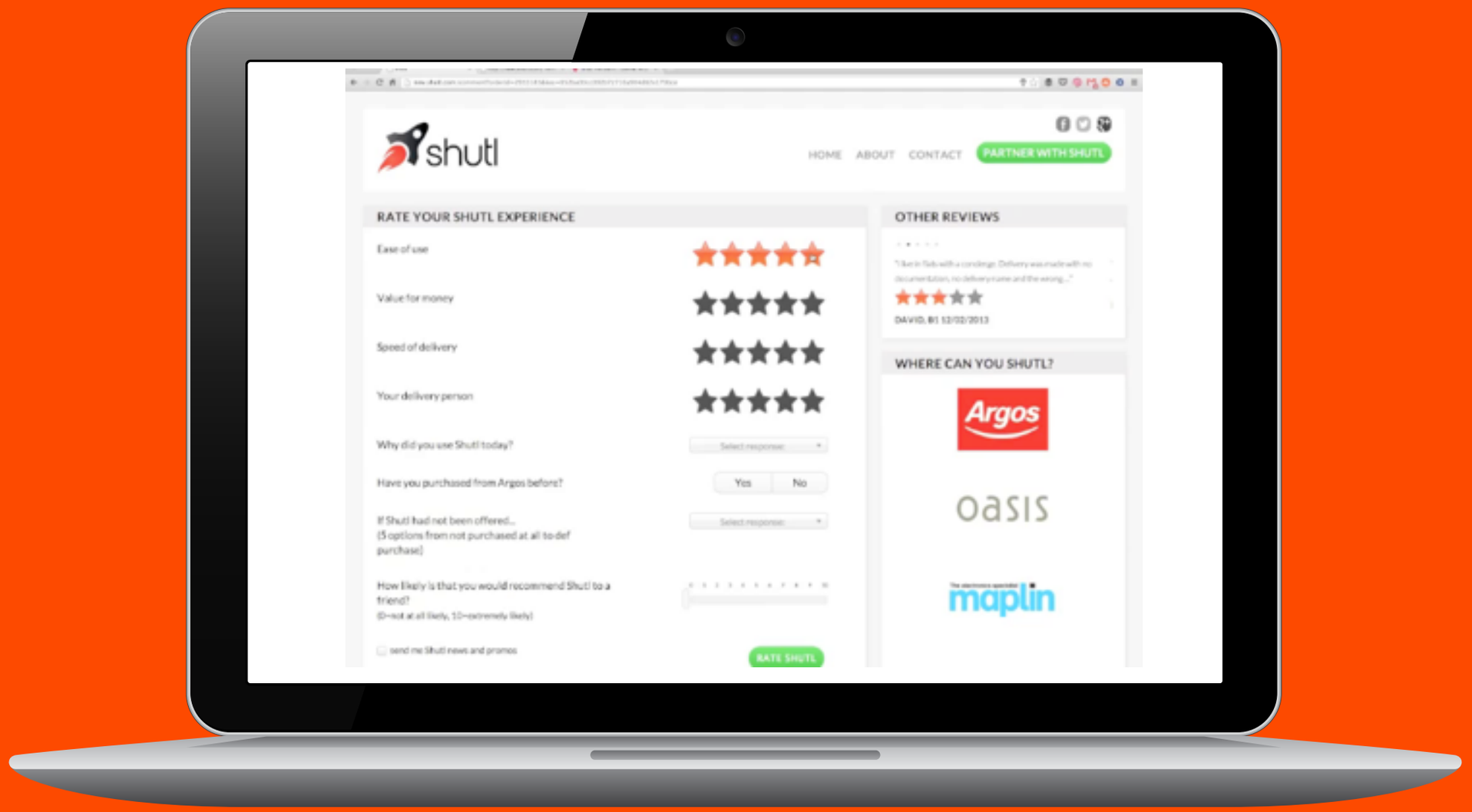


FEEDBACK



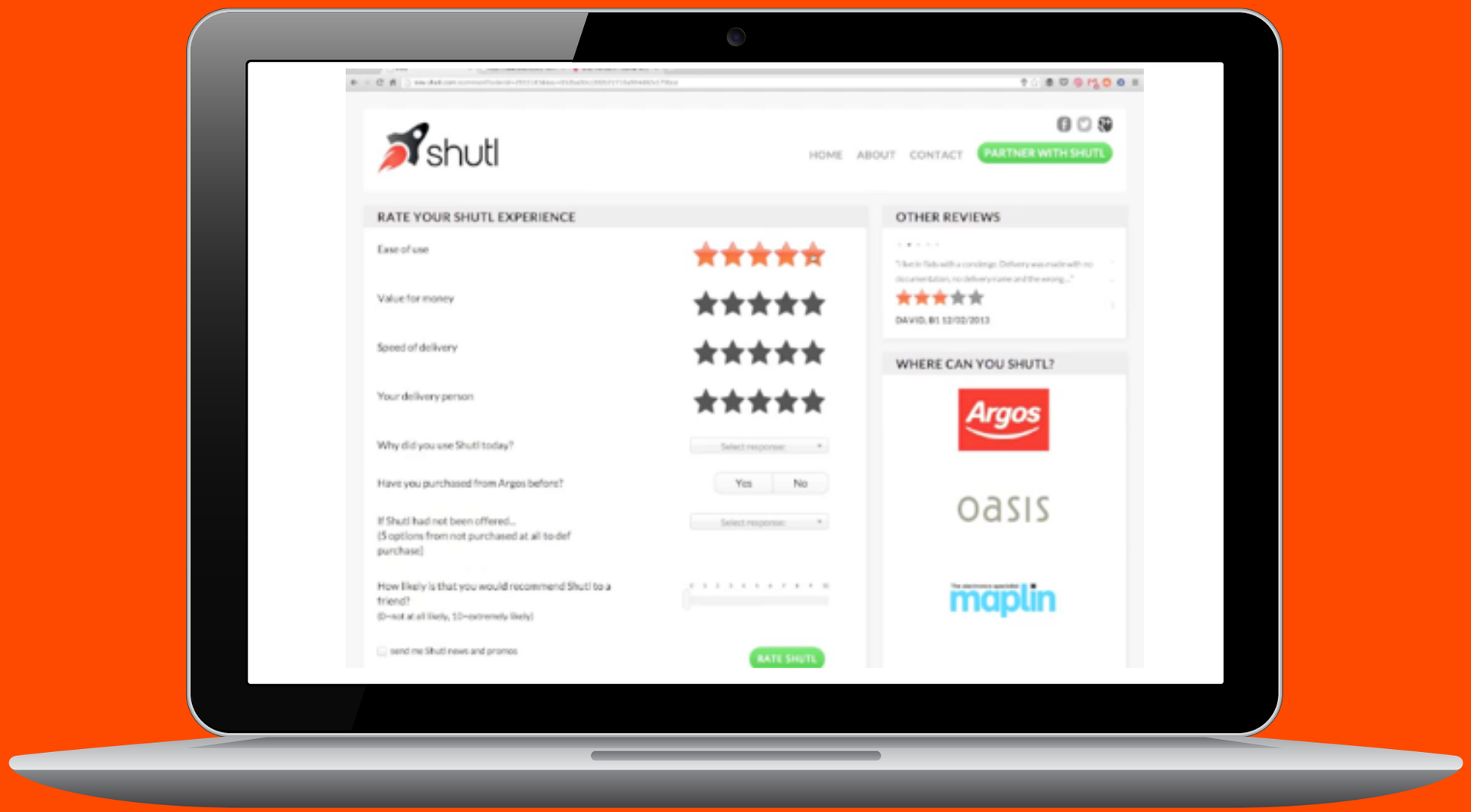
Quality paramount since we are motivated by LTV of shopper

FEEDBACK



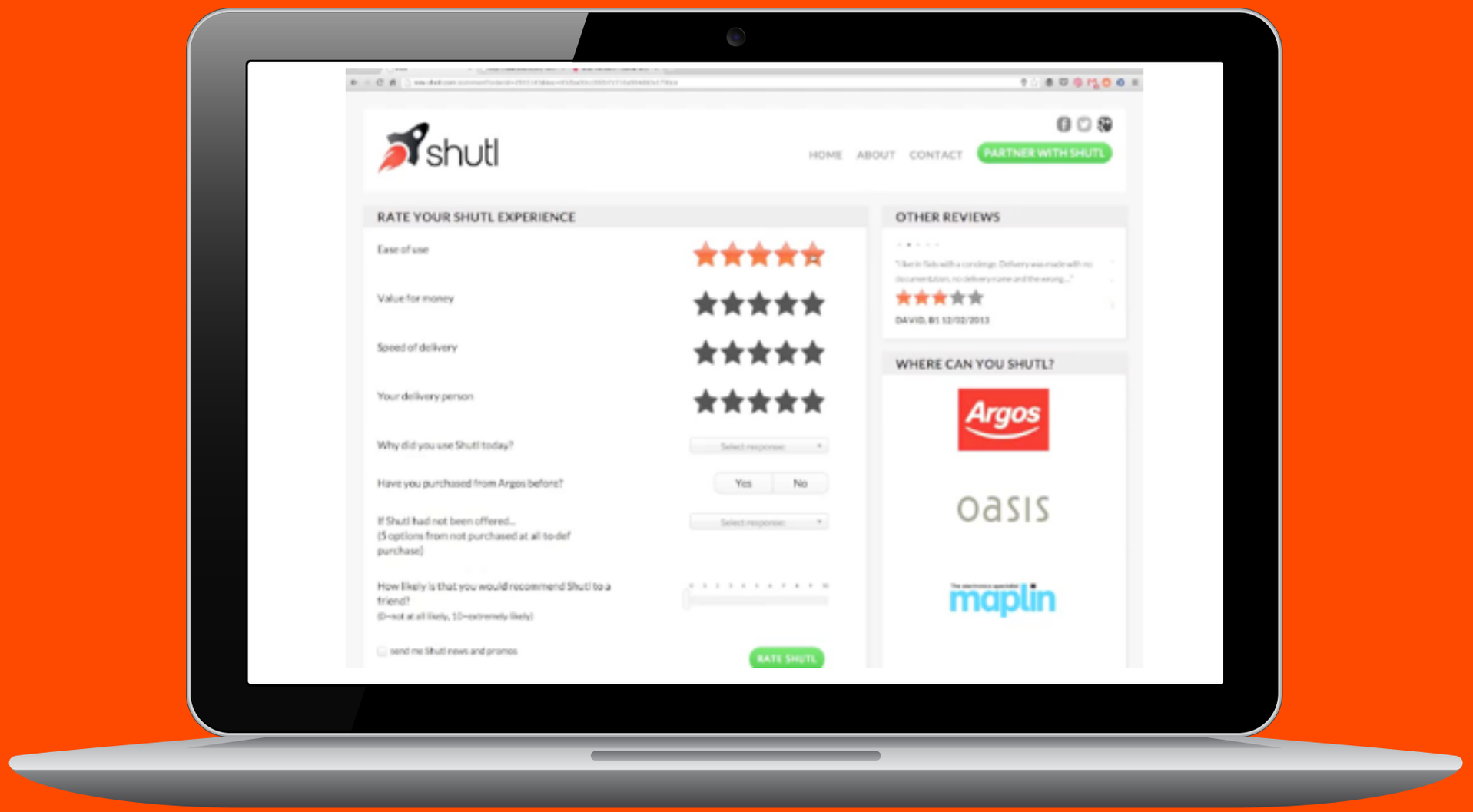
Quality paramount since we are motivated by LTV of shopper

FEEDBACK



Shutl sends feedback email to consumer seconds after they have received delivery asking to rate qualitative aspects of experience

FEEDBACK



Feedback streamed unedited to shutl.com/feedback & facebook

FEEDBACK

1 REVIEW

WILLIAM, LONDON

Order a tv online, two hours late it was in my living room!
Fantastic!

REVIEWED 13:18 ON 06/03/2014

COMMENT

Tweet 0

FEEDBACK

1 REVIEW

STEPHEN, SHEFFIELD

brilliant service from shuti didn't have time to make a cuppa before there was a knock on the door

REVIEWED 11:19 ON 26/02/2014

COMMENT

Tweet 0

FEEDBACK

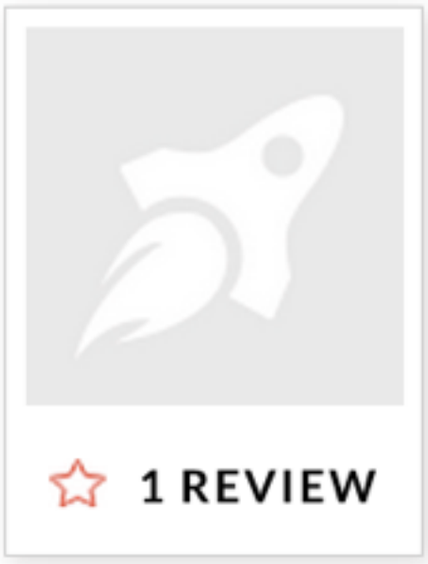
☆ 2 REVIEWS

GUEST, GLASGOW

Absolutely fantastic speed of delivery and the delivery guy was hot also :)

REVIEWED 13:15 ON 03/03/2014

FEEDBACK



☆ 1 REVIEW

JAMES, LONDON

Amazingly fast, got my product when I couldnt leave the house (under house arrest).

REVIEWED 10:53 ON 16/01/2014

Argos

ebay

The electronics specialist maplin

KAREN MILLEN

HOTEL Chocolat.

oasis

coast

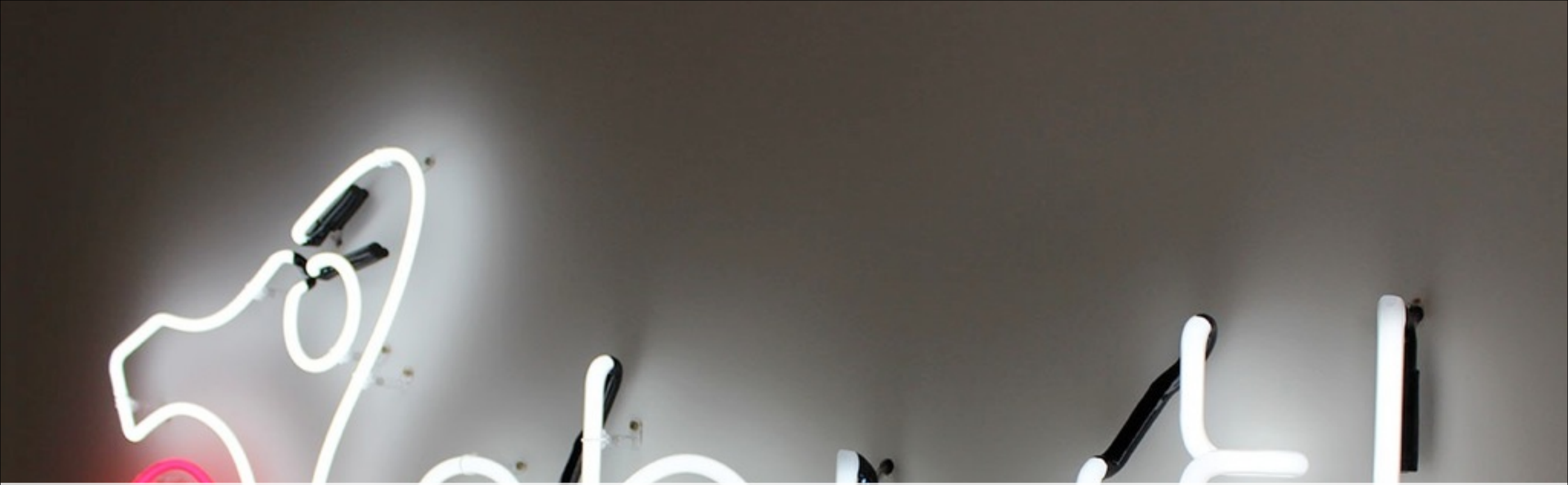
WAREHOUSE

THE PERFUME SHOP

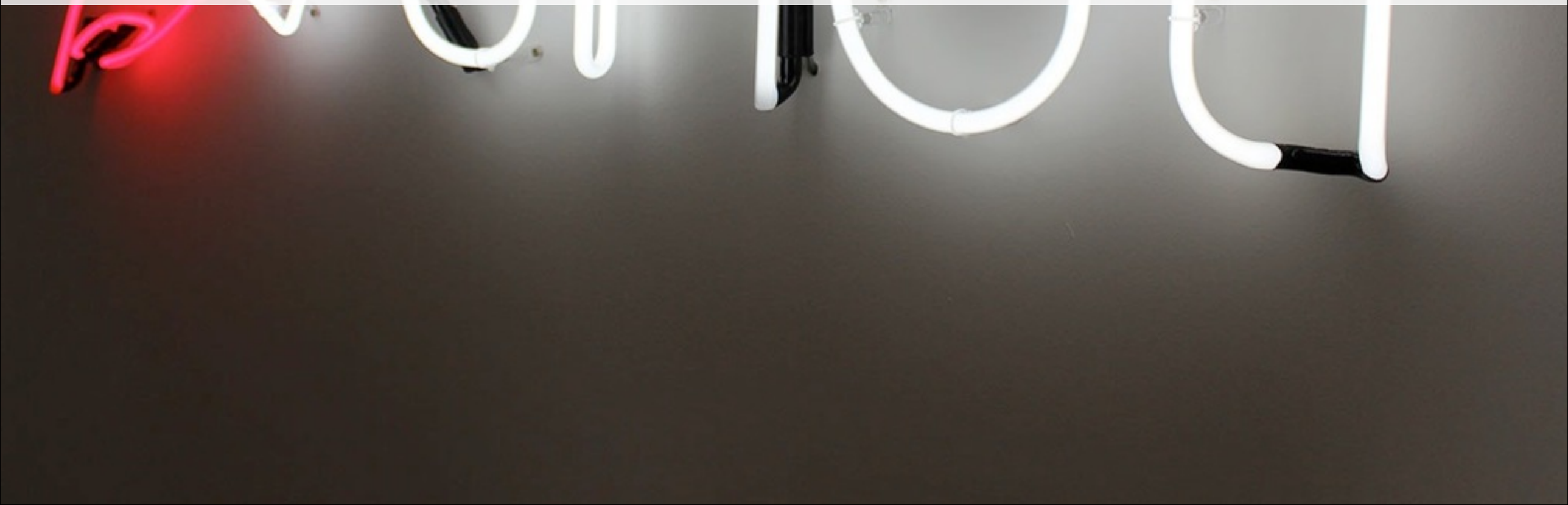
schuh

UGG australia

The Entertainer TheToyShop.com



SHUTL IS NOW AN ebay inc™ COMPANY



Version One

Ruby 1.8, Rails 2.3 and MySQL

Version One

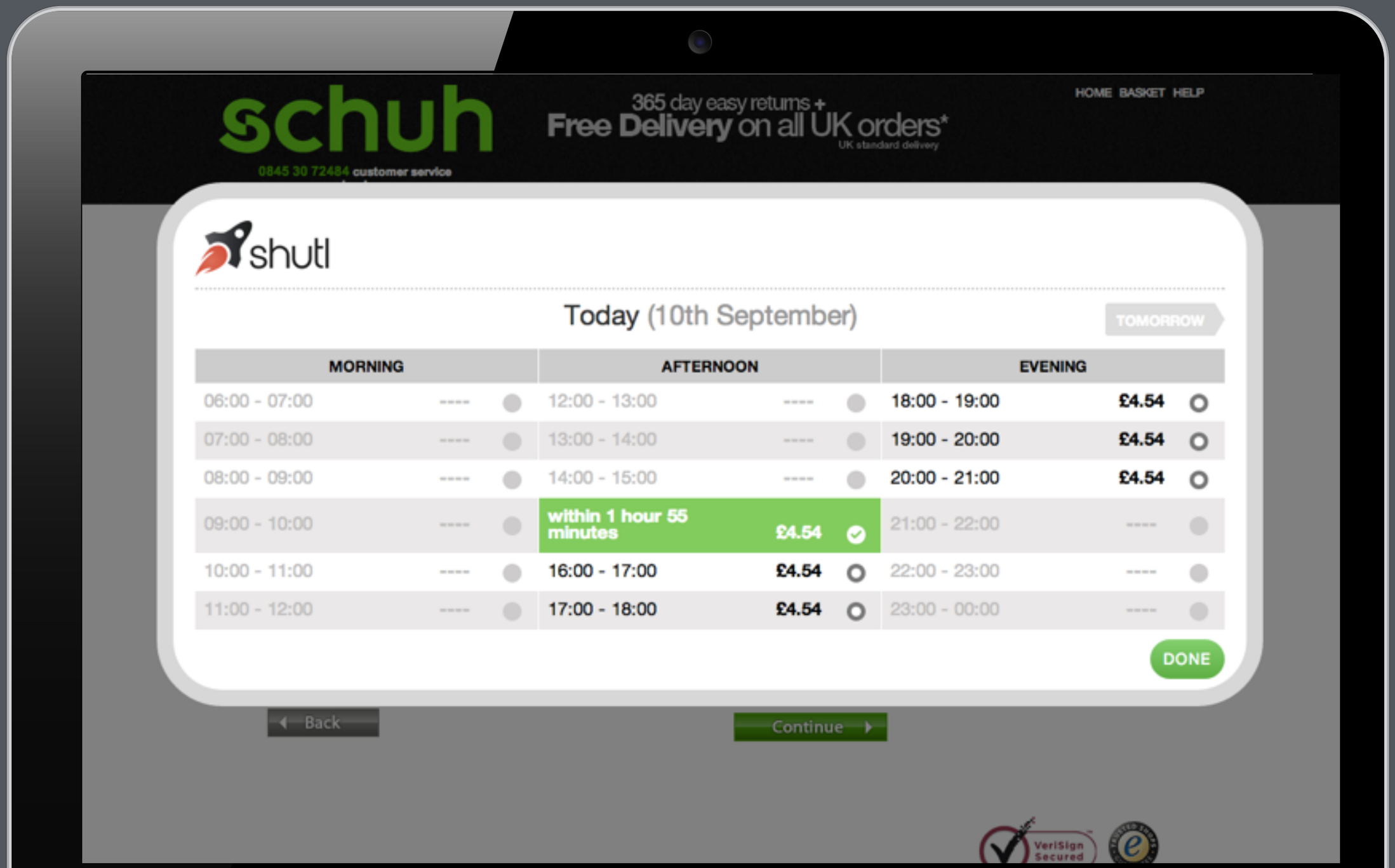
Ruby 1.8, Rails 2.3 and MySQL

Version One

Ruby 1.8, Rails 2.3 and MySQL

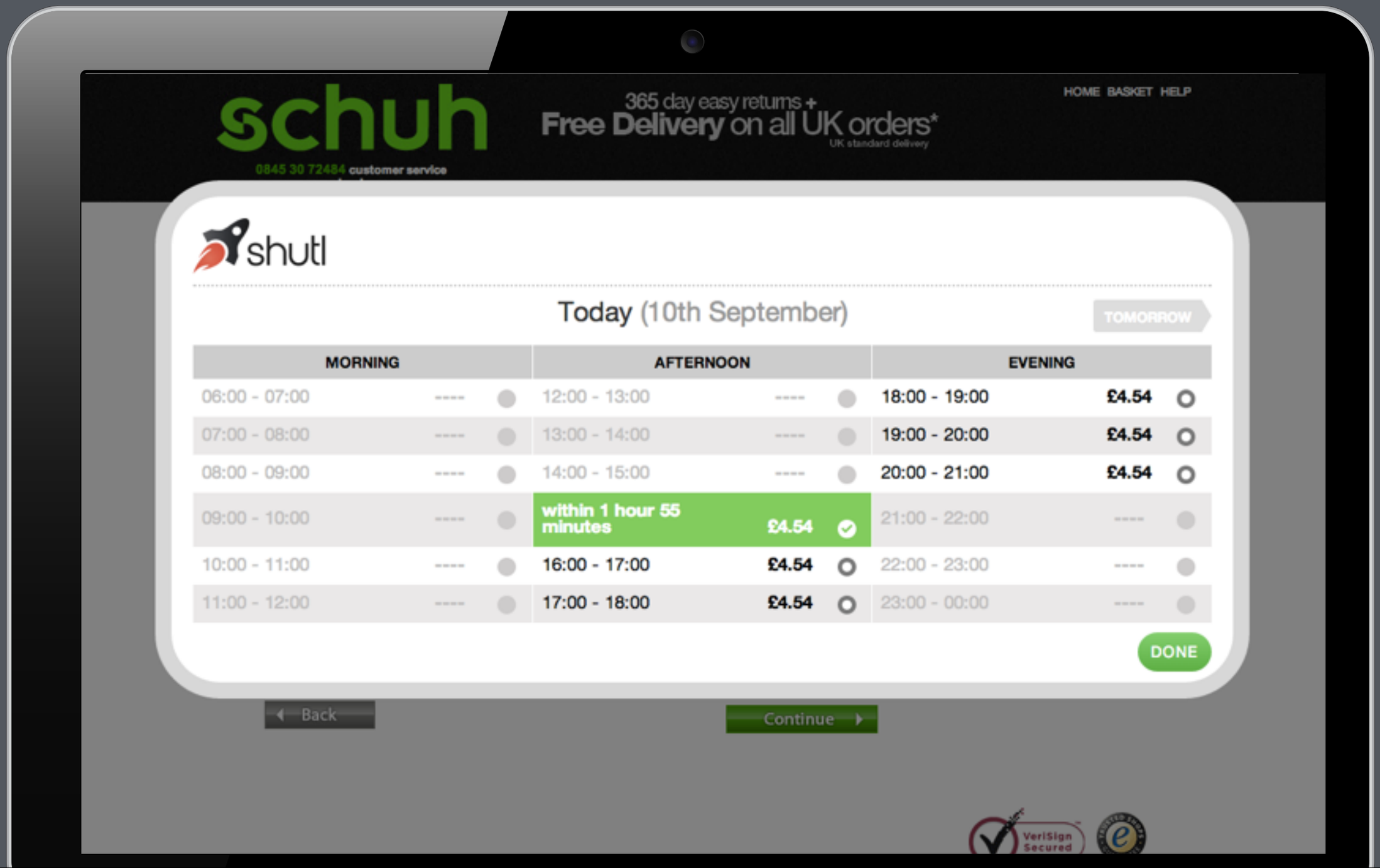
- Well-known tale: built quickly, worked slowly, tough to maintain
- Getting a quote for an hour time-slot took over 4 seconds

