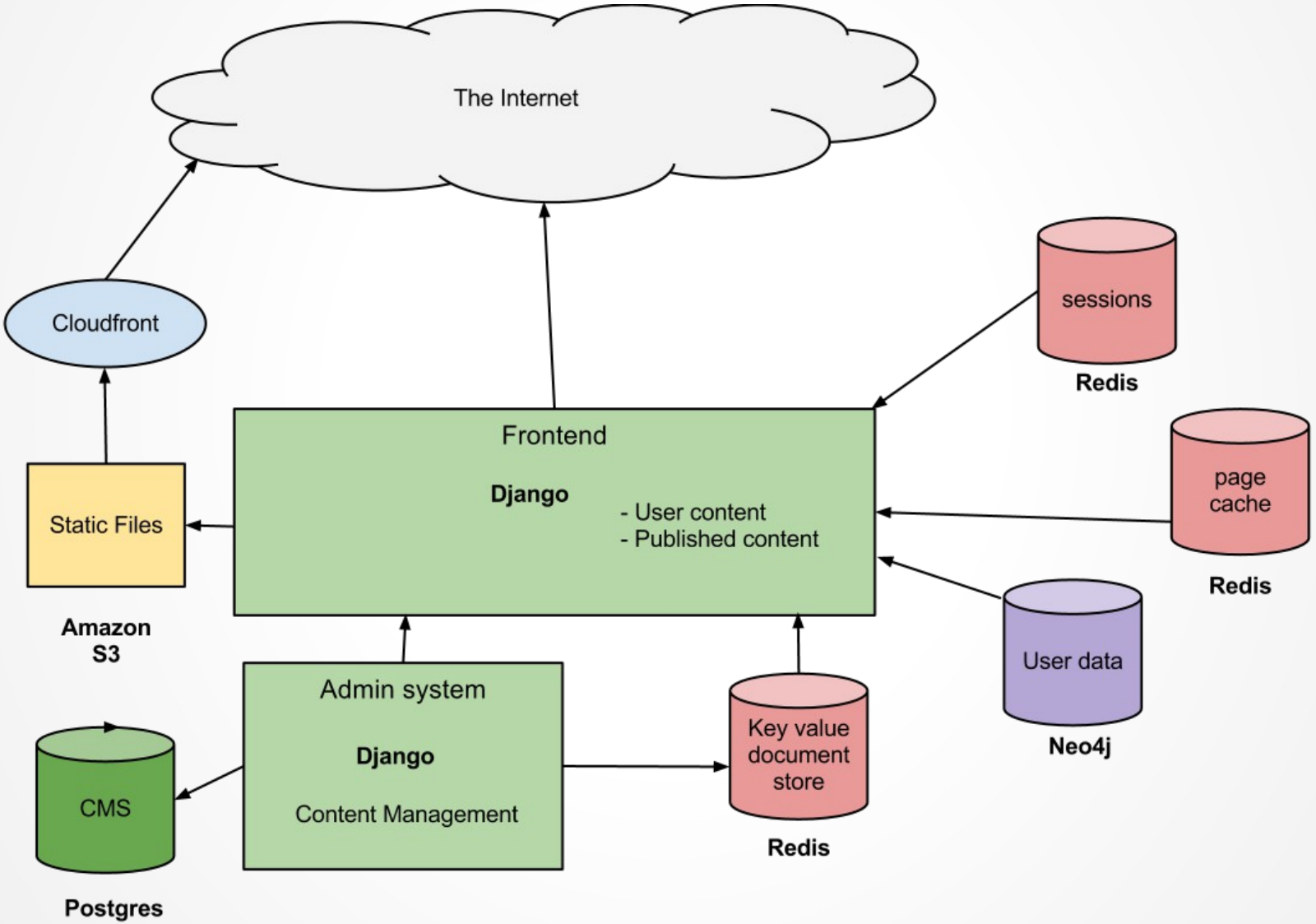
Neo4j at Sharehoods

github.com/robinedwards

Starting out

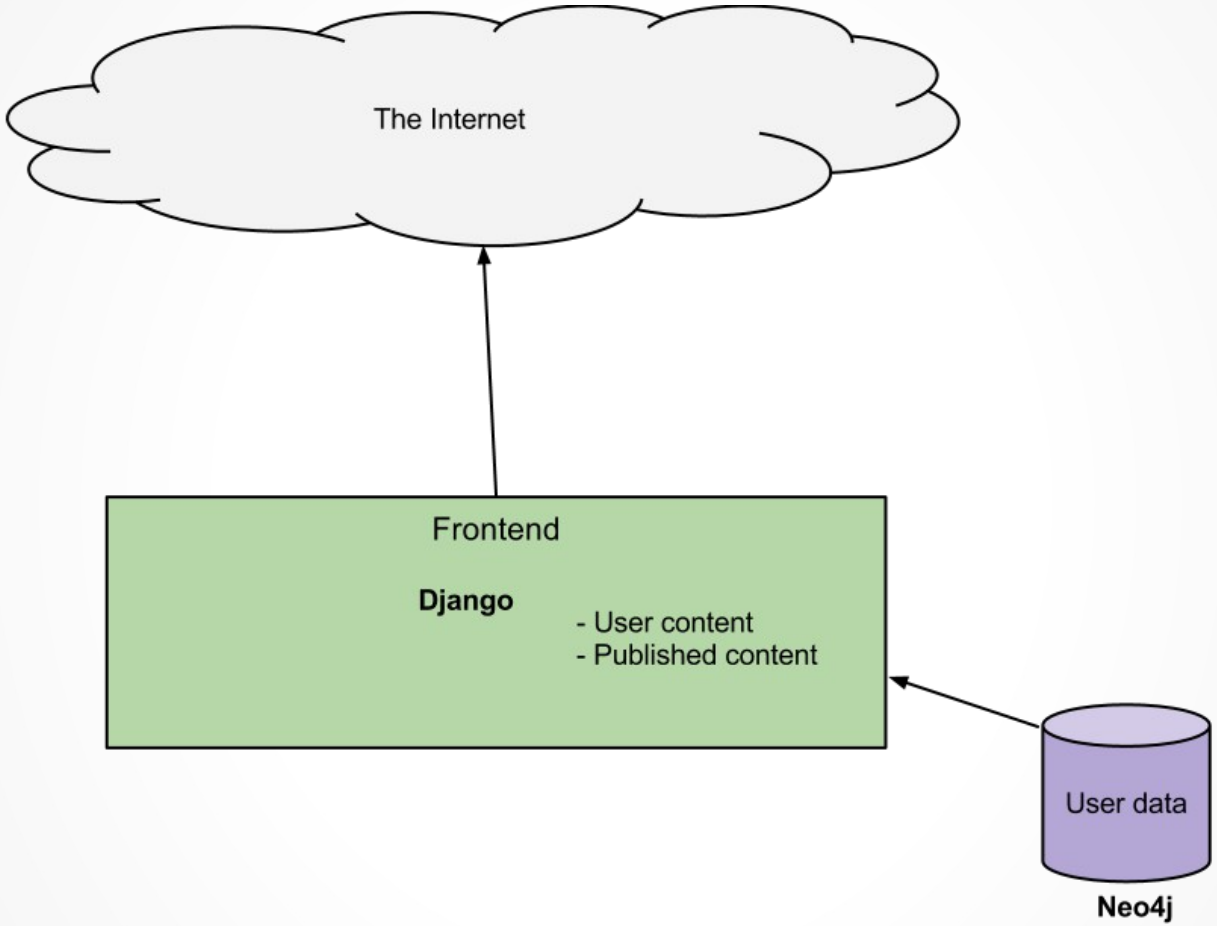
- Python - easy learning curve
- Django – widely used, good docs and support available
- Celery – background tasks
- Two main Django applications
- Front-end and Backend (CMS)
- Heroku – no sysadmins, or late nights (yet!)

