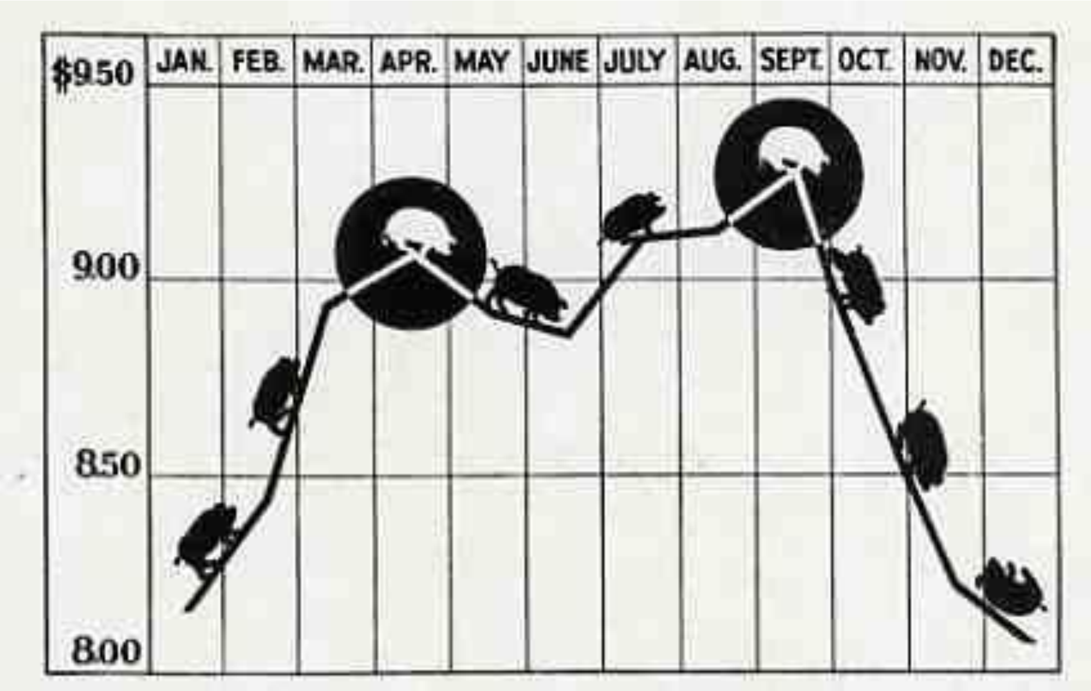
NoSQL Graph Databases



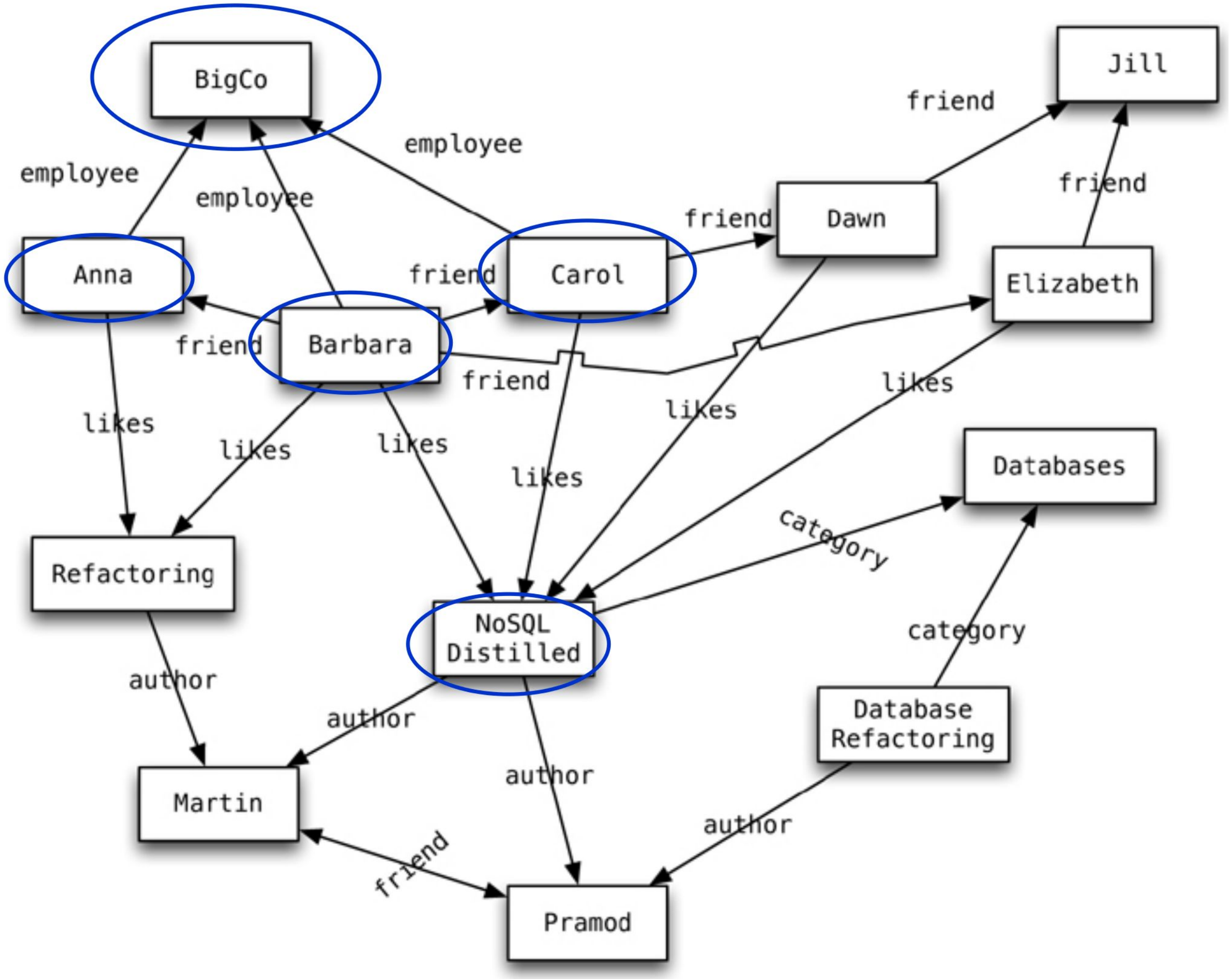
Problem Set #4 is graded
Problem Set #5 is due tonight

Graph Databases: Concept

- To store entities and relationships between them
  - Nodes are instances of objects
  - Nodes have properties, e.g., name
  - Edges connect nodes and are directed
  - Edges have types (e.g., likes, friend, …)

- Nodes are organized by relationships
  - Allow to find interesting patterns
  - **example:** Get all nodes that are "employee" of "Big Company" and that "likes" "NoSQL Distilled"

