Tuesday, May 13, Practical Session II: Neo4j

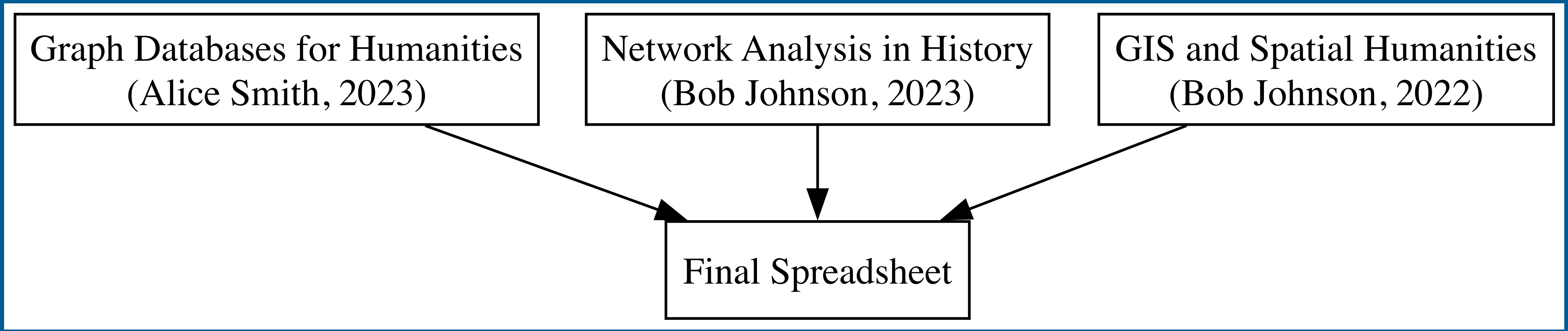
Ryan Horne
Advanced Research Computing, UCLA
ryan.matthew.horne@gmail.com

Agenda

- From data to databases
- Fundamentals of graph databases
- Neo4j operations

Tabular Data

- Familiar to most of us
- Excell, Google Sheets, etc
- *Rows* and *Columns*



ID	Title	Author	Year
1	Graph Databases for Humanities	Alice Smith	2023
2	Network Analysis in History	Bob Johnson	2023
3	GIS and Spatial Humanities	Bob Johnson	2022

Some Issues

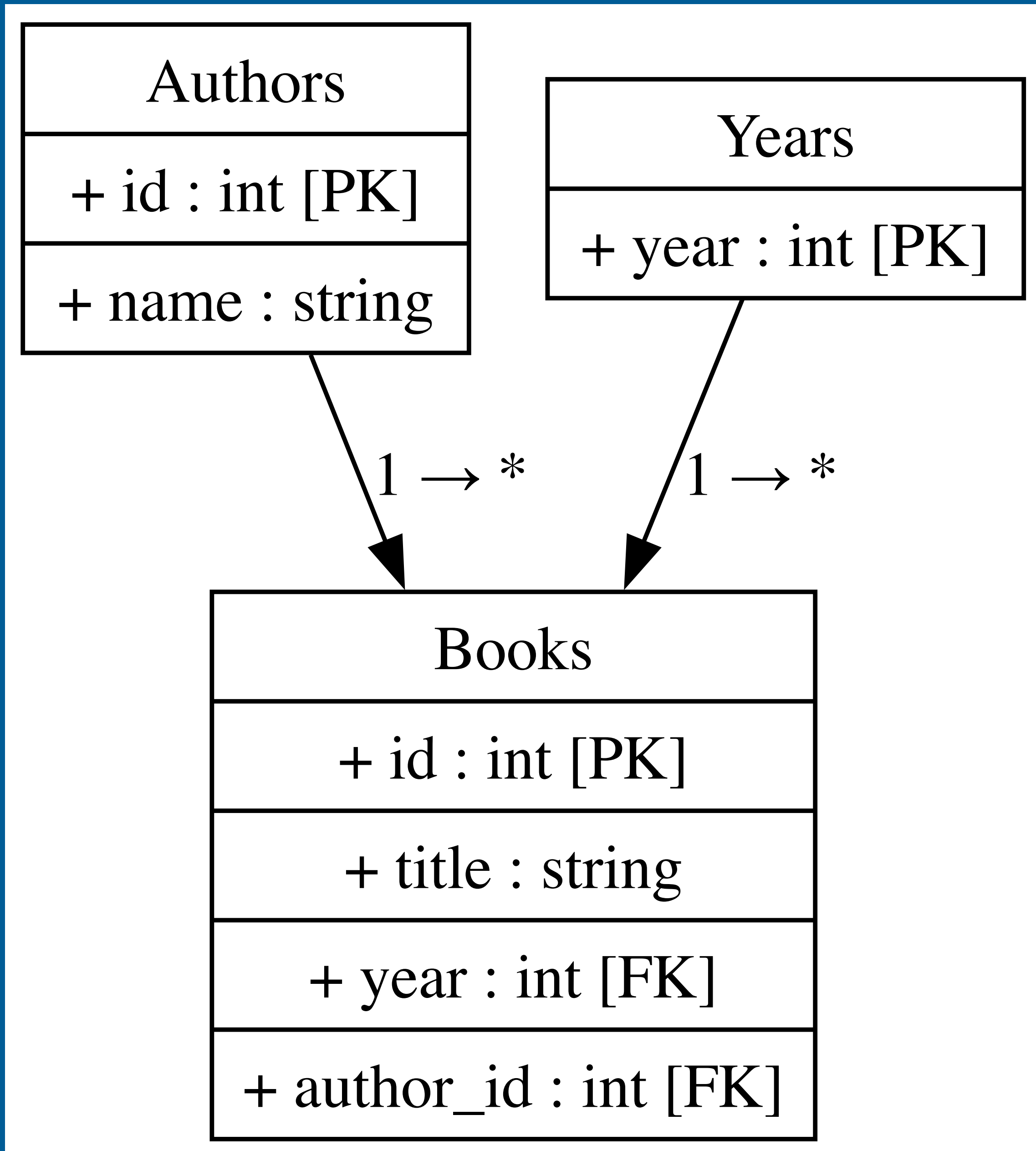
- Not a very flexible data model
- Data redundancy
- Cascading changes
- Not an easy way to link different tables and files

