Graph databases

COSC430—Advanced Databases

David Eyers (thanks to Paul Crane)

Learning objectives

- You should be able to **explain**:
  - What graphs are
  - Key characteristics of graph databases
  - Common uses for graph databases
  - Why specialised graph databases are needed

- You should be able to **describe** existing research:
  - Trinity—the focus of our discussion today

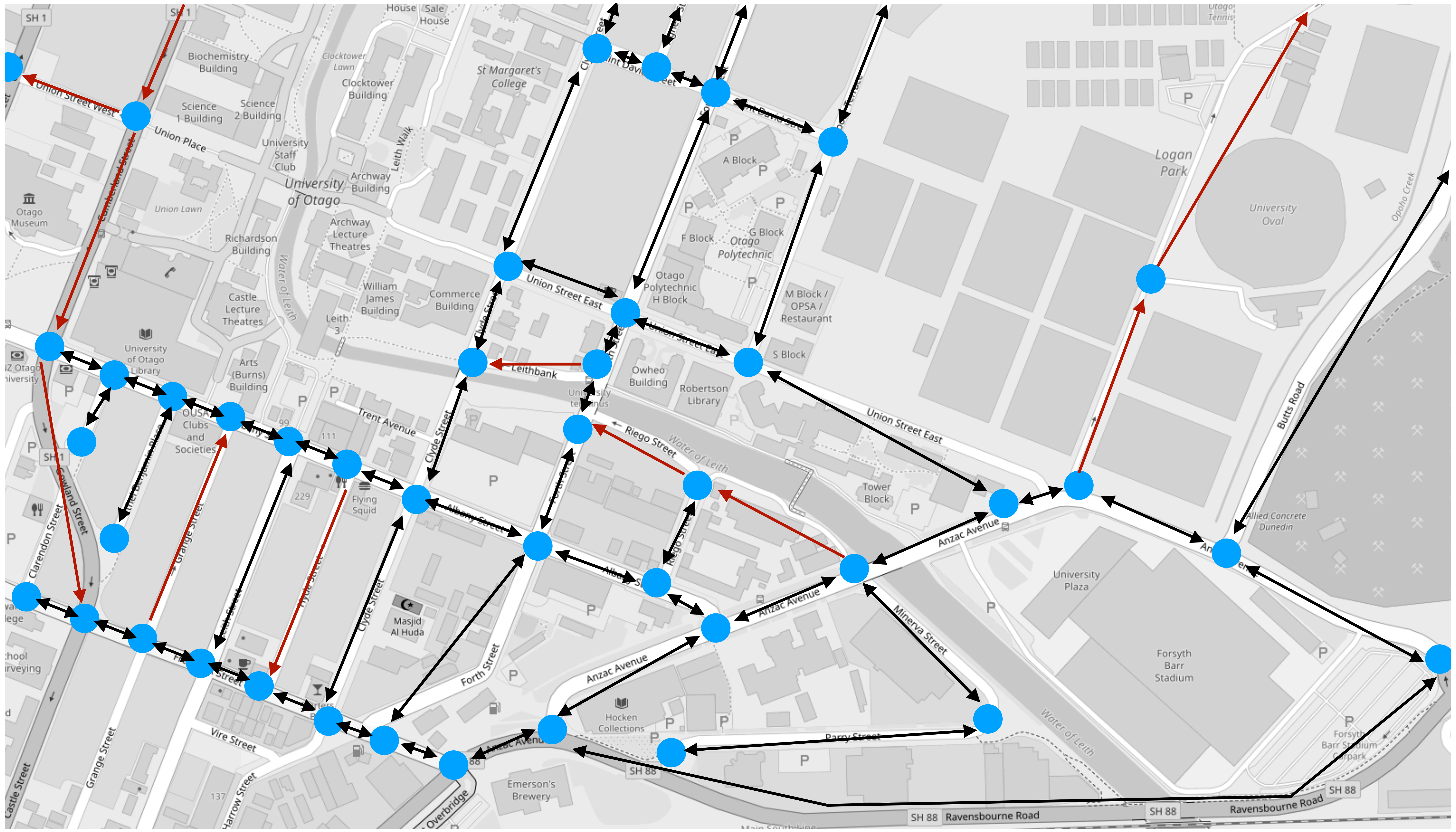
Definitions

- **Node** (or **vertex**)—represents an entity
- **Edge**—represents relationship between nodes
  - Bidirectional (usually illustrated without arrowheads)
  - Unidirectional (usually illustrated with an arrowhead)
- **Properties**—describe attributes of the node or edge
  - Often stored as a key-value set

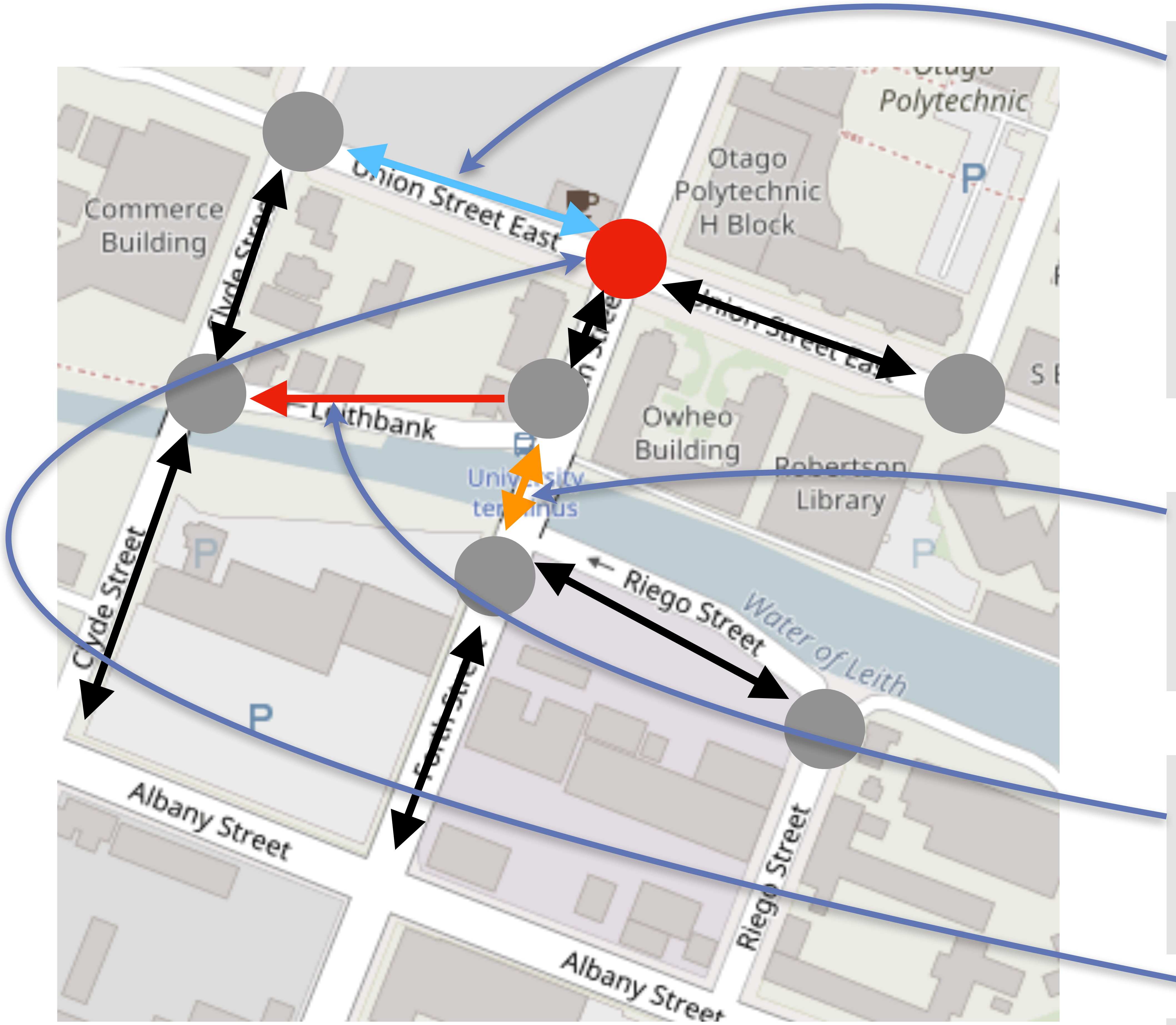
- (We won't explore extended notions like hypergraphs)

Street map connectivity is a graph

- Node
  - traffic junction
- Edge
  - shows traffic flow
  - uni/bidirectional



Edges can have properties



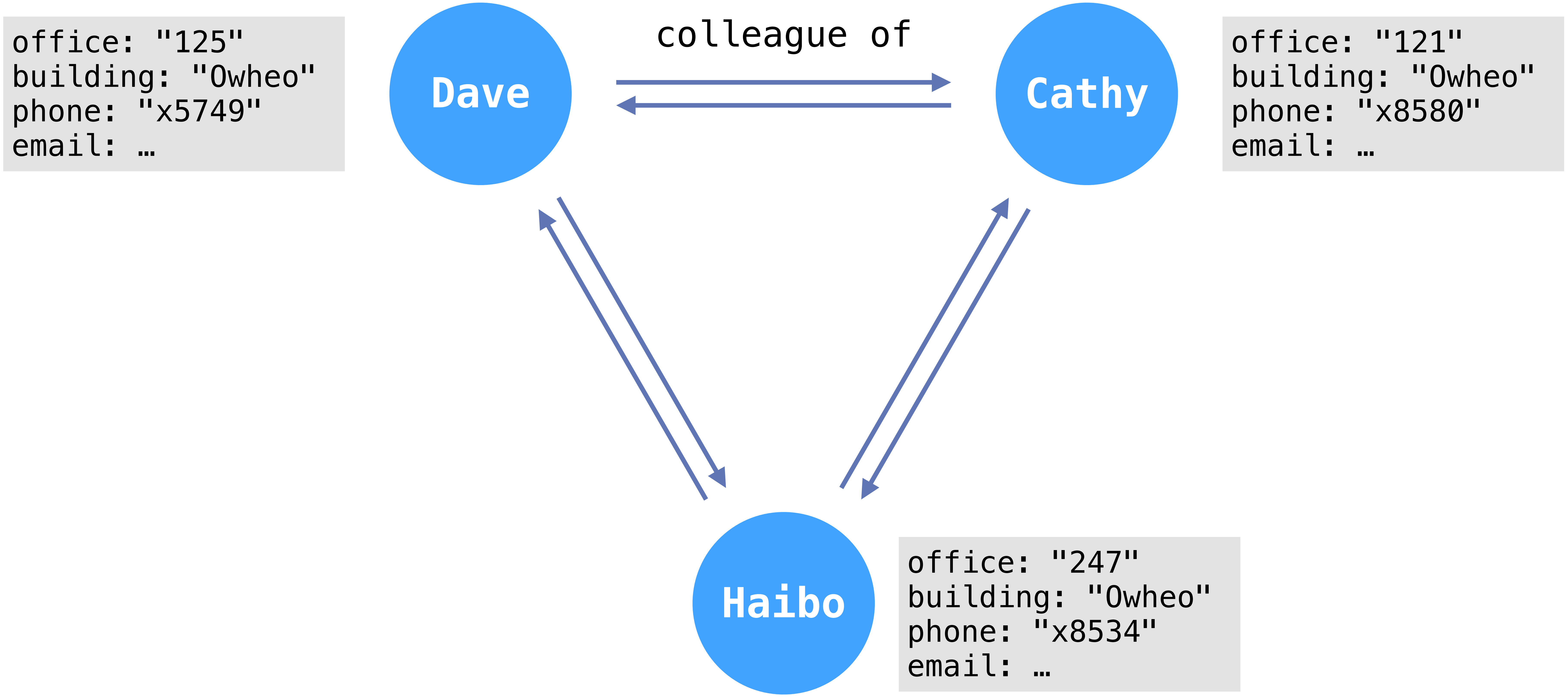

```
name: "Union Street East"
type: "residential"
max_speed: "50"
restrictions: "commercial"
surface: "tarmac"
status: "closed"
reason: "repairs"
length: "250m"
```

```
name: "Forth Street"
type: "residential"
max_speed: "50"
furniture: "bus stop"
```