Graph Databases: Example

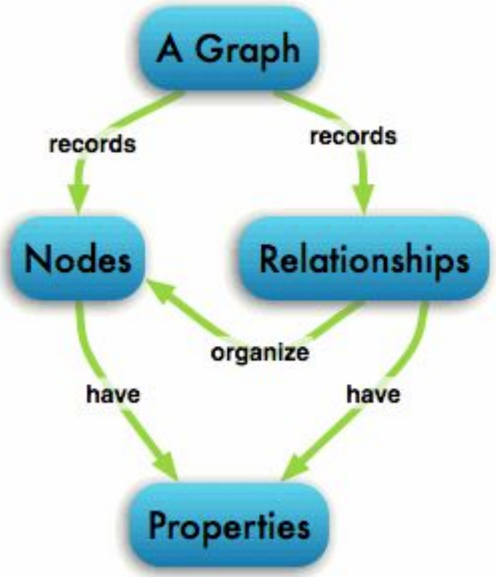


source: Sadalage & Fowler: NoSQL Distilled, 2012

Graph Databases: Representatives

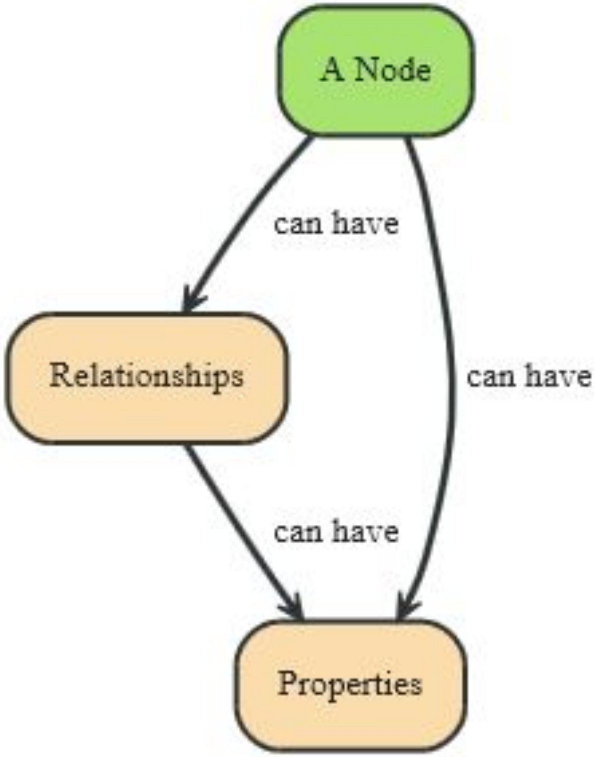
Neo4j: An exemplar Graph database

- Open source graph database
  - The most popular
- Initial release: 2007
- Written in: Java
- OS: cross-platform
- Stores data as nodes connected by directed, typed relationships
  - With properties on both
  - Called the "property graph"



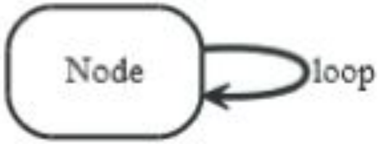
Neo4j: Data Model

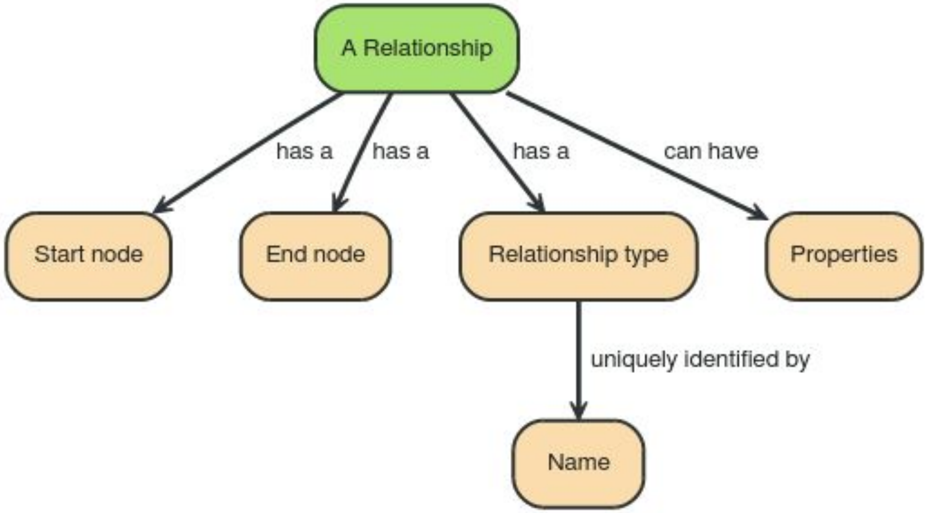
- Fundamental units: nodes + relationships
- Both can contain properties
  - Key-value pairs
  - Value can be of primitive type or an array of primitive type
  - null is not a valid property value
    - nulls can be modelled by the absence of a key