Here is the Shutl **price calendar**



Here is the Shutl **price calendar**

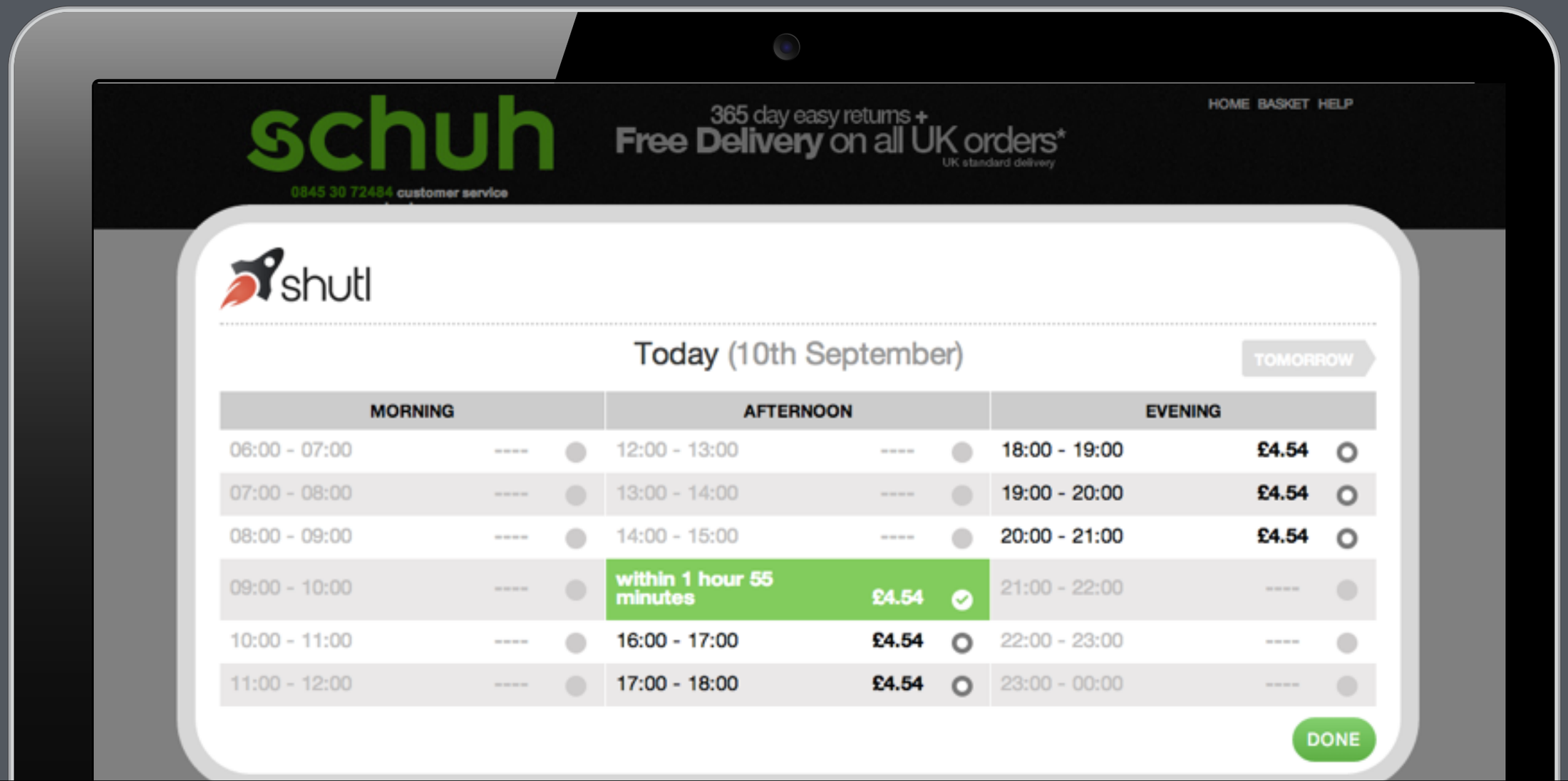
To generate this in V1, the merchant site would have had to call Shutl to get available slots (2 seconds)



Here is the Shutl **price calendar**

To generate this in V1, the merchant site would have had to call Shutl to get available slots (2 seconds)

Then, they would have to call Shutl to generate a quote for each slot - for two days of store opening, that's 20+ slots

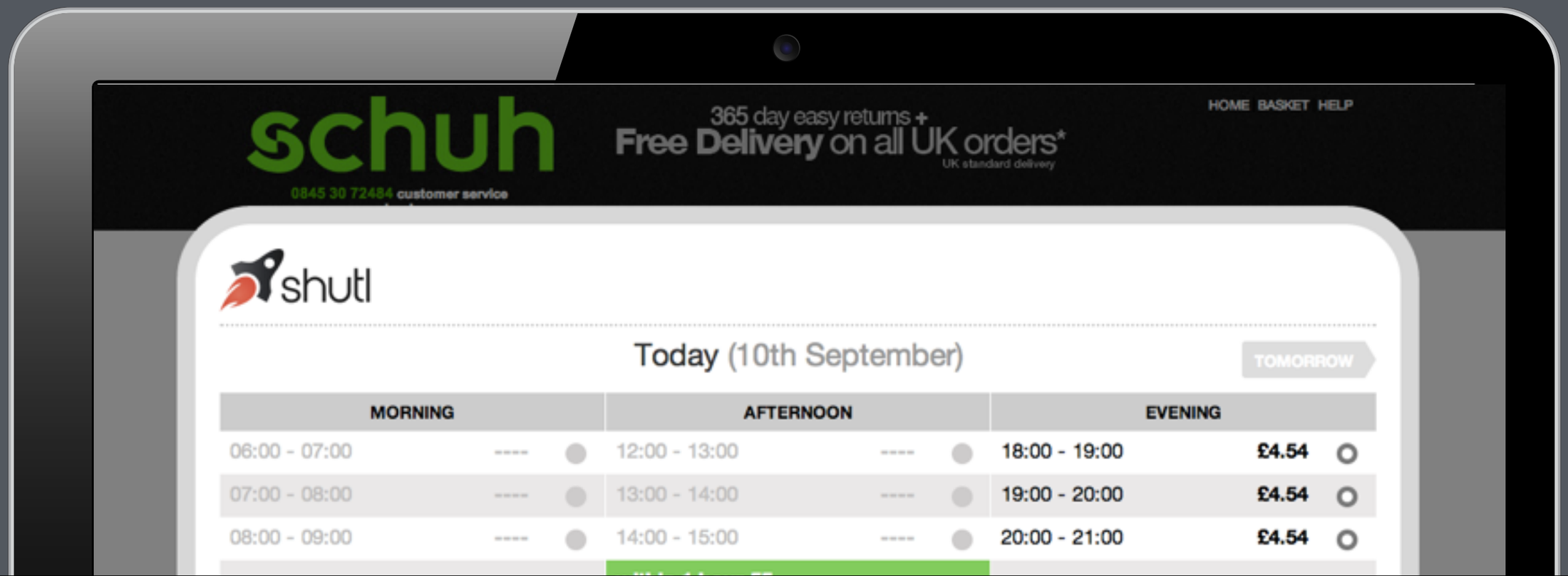


Here is the Shutl **price calendar**

To generate this in V1, the merchant site would have had to call Shutl to get available slots (2 seconds)

Then, they would have to call Shutl to generate a quote for each slot - for two days of store opening, that's 20+ slots

So, that's 2 + (20 x 4) seconds, 1:22 to generate the data for this calendar



Here is the Shutl **price calendar**

To generate this in V1, the merchant site would have had to call Shutl to get available slots (2 seconds)

Then, they would have to call Shutl to generate a quote for each slot - for two days of store opening, that's 20+ slots

So, that's 2 + (20 x 4) seconds, 1:22 to generate the data for this calendar

In V1, this UX could never have happened.

V2

V2

- Broke app into services
- Services focused around functions like quoting, booking, and giving feedback
- Key goal for the project was improving the speed of the quoting operation, which is where we used graph databases

VI

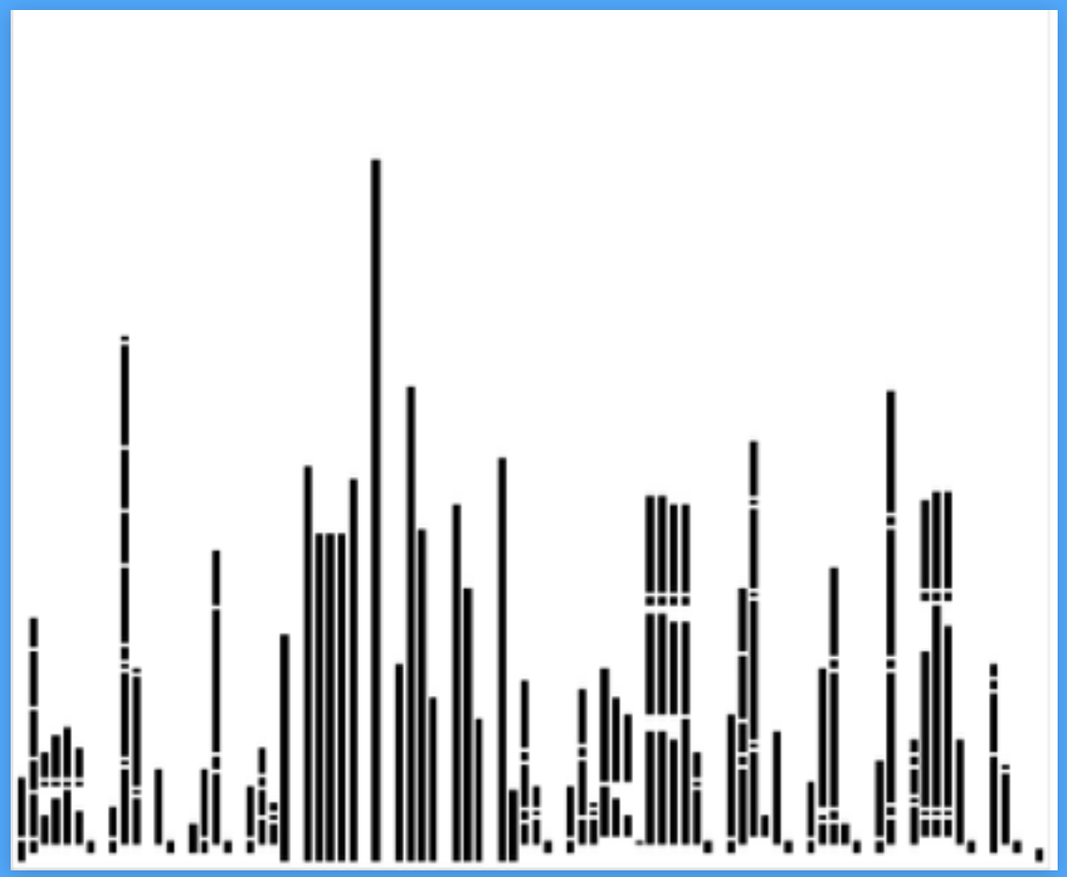
V2

V1

V2

- Quoting for 20 windows down from 82000 ms to 800 ms

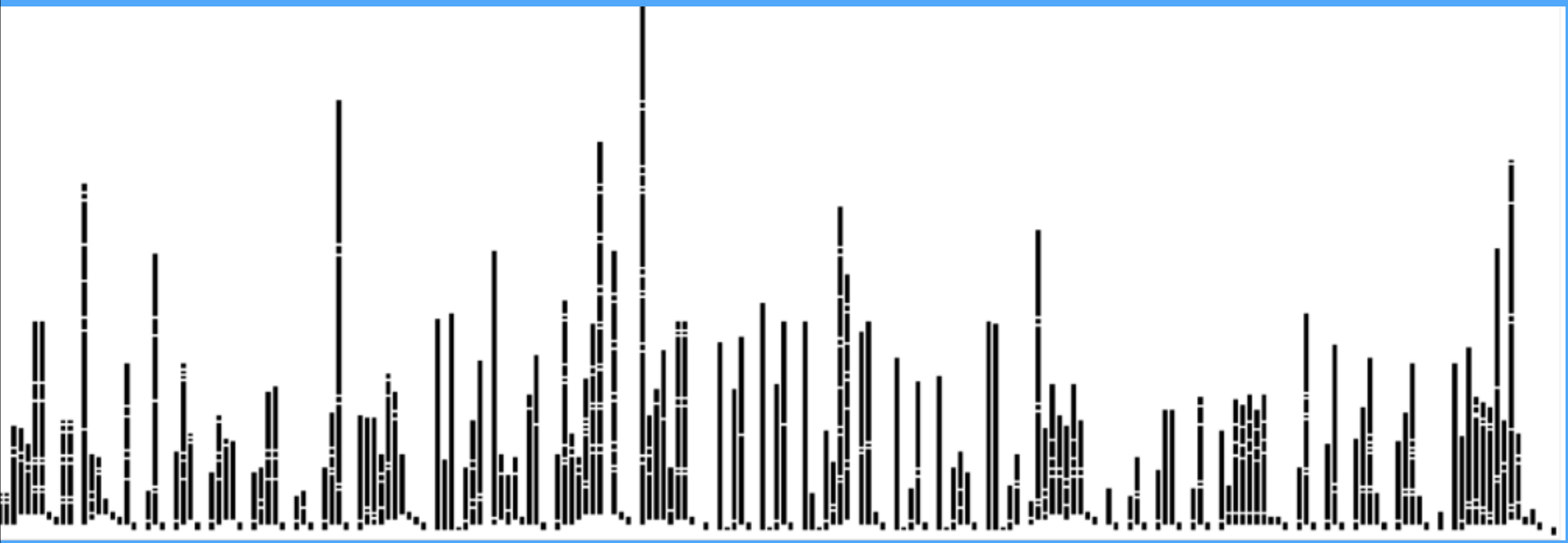
V1



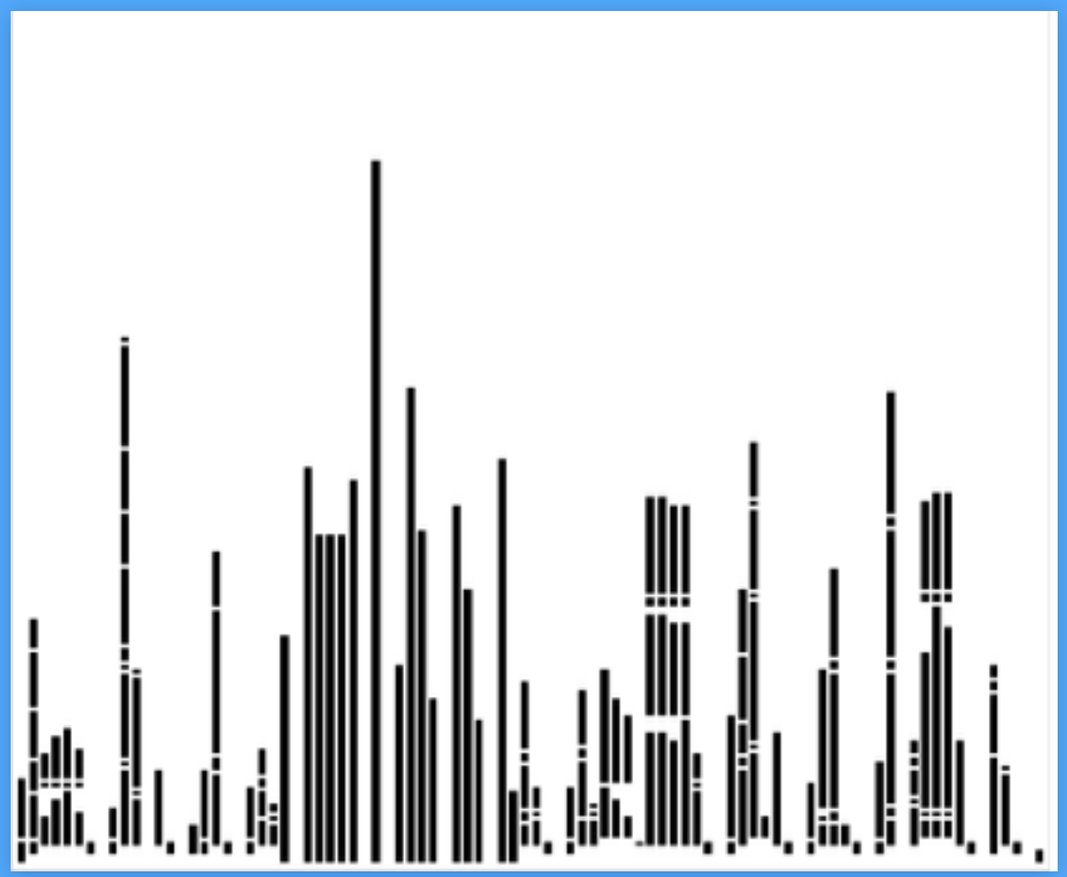
V2

- Quoting for 20 windows down from 82000 ms to 800 ms
- Code complexity much reduced

V1



V2

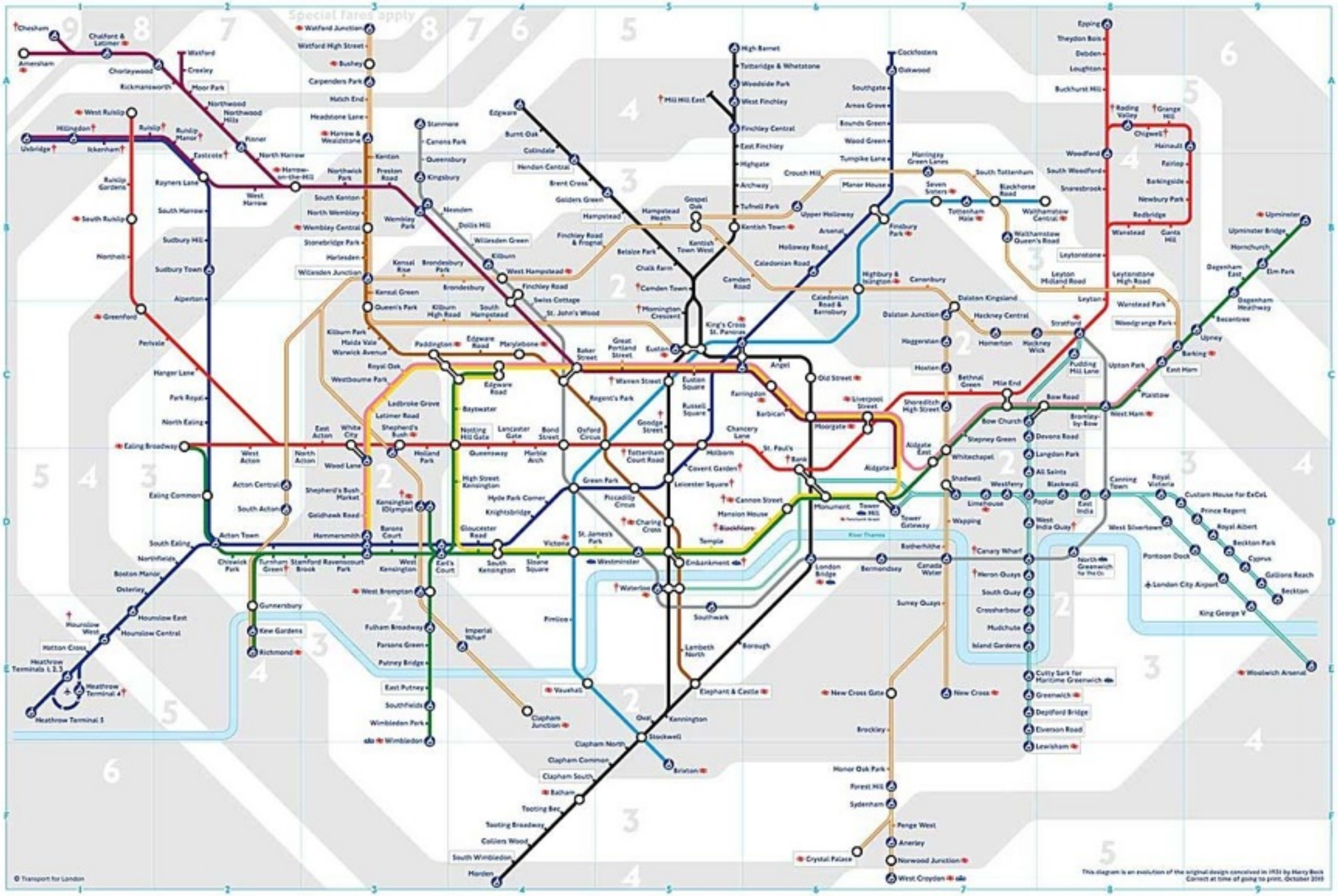


- Quoting for 20 windows down from 82000 ms to 800 ms
- Code complexity much reduced

A large part of the success of our rewrite was down to the graph database.

What is a graph anyway?

Tube map



Key to lines †Check before you travel

Bakerloo		No special arrangements
Central	Chigwell	Served until about 2400
	Grange Hill	
	Roding Valley	
Circle	Blackfriars	Underground station closed until late 20
	Cannon Street	Open until 2100 Mondays to Fridays. Closed Saturdays and Sundays
District	Blackfriars	Underground station closed until late 20
	Cannon Street	Open until 2100 Mondays to Fridays. Closed Saturdays and Sundays
	Kensington (Olympia)	Served 0700 until 23 Mondays to Saturday and 0800 until 2345 Sundays
Hammersmith & City		No special arrangements
Jubilee	Canary Wharf	Step-free interchang between Undergrou Canary Wharf DLR a Heron Quays DLR stations at street le
Metropolitan	Chesham	Change at Chalfont Latimer on most tra
Northern	Camden Town	From 0300 until 0750 Sundays open for interchange and exit only
	Charing Cross branch	Change at Kennington at off-peak times if travelling towards or from Morden
	Mill Hill East	Change at Finchley Central at off-peak times
Piccadilly	Covent Garden	A short walk from either Leicester Squ (6 minutes) or Holbo (9 minutes)
	Eastcote to Uxbridge	Not served by Piccadilly line trains early mornings
	Heathrow Terminal 4	Open until 2400 Mondays to Saturday and until 2330 Sunda Trains may wait for eight minutes before continuing to Terminals 1, 2, 3
	Hounslow West	Step-free access for wheelchair users on
	Turnham Green	Served by Piccadilly trains early mornin and late evenings on
Victoria		No special arrangements
Waterloo & City	Bank to Waterloo	Open 0615 until 2148 Mondays to Fridays and 0800 until 1830 Saturdays. Closed Sundays and public holidays
Overground		No special arrangements
DLR	Heron Quays	Step-free interchang between Heron Qua and Canary Wharf Underground statio street level.
	West India Quay	Not served by DLR trains from Bank towards Lewisham a peak times

MAYOR OF LONDON

Website tfl.gov.uk

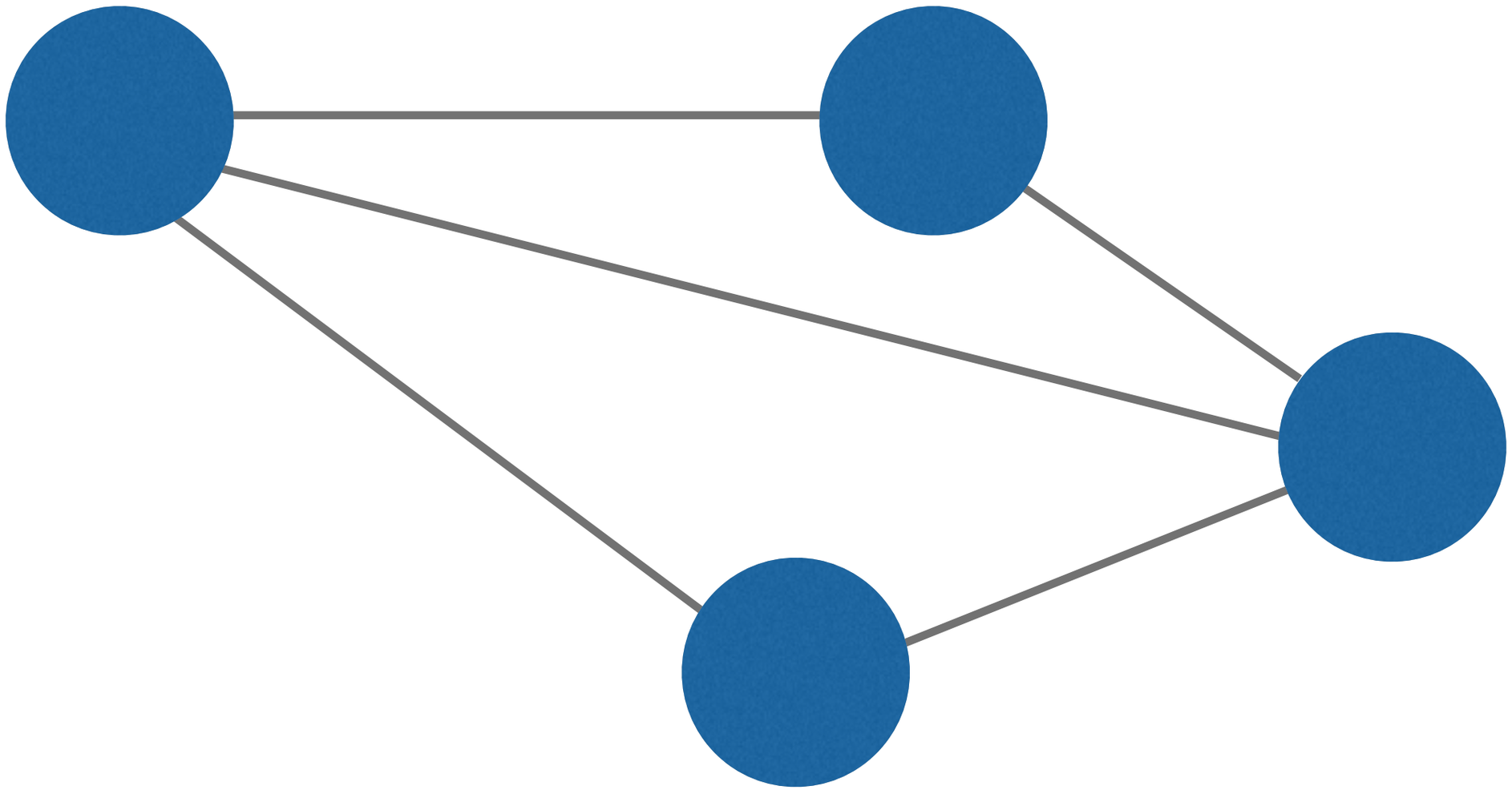
24 hour travel information 0843 222 1234*

*You pay no more than 5p per minute if calling from a BT landline. There may be a connection charge. Charges from mobiles or other landline providers may vary.