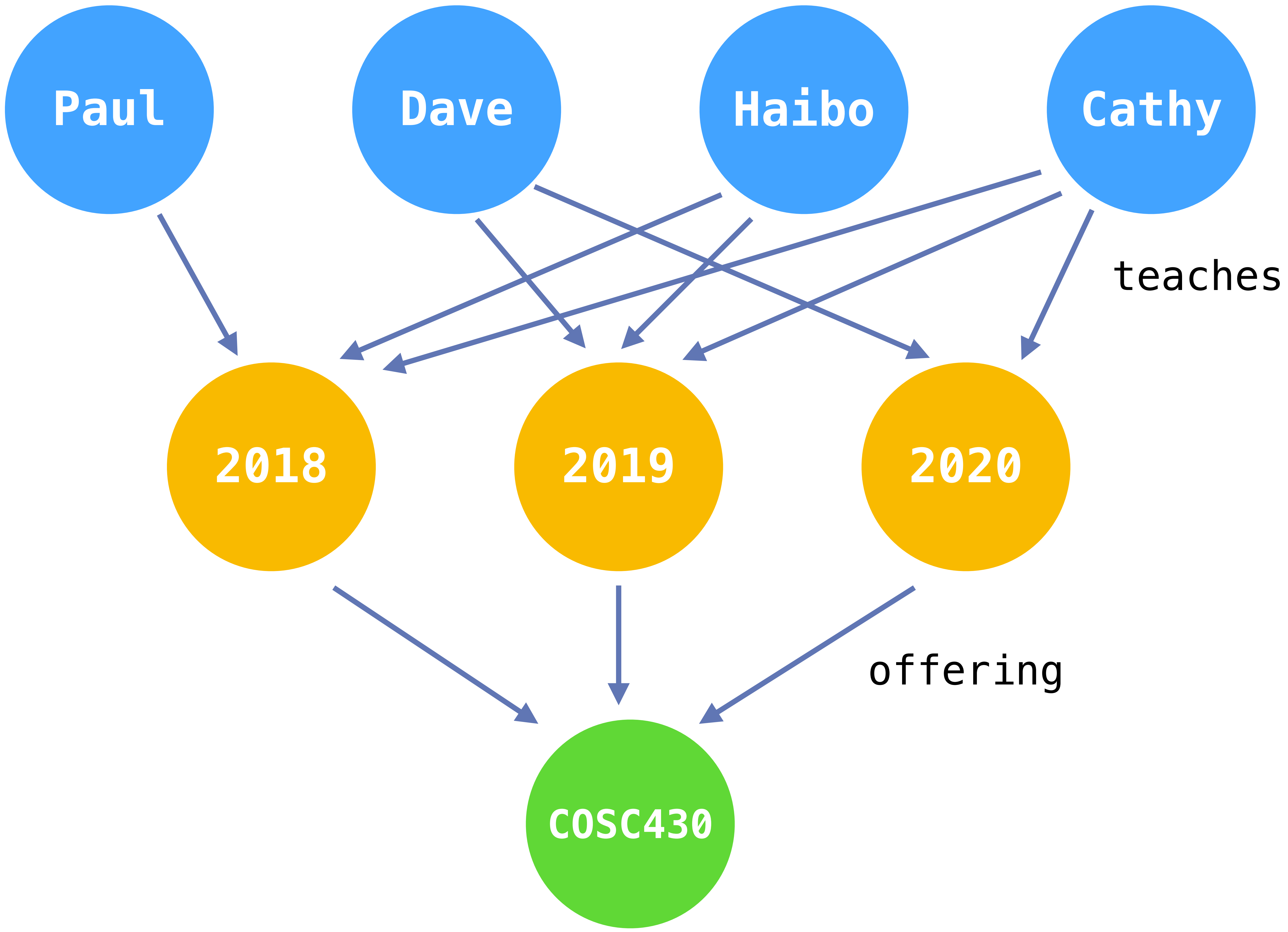
```
name: "Leithbank"
type: "residential"
max_speed: "50"
direction: "one way"
```

```
traffic_control: "traffic lights"
```

Nodes can have properties



Different types of node



Different lineages of graph representation

- Semantic web style triple-store (**RDF**)
  - Subject-Predicate-Object ("Bob knows Fred")
  - Nodes are URIs or values: achieving property lists is arduous
- Our focus: **labelled property graphs**
  - Nodes and edges have internal structure: e.g., a key/value set

- Labelled property graphs are more focused on efficient storage than RDF (which is more focused on interchange of information)

Justifying the need for graph databases

- Can store graphs in RDBMSs, e.g.,
  - Node table
  - Edge table
- But, joins between nodes and edges are common
  - … as the number of hops in a graph increases, this becomes increasingly expensive
- Some problems best suit direct representation in graphs

Type of data in the DB

Count number of nodes 4 deep

Count number of nodes 128 deep

Find orphan nodes

Table 2: Structural query results, in milliseconds.

Database	MySQL S4	Neo4j S4	MySQL S128	Neo4j S128	MySQL S0	Neo4j S0
1000int	38.9	2.8	80.4	15.5	1.5	9.6
5000int	14.3	1.4	97.3	30.5	7.4	10.6
10000int	10.5	0.5	75.5	12.5	14.8	23.5
100000int	6.8	2.4	69.8	18.0	187.1	161.8
1000char8K	1.1	0.1	21.4	1.3	1.1	1.1
5000char8K	1.0	0.1	34.8	1.9	7.6	7.5
10000char8K	1.1	0.6	37.4	4.3	14.9	14.6
100000char8K	1.1	6.5	40.9	13.5	187.1	146.8
1000char32K	1.0	0.1	12.5	0.5	1.3	1.0
5000char32K	2.1	0.5	29.0	1.6	7.6	7.5
10000char32K	1.1	0.8	38.1	2.5	15.1	15.5
100000char32K	6.8	4.4	39.8	8.1	183.4	170.0

Number of nodes in DB

Size of data in DB

C. Vicknair, M. Macias, Z. Zhao, et al. 2010. "A Comparison of a Graph Database and a Relational Database: A Data Provenance Perspective." In Proceedings of the 48th Annual Southeast Regional Conference, 42:1–42:6. ACM SE '10. New York, NY, USA: ACM.

Example uses of graph databases

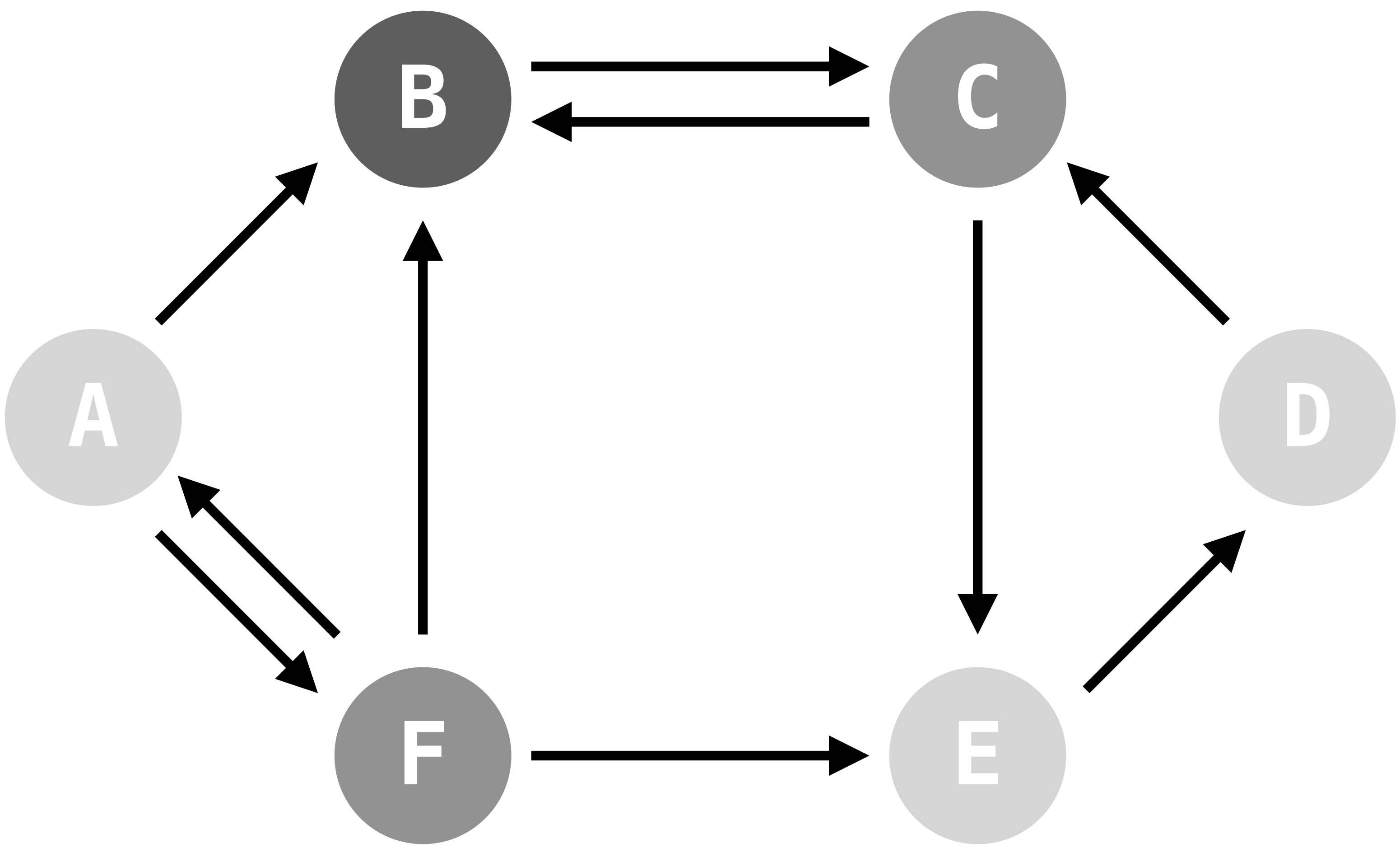
- **Recommendations**—e.g., find most important pages within the World Wide Web
- **Navigation**—e.g., find a route from home to work
- **Social networks**—e.g., Facebook friends, Twitter follows
- **Networks**—e.g., the Internet, WWW interconnections
- **Processes**—e.g., cooking dinner