Architecture

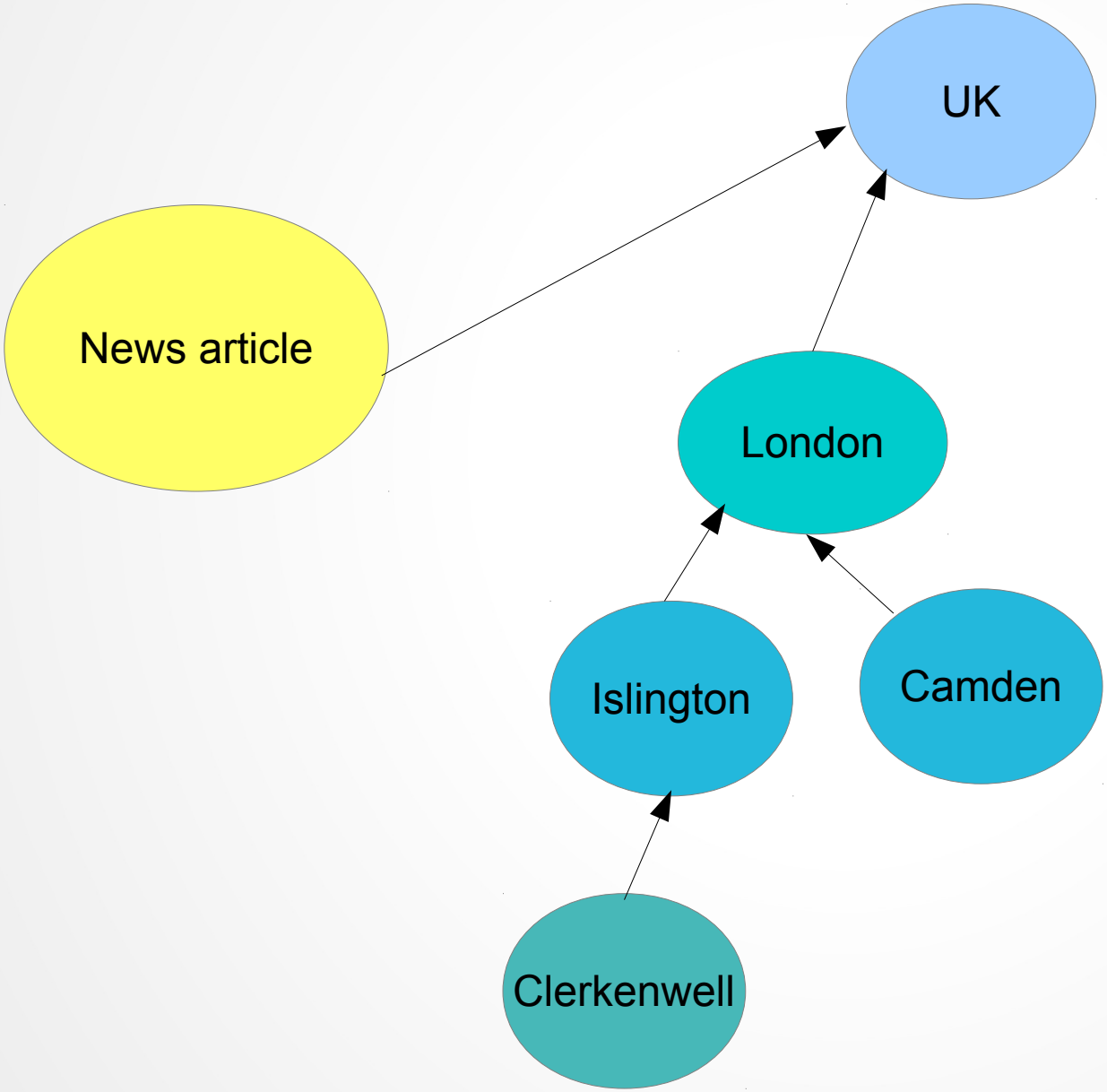


github.com/robinedwards