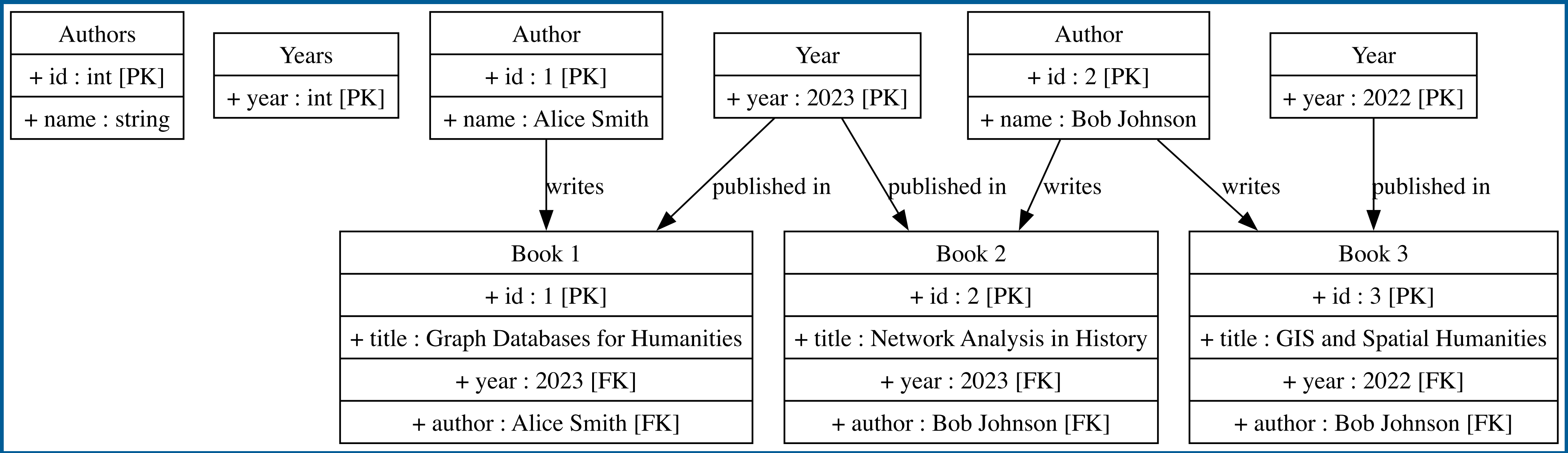
Relational Model

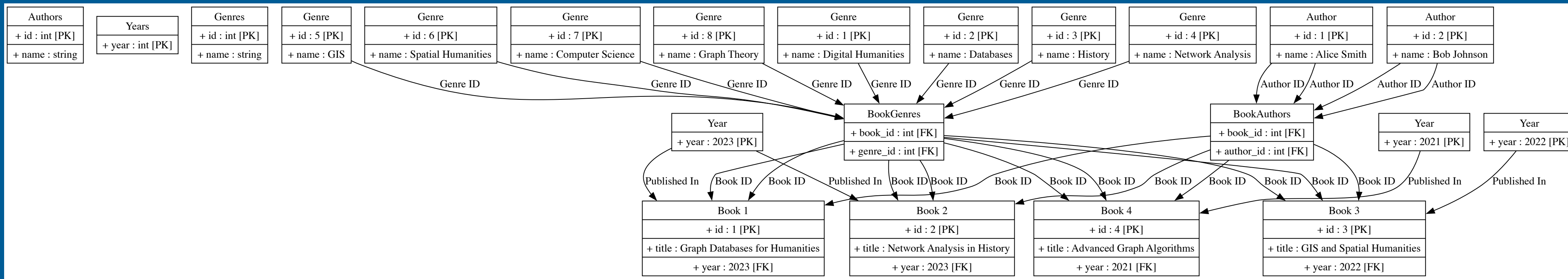
- A full decoupling of the logic of organizing data from the physical structures that house it
- Proposed in 70's; embraced by the 80's
- Schema driven
  - Structure predefined
- Standardized language through SQL
- Same idea as spreadsheets, but we can now link tables through *keys*

Keys to the Kingdom

- *Primary key*: Unique identifier (like ID) which is not repeated in that column
- *Foreign key*: A column whose value is the *primary key* of another row (entity)
- You can have multiple foreign keys in the same table; a linking table can have two columns with the values occurring multiple times
  - It would still have its *own* primary key though!


Authors
+ id : int [PK]
+ name : string
Years
+ year : int [PK]
1 → *
1 → *
Books
+ id : int [PK]
+ title : string
+ year : int [FK]
+ author_id : int [FK]


Authors
+ id : int [PK]
+ name : string
Years
+ year : int [PK]
Author
+ id : 1 [PK]
+ name : Alice Smith
Year
+ year : 2023 [PK]
Author
+ id : 2 [PK]
+ name : Bob Johnson
Year
+ year : 2022 [PK]
writes
published in
published in
writes
writes
published in
Book 1
+ id : 1 [PK]
+ title : Graph Databases for Humanities
+ year : 2023 [FK]
+ author : Alice Smith [FK]
Book 2
+ id : 2 [PK]
+ title : Network Analysis in History
+ year : 2023 [FK]
+ author : Bob Johnson [FK]
Book 3
+ id : 3 [PK]
+ title : GIS and Spatial Humanities
+ year : 2022 [FK]
+ author : Bob Johnson [FK]


Authors
+ id : int [PK]
+ name : string
Years
+ year : int [PK]
Genres
+ id : int [PK]
+ name : string
Genre
+ id : 5 [PK]
+ name : GIS
Genre
+ id : 6 [PK]
+ name : Spatial Humanities
Genre
+ id : 7 [PK]
+ name : Computer Science
Genre
+ id : 8 [PK]
+ name : Graph Theory
Genre
+ id : 1 [PK]
+ name : Digital Humanities
Genre
+ id : 2 [PK]
+ name : Databases
Genre
+ id : 3 [PK]
+ name : History
Genre
+ id : 4 [PK]
+ name : Network Analysis
Author
+ id : 1 [PK]
+ name : Alice Smith
Author
+ id : 2 [PK]
+ name : Bob Johnson
Genre ID
Author ID
BookGenres
+ book_id : int [FK]
+ genre_id : int [FK]
BookAuthors
+ book_id : int [FK]
+ author_id : int [FK]
Year
+ year : 2023 [PK]
Year
+ year : 2021 [PK]
Year
+ year : 2022 [PK]
Published In
Book ID
Book 1
+ id : 1 [PK]
+ title : Graph Databases for Humanities
+ year : 2023 [FK]
Book 2
+ id : 2 [PK]
+ title : Network Analysis in History
+ year : 2023 [FK]
Book 4
+ id : 4 [PK]
+ title : Advanced Graph Algorithms
+ year : 2021 [FK]
Book 3
+ id : 3 [PK]
+ title : GIS and Spatial Humanities
+ year : 2022 [FK]

Problems with the Relational Model

- Structure known *before* data is added
- Complex queries to represent many to 1 and many to many situations
- Despite the name….not really about relations!