Social networks

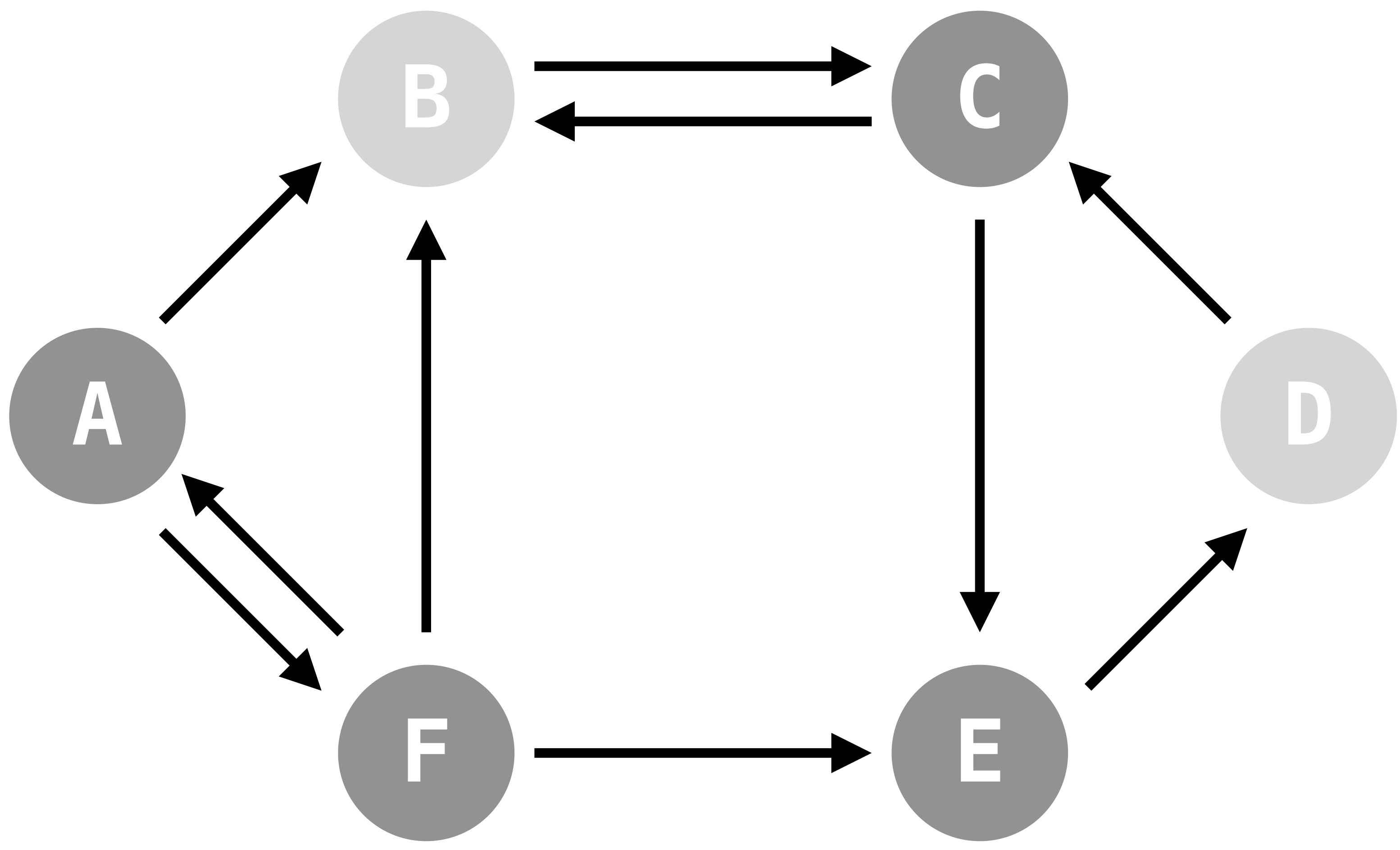
- Want to find important people (nodes) in the network

- Some types of centrality (i.e., “importance”):
  - **Degree** centrality—indegree/outdegree
  - **Closeness** centrality—average length of all shortest paths
  - **Betweenness** centrality—number of times a node acts as a bridge along the shortest paths
  - **Eigenvector** centrality—measures the influence of a node in a network (e.g., Google PageRank)