Transport for London

UNDERGROUND

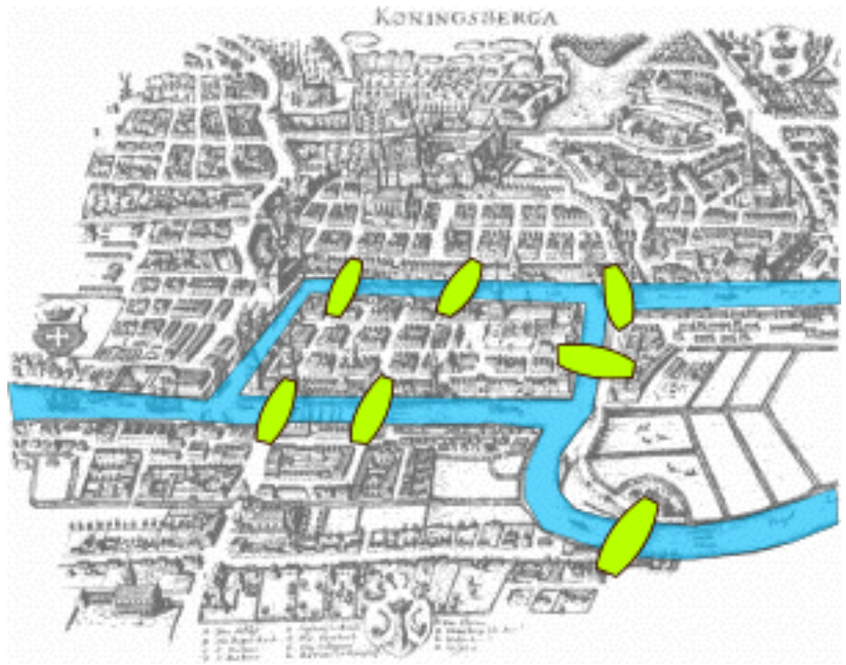
a simple graph



a collection of vertices (nodes)
connected by edges (relationships)

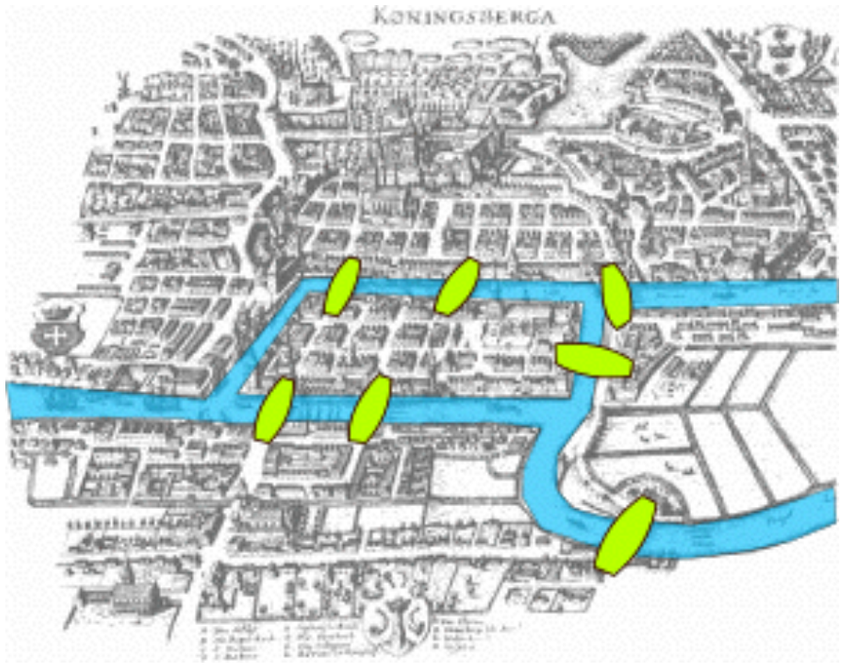
a short history

the seven bridges of Königsberg (1735)

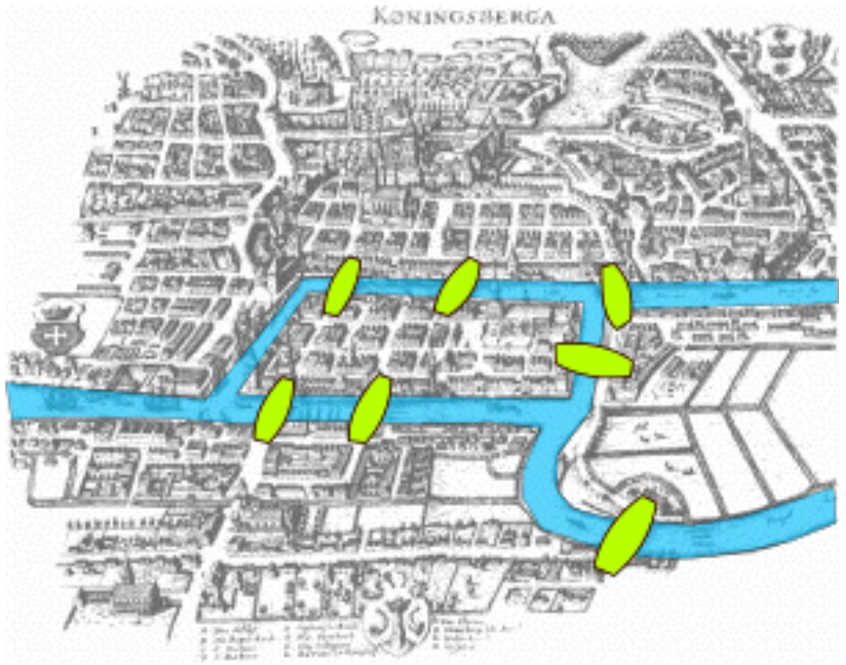


Leonard Euler

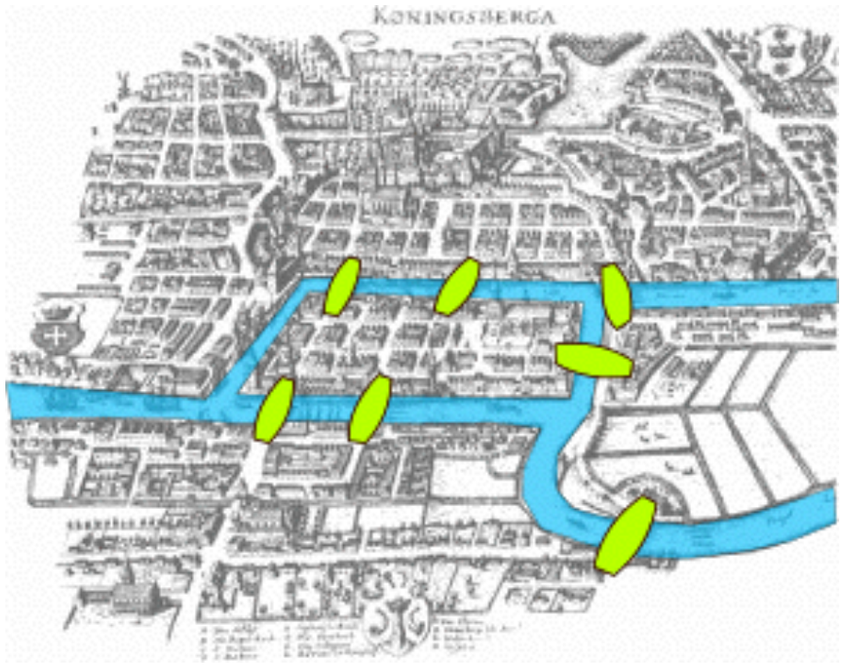
the seven bridges of Königsberg (1735)



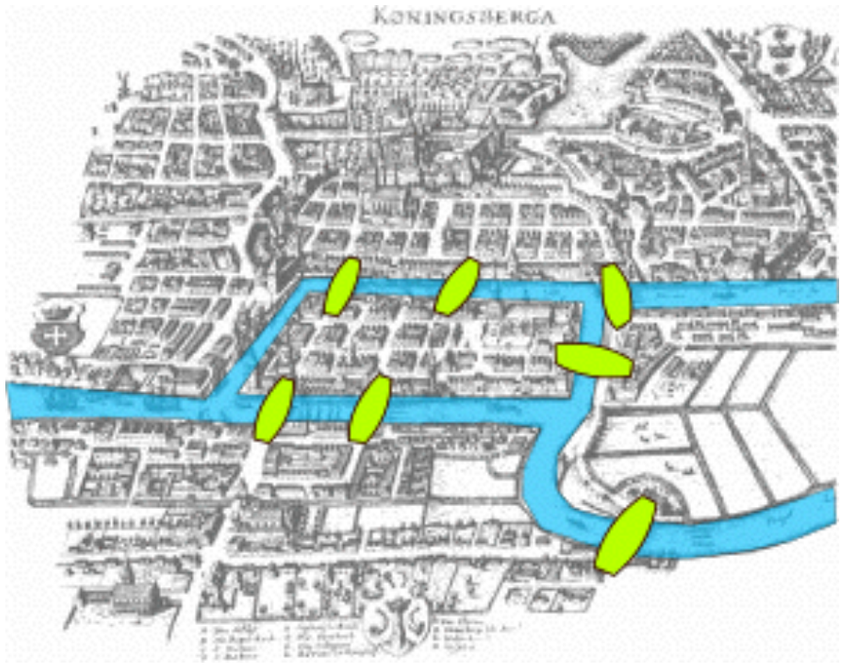
the seven bridges of Königsberg (1735)

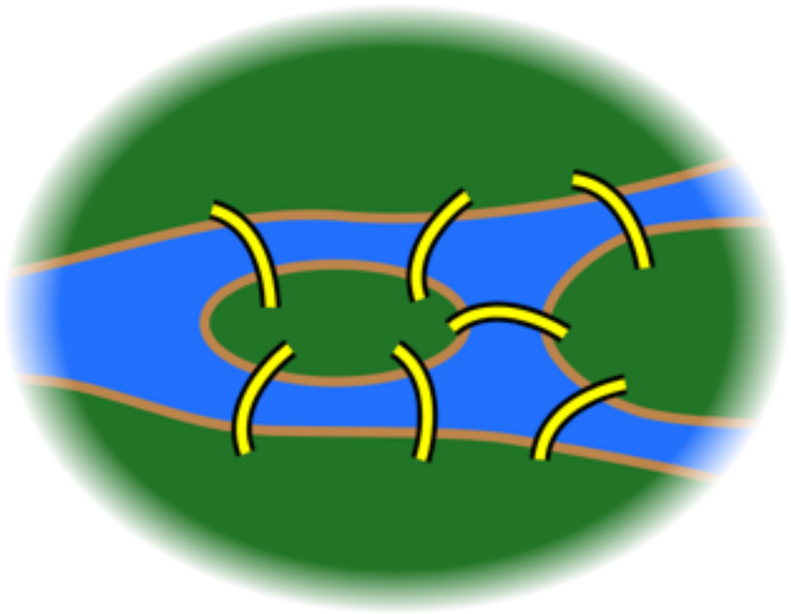


the seven bridges of Königsberg (1735)

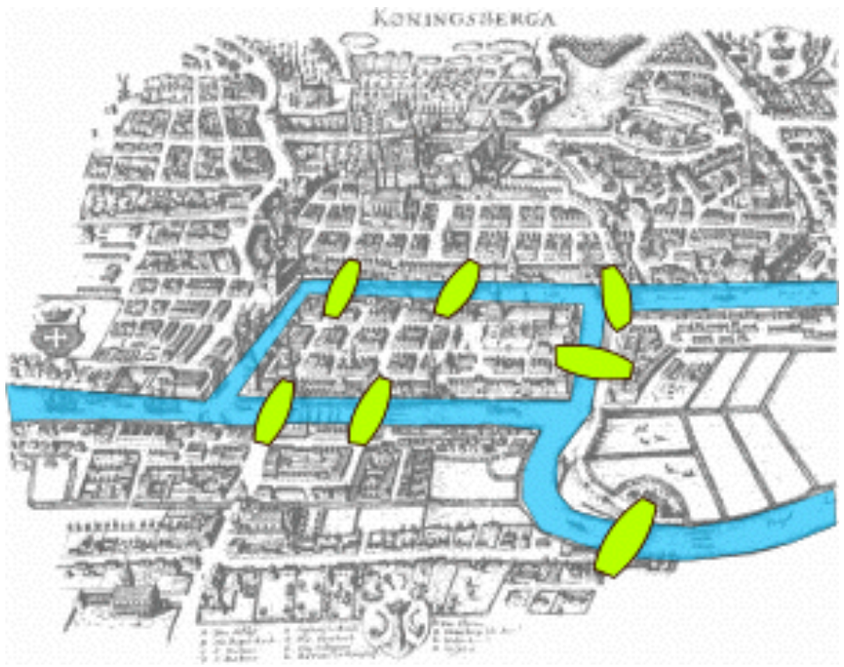


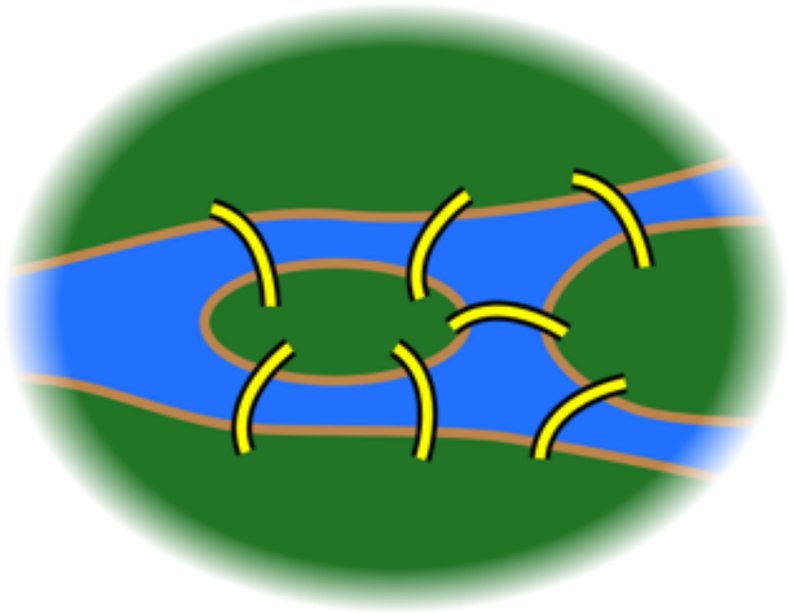
the seven bridges of Königsberg (1735)

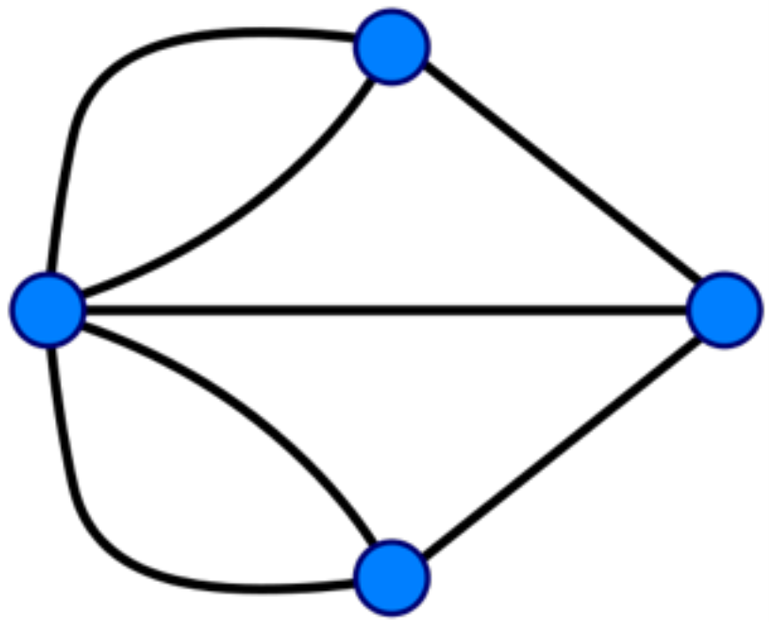




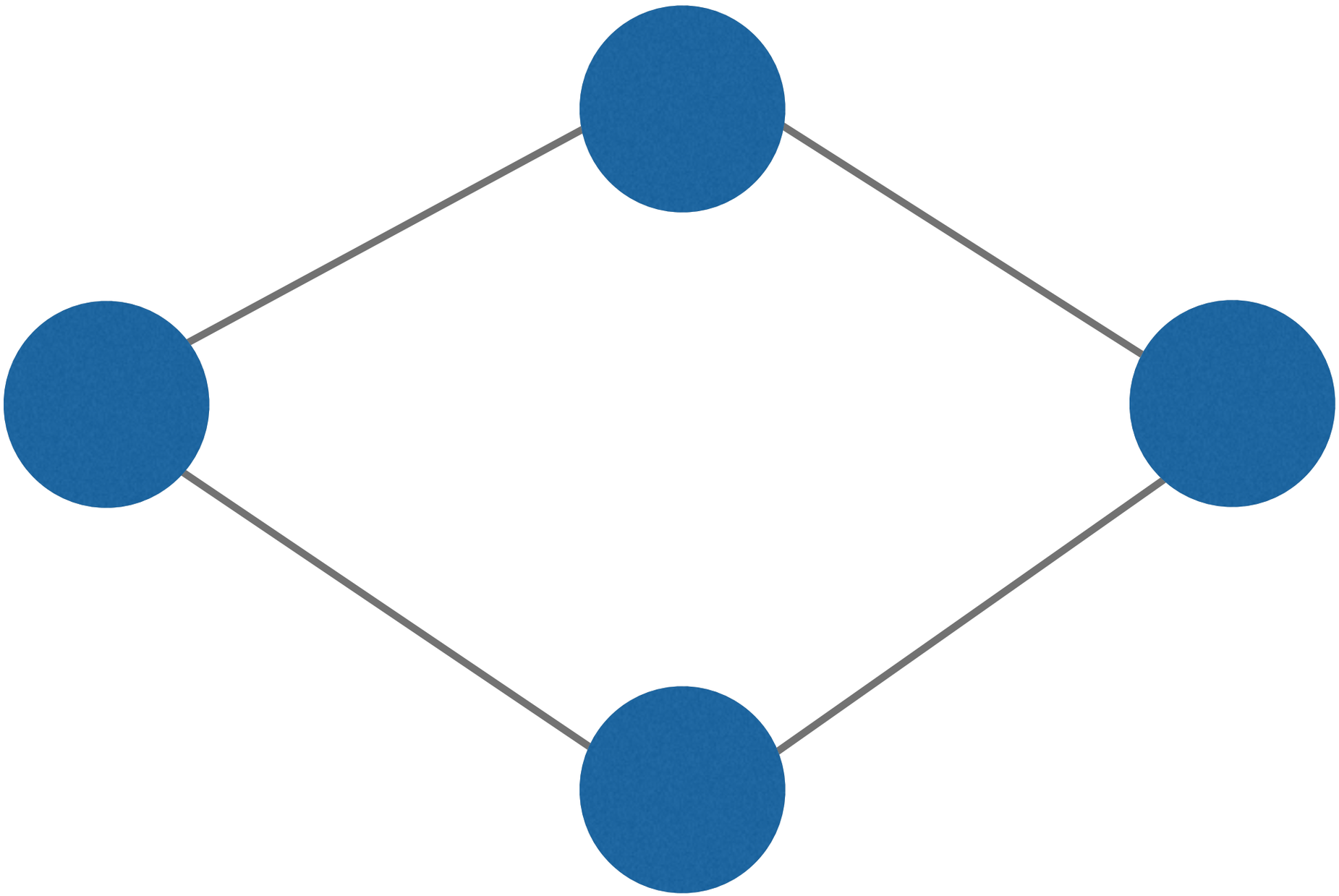
the seven bridges of Königsberg (1735)