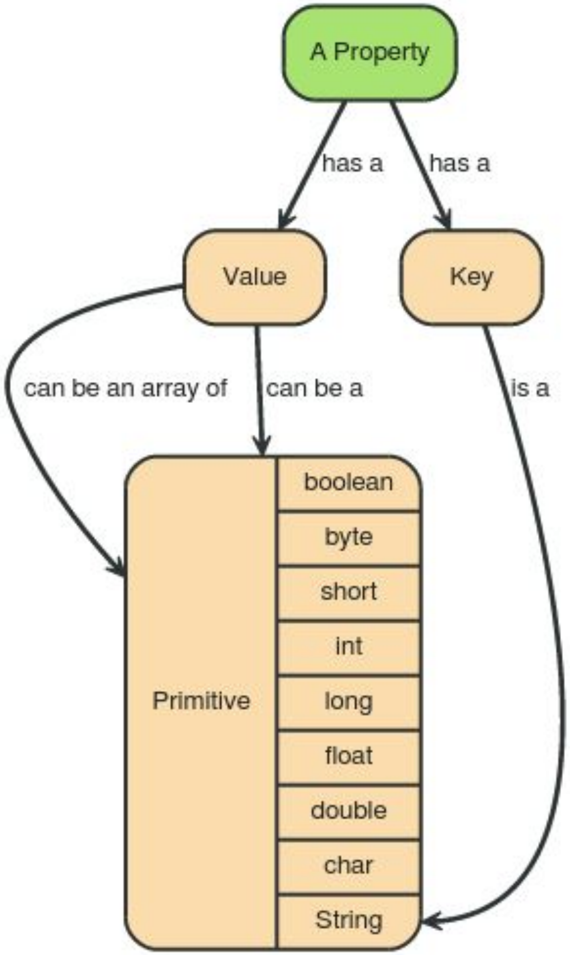
Data Model: Relationships

- Directed relationships (edges)
  - Incoming and outgoing edge
    - Equally efficient traversal in both directions
    - Direction can be ignored
      if not needed by the application
  - Always a start
    and an end node
    - Can be recursive



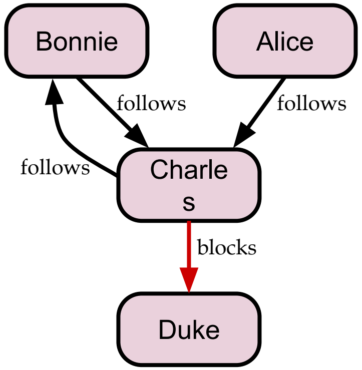


Data Model: Properties



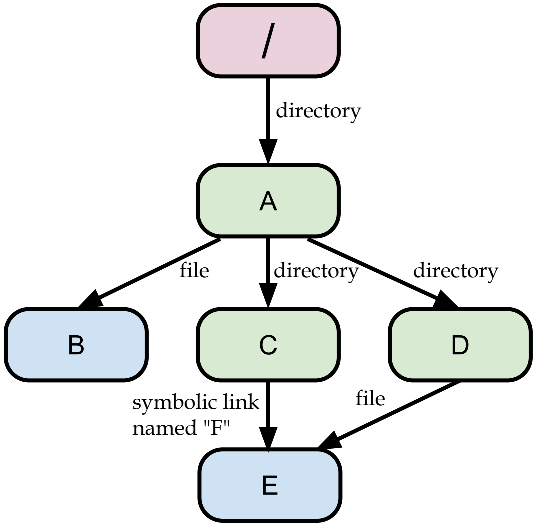
Type	Description
boolean	true/false
byte	8-bit integer
short	16-bit integer
int	32-bit integer
long	64-bit integer
float	32-bit IEEE 754 floating-point number
double	64-bit IEEE 754 floating-point number
char	16-bit unsigned integers representing Unicode characters
String	sequence of Unicode characters

Examples



What	How
get who a person follows	outgoing *follows* relationships, depth one
get the followers of a person	incoming *follows* relationships, depth one
get who a person blocks	outgoing *blocks* relationships, depth one