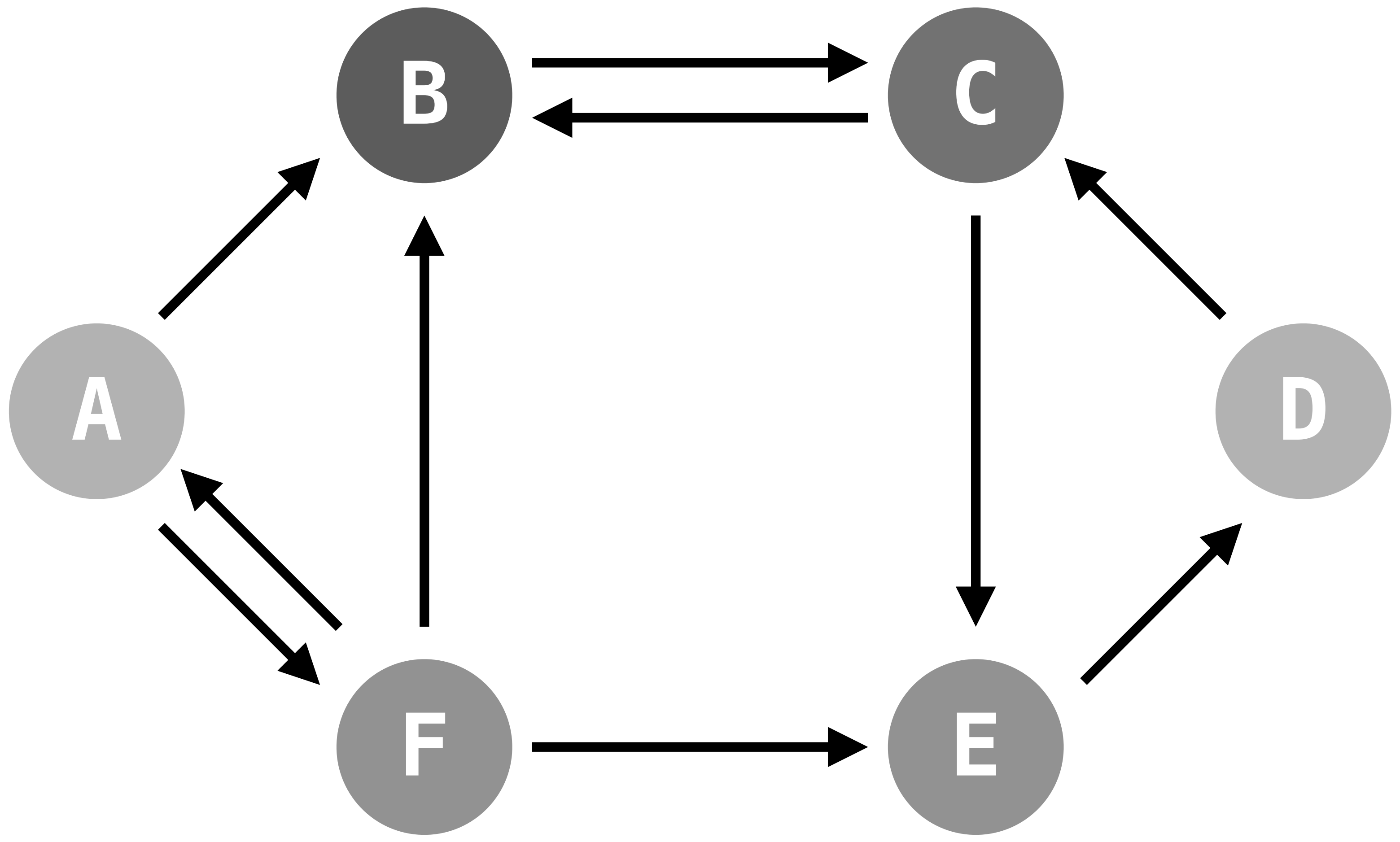
Indegree centrality



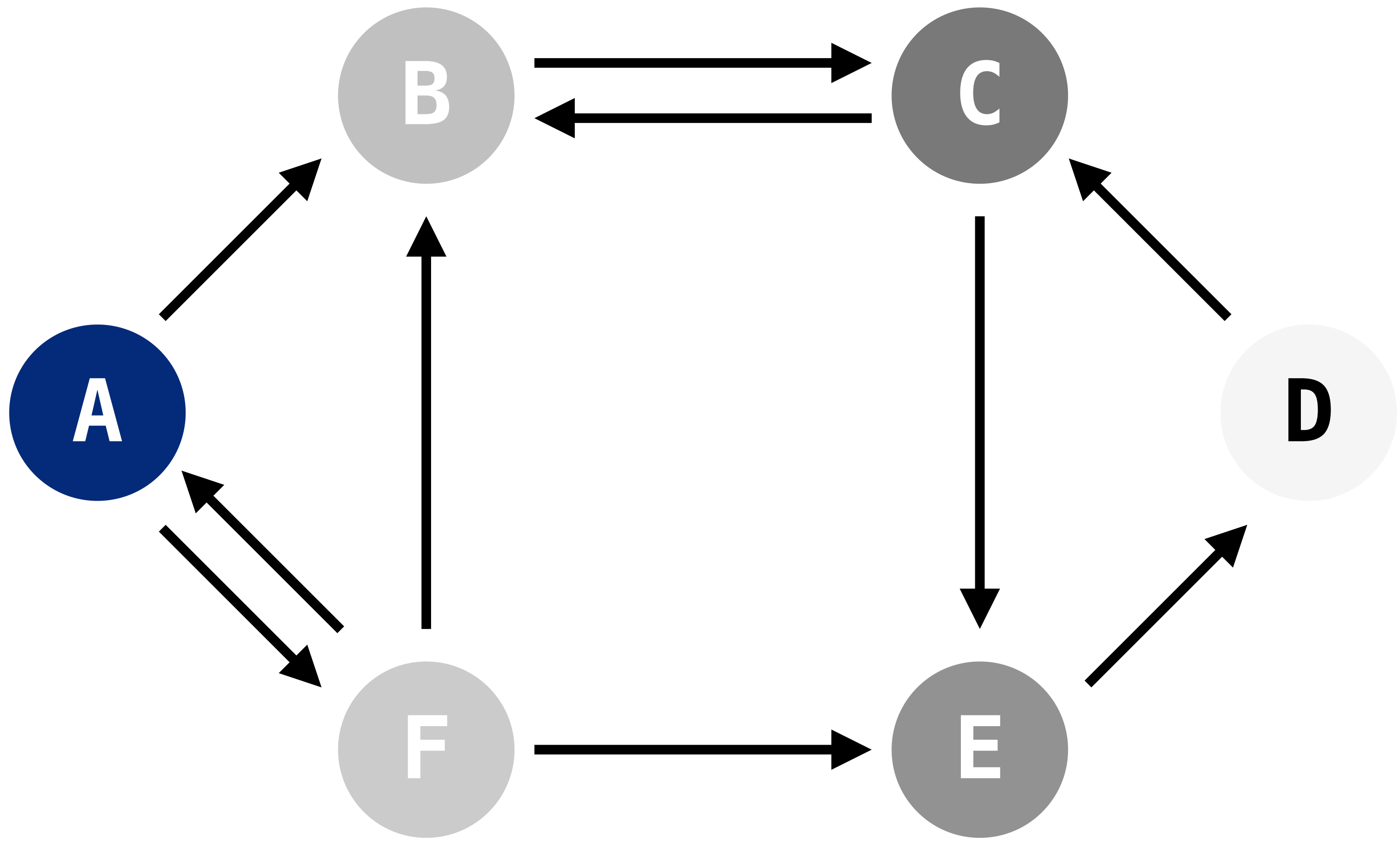
Outdegree centrality



Closeness centrality



Betweenness centrality



Google's PageRank

- PageRank is a "vote" by all other pages on the Web about how important a page is
  - A link to a page counts as a vote of support

$$PR(A) = (1 - d) + d \times \sum_{n \in \text{in\_edges}(A)} \frac{PR(n)}{C(n)}$$

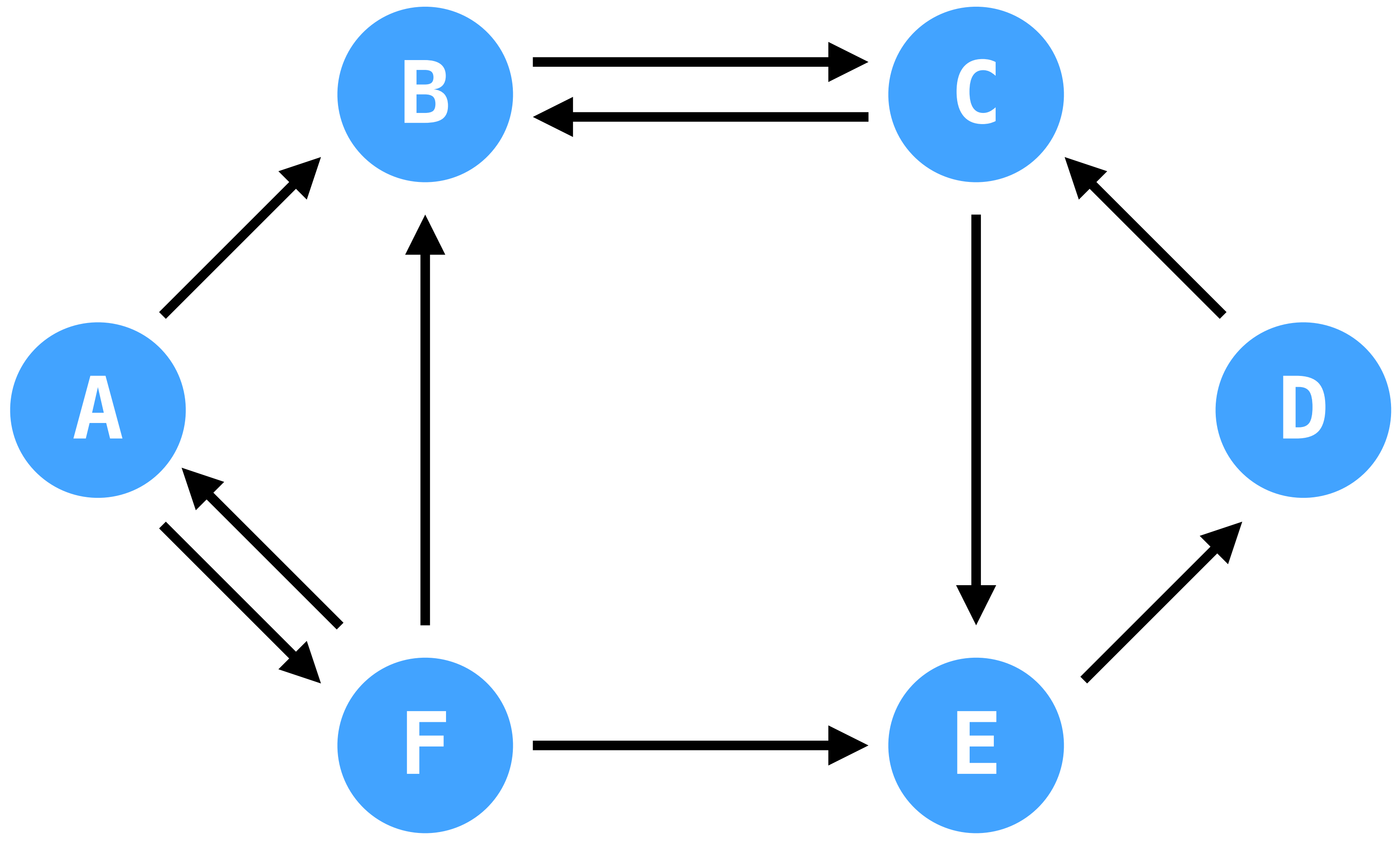
d is a damping factor typically set to 0.85

$C(n)$ is the number of outgoing links or the outdegree

Google's PageRank

- Guess a value of 1 for all initial PageRanks

```
C(A) = 2
C(B) = 1
C(C) = 2
C(D) = 1
C(E) = 1
C(F) = 3
```



Repeated from the previous slide

$$PR(A) = (1-d) + d \times \sum_{n \in \text{in\_edges}(A)} \frac{PR(n)}{C(n)}$$

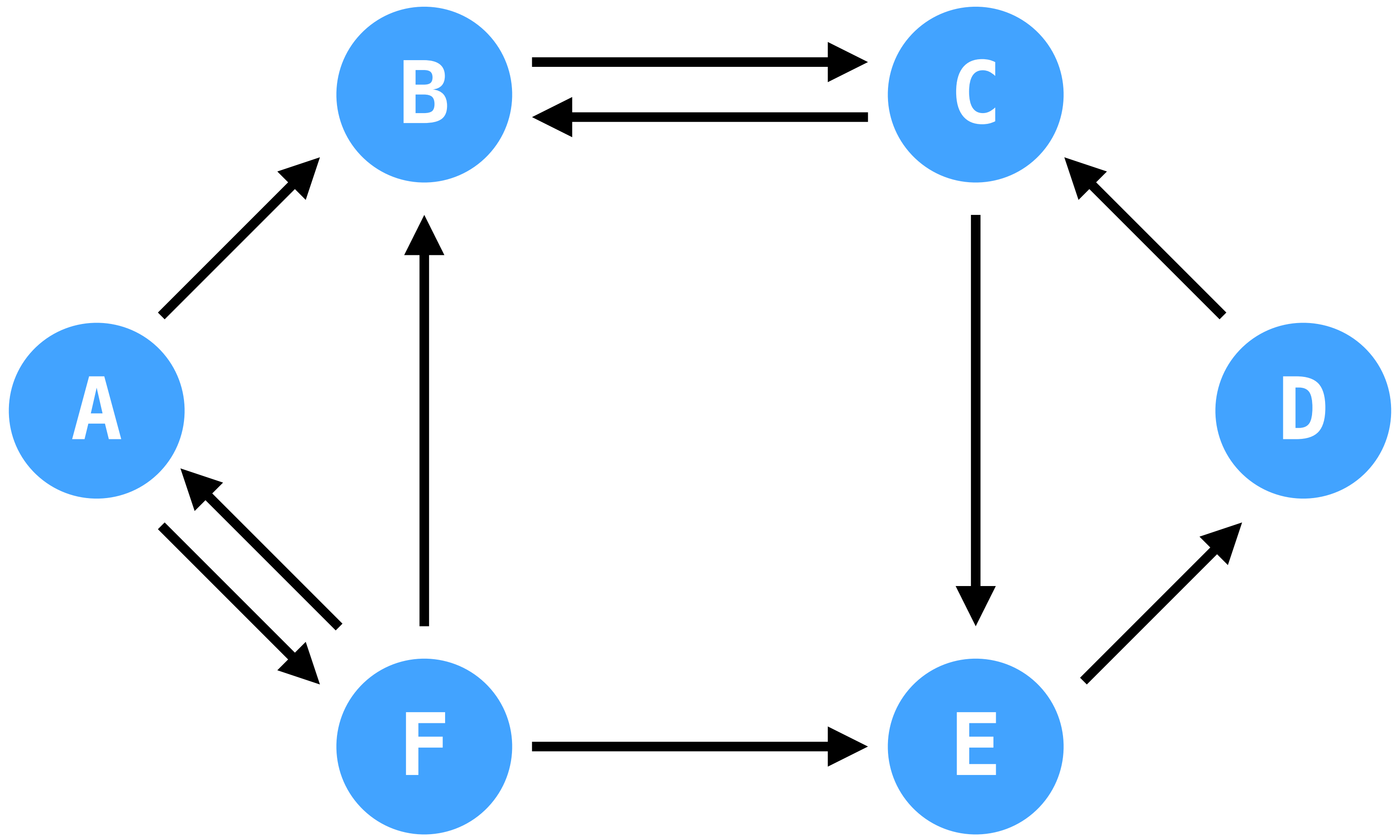
Recall that $C(n)$ is the number of outgoing links or the outdegree

```
PR(A) = 0.15 + 0.85 * (1/3)             = 0.433
PR(B) = 0.15 + 0.85 * (1/2 + 1/2 + 1/3) = 1.283
PR(C) = 0.15 + 0.85 * (1/1 + 1/1)       = 1.85
PR(D) = 0.15 + 0.85 * (1/1)             = 1
PR(E) = 0.15 + 0.85 * (1/2 + 1/3)       = 0.858
PR(F) = 0.15 + 0.85 * (1/2)             = 0.575
```