A Sequel to SQL: NoSQL

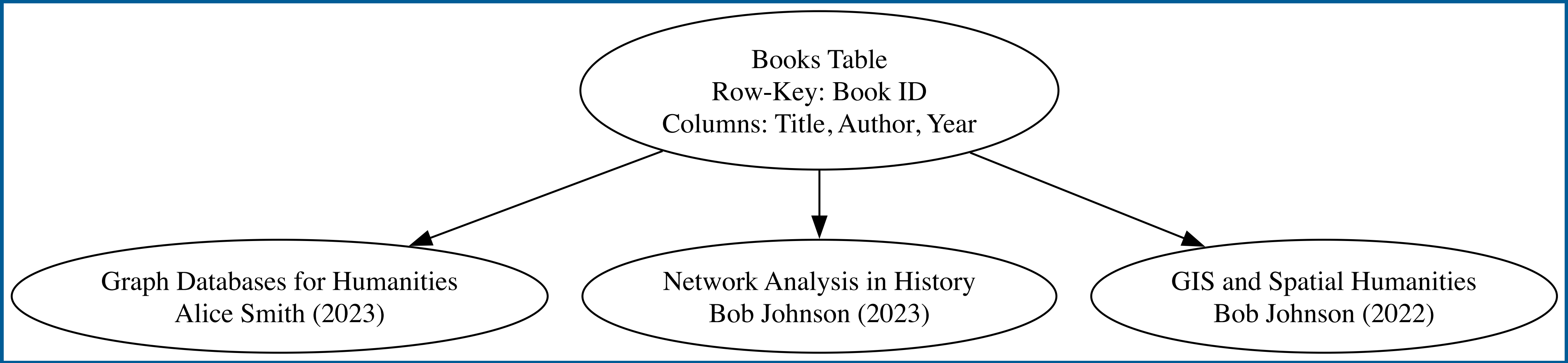
- NoSQL = Not Only SQL
- Umbrella term for databases that:
  - Do not (only) use SQL
  - May not have a fixed schema
  - Might Not use tabular structure
- Key Part: A *flexible* schema

Main Categories

- Wide Column Stores / Column-Family
- Key-Value
- Document
- Graph

Wide Column Stores / Column-Family

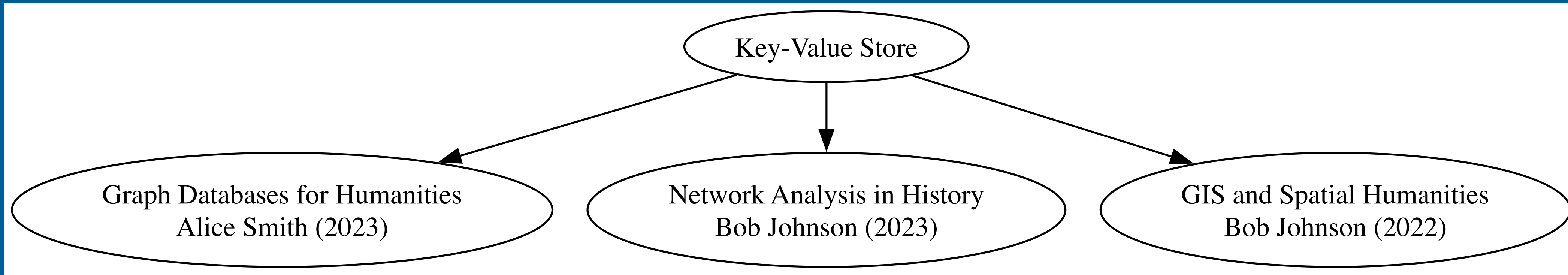
- Rows and Columns in a tabular format
- Row key identifies each record
- Columns grouped into families
- Number of columns for each row is not fixed


Books Table
Row-Key: Book ID
Columns: Title, Author, Year
Graph Databases for Humanities
Alice Smith (2023)
Network Analysis in History
Bob Johnson (2023)
GIS and Spatial Humanities
Bob Johnson (2022)

Feature	Column-Family Database
Structure	Columns grouped into families
Schema	Flexible or semi-structured schema
Query Language	Query language (e.g., CQL in Cassandra)
Relationships	Limited or no relationships
Use Cases	Time-series data, IoT, logs, and analytics

Key-Value

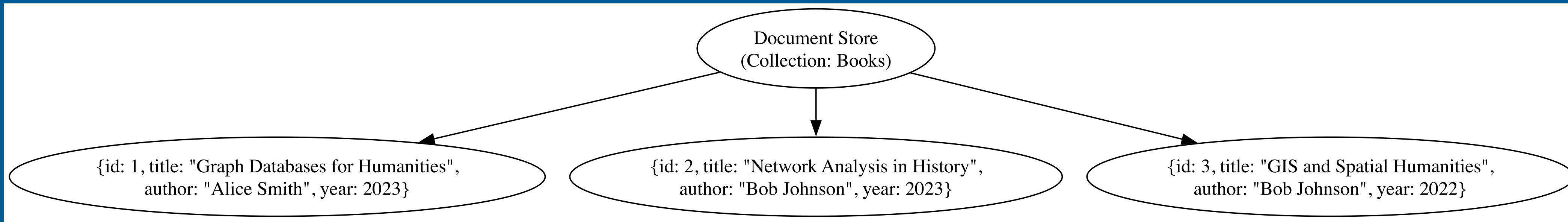
- Key-value pairs, data structures where the *key* is used to identify an entity that is associated with a *value*
- Very Flexible
- Terrible at storing relationships


Key-Value Store
Graph Databases for Humanities
Alice Smith (2023)
Network Analysis in History
Bob Johnson (2023)
GIS and Spatial Humanities
Bob Johnson (2022)

Feature	Key-Value Database
Structure	Key-value pairs
Schema	Schema-free
Query Language	Simple API (get, put, delete)
Relationships	No relationships by default
Use Cases	Caching, session storage, configuration

Document Model

- Extends the idea of Key:Value pairs by introducing the paradigm of storing data as *documents*
- Typically in a JSON, BSON, or XML format
- Each document can have its own structure and schema as required
- Again not great at relationships


Document Store
(Collection: Books)
{id: 1, title: "Graph Databases for Humanities",
author: "Alice Smith", year: 2023}
{id: 2, title: "Network Analysis in History",
author: "Bob Johnson", year: 2023}
{id: 3, title: "GIS and Spatial Humanities",
author: "Bob Johnson", year: 2022}

Feature	Document Database
Structure	JSON or BSON documents
Schema	Schema-free or dynamic schema
Query Language	JSON-based query API (e.g., MongoDB Query Language)
Relationships	Embedded documents or references
Scalability	Horizontal scaling
Use Cases	Content management, catalogs, user profiles

On to Graph Databases!