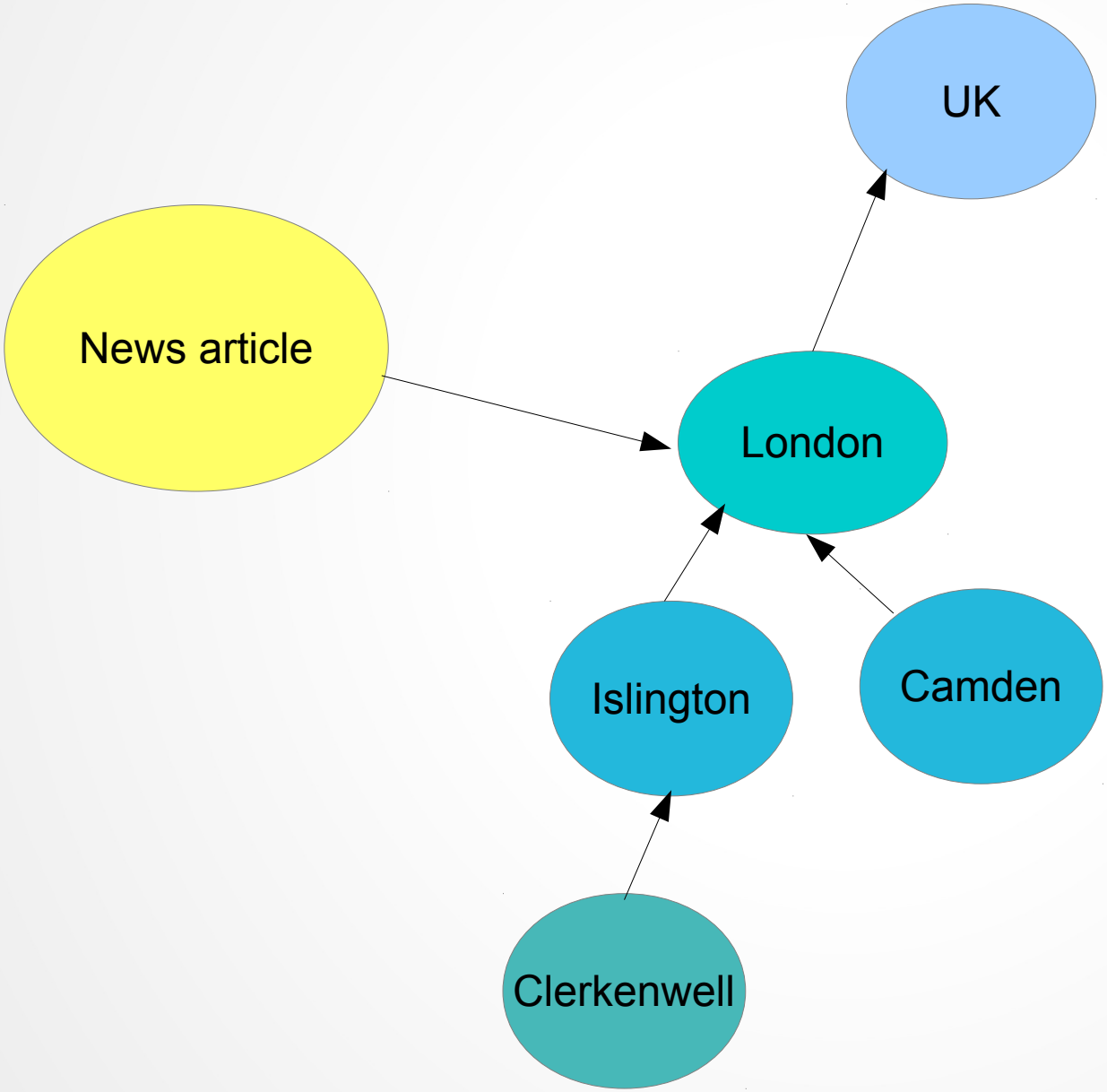
Architecture



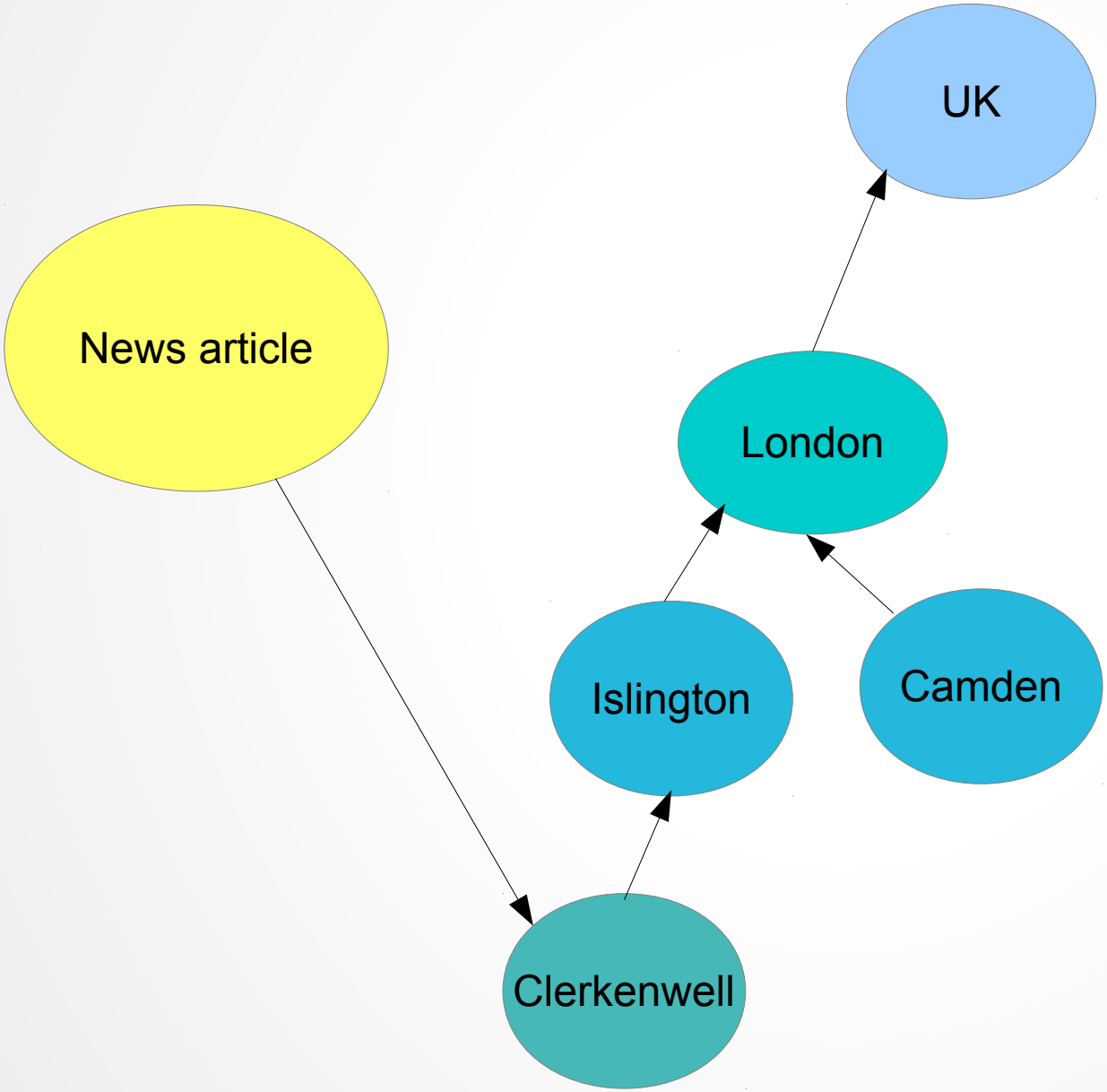
Why Neo4j?