Google's PageRank

- Use previous values for new PR calculation …

```
C(A) = 2
C(B) = 1
C(C) = 2
C(D) = 1
C(E) = 1
C(F) = 3
```

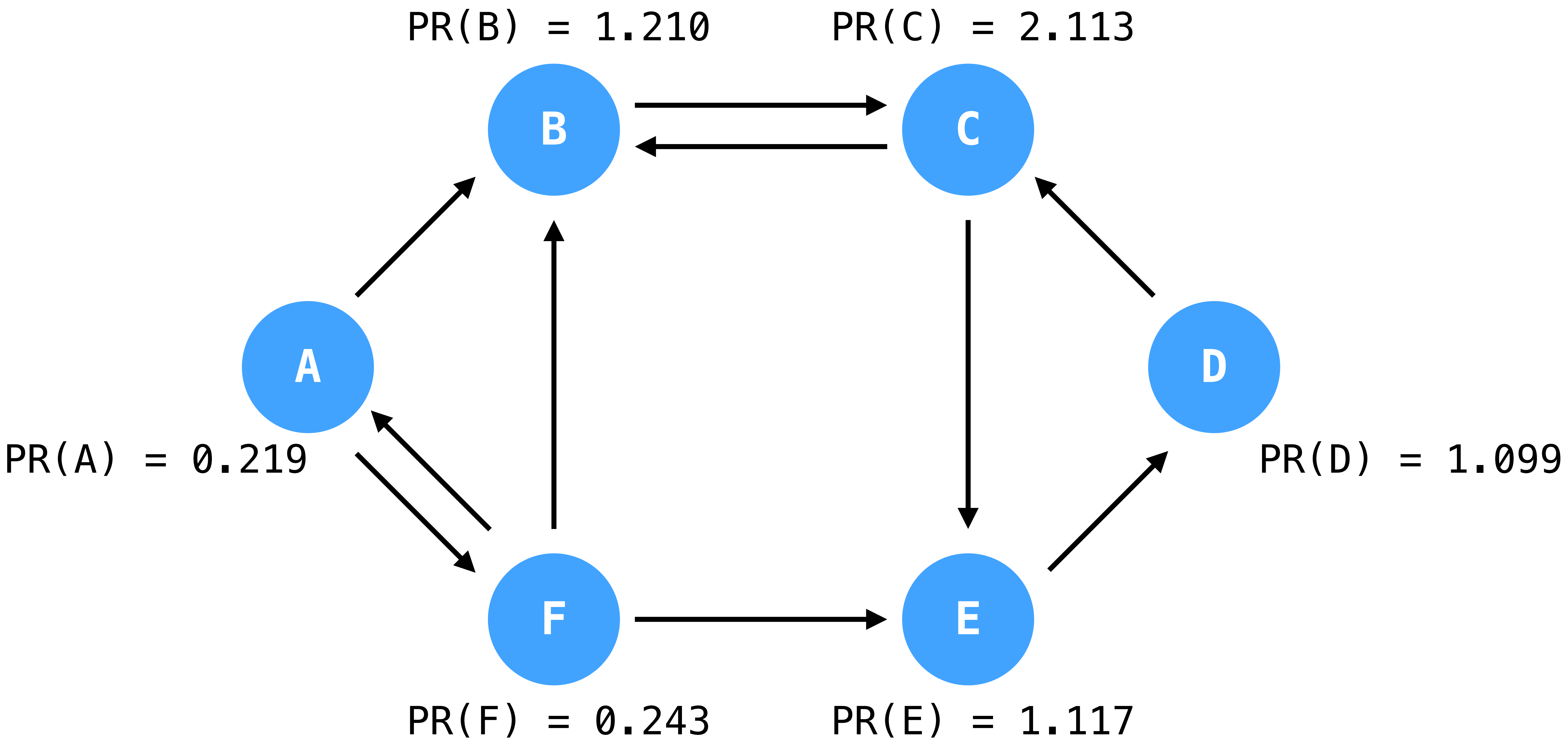


$$PR(A) = (1 - d) + d \times \sum_{n \in \text{in\_edges}(A)} \frac{PR(n)}{C(n)}$$

```
PR(A) = 0.15 + 0.85 * (0.575/3)                      = 0.313
PR(B) = 0.15 + 0.85 * (0.433/2 + 1.850/2 + 0.575/3)  = 1.283
PR(C) = 0.15 + 0.85 * (1.283/1 + 1/1)                = 2.091
PR(D) = 0.15 + 0.85 * (0.858/1)                      = 0.880
PR(E) = 0.15 + 0.85 * (1.850/2 + 0.575/3)            = 1.099
PR(F) = 0.15 + 0.85 * (0.433/2)                      = 0.334
```

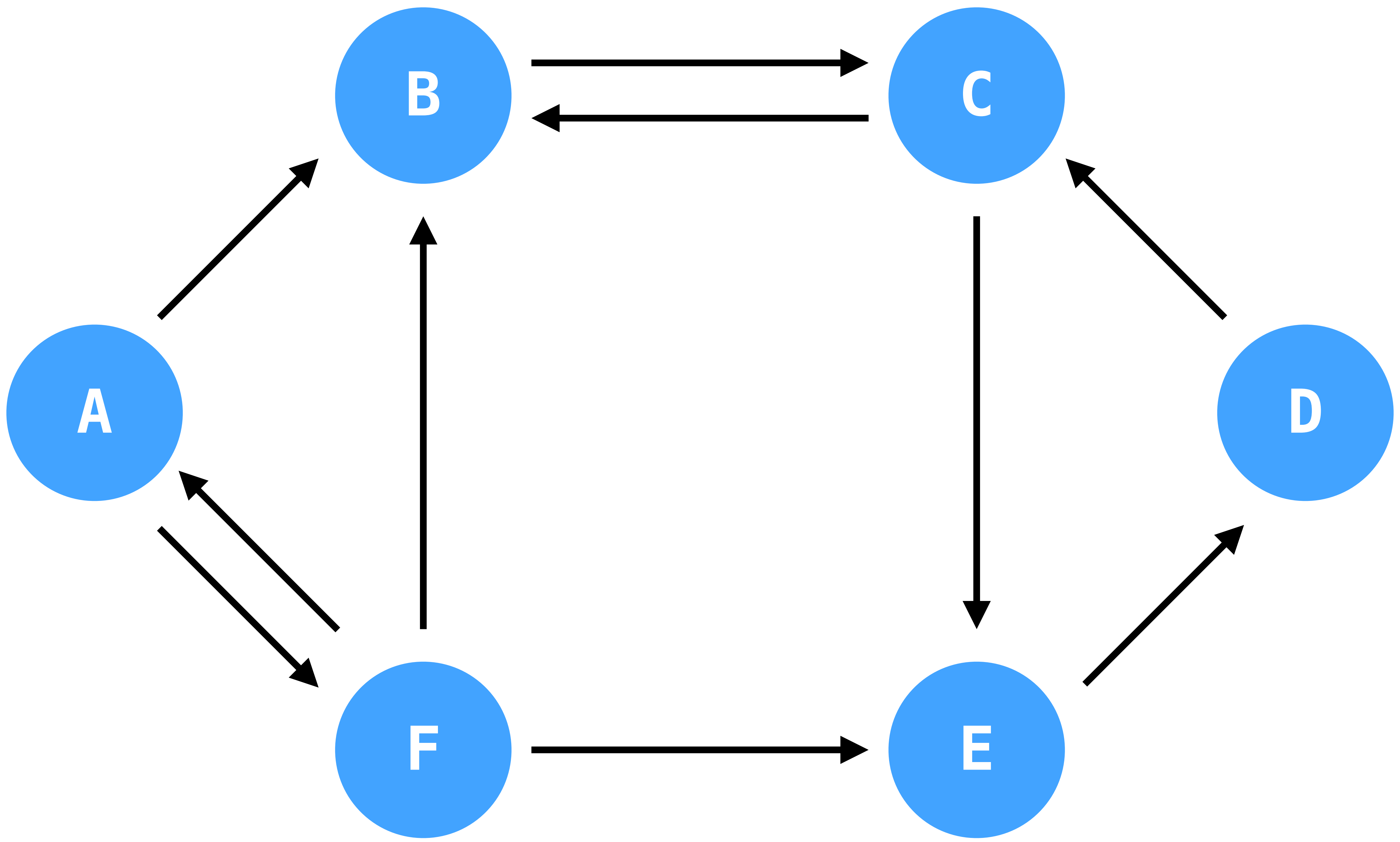
Google's PageRank

- Iterate PR calculation until results stabilise (or bail out)