What is a Graph Database?

- Key components: nodes, relationships, and properties
- Built on a network data model

Basic Idea Revisited

- Networks are a collection of entities
- At least some are *linked*
- All kinds of subject domains
- Very flexible definition

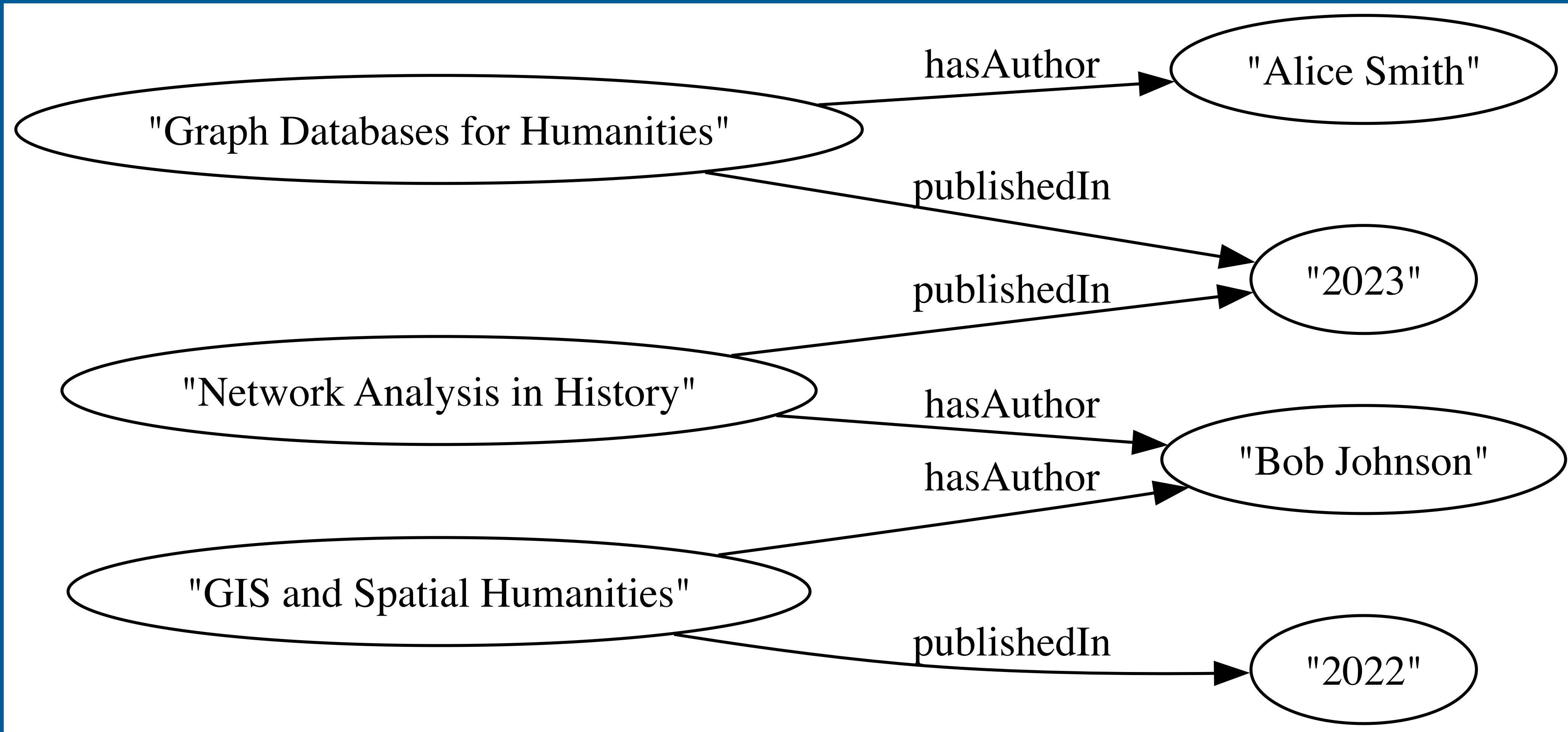
Types of Graph Databases

There are two main types:

- RDF / Triple Store
- Linked Property
- Also Wikidata/Wikibase which is not a *true* graph database

RDF/Triple Store Graphs

- Backbone of the semantic web and almost all Linked Open Data
- Key technology in Big Data
- RDF is a data exchange standard from W3C
  - Data model of subject (the entity being described)-predicate-(the relationship between the subject and the object) and the object (the target or value of the statement).
  - `<This person><wrote><that book>`

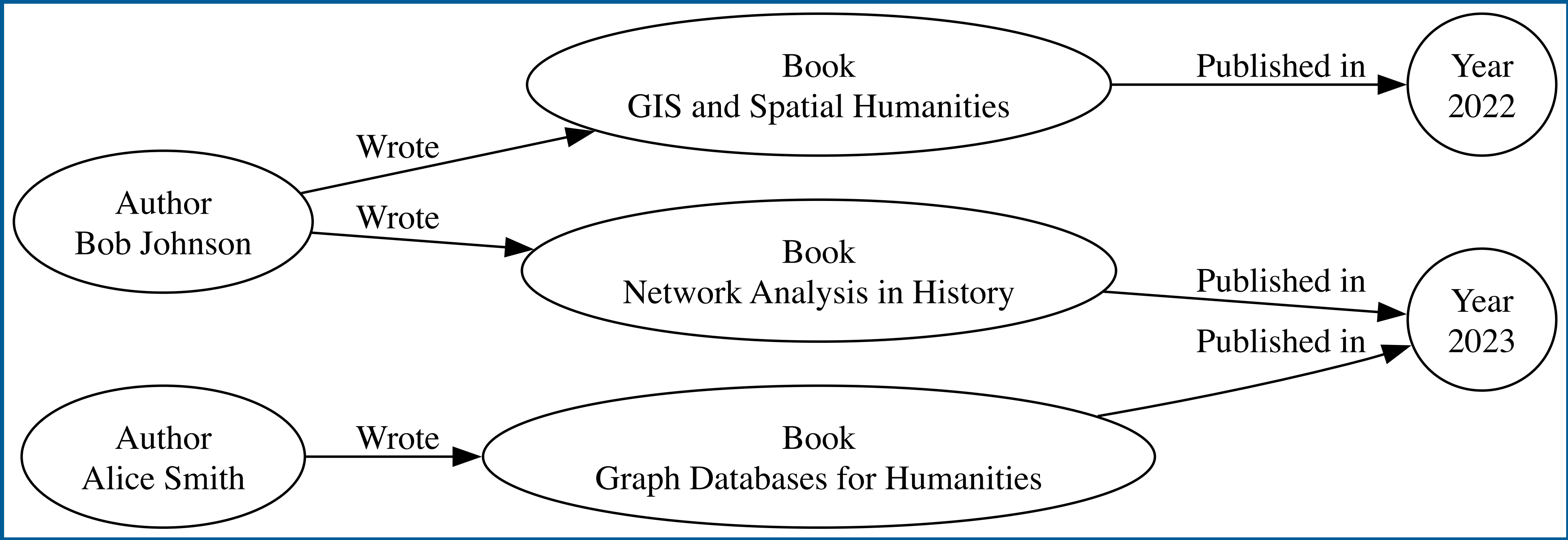


"Graph Databases for Humanities"
hasAuthor
"Alice Smith"
publishedIn
"2023"
"Network Analysis in History"
publishedIn
hasAuthor
"Bob Johnson"
"GIS and Spatial Humanities"
hasAuthor
publishedIn
"2022"