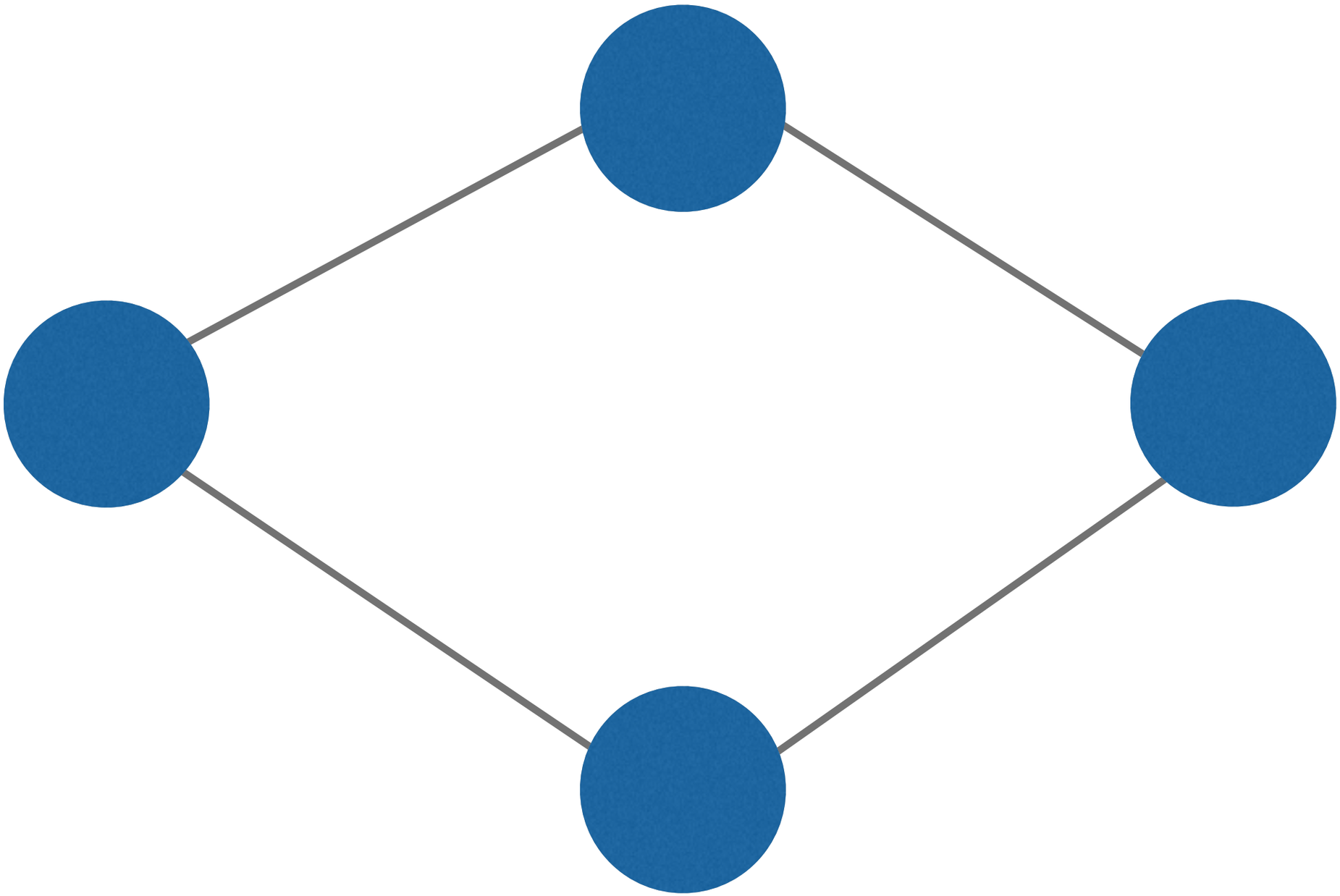


Euler walk

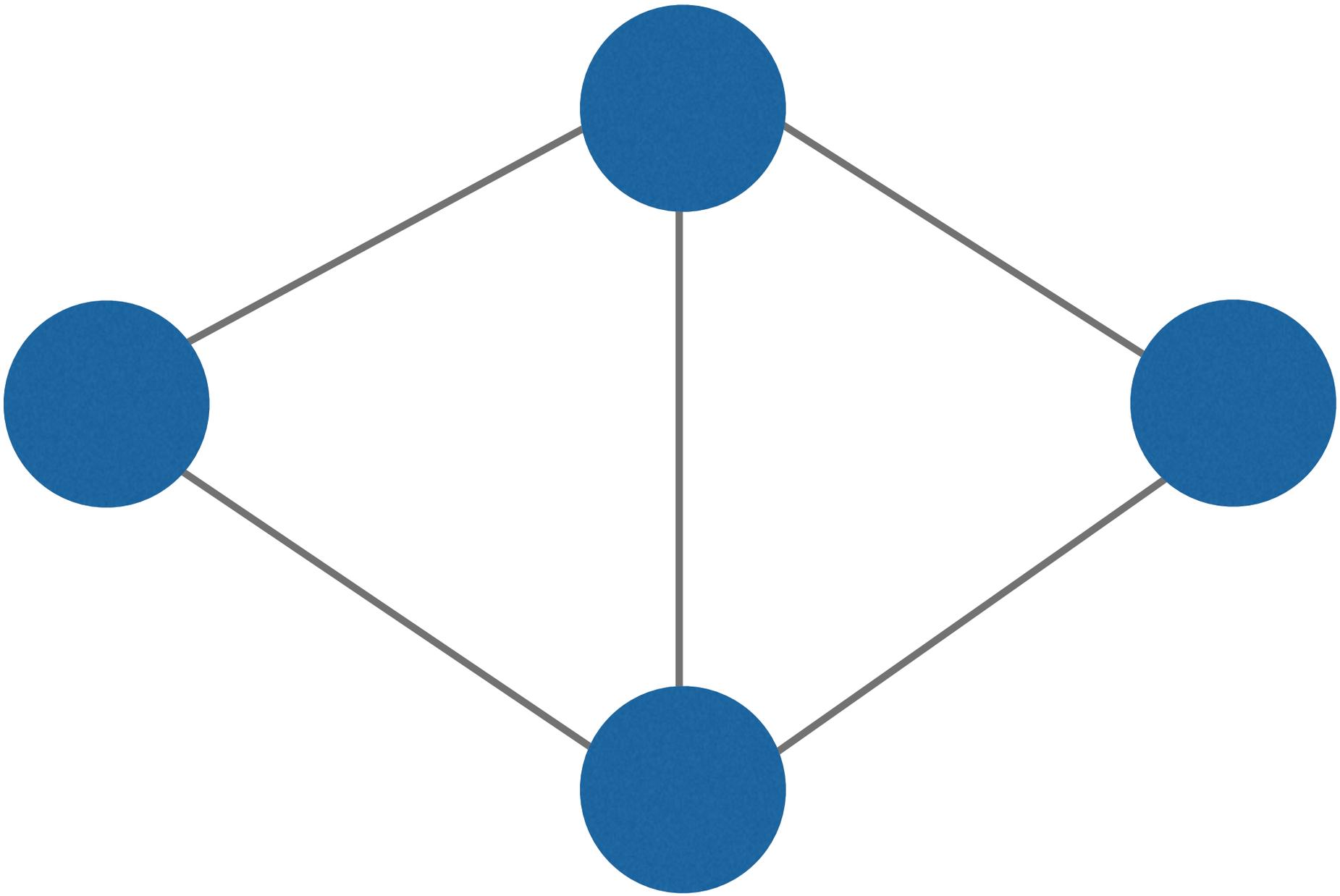


each node has an even degree

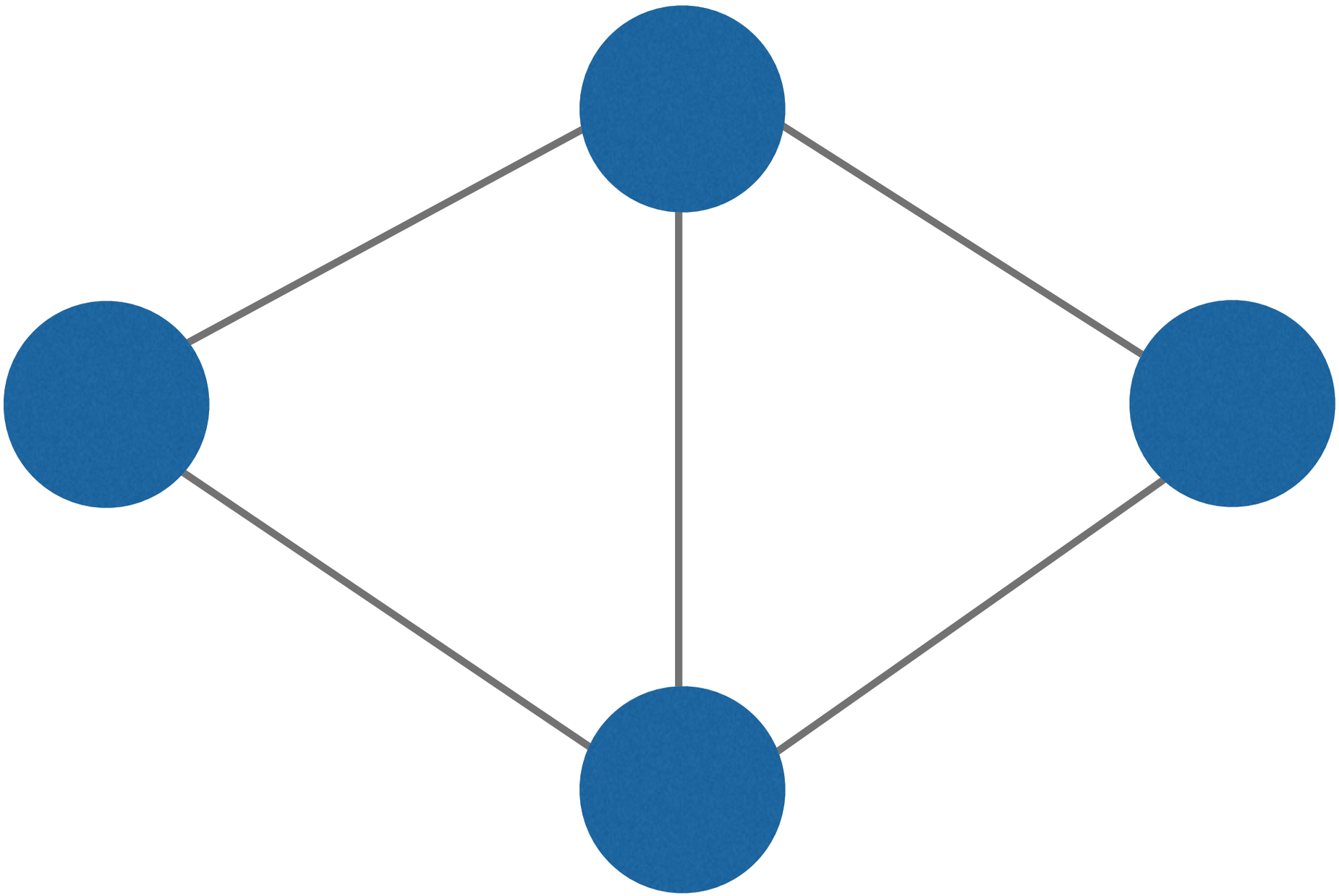
Euler walk



Euler walk

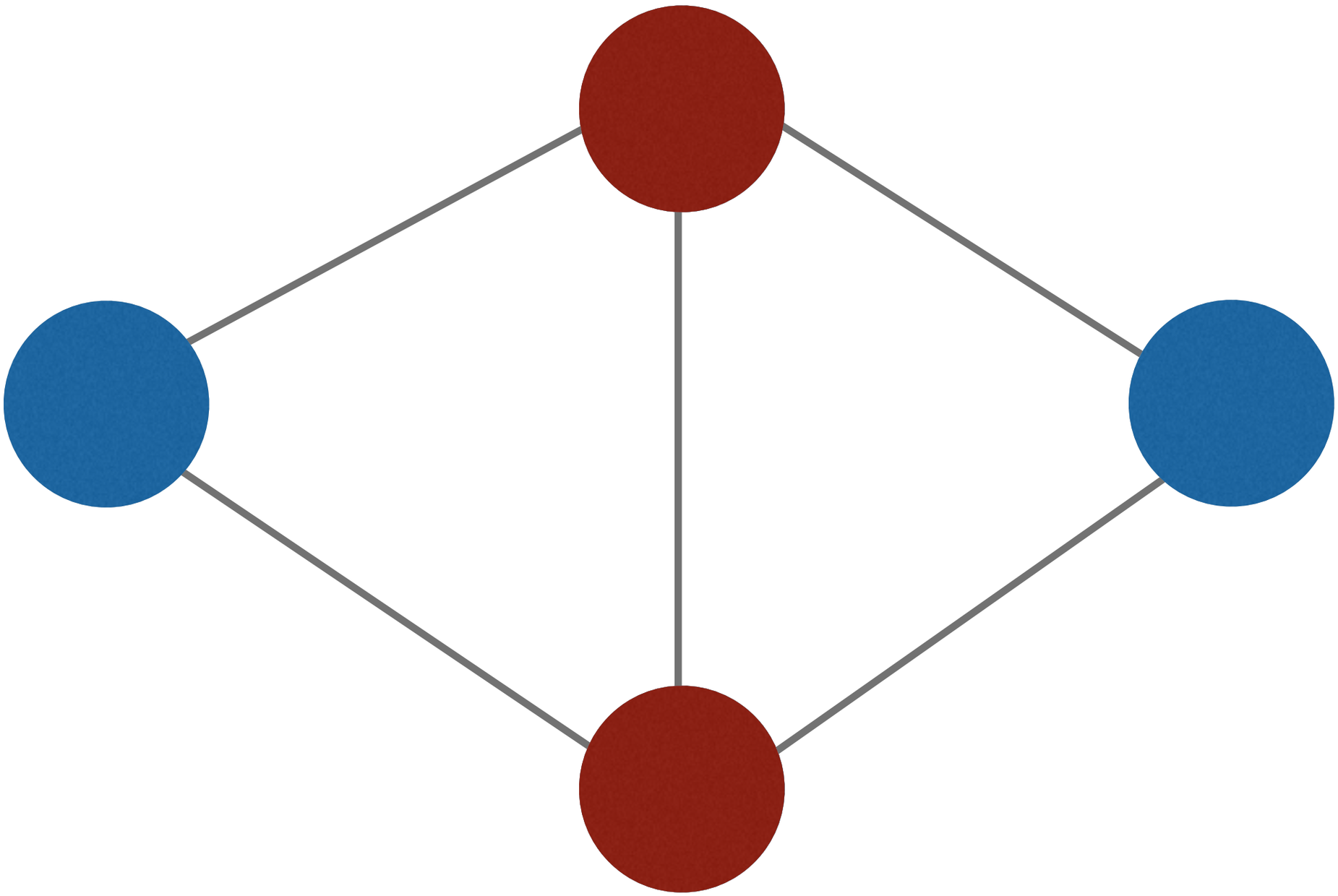


Euler walk



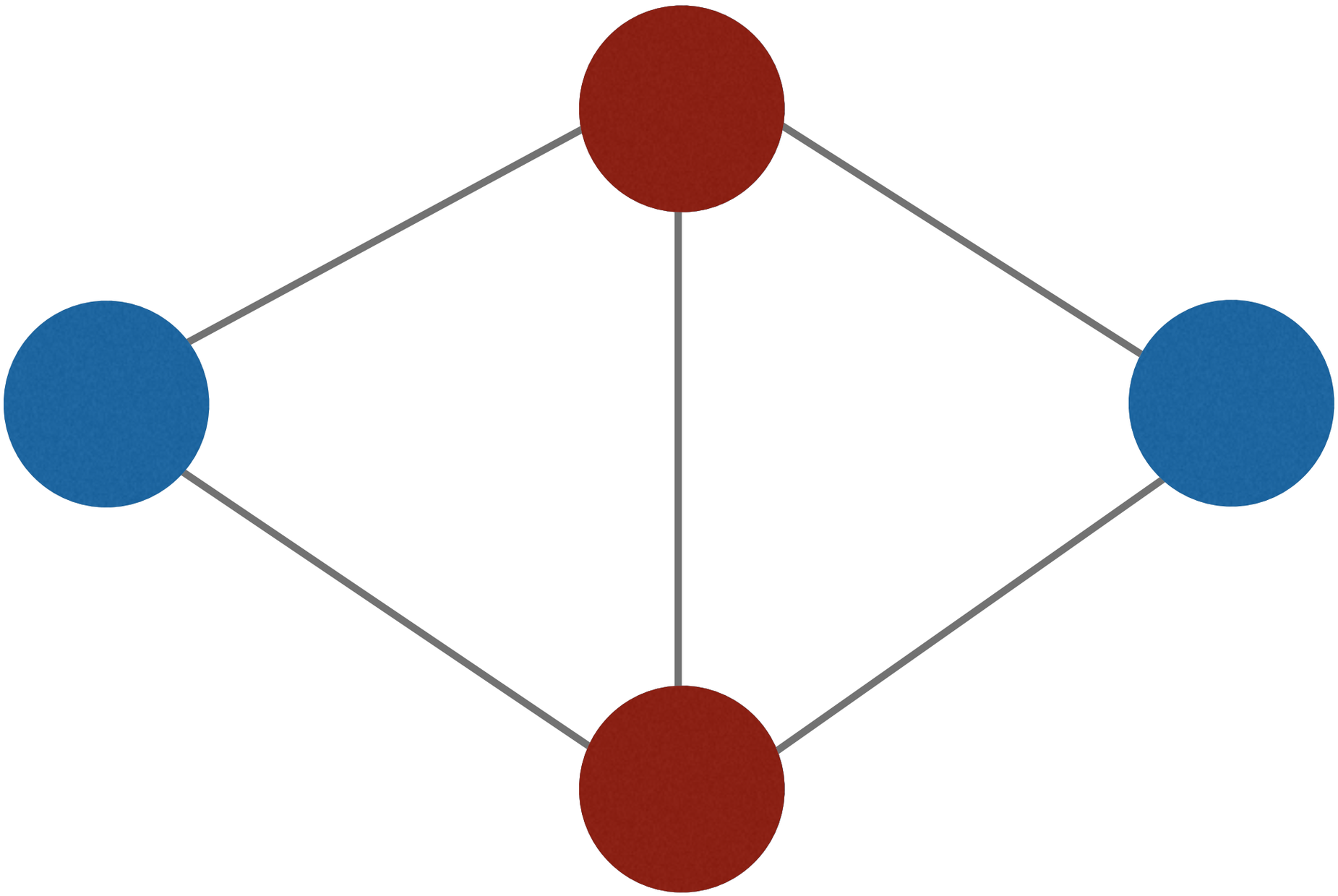
two nodes have an odd degree

Euler walk



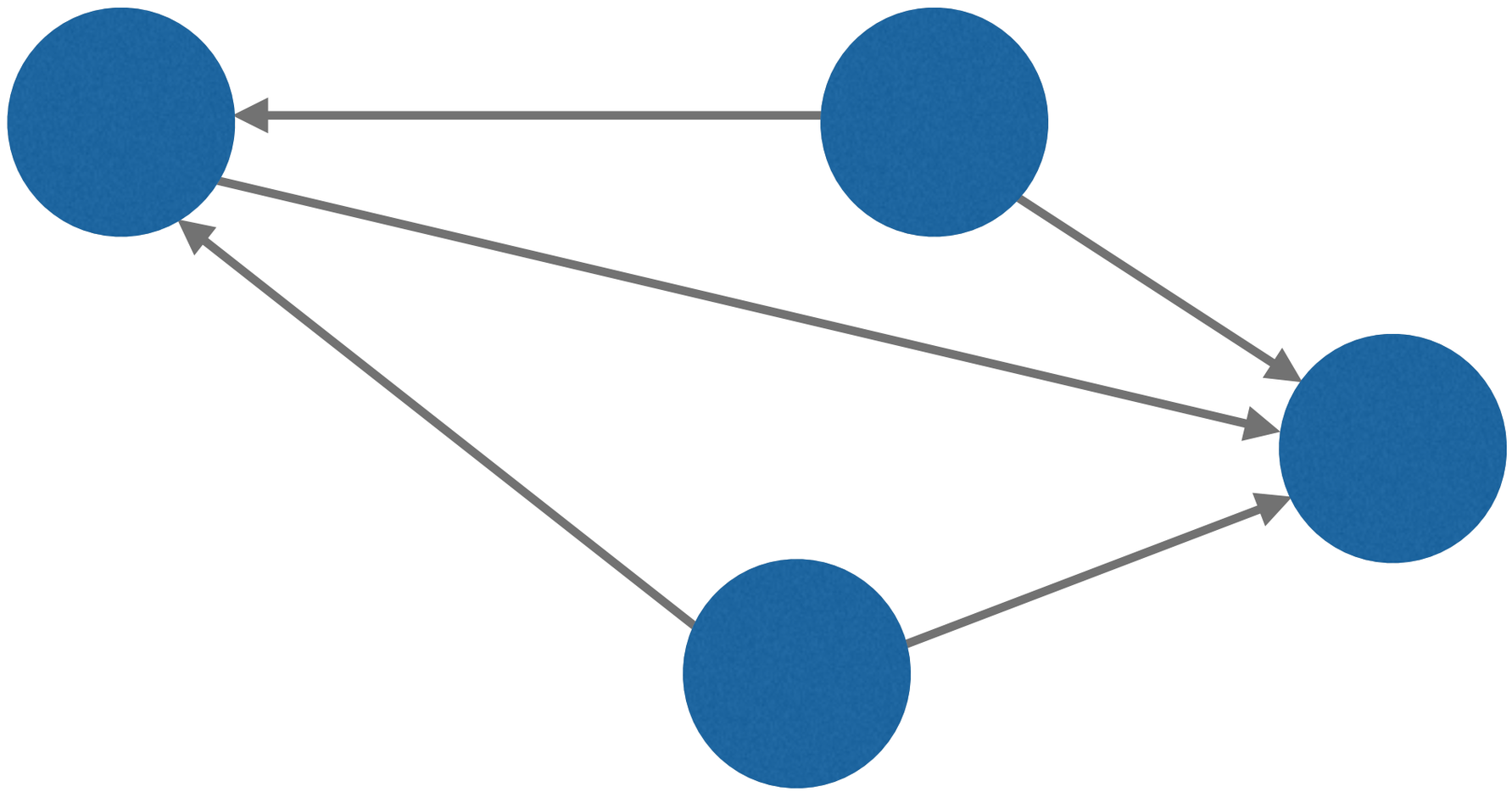
two nodes have an odd degree

no Euler walk



two nodes have an odd degree

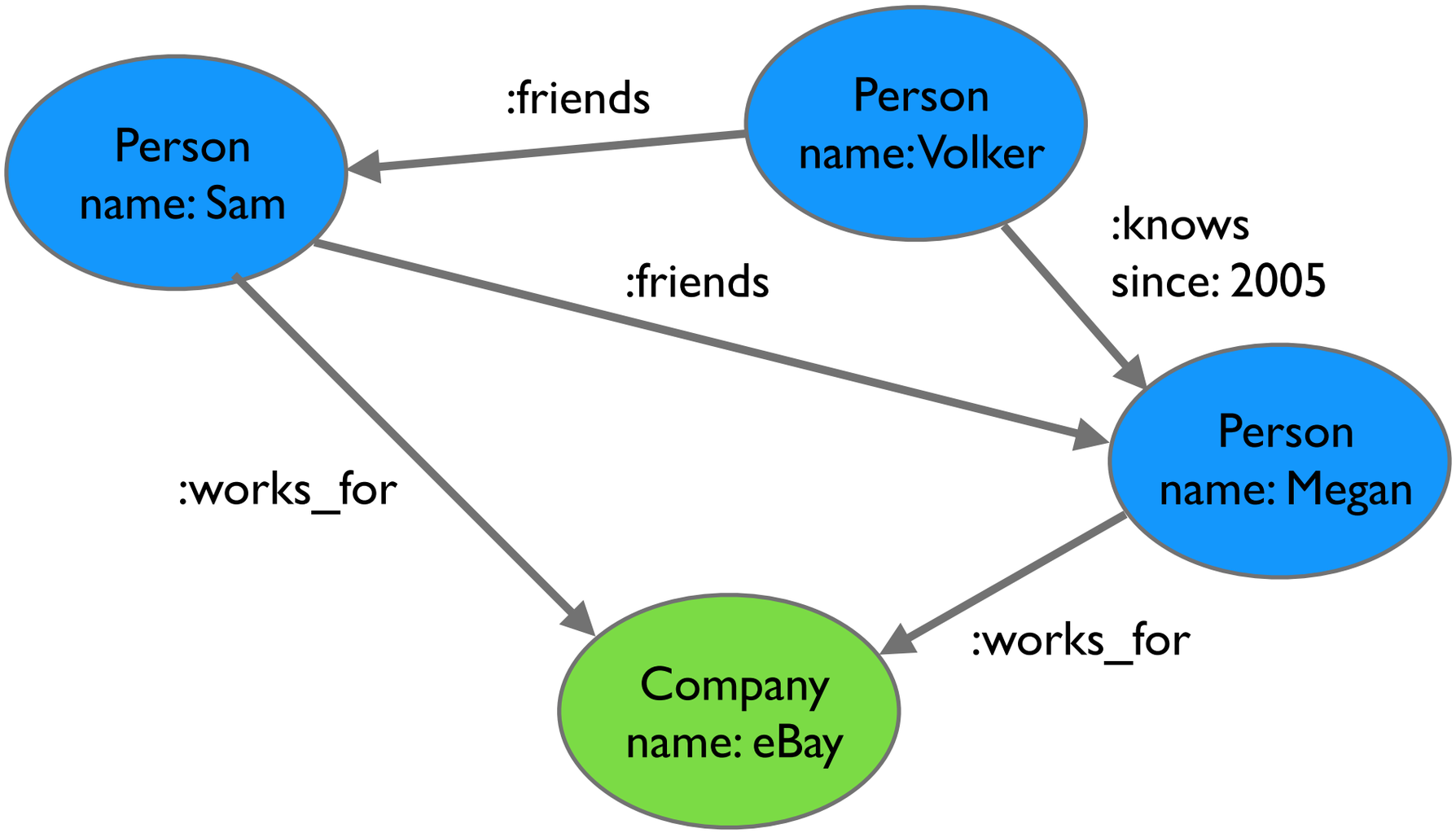
directed graph



each relationship has a direction or
one start node and one end node

property graph

nodes contain properties (key, value)
relationships have a type and are always directed
relationships can contain properties too



The Case for Graph Databases

relationships are explicit stored

additive domain modelling

whiteboard friendly

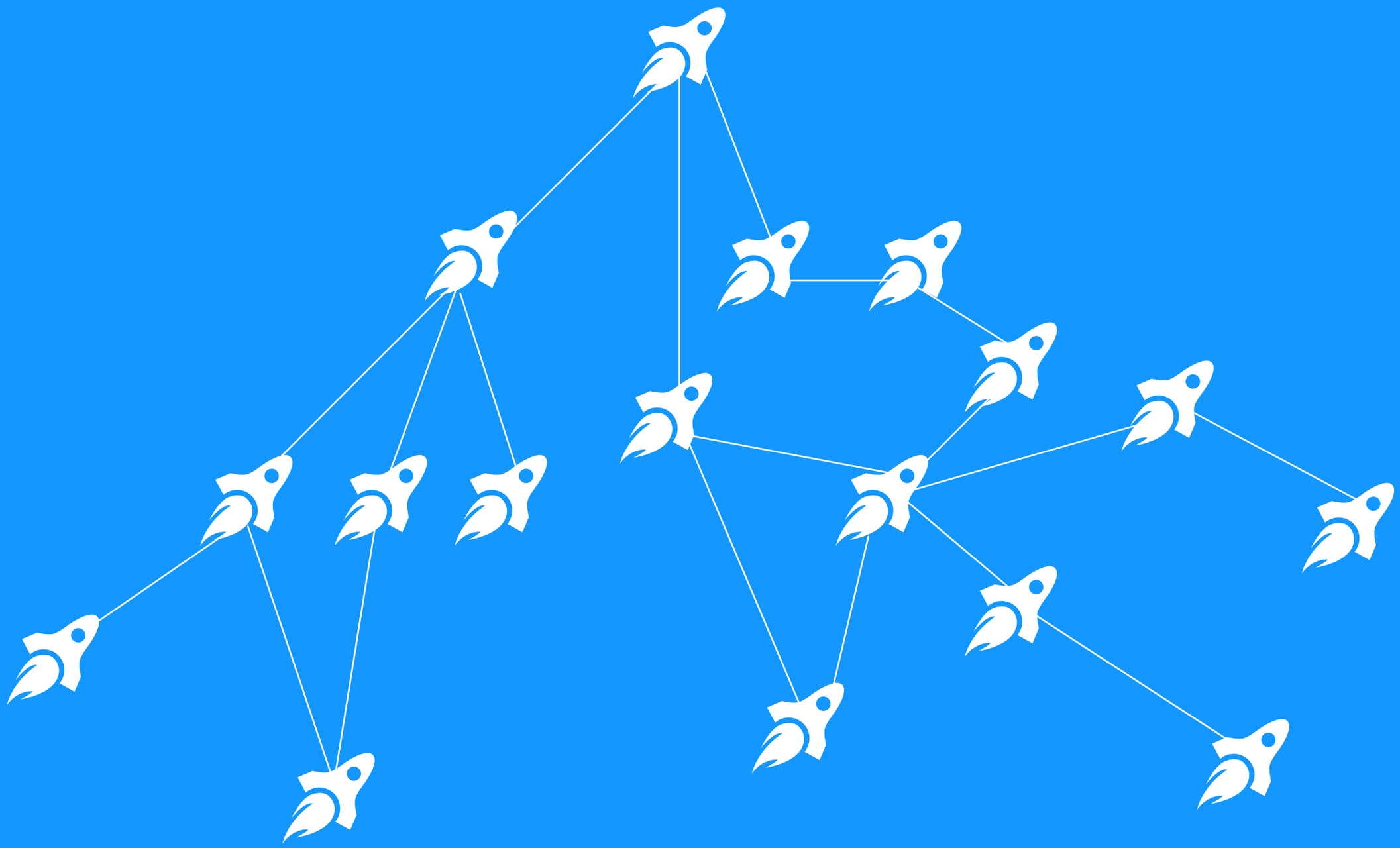
traversals of relationships are easy and very fast

DB performance remains relatively constant as queries are localised to its portion of the graph.