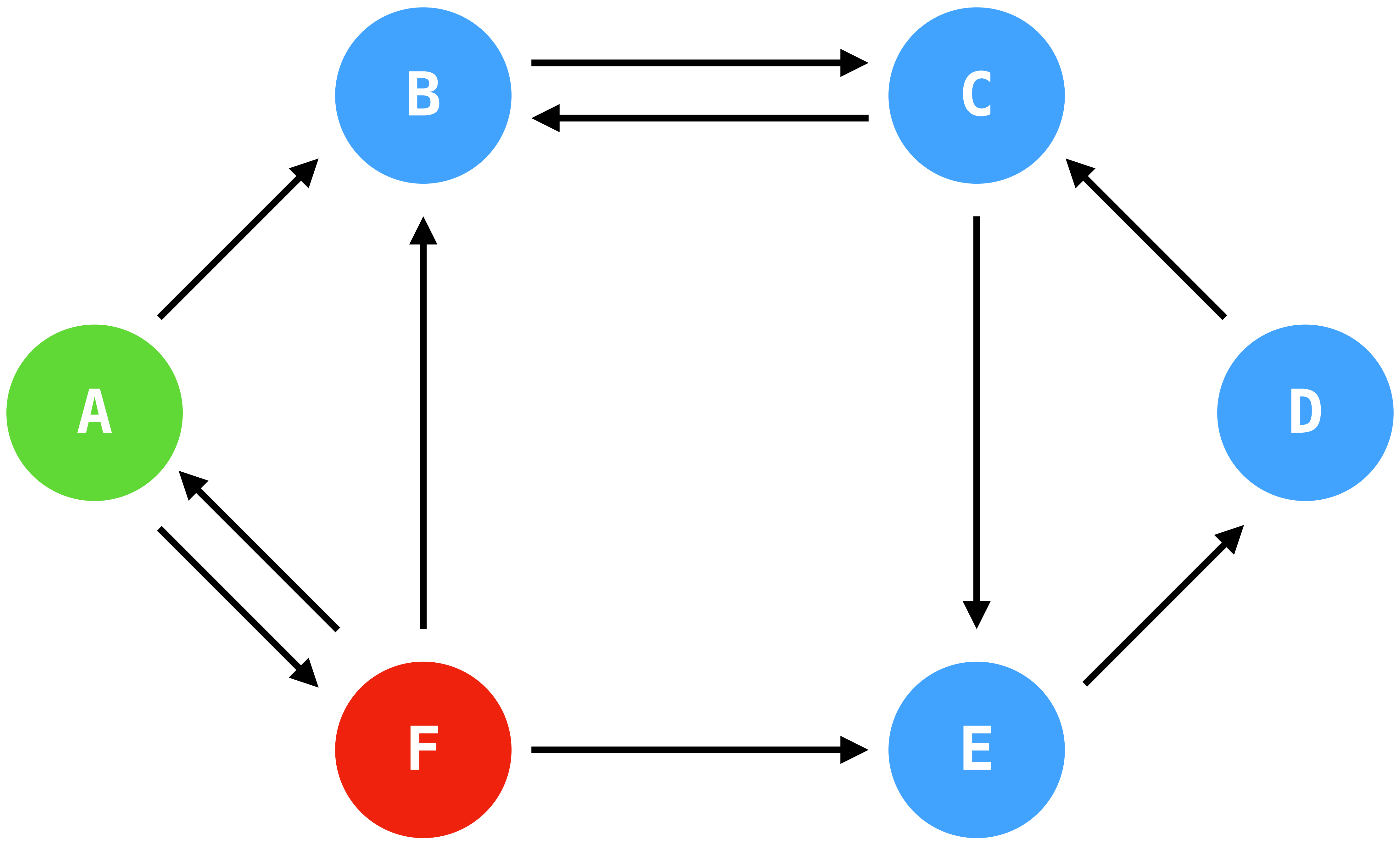
Navigation / route planning

- **Depth-first search** visits nodes along branches before back-tracking
- **Breadth-first search** visits neighbours before visiting descendants
- Others: Dijkstra's Shortest Path (SP), Minimum Spanning Tree (MST)



Navigation / route planning

- Consider determining a route from A to F:
  - **Depth-first search**: A, B, C, E, D, F
  - **Breadth-first search**: A, B, F



Designing graph databases

- Typical mapping from application's data to a graph:
  - **Entities** are represented as nodes
  - **Connections** are represented as edges between nodes
  - **Connection semantics** dictate directions of edges
  - **Entity attributes** become node properties
  - Link **strength / weight / quality** maps to relationship properties

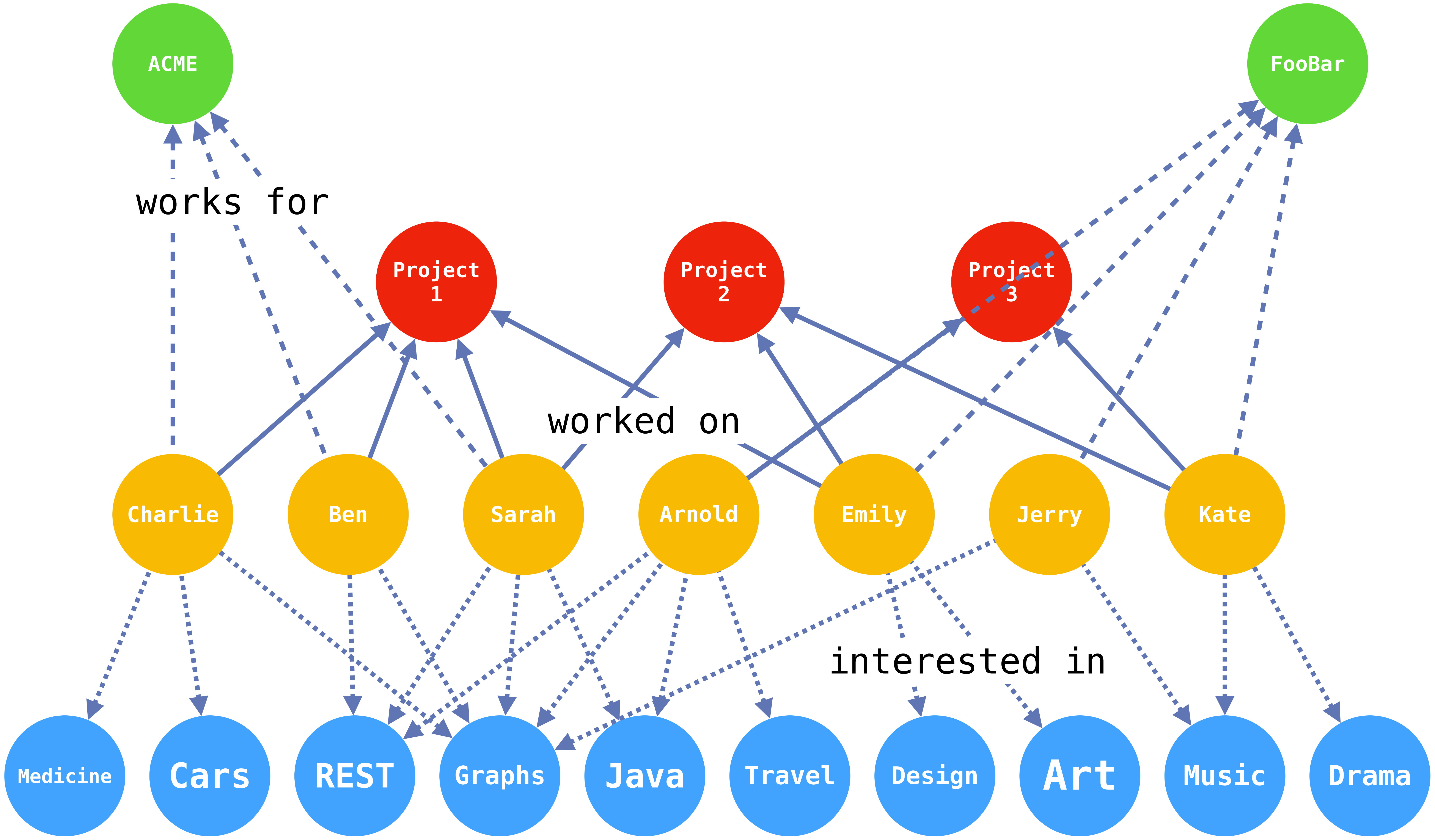
- Other metadata will also be include in property sets
  - e.g., information about data entry and revision

Use case: professional social networks

- Enable users to discover others with particular skillsets
  - Users work for companies
  - Users work on projects
  - Users have one or more interests/skills

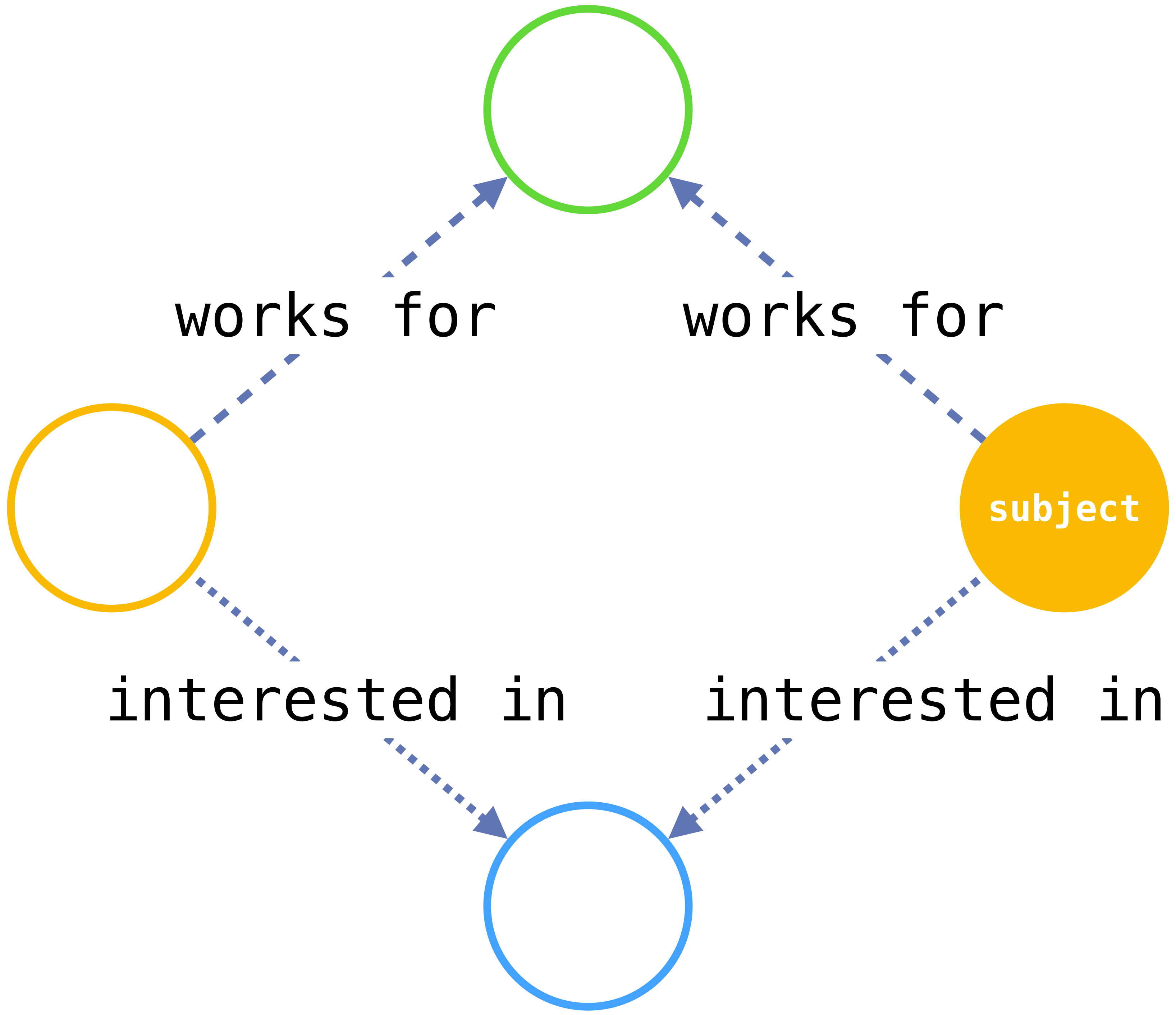
- Similarity between users is based on the number of their common interests/skills