Feature	RDF Graph Database
Structure	Triples (subject-predicate-object)
Schema	Schema-optional (RDFS/OWL for semantics)
Query Language	SPARQL (Semantic query language)
Relationships	Relationships via triples (semantic links)
Use Cases	Linked data, semantic web, ontologies

Labeled Property Graphs

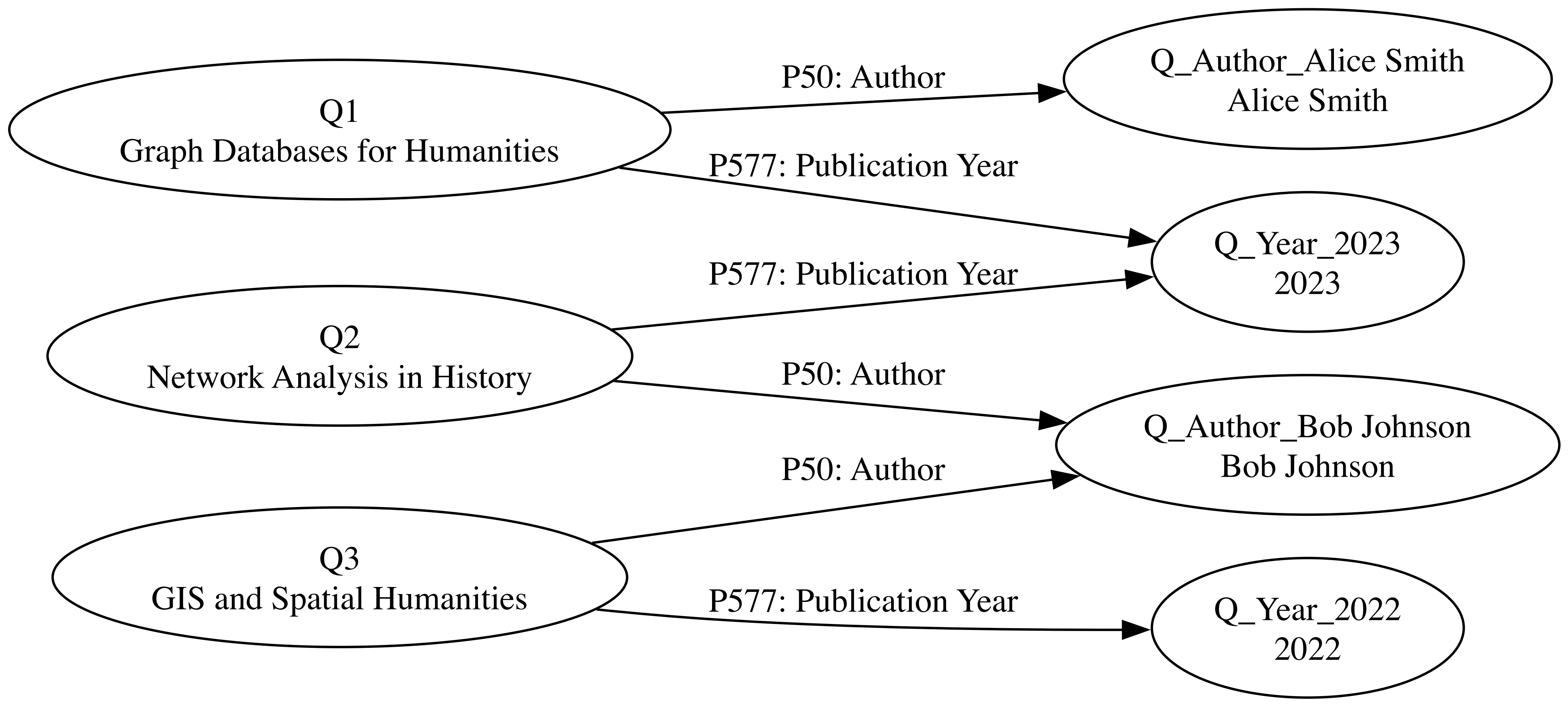
- Allow labeling for nodes (sometimes called vertices or actors) and edges, along with any number of properties for each
- Relationships are inherently directed, with a start and an end node
- Nodes and edges are primitive types; no need to linking tables
- No industry standard querying language
- Neo4j is currently the most popular implementation


Author
Bob Johnson
Wrote
Wrote
Book
GIS and Spatial Humanities
Published in
Year
2022
Book
Network Analysis in History
Published in
Year
2023
Published in
Author
Alice Smith
Wrote
Book
Graph Databases for Humanities