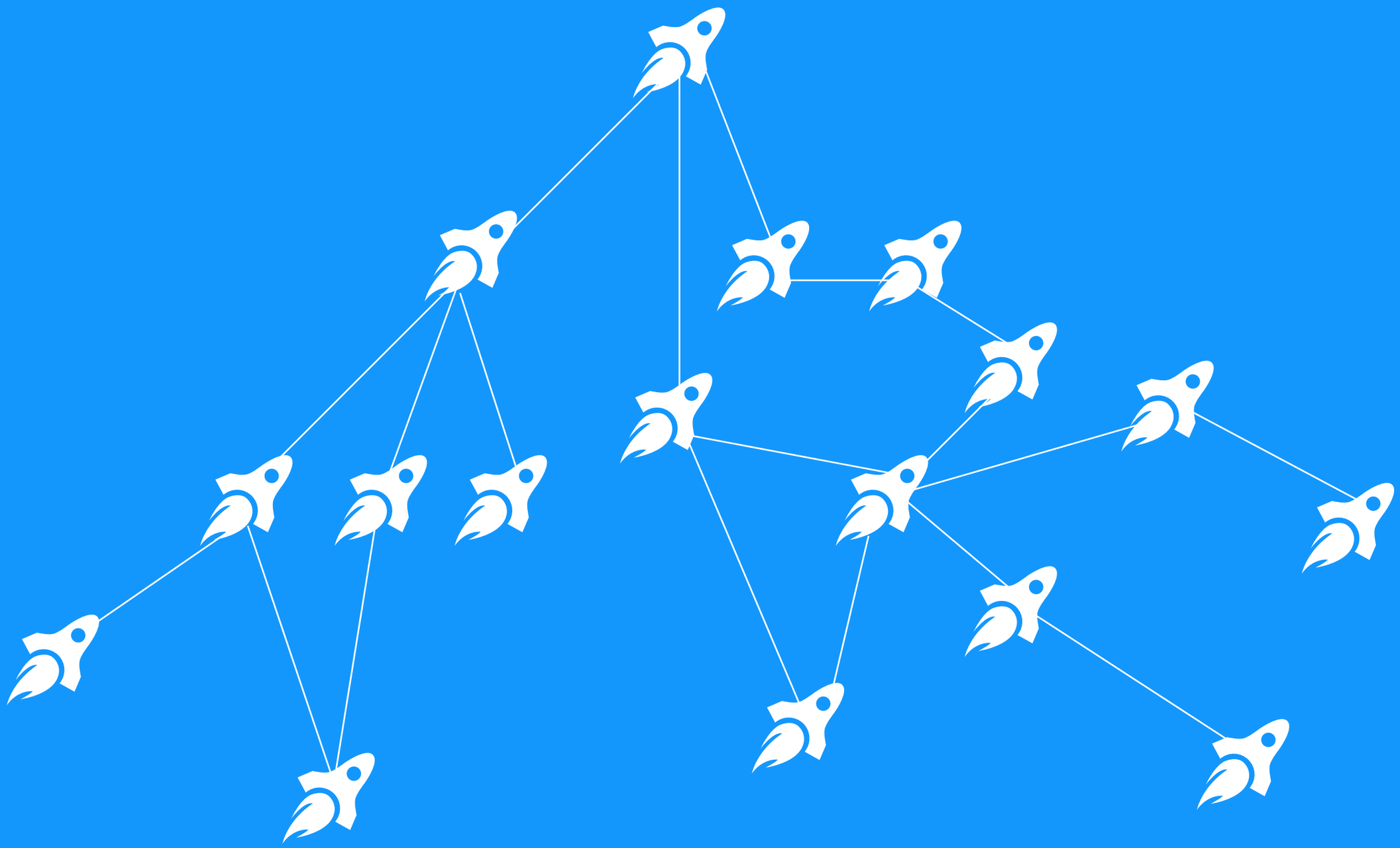
O(1) for same query

a graph is its own index (constant query performance)

a graph is its own index (constant query performance)



a graph is its own index (constant query performance)



the case for Neo4j

standalone or embedded in jvm

ruby/jruby

ruby libraries - neo4j gem by Andreas Ronge (https://github.com/andreasronge/neo4j)

cypher

the neotech guys are awesome

Querying the graph: Cypher

declarative query language specific to neo4j

easy to learn and intuitive

use specific patterns to query for (something that looks like ‘this’)

inspired partly by SQL (WHERE and ORDER BY) and SPARQL (pattern matching)

focuses on what to query for and not how to query for it

switch from a mySQl world is made easier by the use of cypher instead of having to learn a traversal framework straight away

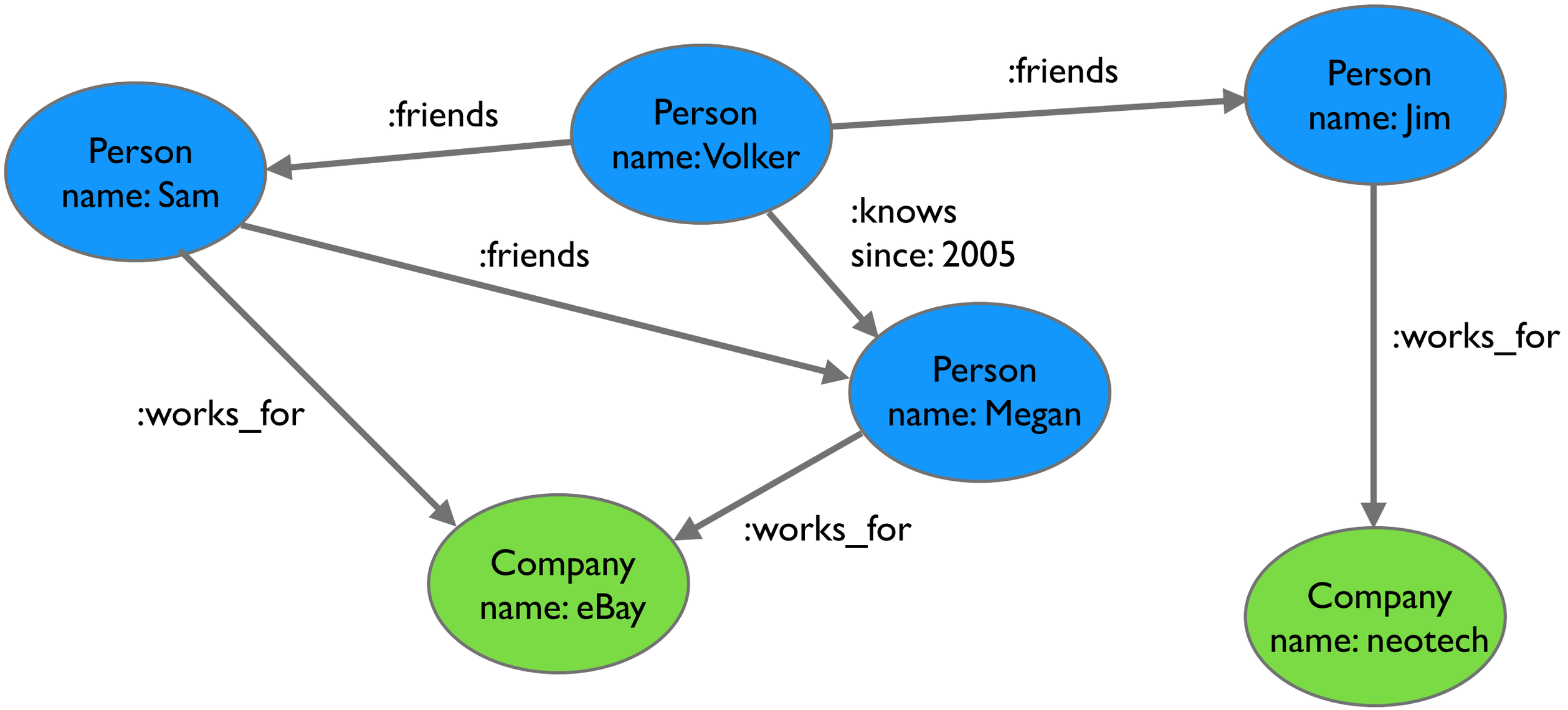
cypher clauses

START: Starting points in the graph, obtained via index lookups or by element IDs.
MATCH: The graph pattern to match, bound to the starting points in START.
WHERE: Filtering criteria.
RETURN: What to return.
CREATE: Creates nodes and relationships.
DELETE: Removes nodes, relationships and properties.
SET: Set values to properties.
FOREACH: Performs updating actions once per element in a list.
WITH: Divides a query into multiple, distinct parts

cypher clauses

~~START: Starting points in the graph, obtained via index lookups or by element IDs.~~
MATCH: The graph pattern to match, bound to the starting points in START.
WHERE: Filtering criteria.
RETURN: What to return.
CREATE: Creates nodes and relationships.
DELETE: Removes nodes, relationships and properties.
SET: Set values to properties.
FOREACH: Performs updating actions once per element in a list.
WITH: Divides a query into multiple, distinct parts

an example


Person
name: Sam
:friends
Person
name:Volker
:friends
Person
name: Jim
:friends
:knows
since: 2005
:works_for
Person
name: Megan
:works_for
:works_for
Company
name: eBay
Company
name: neotech

find all the companies my friends work for

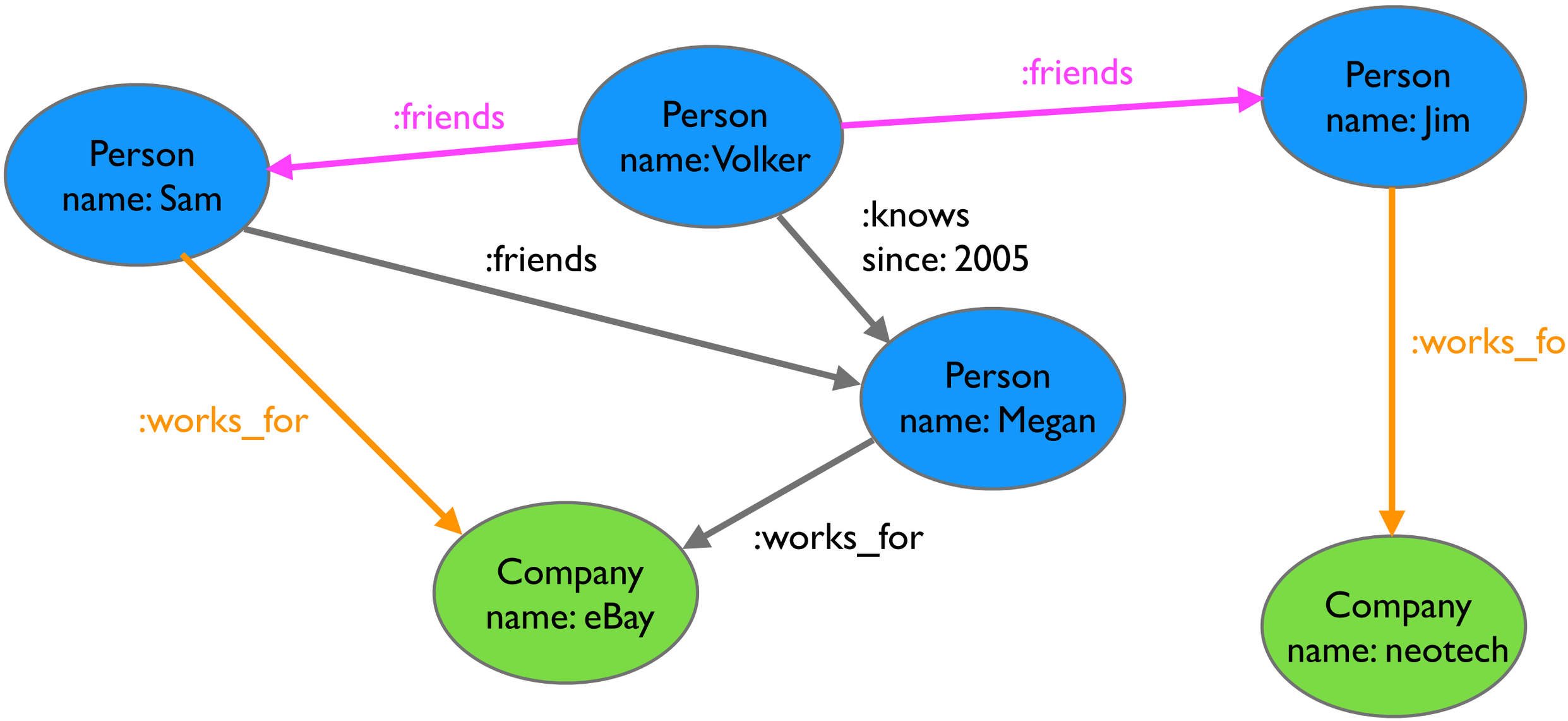
```
MATCH (person{ name:'Volker' }) -[:friends]
      - (person) - [:works_for]-> company
RETURN company
```

find all the companies my friends work for

```
MATCH (person{ name:'Volker' }) -[:friends]
      - (person) - [:works_for]-> company
RETURN company
```

find all the companies my friends work for

```
MATCH (person{ name:'Volker' }) -[:friends]
      - (person) - [:works_for]-> company
RETURN company
```



find all the companies my friend's friends work for

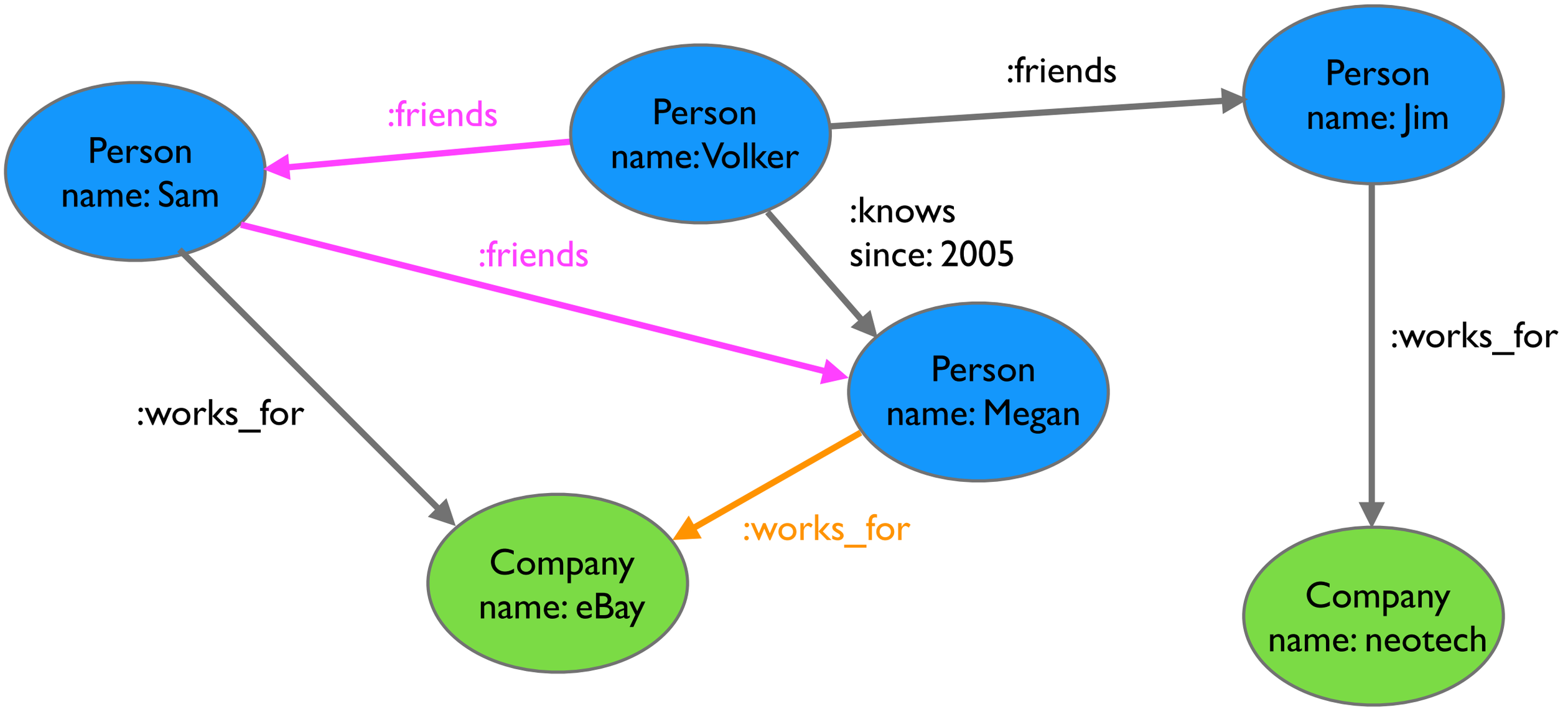
```
MATCH (person{ name:'Volker' }) -
      [:friends*2..2]-(person) - [:works_for]
      -> company
RETURN company
```

find all the companies my friend's friends work for

```
MATCH (person{ name:'Volker' }) -
      [:friends*2..2]-(person) - [:works_for]
      -> company
RETURN company
```

find all the companies my friend's friends work for

```
MATCH (person{ name:'Volker' }) -
      [:friends*2..2]-(person) - [:works_for]
      -> company
RETURN company
```



find all my friends who work for neotech

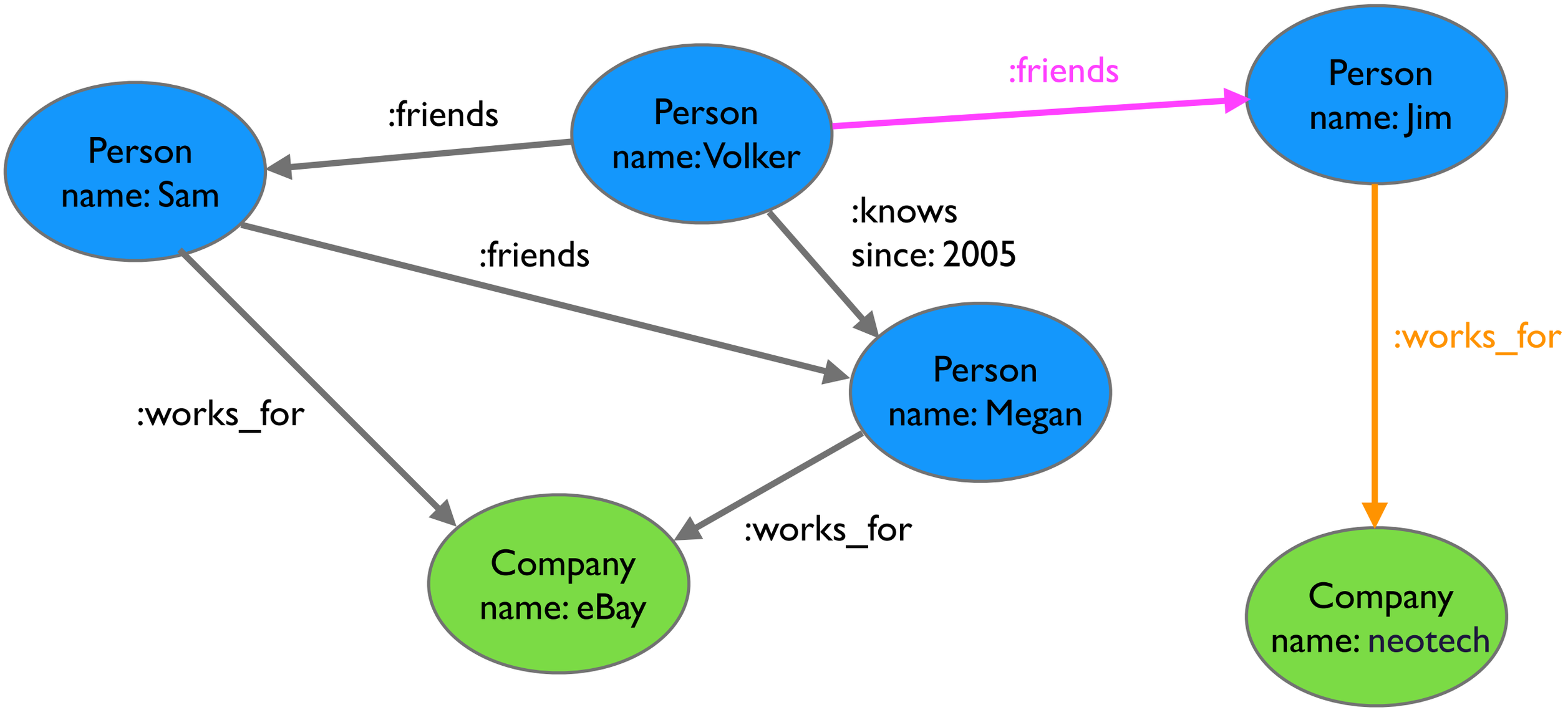
```
MATCH (person{ name:'Volker' }) -[:friends]
      -(friends) - [:works_for]-> company
WHERE   company.name = 'neotech'
RETURN friends
```

find all my friends who work for neotech

```
MATCH (person{ name:'Volker' }) -[:friends]
      -(friends) - [:works_for]-> company
WHERE  company.name = 'neotech'
RETURN friends
```

find all my friends who work for neotech

```
MATCH (person{ name:'Volker' }) -[:friends]
      -(friends) - [:works_for]-> company
WHERE  company.name = 'neotech'
RETURN friends
```