Professional social network (PSN) example



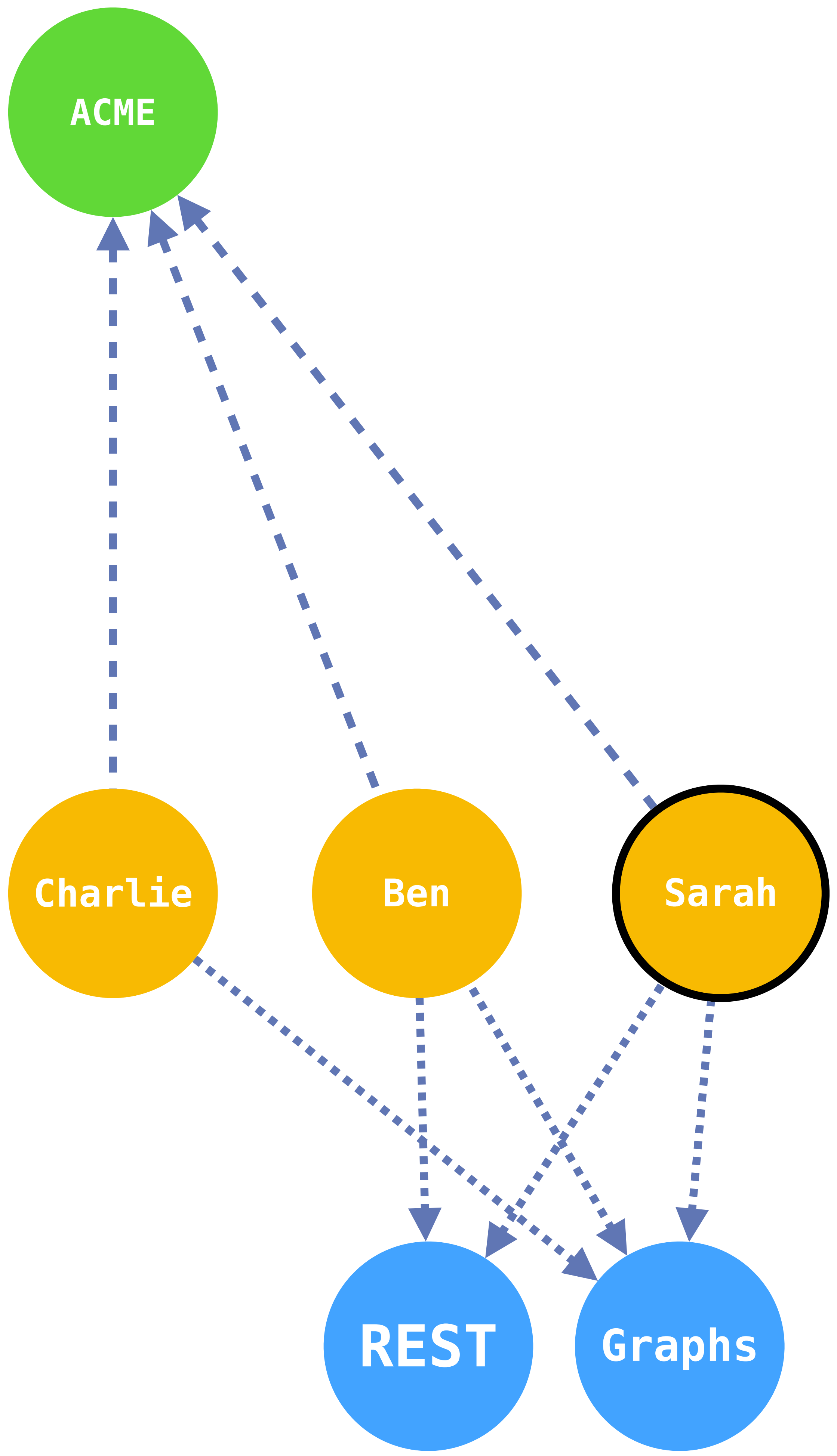
Simple, small-scale PSN query

- Find people who work for the same company with the same interests as the given subject (person)



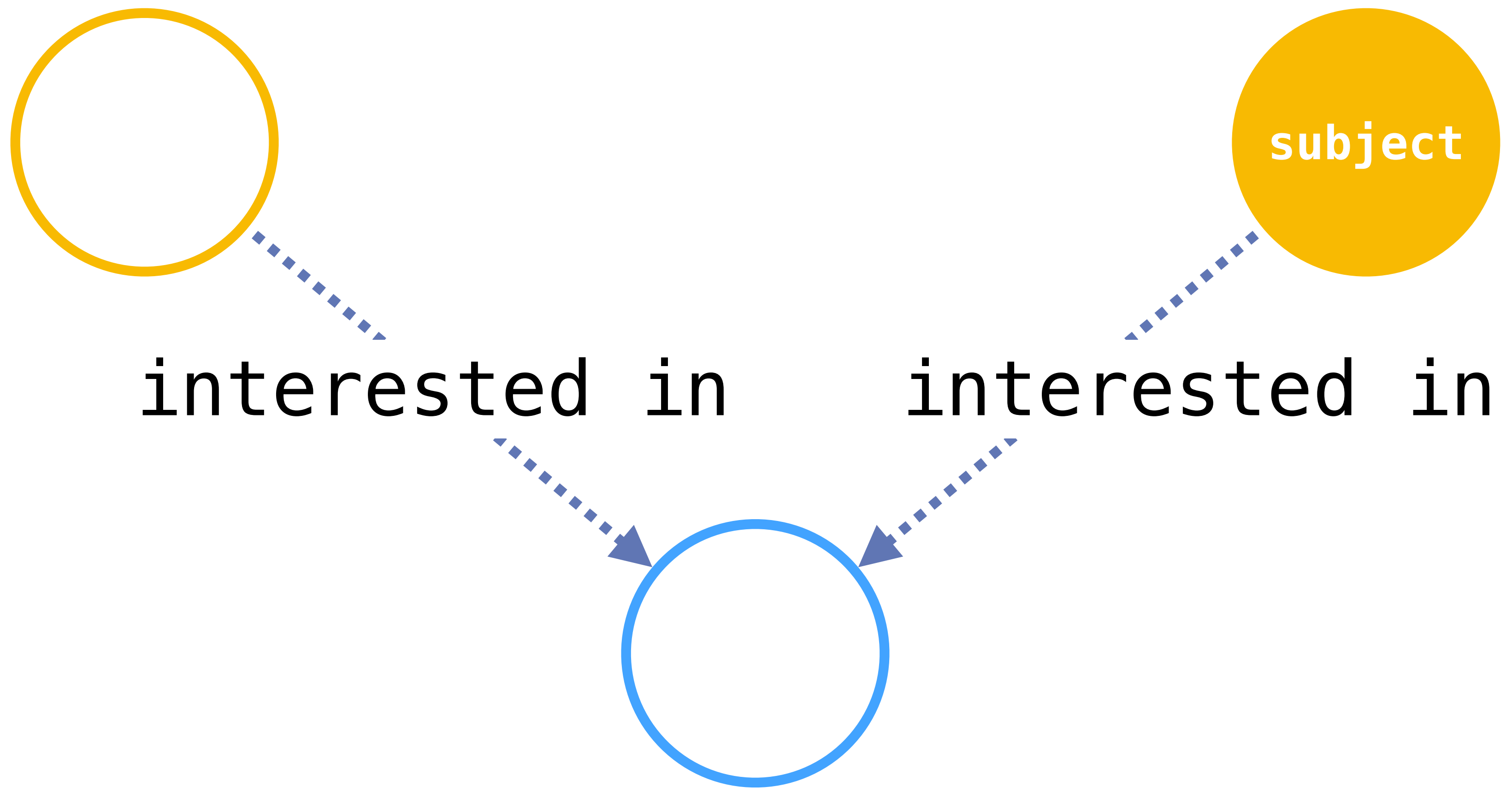
Result of simple, small-scale PSN query

- With Sarah as the subject we get this subgraph
- Ordered by number of shared interests:
  - Ben: 2
  - Charlie: 1
- Only takes into account people in the same company…