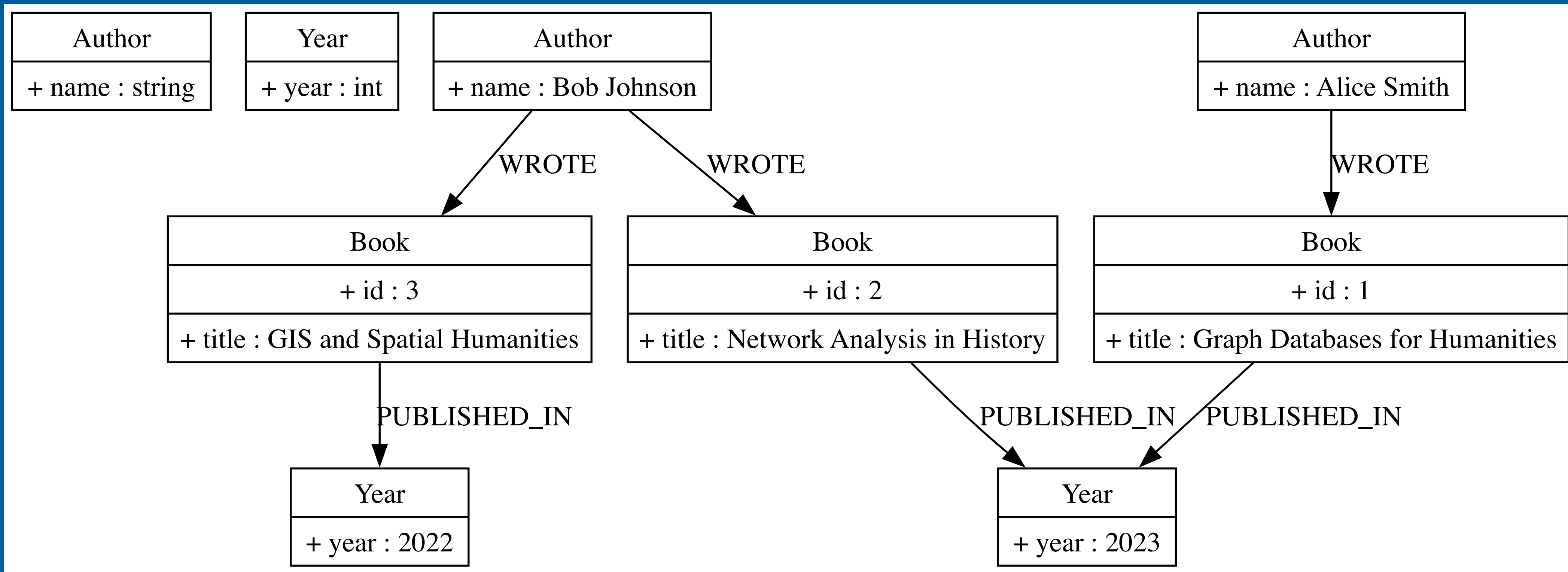
Feature	Property Graph Database
Structure	Nodes, edges, and properties
Schema	Schema-optional or dynamic
Query Language	Graph-specific (e.g., Cypher, Gremlin)
Relationships	Native relationships with edges
Use Cases	Social networks, recommendation engines, fraud detection

A Hybrid Approach: The Case of Wikidata/ Wikibase

- Not a “true” graph database
  - Underlying database is relational, but uses graph-like structures
- Entity-Relationship model
- *Items* have *properties* which can connect to other items
- Primarily intended to work with semantic web so uses RDF like structures and encoding

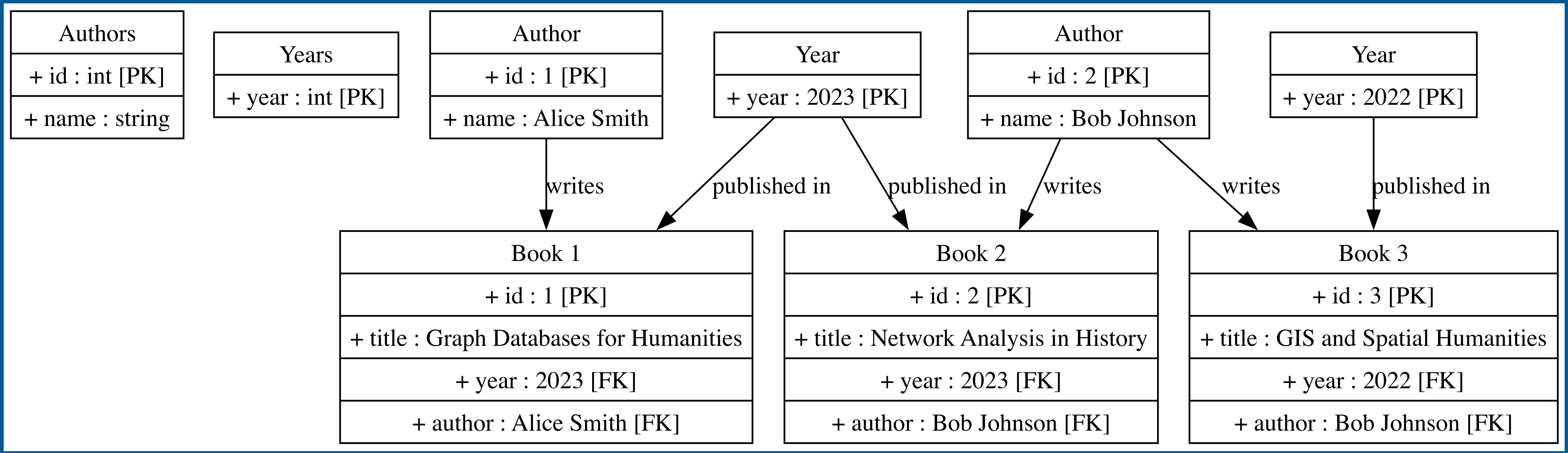

Q1
Graph Databases for Humanities
Q2
Network Analysis in History
Q3
GIS and Spatial Humanities
P50: Author
P577: Publication Year
P577: Publication Year
P50: Author
P50: Author
P577: Publication Year
Q_Author_Alice Smith
Alice Smith
Q_Year_2023
2023
Q_Author_Bob Johnson
Bob Johnson
Q_Year_2022
2022

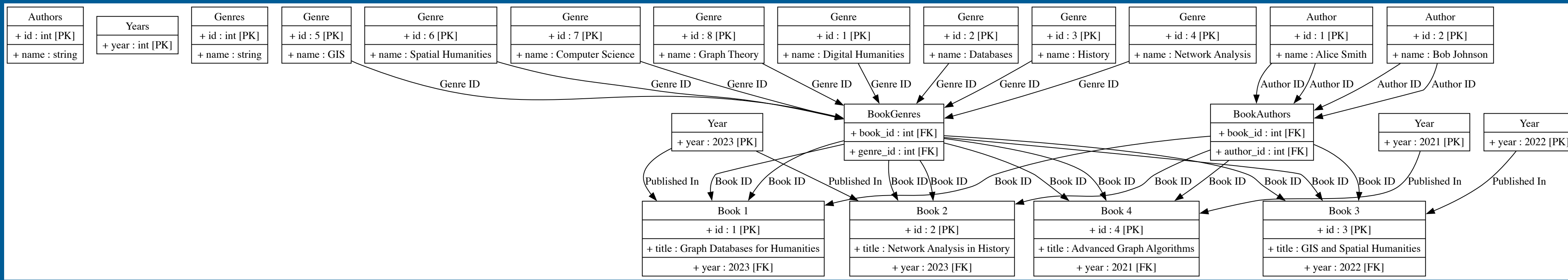
Feature	Wikibase	Graph Database	SQL Database
Structure	Entities (items, properties, values), statements	Nodes, edges, and properties	Tables with rows and columns
Schema	Schema-optional (but predefined entities like items, properties)	Schema-optional or dynamic	Predefined schema required
Query Language	SPARQL, Wikibase Query Service (WQS)	Graph-specific (e.g., Cypher, Gremlin)	SQL
Relationships	Relationships via statements (subject, predicate, object)	Native relationships with edges	Relationships via foreign keys