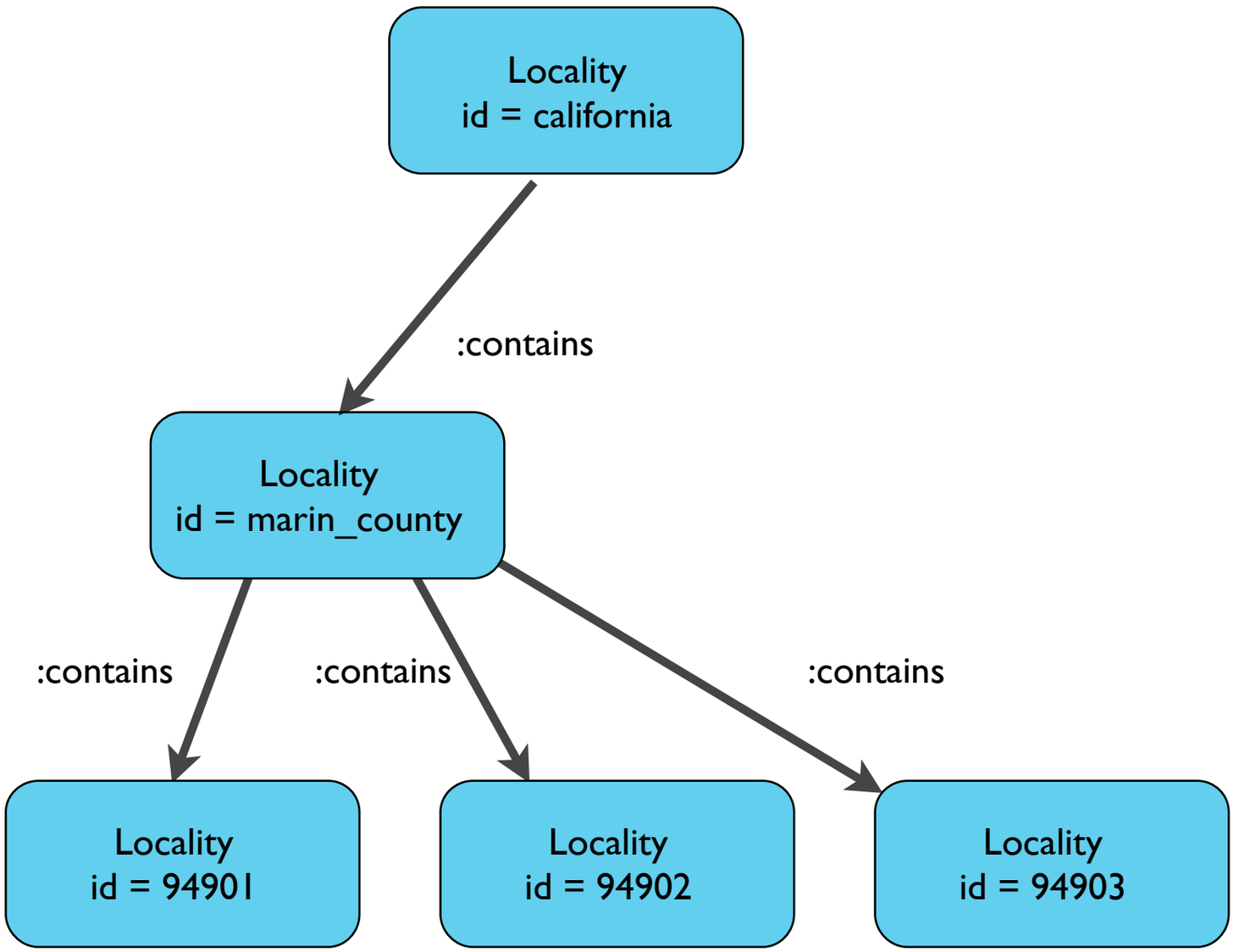


a good place to try it out:

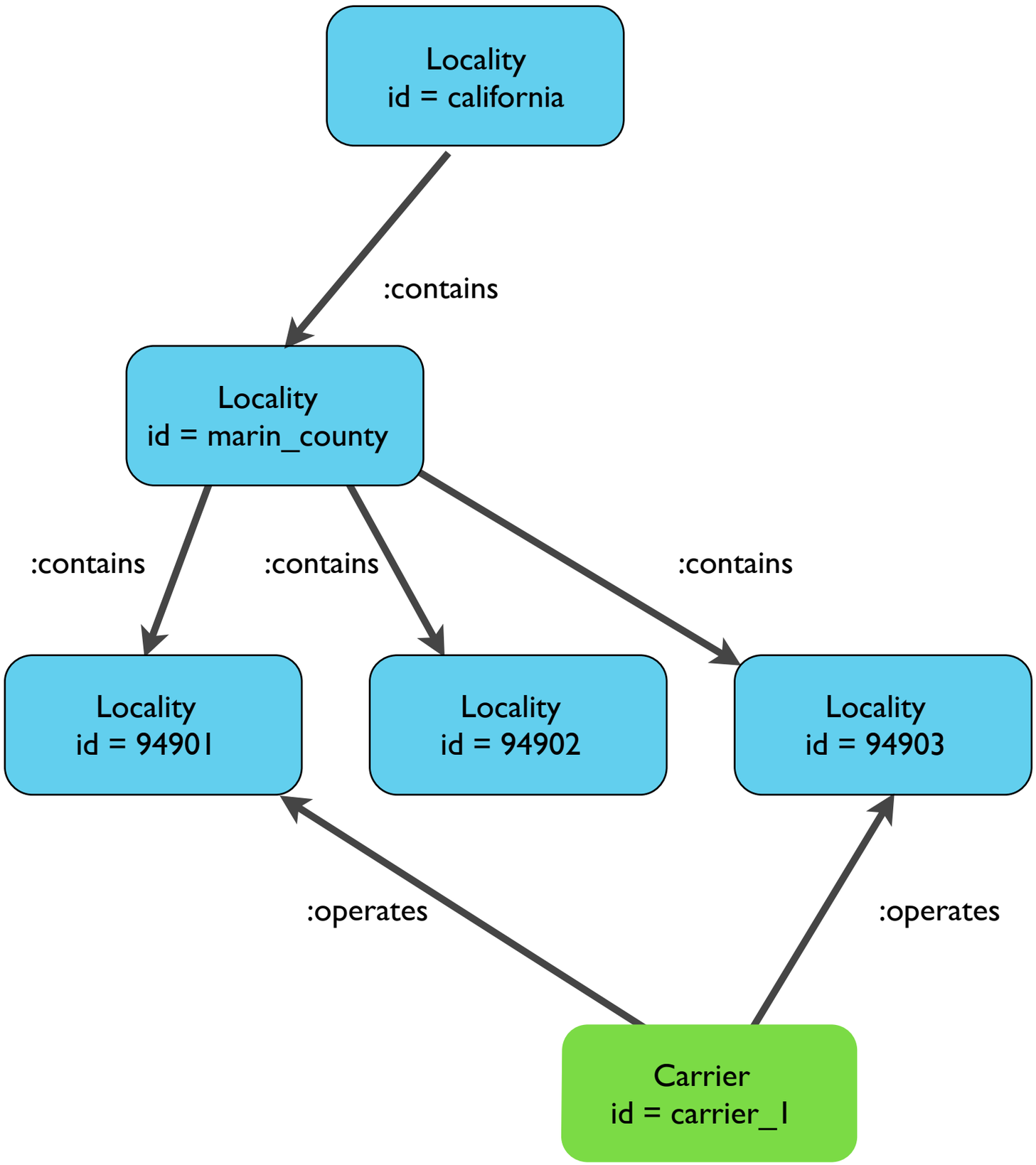
http://console.neo4j.org/

http://gist.neo4j.org/

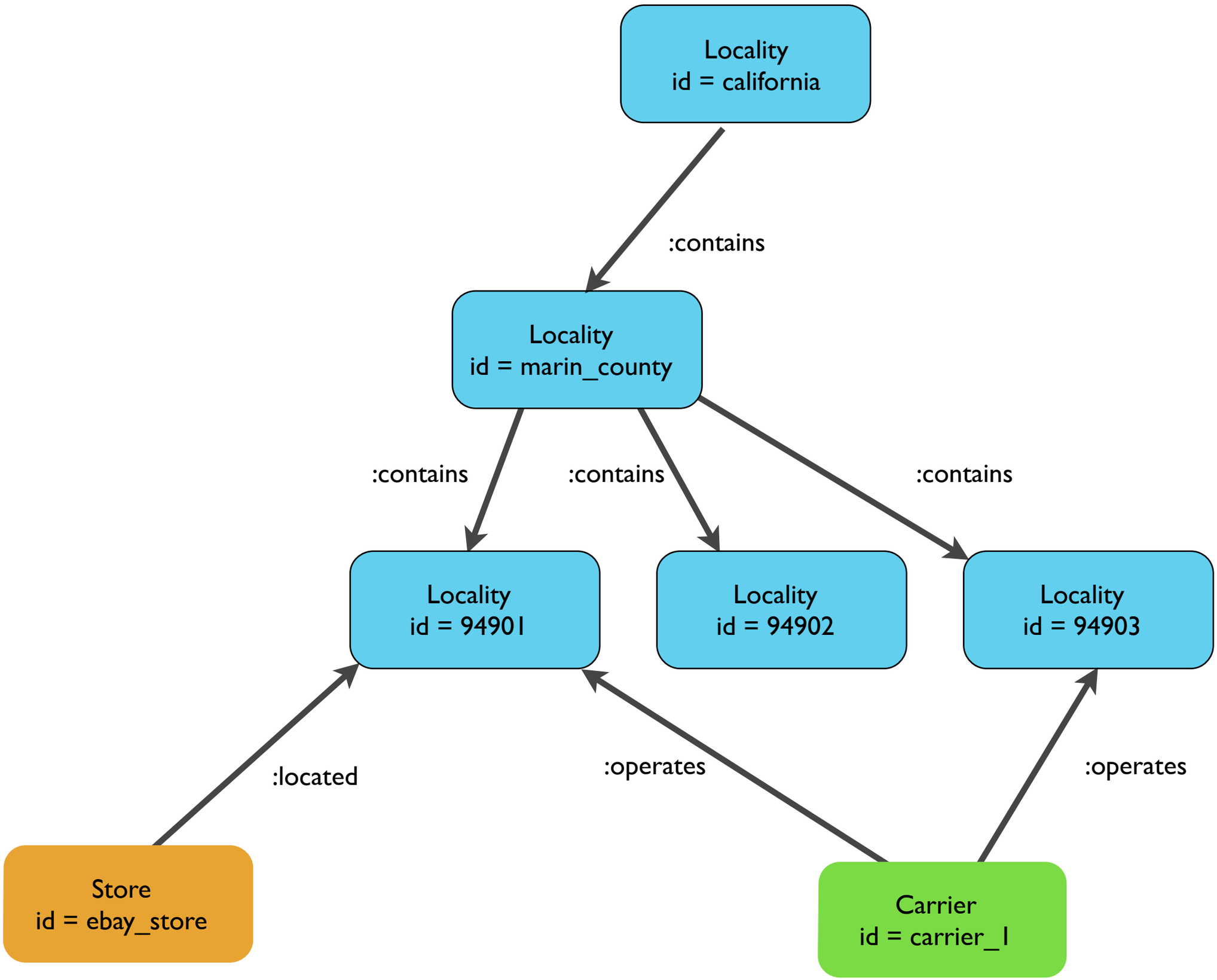
coverage example



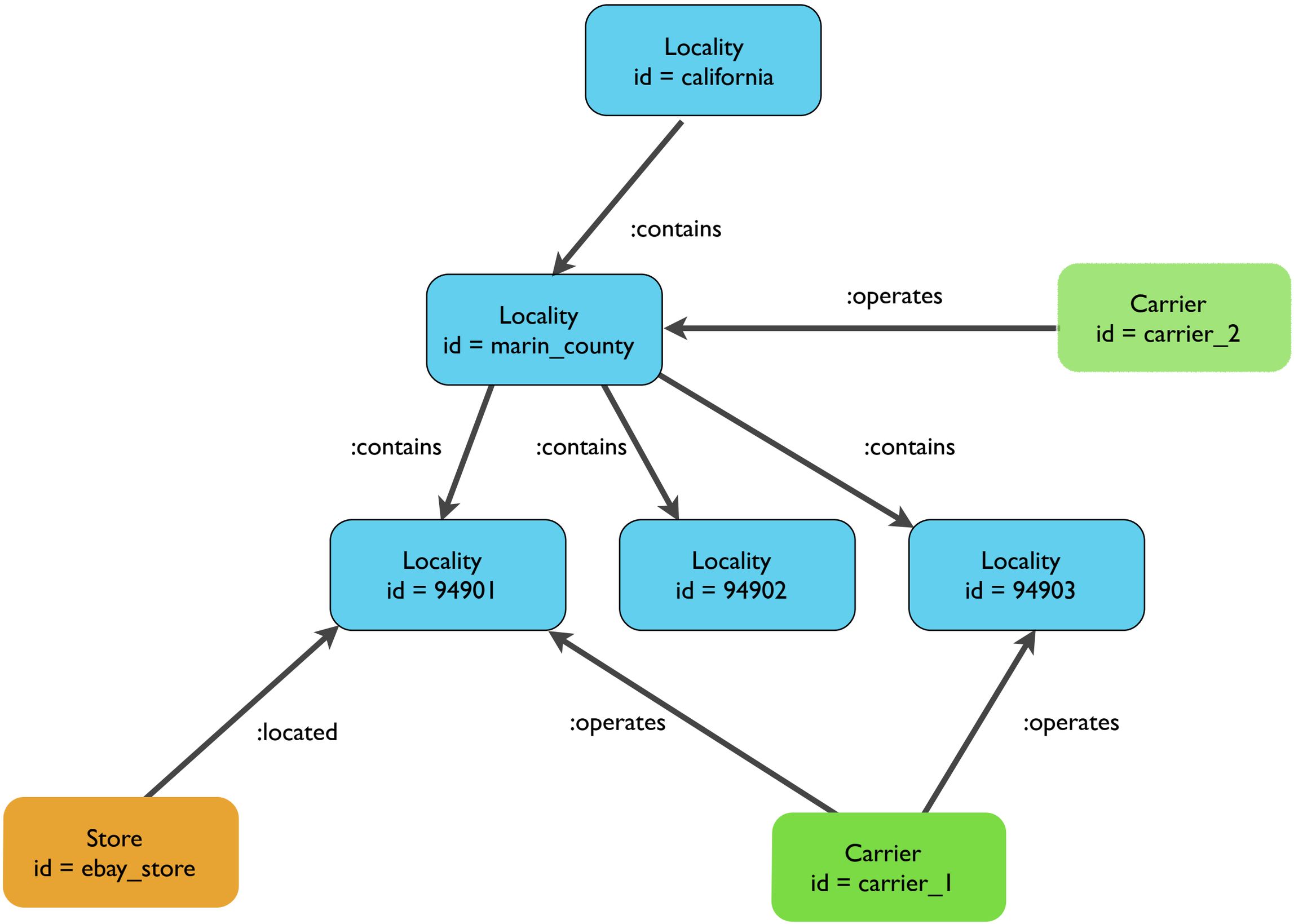
coverage example



coverage example



coverage example



the query

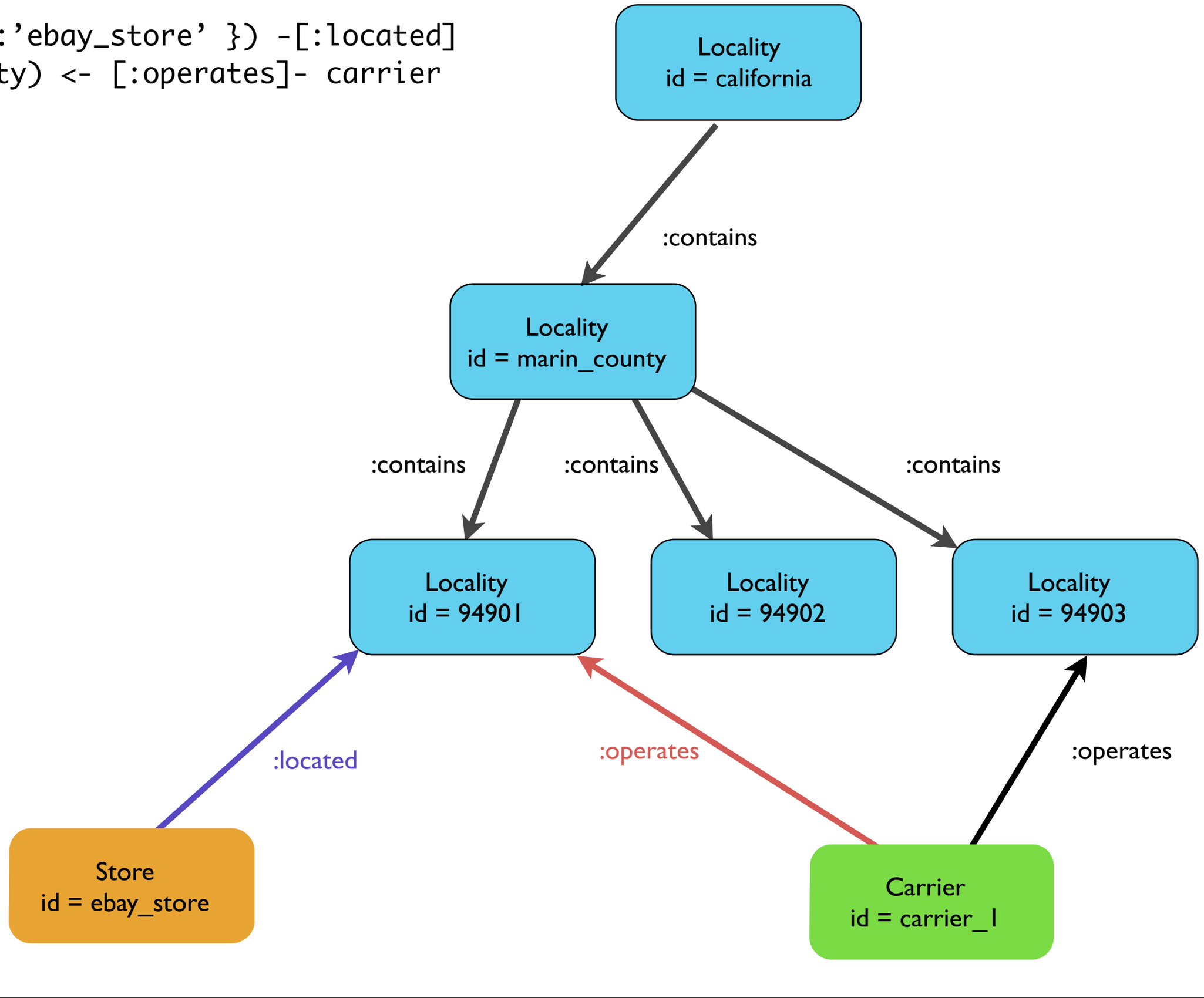
```
MATCH (store{ id:'ebay_store' }) -[:located]
      -> (locality) <- [:operates]- carrier
RETURN carrier
```

the query

```
MATCH (store{ id:'ebay_store' }) -[:located]
      -> (locality) <- [:operates]- carrier
RETURN carrier
```

the query

```
MATCH (store{ id:’ebay_store’ }) -[:located]
      -> (locality) <- [:operates]- carrier
RETURN carrier
```



the query

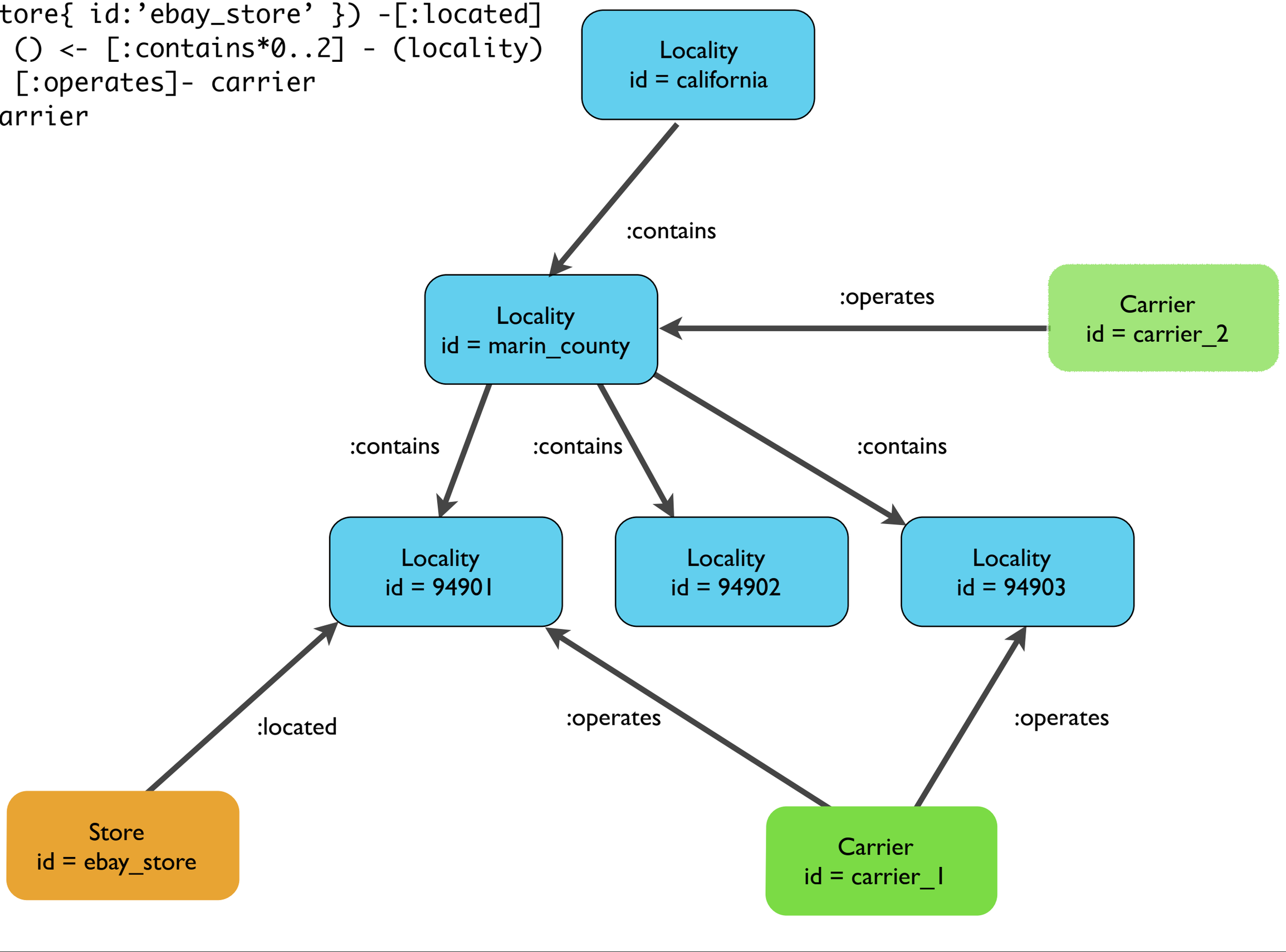
```
MATCH (store{ id:'ebay_store' }) -[:located]
      -> () <- [:contains*0..2] - (locality)
      <- [:operates]- carrier
RETURN carrier
```

the query

```
MATCH (store{ id:’ebay_store’ }) -[:located]
      -> () <- [:contains*0..2] - (locality)
      <- [:operates]- carrier
RETURN carrier
```

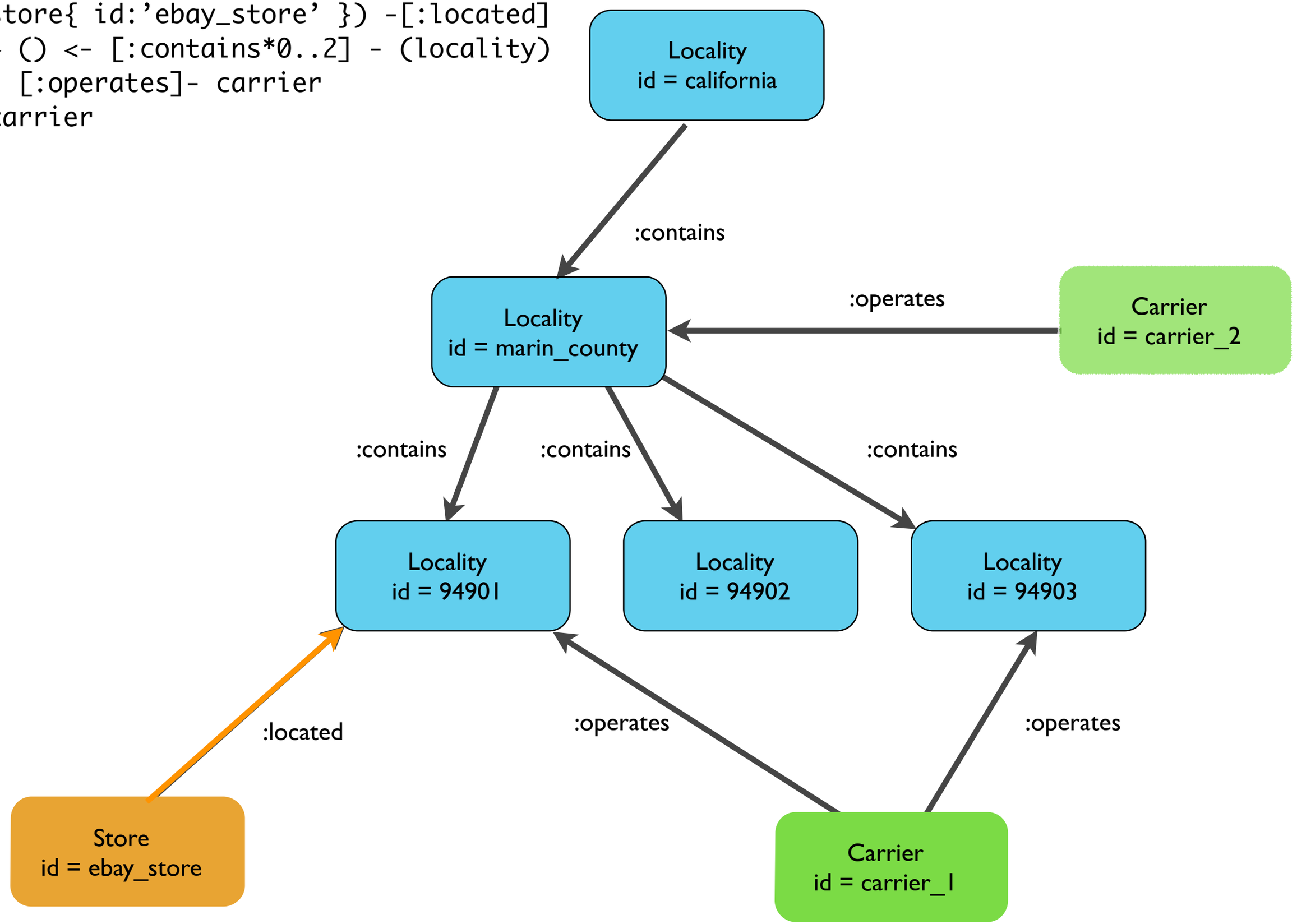
the query

```
MATCH (store{ id:'ebay_store' }) -[:located]
      -> () <- [:contains*0..2] - (locality)
      <- [:operates]- carrier
RETURN carrier
```