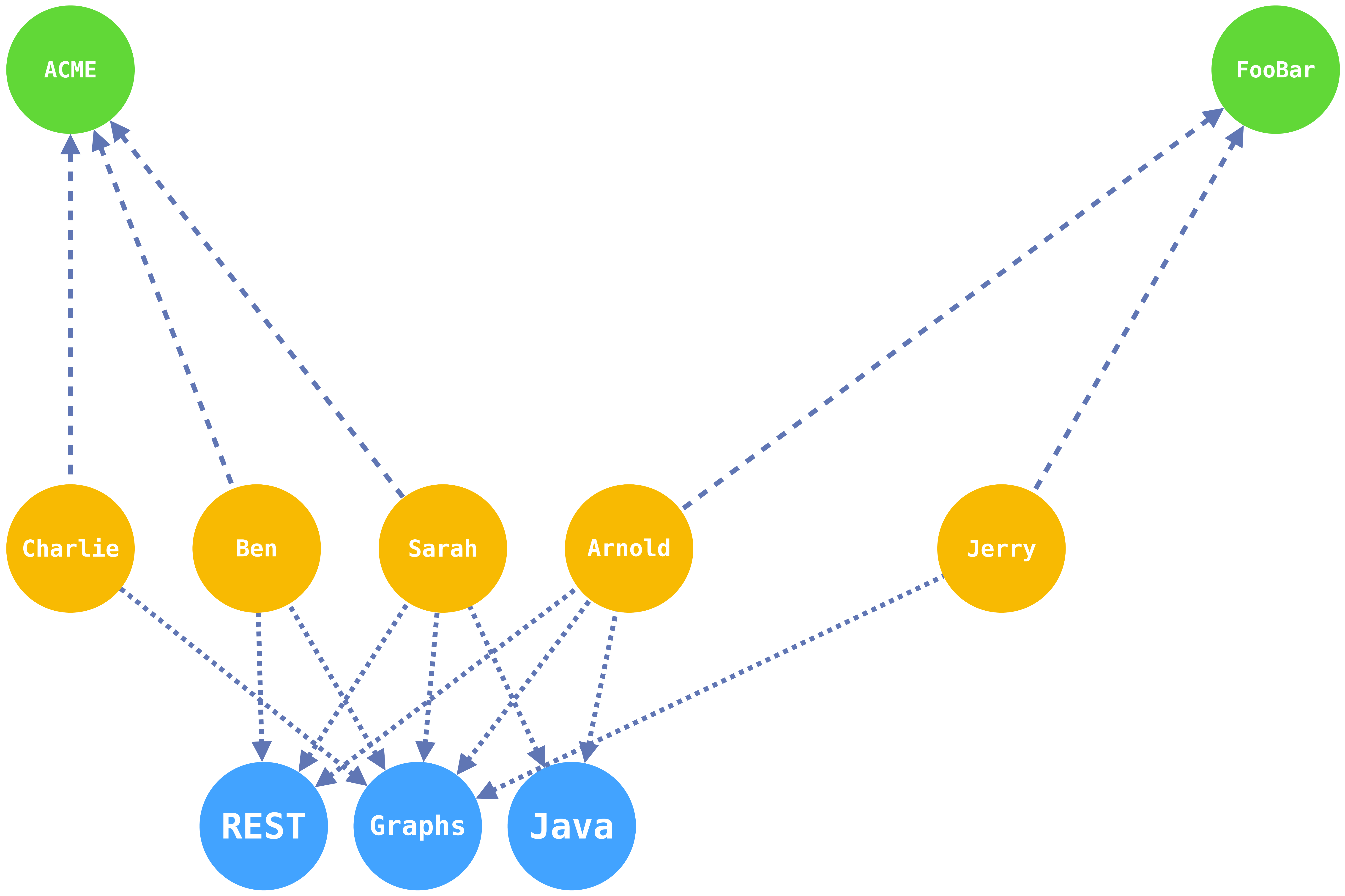
Larger-scale PSN query

- Find people with the same interests as the subject



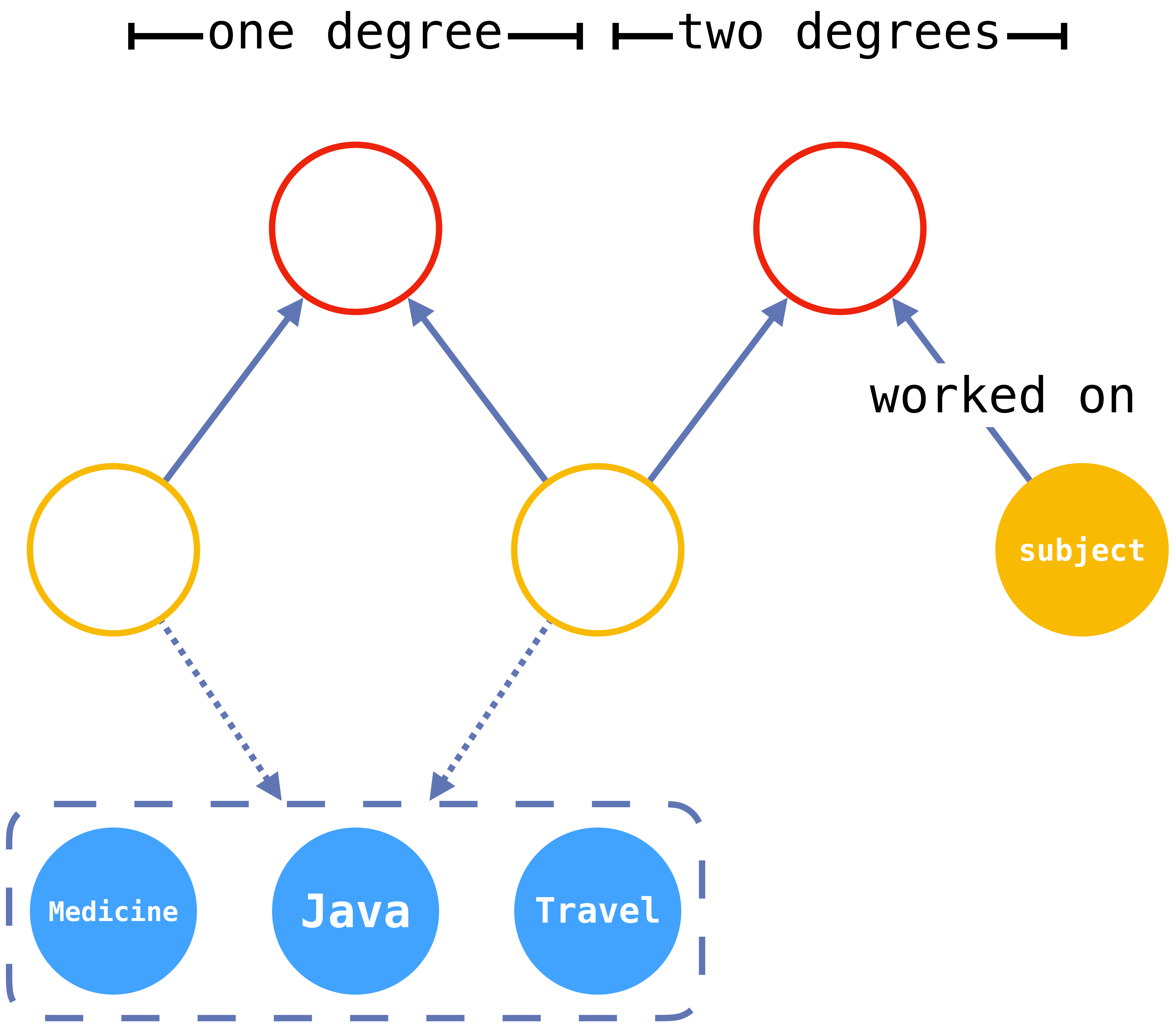
Result of larger-scale PSN query

- Ordered by number of shared interests:
  - Arnold: 3
  - Ben: 2
  - Charlie: 1
  - Jerry: 1



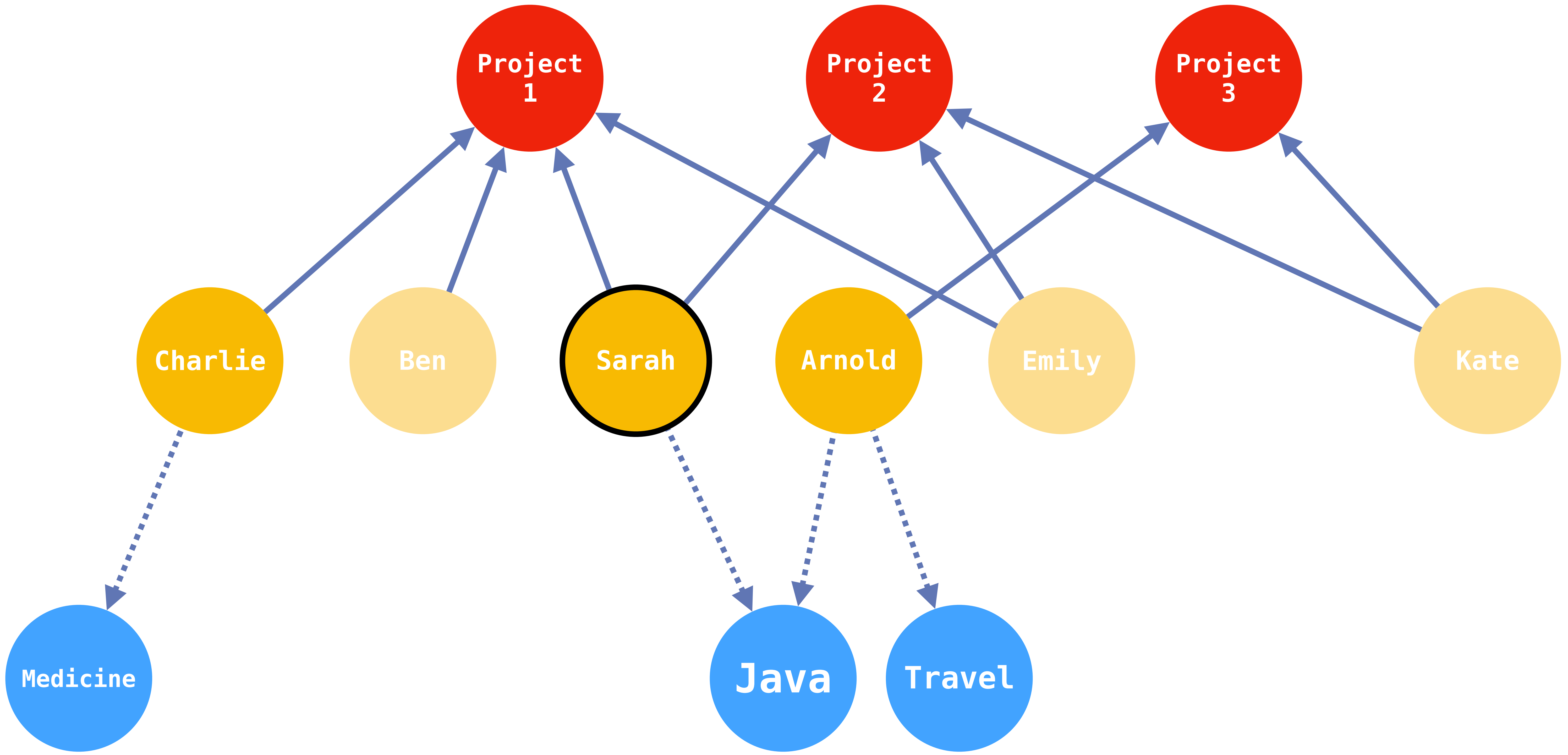
A PSN query that traverses the graph

- Find people who have worked with the subject up to two degrees of separation away, that are interested in at least one of medicine, Java, and travel



Result from graph traversing query

- Results ordered by distance:
  - Charlie: 1
  - Arnold: 2



Summary

- Introduced graph terminology
  - nodes
  - edges
  - properties
- Illustrated some applications in which graphs are useful
- Explored graph database design and querying

Trinity—paper review

- Key problem under investigation?
- Key idea of the proposed solution?
- How does it solve the problem?
- Evaluation?
- Drawbacks?