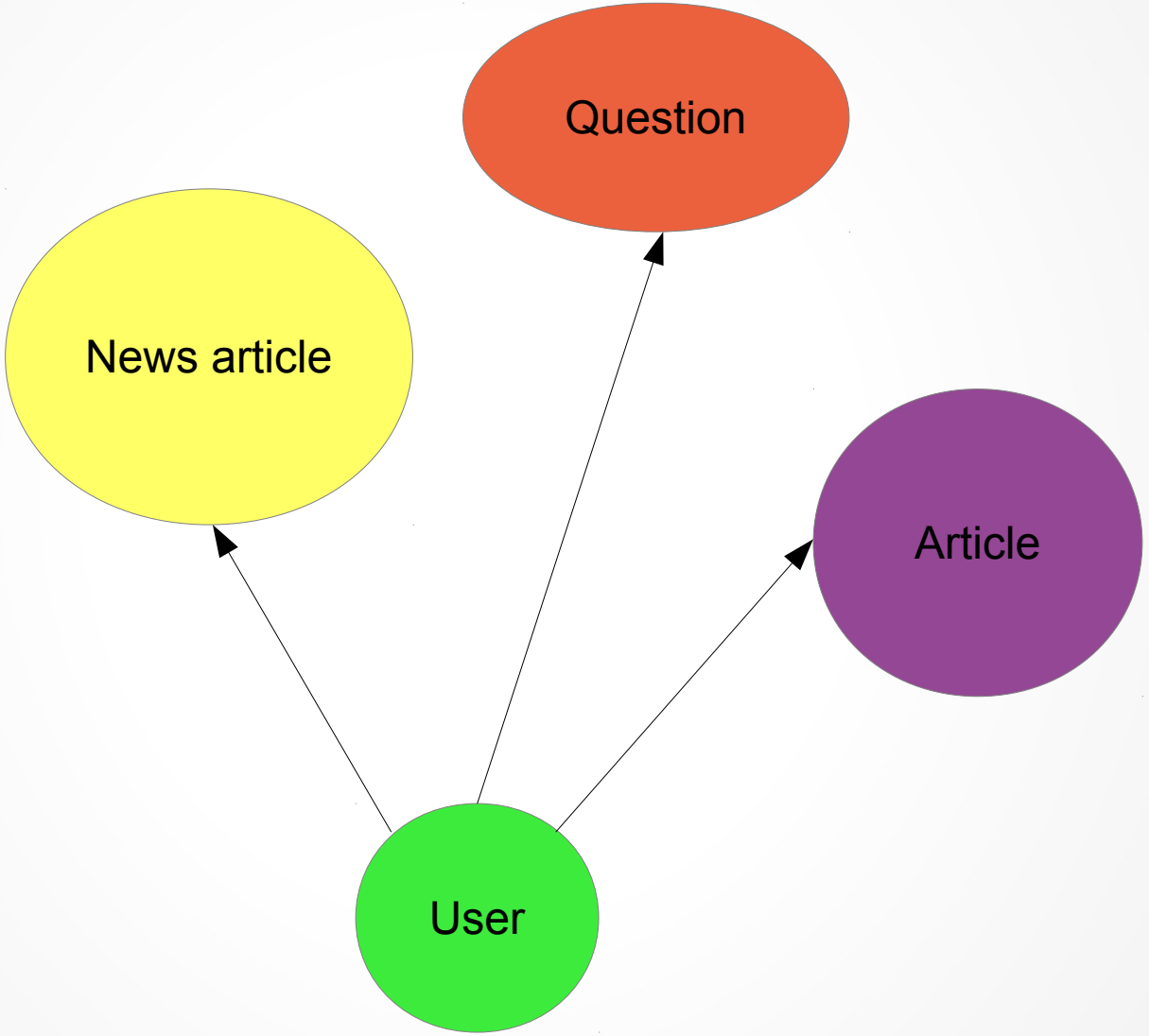
Location Tree



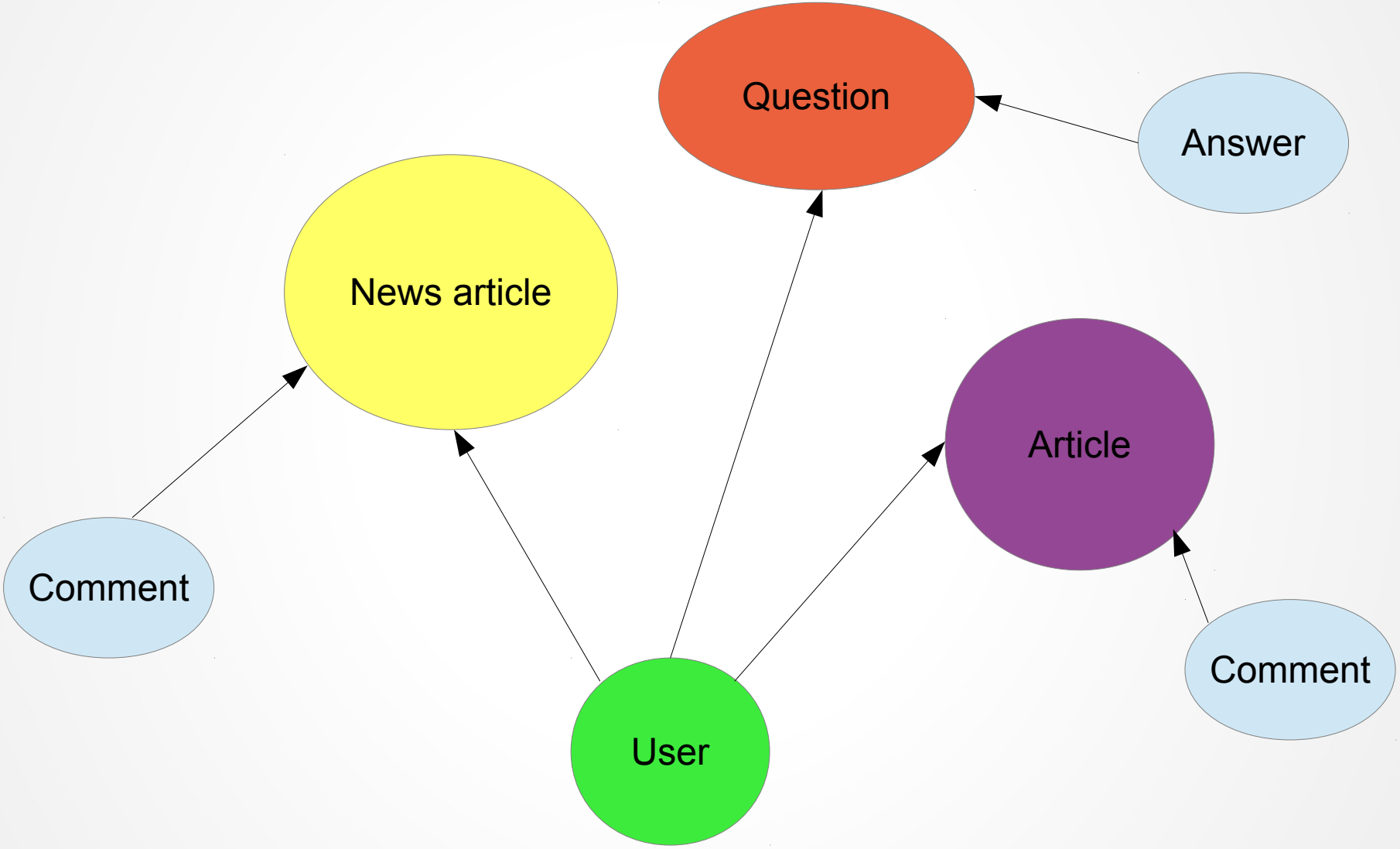
Location Tree



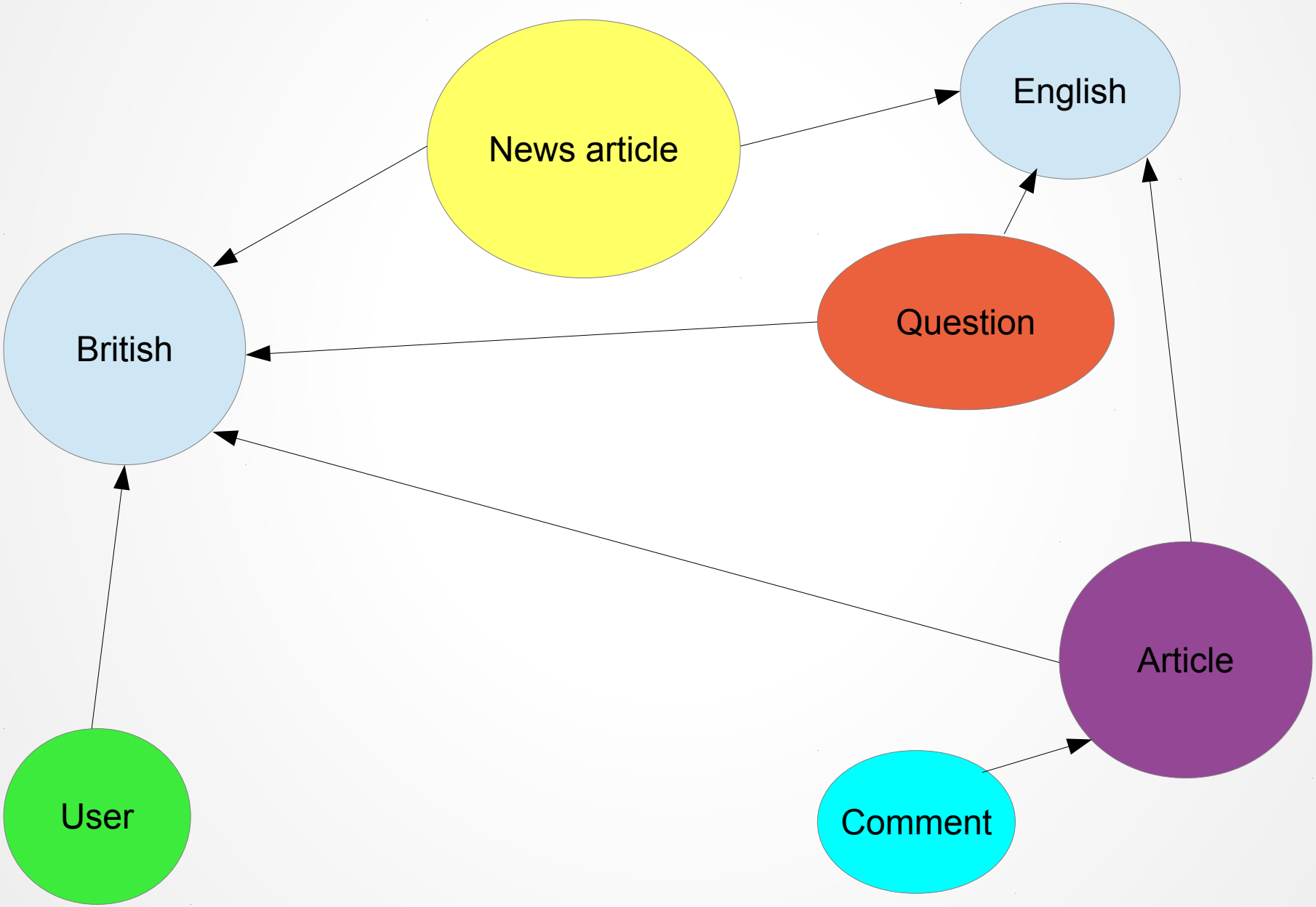
'Follow' system



'Follow' system



Context sensitive content



py2neo

- Great python client for neo4j
- Lots of good documentation
- Easy to read code
- Go check it out!
- http://py2neo.org

github.com/robinedwards