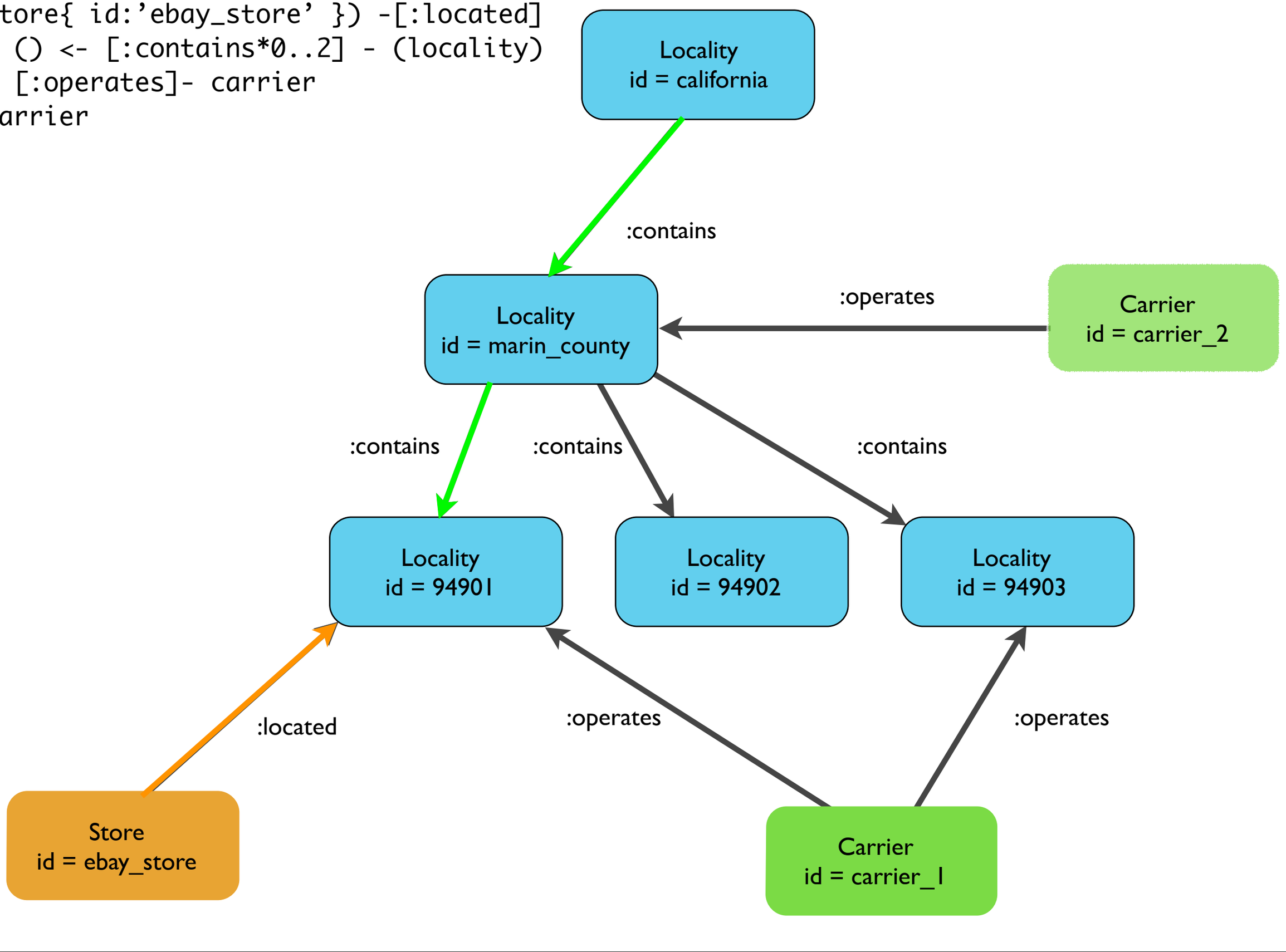
the query

```
MATCH (store{ id:'ebay_store' }) -[:located]
      -> () <- [:contains*0..2] - (locality)
      <- [:operates]- carrier
RETURN carrier
```



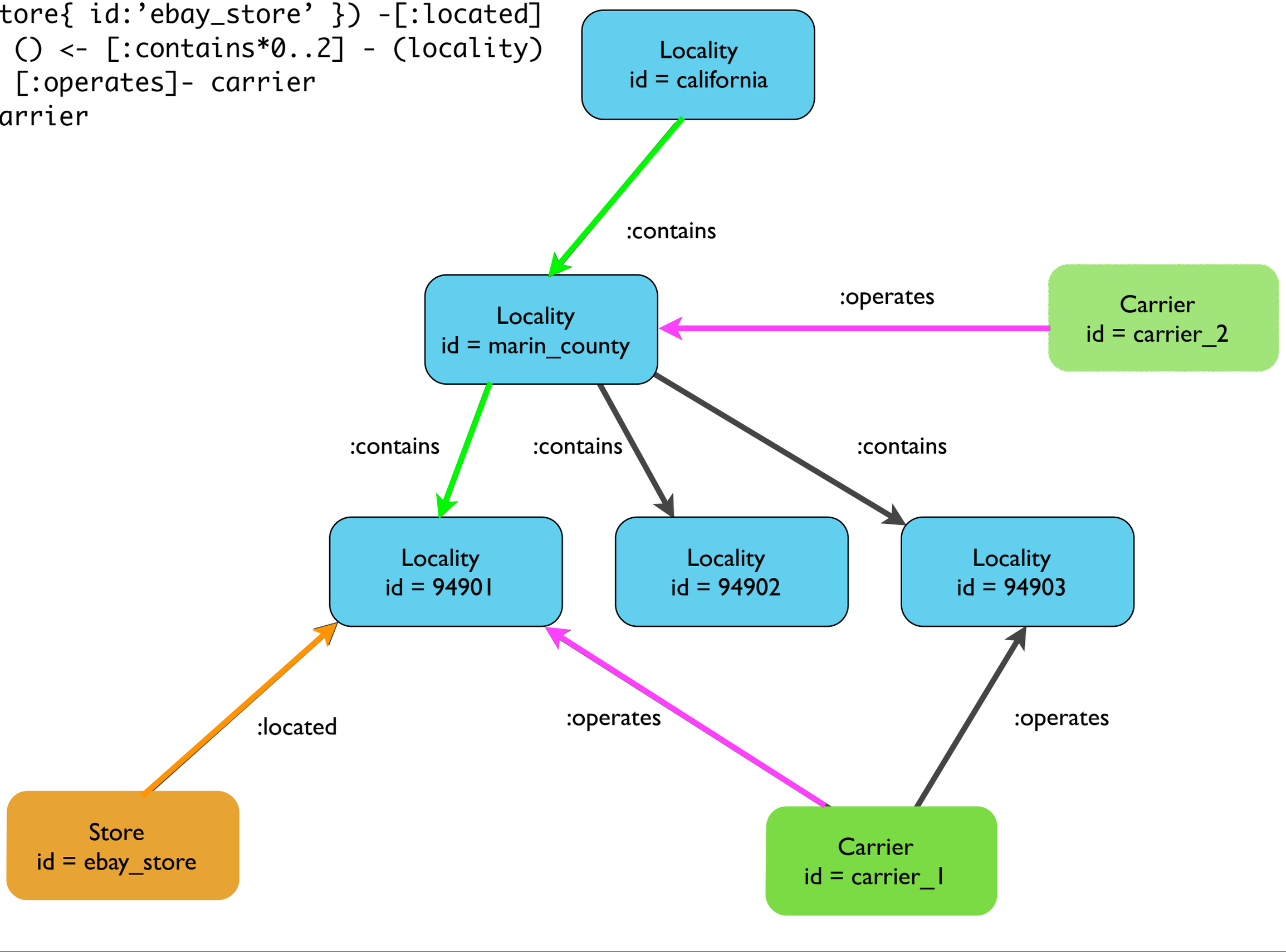
the query

```
MATCH (store{ id:'ebay_store' }) -[:located]
      -> () <- [:contains*0..2] - (locality)
      <- [:operates]- carrier
RETURN carrier
```



the query

```
MATCH (store{ id:'ebay_store' }) -[:located]
      -> () <- [:contains*0..2] - (locality)
      <- [:operates]- carrier
RETURN carrier
```



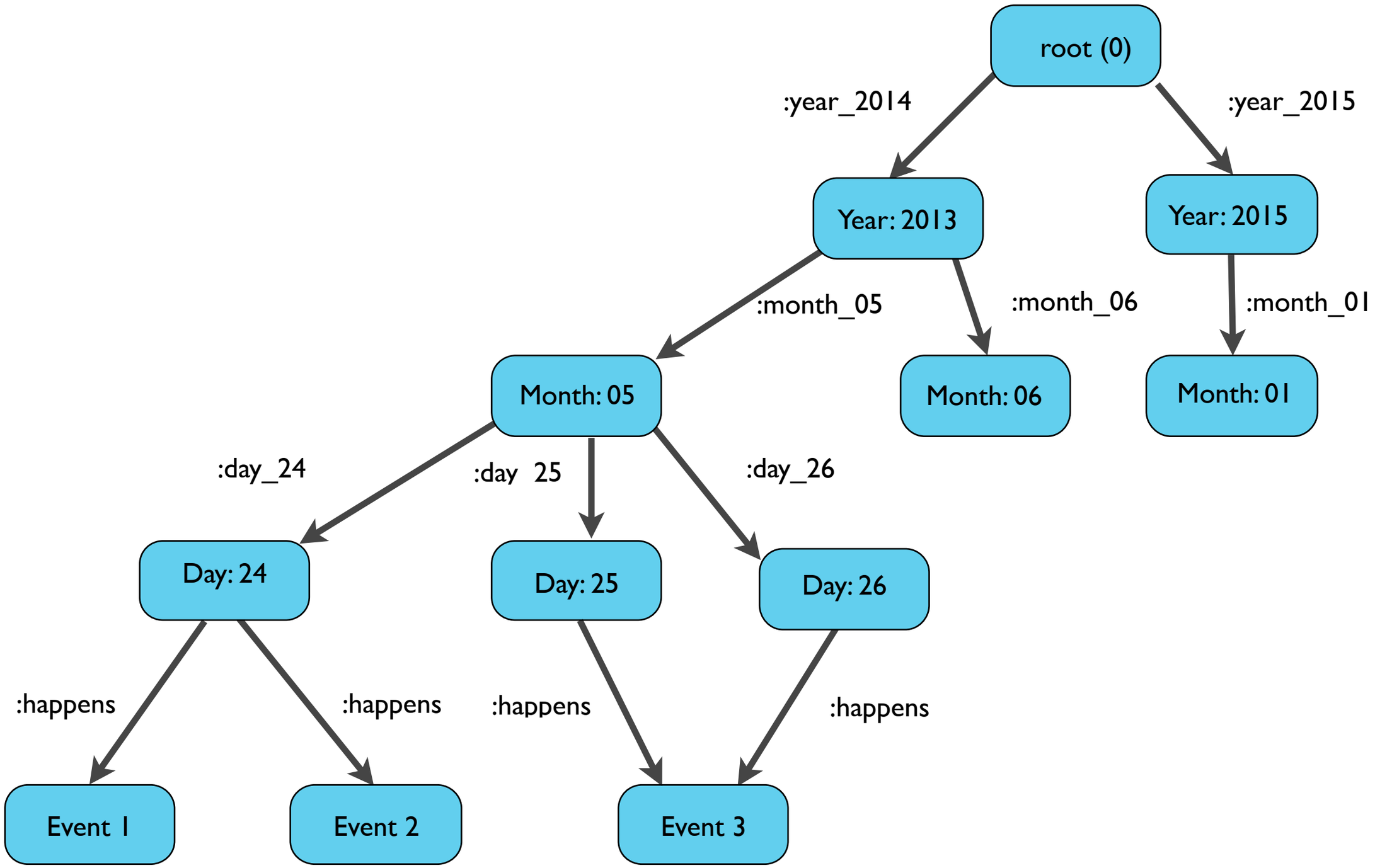
```
SELECT *  FROM carriers
LEFT JOIN locations ON carrier.location_id = location.id
LEFT JOIN stores ON stores.location_id = carrier.location_id
WHERE      stores.name = ‘ebay_store’
```

```
SELECT * FROM carriers
LEFT JOIN locations ON carrier.location_id = location.id  OR
                       carrier.location_id = location.parent_id
LEFT JOIN stores   ON stores.location_id = carrier.location_id
WHERE      stores.name = ‘ebay_store’
```

?

```
MATCH (store{ id:’ebay_store’ }) -[:located]
      -> () <- [:contains*0..2] - (locality)
      <- [:operates]- carrier
RETURN carrier
```

representing dates/times



find all events on a specific day

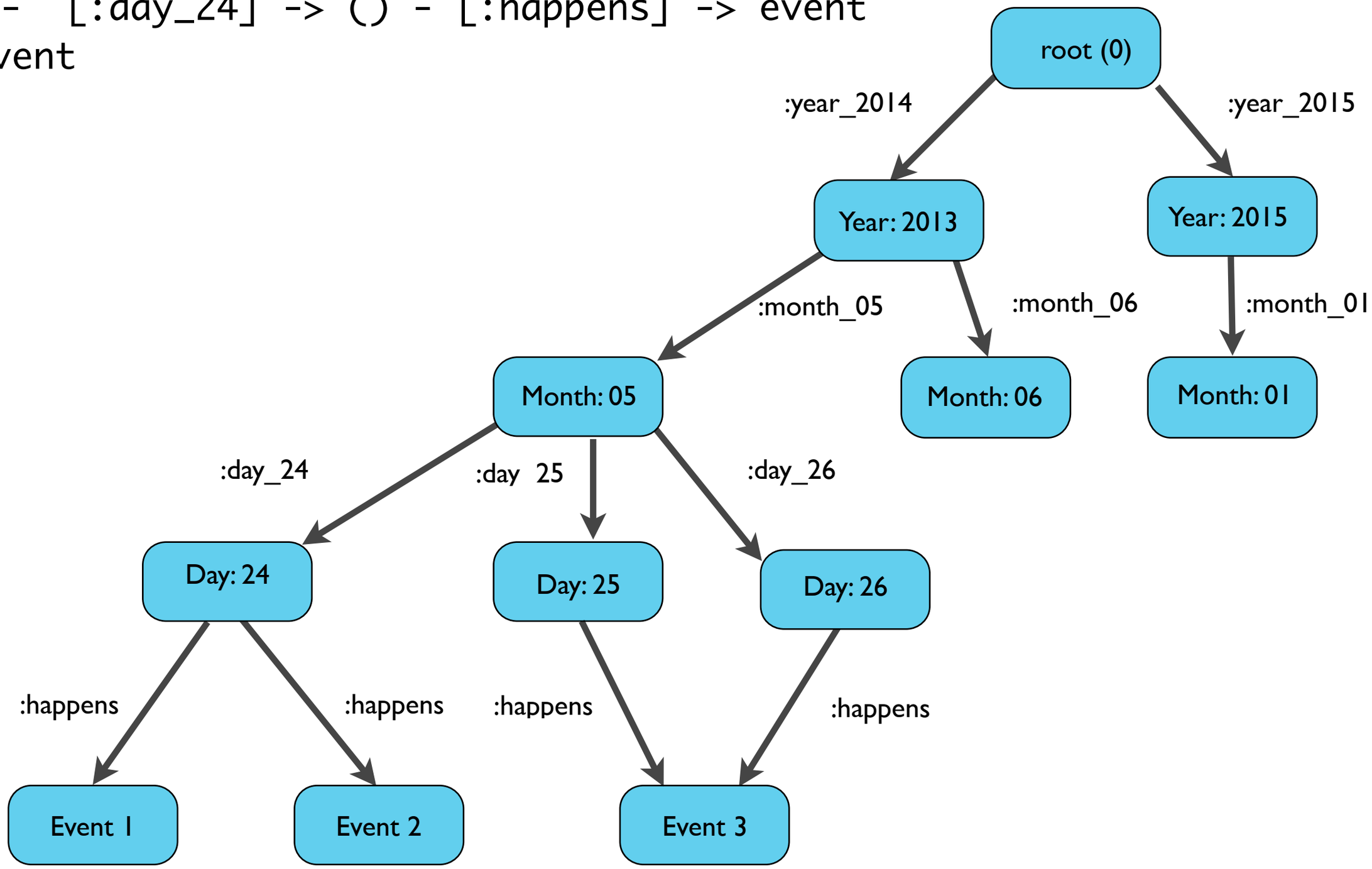
```
START  root=node(0)
MATCH  root - [:year_2014] -> () -[:month_05] ->
       ()-  [:day_24] -> () - [:happens] -> event
RETURN event
```

find all events on a specific day

```
START  root=node(0)
MATCH  root - [:year_2014] -> () -[:month_05] ->
       ()-  [:day_24] -> () - [:happens] -> event
RETURN event
```

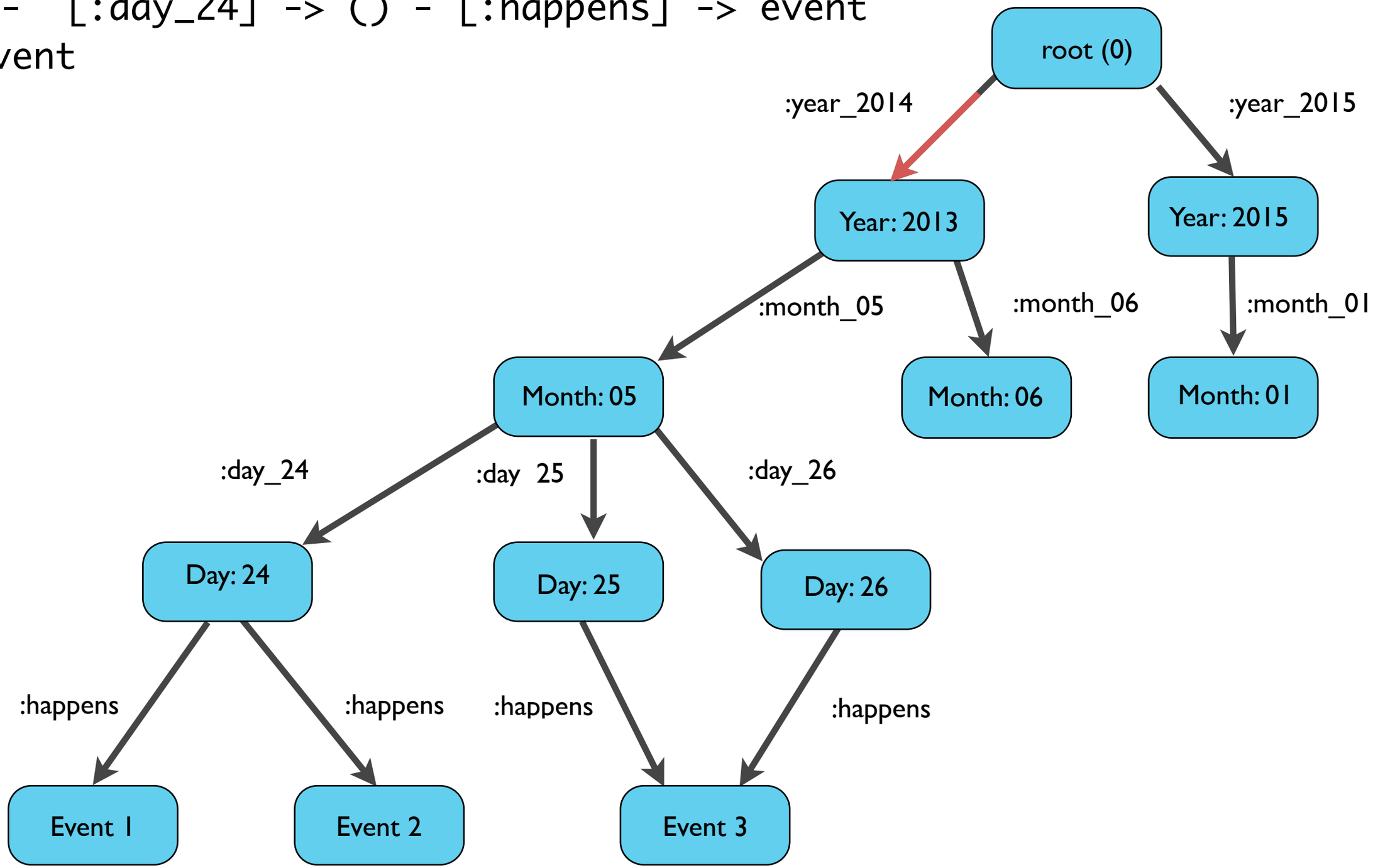
find all events on a specific day

```
START   root=node(0)
MATCH   root - [:year_2014] -> () -[:month_05] ->
       ()-  [:day_24] -> () - [:happens] -> event
RETURN event
```



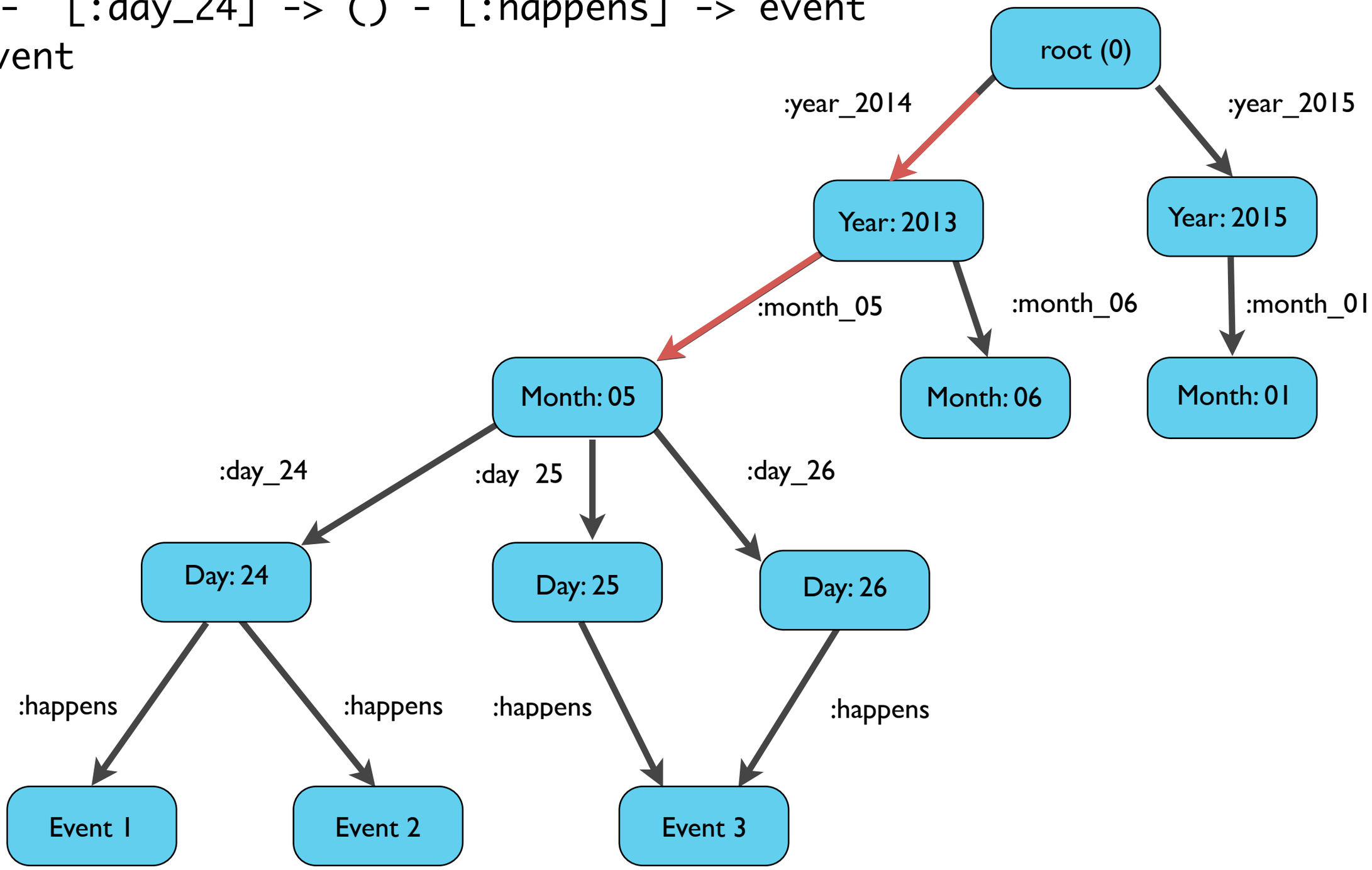
find all events on a specific day

```
START   root=node(0)
MATCH   root - [:year_2014] -> () -[:month_05] ->
       ()-  [:day_24] -> () - [:happens] -> event
RETURN event
```