What	How
get the full path of a file	incoming *file* relationships
get all paths for a file	incoming *file* and *symbolic link* relationships
get all files in a directory	outgoing *file* and *symbolic link* relationships, depth one
get all files in a directory, excluding symbolic links	outgoing *file* relationships, depth one
get all files in a directory, recursively	outgoing *file* and *symbolic link* relationships



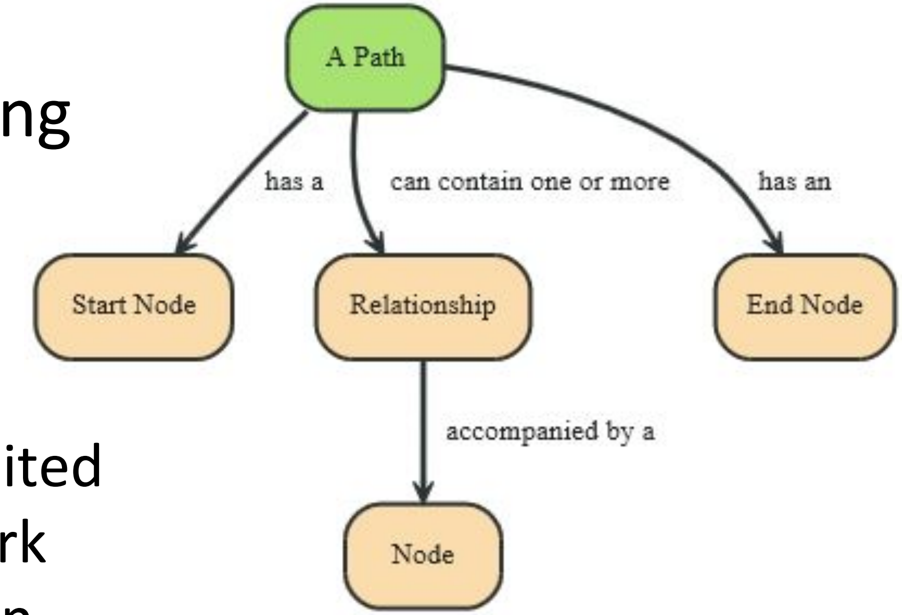
Native Java Interface: Example

```
Node alice = graphDb.createNode();
alice.setProperty("name", "Alice");
Node bonnie = graphDb.createNode();
bonnie.setProperty("name", "Bonnie");

Relationship a2b = alice.createRelationshipTo(bonnie,
FRIEND);
Relationship b2a = bonnie.createRelationshipTo(alice,
FRIEND);

a2b.setProperty("quality", "a good one");
b2a.setProperty("since", 2003);
```

- Undirected edge:
  - Relationship between the nodes in both directions
  - **INCOMING** and **OUTGOING** relationships from a node

Data Model: Traversal + Path

- Path = one or more nodes + connecting relationships
  - Typically retrieved as a result of a query or a traversal



- Traversing a graph = visiting its nodes, following relationships according to some rules
  - Typically, a subgraph is visited
  - Neo4j: Traversal framework + Java API, Cypher, Gremlin

Traversal Framework

- A traversal is influenced by
  - Starting node(s) where the traversal will begin
  - Expanders – defines what edges there are to traverse
    - i.e., relationship direction and type
  - Order – depth-first / breadth-first
  - Uniqueness – visit nodes (relationships, paths) only once
  - Evaluator – what to return and whether to stop or continue traversal beyond a current position

Traversal = TraversalDescription + starting node(s)

Traversal Framework – Java API

- `org.neo4j...TraversalDescription`
  - The main interface for defining traversals
    - Can specify branch ordering `breadthFirst()` / `depthFirst()`

- `.relationships()`
  - Adds the relationship type to traverse
    - e.g., traverse only edge types: FRIEND, RELATIVE
    - Empty (default) = traverse all relationships
  - Can also specify direction
    - `Direction.BOTH`
    - `Direction.INCOMING`
    - `Direction.OUTGOING`