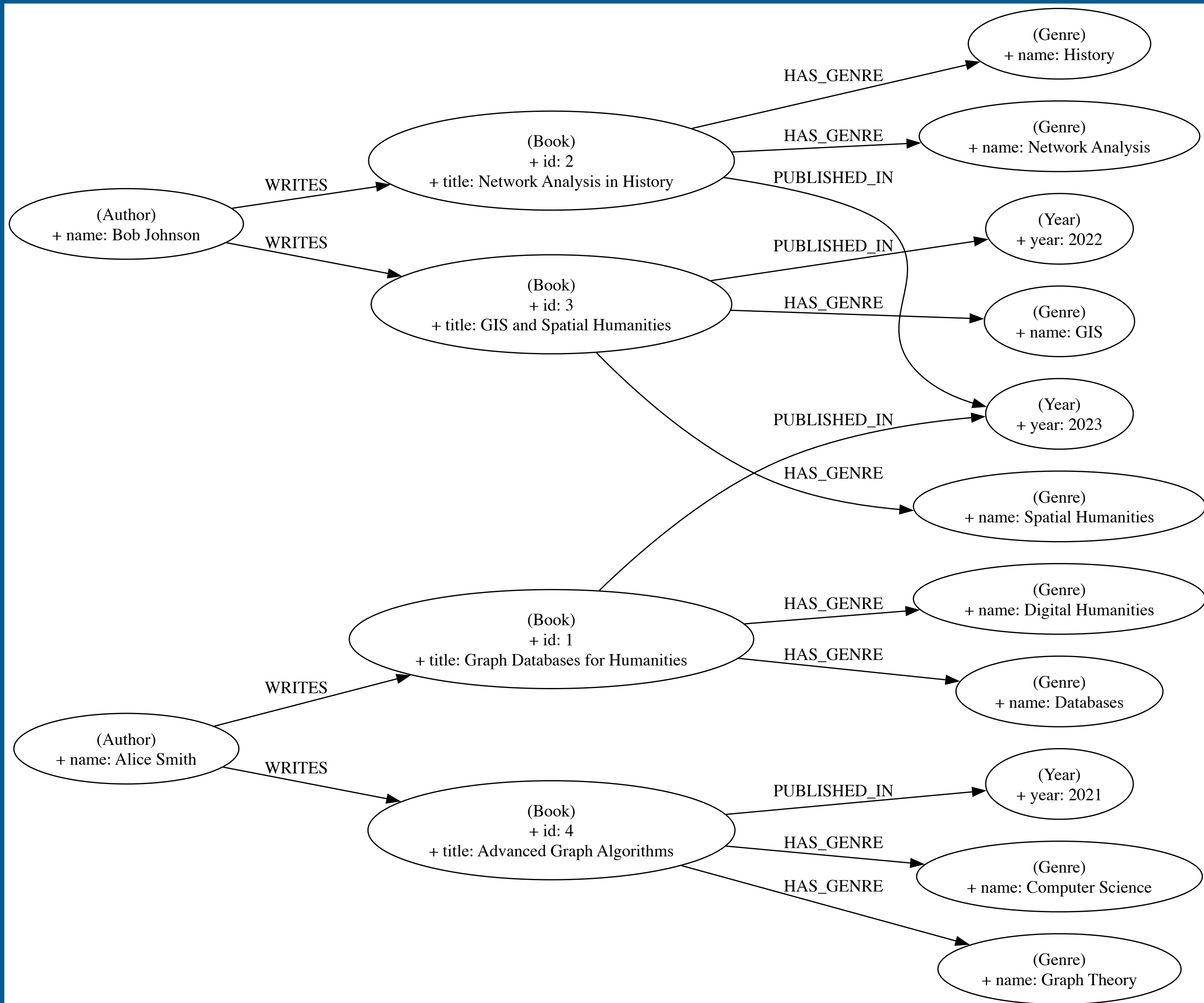

Author
+ name : string
Year
+ year : int
Author
+ name : Bob Johnson
Author
+ name : Alice Smith
WROTE
WROTE
WROTE
Book
+ id : 3
+ title : GIS and Spatial Humanities
Book
+ id : 2
+ title : Network Analysis in History
Book
+ id : 1
+ title : Graph Databases for Humanities
PUBLISHED_IN
PUBLISHED_IN
PUBLISHED_IN
Year
+ year : 2022
Year
+ year : 2023