neomodel

github.com/robinedwards/neomodel

Shopping list

- Property validation and indexing
- Easy access to indexes
- Categorisation of nodes
- Simple relationship operations
- connect / disconnect / search etc
- Easy access to cypher

The basics

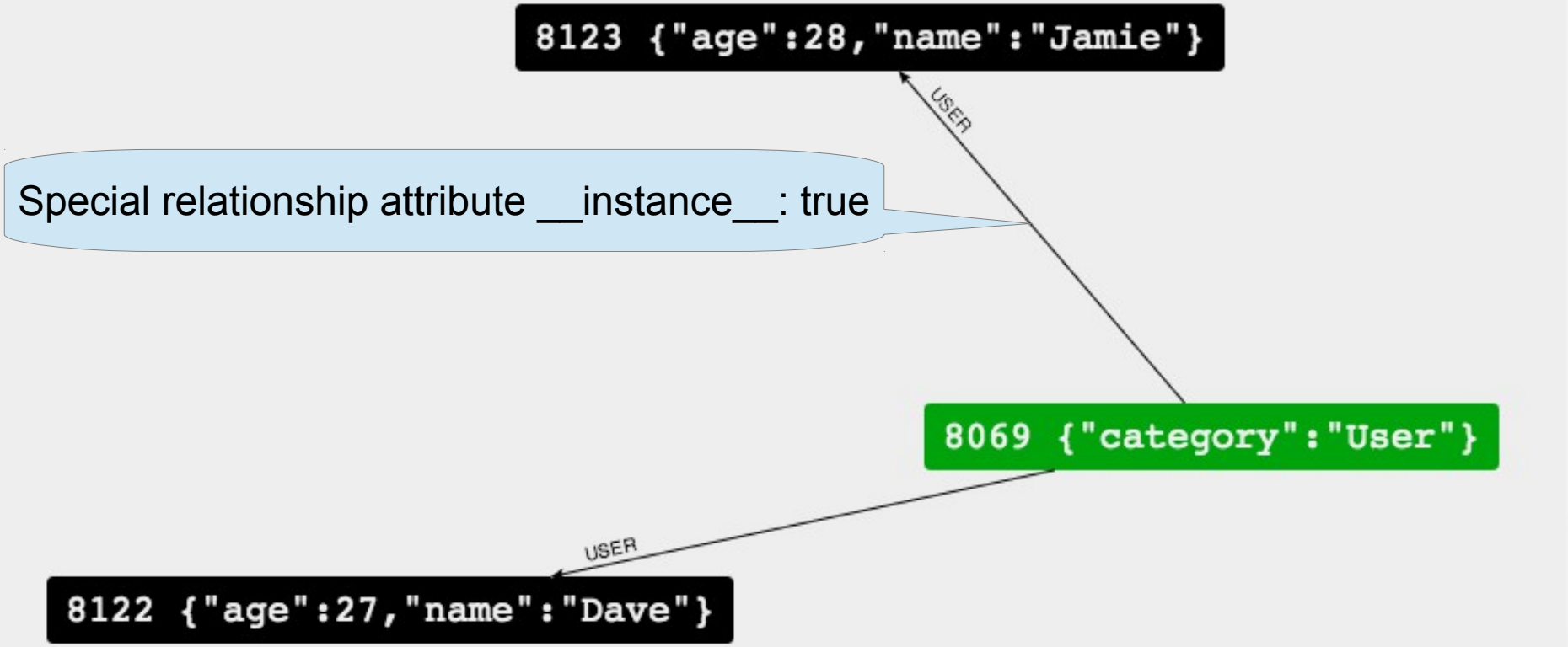
```python
from neomodel import StringProperty, IntegerProperty, StructuredNode

class User(StructuredNode):
    name = StringProperty()
    age = IntegerProperty()


dave = User(name='Dave', age=27).save()
jamie = User(name='Jamie', age="28").save()

stevan = User(name='Stevan', age="twenty nine").save()
# DeflateError: Attempting to deflate property 'age' on object of class 'User':
#               invalid literal for int() with base 10: 'twenty nine'
```