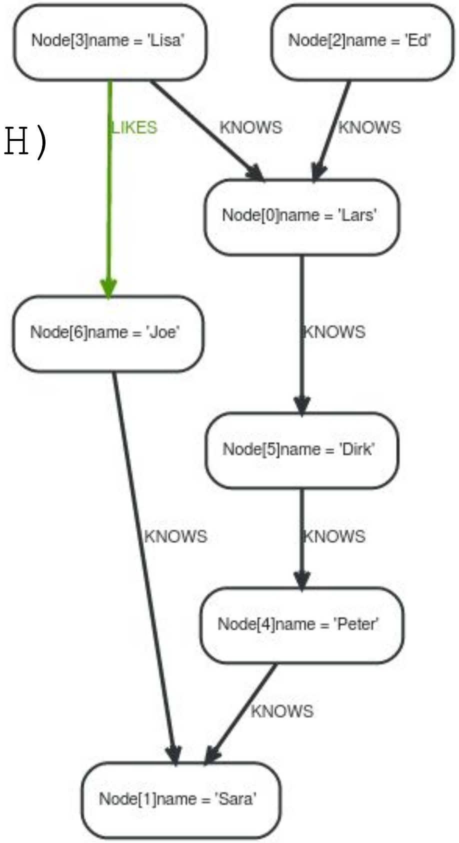
Traversal Framework – Java API (3)

- `org.neo4j...Uniqueness`
  - Can be supplied to the TraversalDescription
  - Indicates under what circumstances a traversal may revisit the same position in the graph

- `Traverser`
  - Starts actual traversal given a TraversalDescription and starting node(s)
  - Returns an iterator over “steps” in the traversal
    - Steps can be: Path (default), Node, Relationship
  - The graph is actually traversed “lazily” (on request)

Example of Traversal

```
TraversalDescription desc =
  db.traversalDescription()
    .depthFirst()
    .relationships(Rels.KNOWS, Direction.BOTH)
    .evaluator(Evaluators.toDepth(3));

// node is ‘Ed’ (Node[2])
for (Node n : desc.traverse(node).nodes()) {
    output += n.getProperty("name") + ", ";
}
```



```
Output:  Ed, Lars, Lisa, Dirk, Peter,
```

Graph Database Summary

- Graph databases excel when objects are "indirectly" related to each other. Friends of friends, Cousins, your boss's boss's boss.

- Graph databases are suited for finding "structural patterns" in data.
  - If "X" buys "A", "B", "C" are they likely to buy "D"?

- When entites and their relationships are clustered

A Farewell to Files and Databases

Final Exam: 11/19 from 12pm-3pm
I will be available on Zoom, but you can leave if you want.
Open book, open notes, open-internet
No human communication

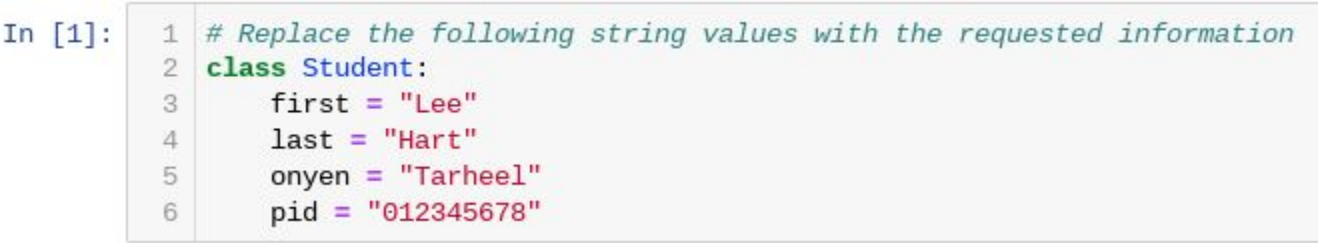
15 questions Jupyter Notebook
10 covering materials since the last midterm;
5 comprehensive

Grading Status

- Midterm
  - To my knowledge all issues are resolved and exams are graded
- Problem Sets (lowest score is dropped)
  - Problem Set #5 graded soon!
  - All *issues* with other problem sets are resolves
- Exercises
  - Everyone will get 100%
- If you still have any issues see me after class today or during my office hours tomorrow

Don't Mess Up!