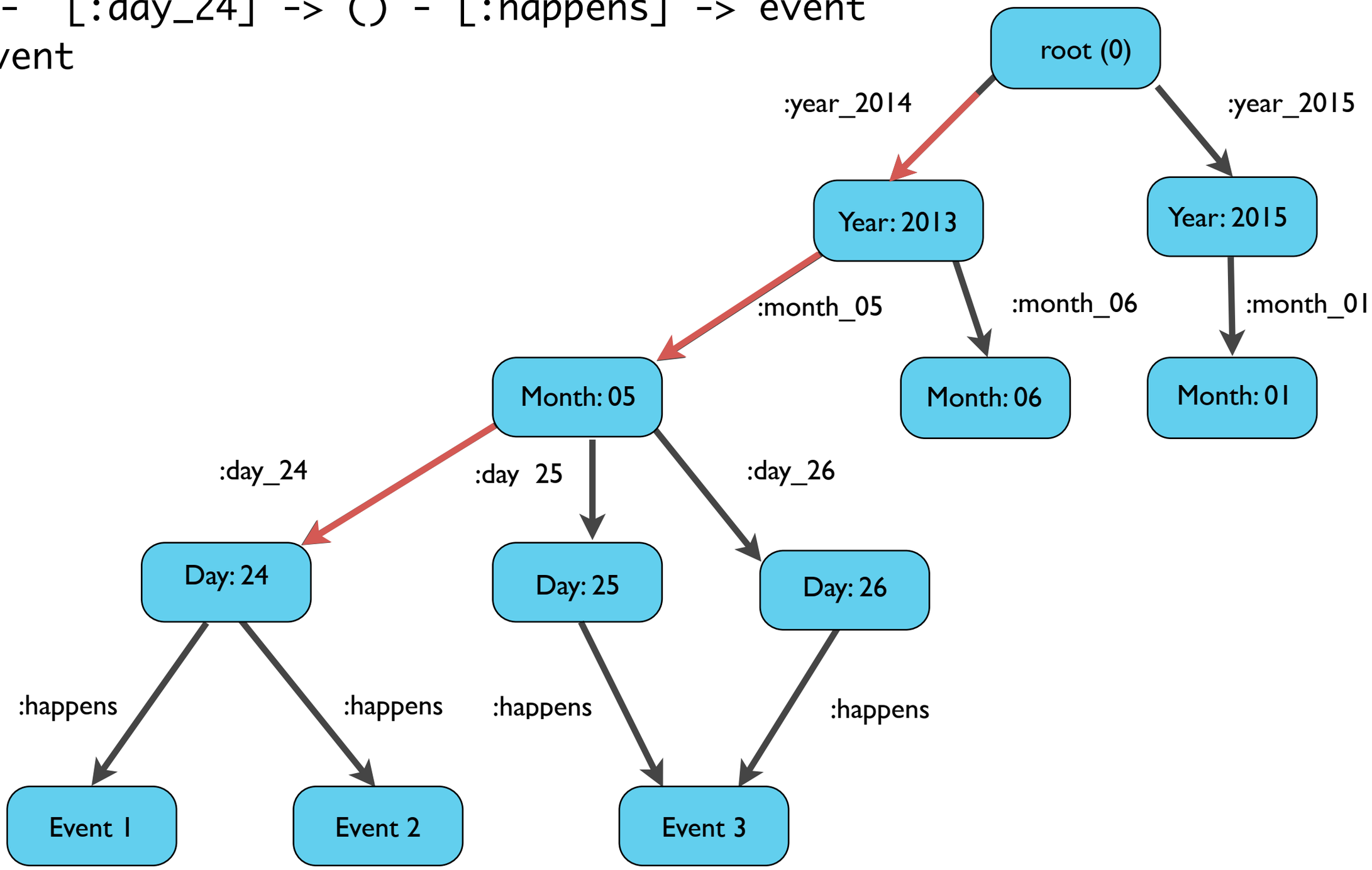
find all events on a specific day

```
START   root=node(0)
MATCH   root - [:year_2014] -> () -[:month_05] ->
       ()-  [:day_24] -> () - [:happens] -> event
RETURN event
```



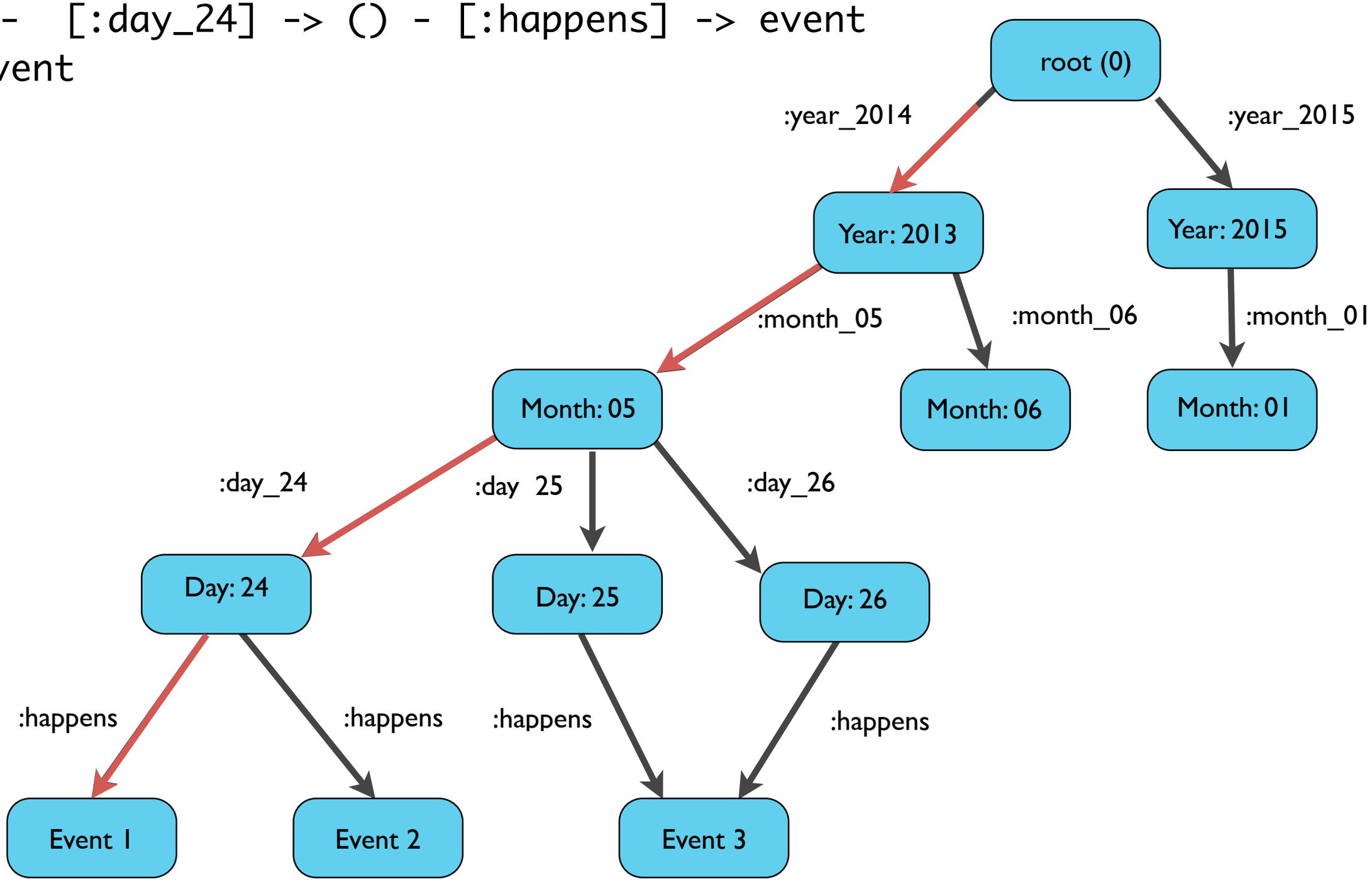
find all events on a specific day

```
START   root=node(0)
MATCH   root - [:year_2014] -> () -[:month_05] ->
       ()-  [:day_24] -> () - [:happens] -> event
RETURN event
```



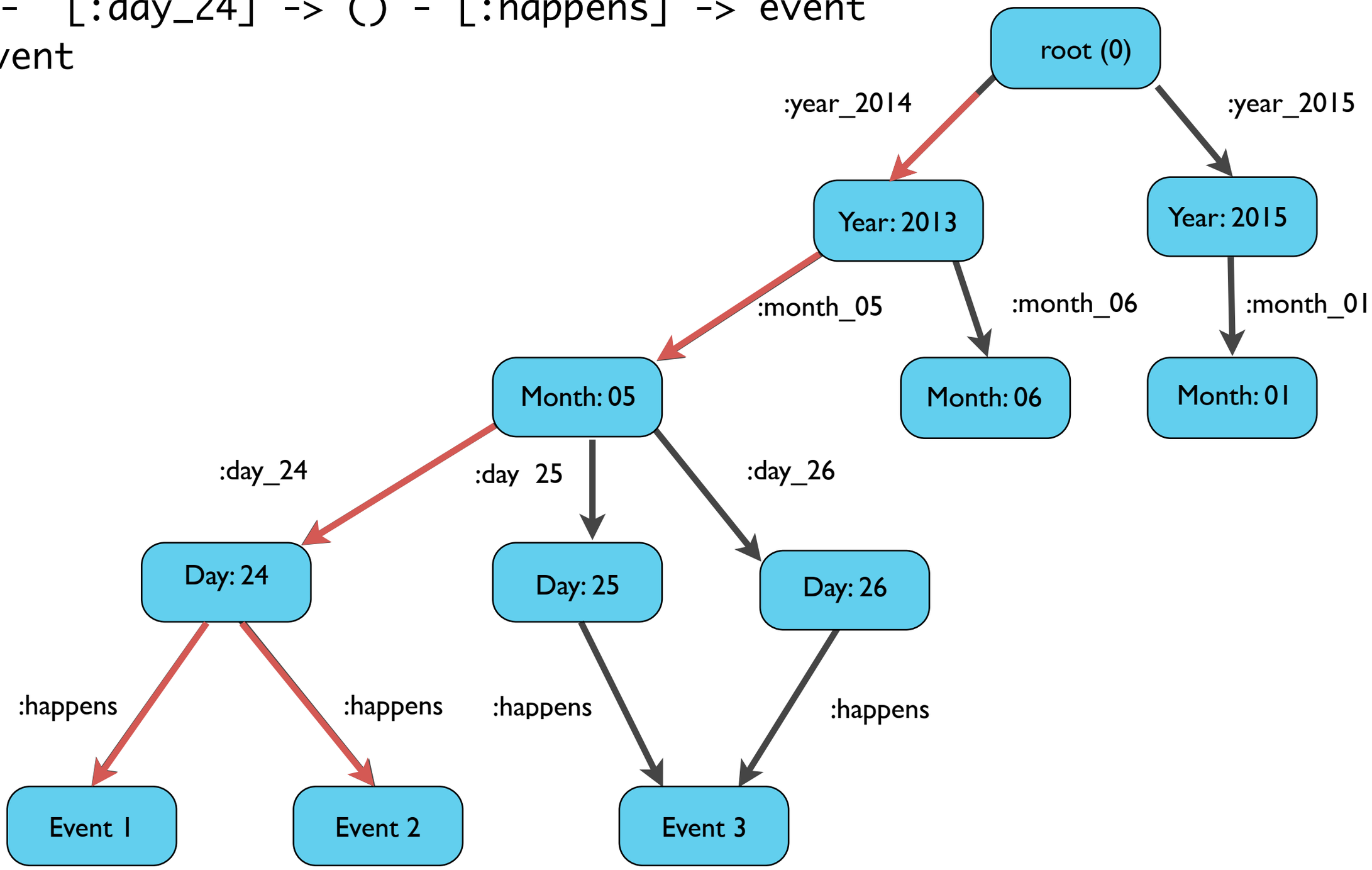
find all events on a specific day

```
START   root=node(0)
MATCH   root - [:year_2014] -> () -[:month_05] ->
       ()-  [:day_24] -> () - [:happens] -> event
RETURN event
```

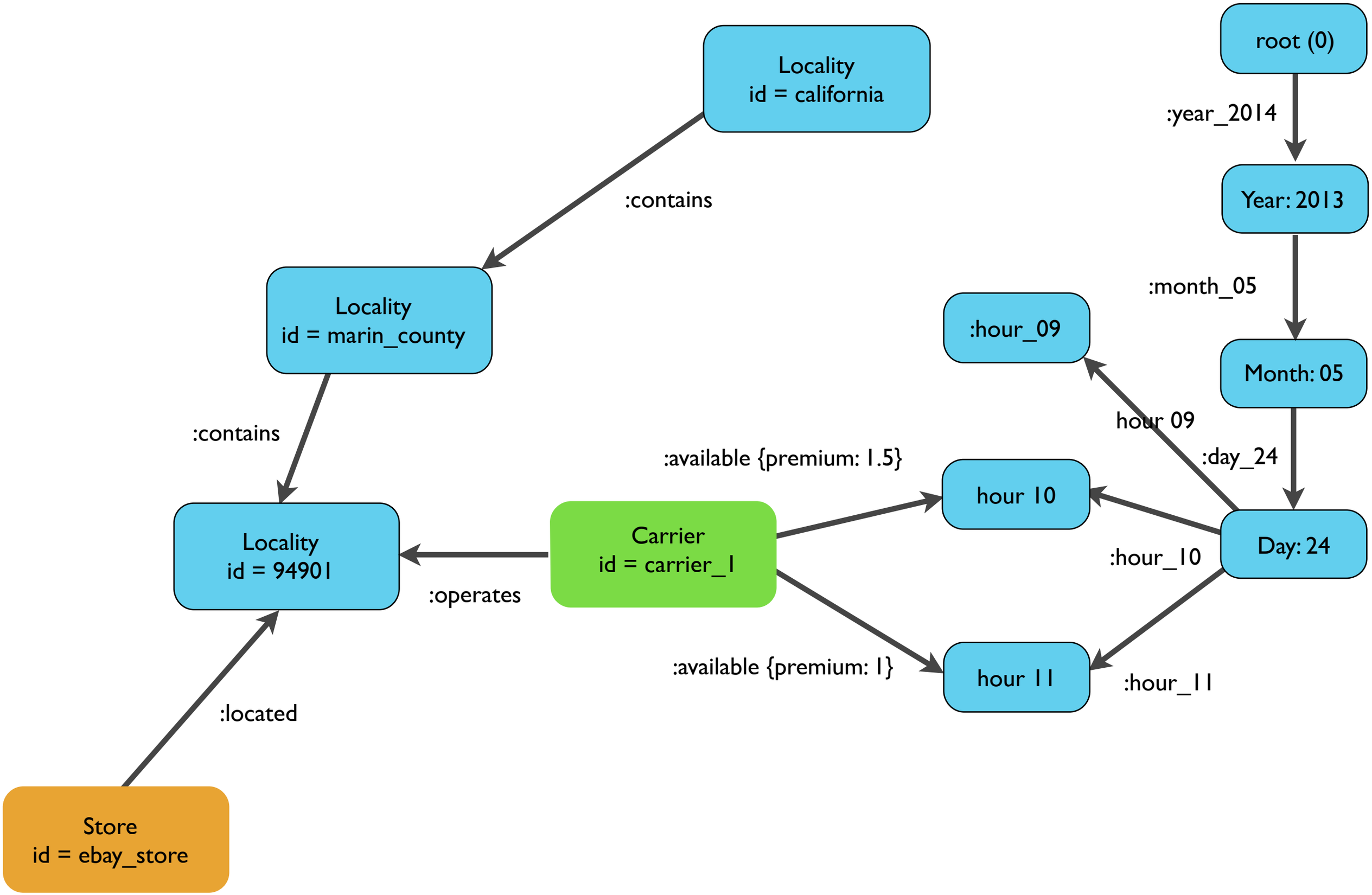


find all events on a specific day

```
START   root=node(0)
MATCH   root - [:year_2014] -> () -[:month_05] ->
       ()-  [:day_24] -> () - [:happens] -> event
RETURN event
```



all together



all together

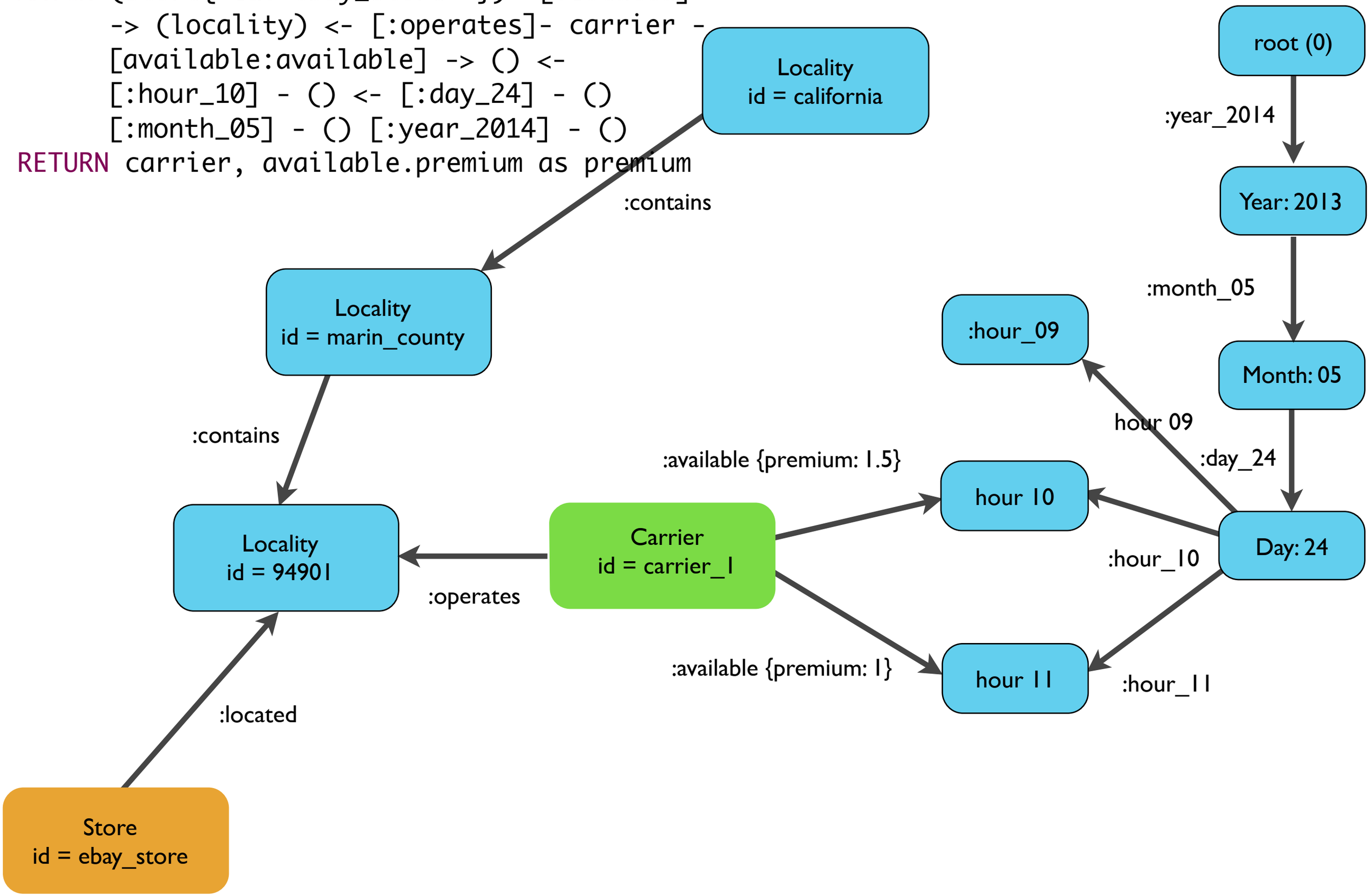
```
MATCH (store{ id:'ebay_store' }) -[:located]
      -> (locality) <- [:operates]- carrier -
      [available:available] -> () <-
      [:hour_10] - () <- [:day_24] - ()
      [:month_05] - () [:year_2014] - ()
RETURN carrier, available.premium as premium
```

all together

```
MATCH (store{ id:’ebay_store’ }) -[:located]
      -> (locality) <- [:operates]- carrier -
      [available:available] -> () <-
      [:hour_10] - () <- [:day_24] - ()
      [:month_05] - () [:year_2014] - ()
RETURN carrier, available.premium as premium
```

all together

```
MATCH (store{ id:'ebay_store' }) -[:located]
      -> (locality) <- [:operates]- carrier -
      [available:available] -> () <-
      [:hour_10] - () <- [:day_24] - ()
      [:month_05] - () [:year_2014] - ()
RETURN carrier, available.premium as premium
```



Other graph uses

Other graph uses

- Recommendation engines

Other graph uses

- Recommendation engines
- Organisational analysis

Other graph uses

- Recommendation engines
- Organisational analysis
- Graphing your infrastructure

Some gotchas

Some gotchas

- There was a learning curve in switching from a relational mentality to a graph one

Some gotchas

- There was a learning curve in switching from a relational mentality to a graph one

- Tooling not as mature as in the relational world

Some gotchas

- There was a learning curve in switching from a relational mentality to a graph one

- Tooling not as mature as in the relational world

- No out of the box solution for db migrations

Some gotchas

- There was a learning curve in switching from a relational mentality to a graph one
- Tooling not as mature as in the relational world
- No out of the box solution for db migrations
- Seeding an embedded database was unfamiliar

Testing was a challenge

Testing was a challenge

- Setting up scenarios for tests was tedious

Testing was a challenge

- Setting up scenarios for tests was tedious
- Built our own tool based on the geoff syntax developed by Nigel Small

Testing was a challenge

- Setting up scenarios for tests was tedious
- Built our own tool based on the geoff syntax developed by Nigel Small
- Geoff allows modelling of graphs in textual form and provides an interface to insert them into an existing graph

Testing was a challenge

- Setting up scenarios for tests was tedious
- Built our own tool based on the geoff syntax developed by Nigel Small
- Geoff allows modelling of graphs in textual form and provides an interface to insert them into an existing graph

(A) {“name”: “Alice”}

Testing was a challenge

- Setting up scenarios for tests was tedious
- Built our own tool based on the geoff syntax developed by Nigel Small
- Geoff allows modelling of graphs in textual form and provides an interface to insert them into an existing graph

(A) {“name”: “Alice”}

(B) {“name”: “Bob”}

Testing was a challenge

- Setting up scenarios for tests was tedious
- Built our own tool based on the geoff syntax developed by Nigel Small
- Geoff allows modelling of graphs in textual form and provides an interface to insert them into an existing graph

(A) {“name”: “Alice”}

(B) {“name”: “Bob”}

(A) -[:KNOWS] -> (B)

Testing was a challenge

- Setting up scenarios for tests was tedious
- Built our own tool based on the geoff syntax developed by Nigel Small
- Geoff allows modelling of graphs in textual form and provides an interface to insert them into an existing graph

(A) {“name”: “Alice”}

(B) {“name”: “Bob”}

(A) -[:KNOWS] -> (B)

- We created a Ruby dsl for modelling a graph and inserting it into the db that works with factory_girl

Testing was a challenge

- Setting up scenarios for tests was tedious
- Built our own tool based on the geoff syntax developed by Nigel Small
- Geoff allows modelling of graphs in textual form and provides an interface to insert them into an existing graph

(A) {“name”: “Alice”}

(B) {“name”: “Bob”}

(A) -[:KNOWS] -> (B)

- We created a Ruby dsl for modelling a graph and inserting it into the db that works with factory_girl
- Open source - https://github.com/shutl/geoff

Wrap Up

Wrap Up

- Neo4j and graph theory enabled Shutl to achieve big performance increases in its most important operation - calculating delivery prices

Wrap Up

- Neo4j and graph theory enabled Shutl to achieve big performance increases in its most important operation - calculating delivery prices
- It’s a new tool based on tested theory, and cypher is the first language that allows you to query graphs in a declarative way (like SQL)

Wrap Up

- Neo4j and graph theory enabled Shutl to achieve big performance increases in its most important operation - calculating delivery prices

- It's a new tool based on tested theory, and cypher is the first language that allows you to query graphs in a declarative way (like SQL)

- Tooling and adoption is immature but getting better all the time

Thank you!

Any questions?

Sam Phillips
Head of Engineering

@samsworldofno
http://samsworldofno.com
sam@shutl.com

Volker Pacher
Senior Developer

@vpacher
https://github.com/vpacher
volker@shutl.com

QCon

Please evaluate my talk via the mobile app!

QCon

Please evaluate *our* talk via the mobile app!