- Type check on save()
- Properties not defined on model are discarded

github gist

github.com/robinedwards

Behind the scenes



Category nodes

```
15  user_category = User.category()
16
17  for user in user_category.instance.all():
18      print user.name
19
20  results = user_category.instanch.search(name="Dave", age=23)
```

- Retrieve or search all instances of a class
- Think Model.objects.all() in Django
- Better to search using the graph than the index

github gist

github.com/robinedwards

Indexing

```python
class User(StructuredNode):
    name = StringProperty(unique_index=True)
    age = IntegerProperty(index=True)

shelia = User.index.get(name="Shelia")

for user in User.index.search(age=23):
    print user.name # Tim, Karen

from lucenequerybuilder import Q

for user in User.index.search(Q('age', inrange=[30, 40])):
    print user.name # Bob, Shelia

darren = User.index.get(name="Darren")
# DoesNotExist: Can't find node in index matching query
```

- neomodel creates the 'User' index for you

Adding the Q+A

```
2  from uuid import uuid4
3
4
5  class User(StructuredNode):
6      uuid = StringProperty(unique_index=True, default=uuid4)
7      name = StringProperty(unique_index=True)
8      age = IntegerProperty(index=True)
9
10
11 class Question(StructuredNode):
12     uuid = StringProperty(unique_index=True, default=uuid4)
13     text = StringProperty()
14
15
16 class Answer(StructuredNode):
17     uuid = StringProperty(unique_index=True, default=uuid4)
18     text = StringProperty()
```

github gist

github.com/robinedwards

Defining relationships

```python
from neomodel import RelationshipTo, RelationshipFrom, One


class User(StructuredNode):
    # ...
    questions = RelationshipFrom('Question', 'AUTHOR')
    answers = RelationshipFrom('Answer', 'AUTHOR')


class Question(StructuredNode):
    # ...
    author = RelationshipTo('User', 'AUTHOR', cardinality=One)
    answers = RelationshipTo('Answer', 'ANSWER')


class Answer(StructuredNode):
    # ...
    author = RelationshipTo('User', 'AUTHOR', cardinality=One)
    question = RelationshipFrom('Question', 'ANSWER')
```

Incoming relationship of type 'AUTHOR'