1. Fill in your signature correctly!

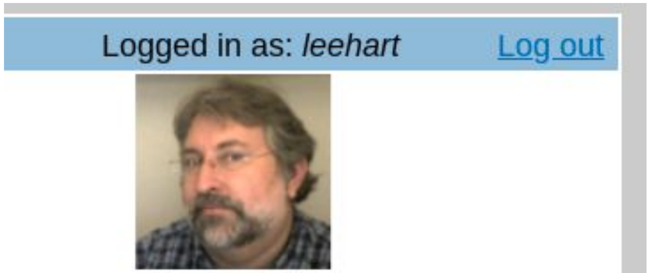


In [1]:

```
# Replace the following string values with the requested information
class Student:
    first = "Lee"
    last = "Hart"
    onyen = "Tarheel"
    pid = "012345678"
```

2. Make sure you are logged in when you submit!

(Your cookie lasts for more than 3 hours, so you should logout and then back in just before the exam)



Logged in as: *leehart* Log out

3. Don't submit the empty copy of the exam that you downloaded!

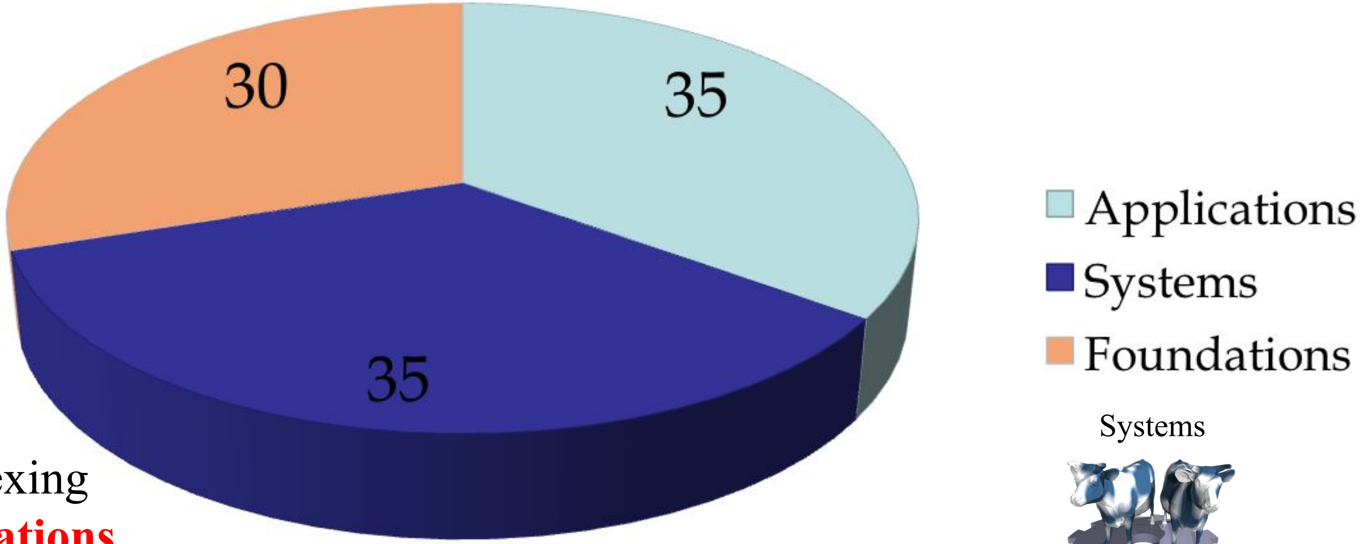
4. Use a local copy of Jupyter if possible.

Summary and What to study

- Relational Model
- **Out-of-core sorting**
- **Normal Forms**

Emphasis

- Structured Query Language
- Integrating Dbases & programs
- **NoSQL**
    - ○ **BASE, MapReduce, Hadoo**
    - ○ **Document Model**



- Database Indexing
- **Query Evaluations**
- **Query Optimization**
- **Transactions and Concurrency**