Outgoing relationship

Expecting single relation to a node

github gist

github.com/robinedwards

Using relationships

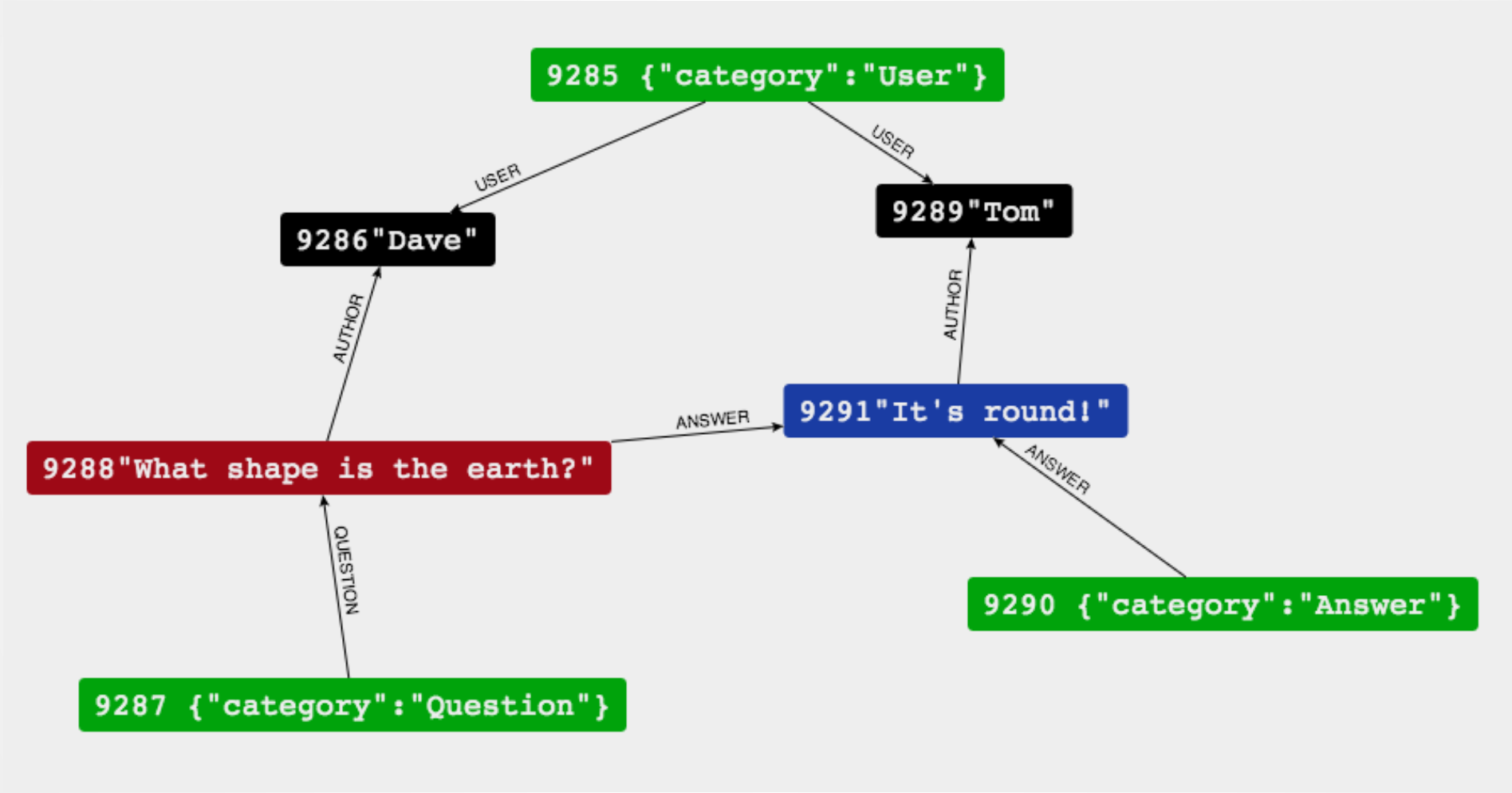
```
1  dave = User(name="Dave", age=45).save()
2  q1 = Question(text="What shape is the earth?").save()
3  dave.questions.connect(q1)
4
5  print q1.author.single().name # Dave
6
7  for question in dave.questions.all():
8      print question.text
9
10 tom = User(name="Tom", age=32).save()
11
12 if not q1.answers:
13     # Answer a question!
14     a1 = Answer(text="It's round!").save()
15     a1.author.connect(tom)
16     a1.question.connect(q1)
17
18 print "Question has {0} answers".format(len(q1.answers))
```

Create unique relationship

github gist

github.com/robinedwards

Viewing the graph



DateTime!

```python
from neomodel import StructuredNode, DateTimeProperty
from datetime import datetime

class Question(StructuredNode):
    # ...
    created = DateTimeProperty(default=datetime.utcnow)


class Answer(StructuredNode):
    # ...
    created = DateTimeProperty(default=datetime.utcnow)
```

- Auto inflate / deflate to datetime object
- Stored as a UTC unix epoch
- Checkout Nigel's blog post on calendars in neo

Sorting answers by date

```
1  class Question(StructuredNode):
2      # ...
3      answers = RelationshipTo('Answer', 'ANSWER')
4
5      def answers_by_date(self):
6          results, _ = self.cypher("""
7              START self = node({self})
8              MATCH (self)-[:ANSWER]->(answer)
9              RETURN answer ORDER BY answer.created""")
10         return [Answer.inflate(row[0]) for row in results]
```

- No order support in RelationshipManager
- Solution: Write a cypher query

Query building

- How do other ORMs do this?
- ActiveRecord DBIx::Class etc
- They use an Abstract Syntax Tree
- Easy to build AST through chaining methods
- Leave producing SQL to the last minute

Traversals

- New **EXPERIMENTAL** feature
- Uses an AST to build the query
- Cypher produced at last minute
- answers_by_date now much simpler

```
1  for answer in question.traverse('answers').order_by('answers.created'):
2      print answer.text
```

Order and limit

```
1  class User(StructuredNode):
2      # either direction
3      friends = Relationship('User', 'FRIEND')
4
5  # Tom and Dave are now friends
6  tom.friends.connect(dave)
7
8  # Tom's friends 10 most recent question
9  questions = [
10     q.text for q in tom.traverse('friends')
11     .traverse('questions')
12     .order_by('questions.created')[0:10]:
13 ]
```

github gist

github.com/robinedwards

Generated Cypher

```
1  START origin=node({self})
2  MATCH
3    (origin)-[:FRIEND]-(friends),
4    (friends)<-[r1:USER]-(),
5    (friends)<-[:AUTHOR]-(questions),
6    (questions)<-[r2:QUESTION]-()
7  WHERE r1.__instance__! = true AND r2.__instance__! = true
8  RETURN questions, r2
9  ORDER BY questions.created
10 LIMIT 10
```

Coming soon:

- filter traversals by rel and node properties
- Documentation!

github.com/robinedwards

Retrospective on Neo4j

- Very adaptable responds well to changing requirements
- Keeping indexes in sync is difficult (leave this to a module)
- Beware of stale Neo4j id's (when serialising objects)
- Absence of schema empowers applications
- Be careful though, Check your data integrity

Thank you

Sharehoods.com

github.com/robinedwards/neomodel/