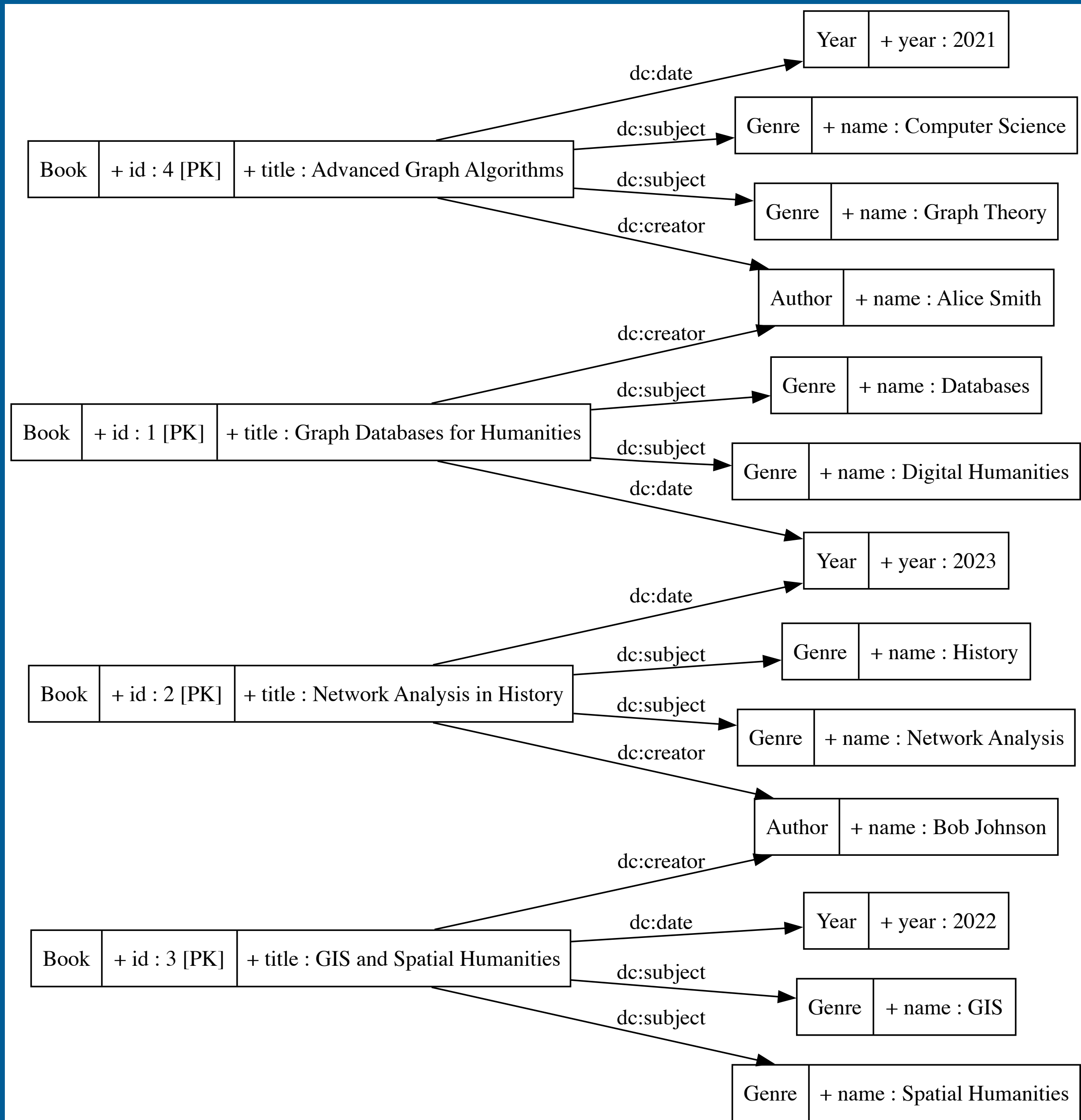
When to use a Graph Database?

- Modeling complex relationships (e.g., networks of people, places, and texts).
- Supporting research questions related to connectivity, lineage, and influence.


Authors
+ id : int [PK]
+ name : string
Years
+ year : int [PK]
Author
+ id : 1 [PK]
+ name : Alice Smith
Year
+ year : 2023 [PK]
Author
+ id : 2 [PK]
+ name : Bob Johnson
Year
+ year : 2022 [PK]
writes
published in
published in
writes
writes
published in
Book 1
+ id : 1 [PK]
+ title : Graph Databases for Humanities
+ year : 2023 [FK]
+ author : Alice Smith [FK]
Book 2
+ id : 2 [PK]
+ title : Network Analysis in History
+ year : 2023 [FK]
+ author : Bob Johnson [FK]
Book 3
+ id : 3 [PK]
+ title : GIS and Spatial Humanities
+ year : 2022 [FK]
+ author : Bob Johnson [FK]


Authors
+ id : int [PK]
+ name : string
Years
+ year : int [PK]
Genres
+ id : int [PK]
+ name : string
Genre
+ id : 5 [PK]
+ name : GIS
Genre
+ id : 6 [PK]
+ name : Spatial Humanities
Genre
+ id : 7 [PK]
+ name : Computer Science
Genre
+ id : 8 [PK]
+ name : Graph Theory
Genre
+ id : 1 [PK]
+ name : Digital Humanities
Genre
+ id : 2 [PK]
+ name : Databases
Genre
+ id : 3 [PK]
+ name : History
Genre
+ id : 4 [PK]
+ name : Network Analysis
Author
+ id : 1 [PK]
+ name : Alice Smith
Author
+ id : 2 [PK]
+ name : Bob Johnson
Genre ID
Author ID
BookGenres
+ book_id : int [FK]
+ genre_id : int [FK]
BookAuthors
+ book_id : int [FK]
+ author_id : int [FK]
Year
+ year : 2023 [PK]
Year
+ year : 2021 [PK]
Year
+ year : 2022 [PK]
Published In
Book ID
Book 1
+ id : 1 [PK]
+ title : Graph Databases for Humanities
+ year : 2023 [FK]
Book 2
+ id : 2 [PK]
+ title : Network Analysis in History
+ year : 2023 [FK]
Book 4
+ id : 4 [PK]
+ title : Advanced Graph Algorithms
+ year : 2021 [FK]
Book 3
+ id : 3 [PK]
+ title : GIS and Spatial Humanities
+ year : 2022 [FK]


(Author)
+ name: Bob Johnson
WRITES
WRITES
(Book)
+ id: 2
+ title: Network Analysis in History
(Book)
+ id: 3
+ title: GIS and Spatial Humanities
HAS_GENRE
HAS_GENRE
PUBLISHED_IN
PUBLISHED_IN
HAS_GENRE
PUBLISHED_IN
HAS_GENRE
(Genre)
+ name: History
(Genre)
+ name: Network Analysis
(Year)
+ year: 2022
(Genre)
+ name: GIS
(Year)
+ year: 2023
(Genre)
+ name: Spatial Humanities
(Author)
+ name: Alice Smith
WRITES
WRITES
(Book)
+ id: 1
+ title: Graph Databases for Humanities
(Book)
+ id: 4
+ title: Advanced Graph Algorithms
HAS_GENRE
HAS_GENRE
PUBLISHED_IN
HAS_GENRE
HAS_GENRE
(Genre)
+ name: Digital Humanities
(Genre)
+ name: Databases
(Year)
+ year: 2021
(Genre)
+ name: Computer Science
(Genre)
+ name: Graph Theory


Year | + year : 2021
dc:date
Genre | + name : Computer Science
dc:subject
Book | + id : 4 [PK] | + title : Advanced Graph Algorithms
dc:subject
Genre | + name : Graph Theory
dc:creator
Author | + name : Alice Smith
dc:creator
Genre | + name : Databases
dc:subject
Book | + id : 1 [PK] | + title : Graph Databases for Humanities
dc:subject
Genre | + name : Digital Humanities
dc:date
Year | + year : 2023
dc:date
Genre | + name : History
dc:subject
Book | + id : 2 [PK] | + title : Network Analysis in History
dc:subject
Genre | + name : Network Analysis
dc:creator
Author | + name : Bob Johnson
dc:creator
Year | + year : 2022
dc:date
Book | + id : 3 [PK] | + title : GIS and Spatial Humanities
dc:subject
Genre | + name : GIS
dc:subject
Genre | + name : Spatial Humanities

Why a Graph Database?

- Operations directly on a graph schema
    - Constraints can be used
- Relationships are seamless and can be changed when necessary
- Multi-layer intersecting domains and fields
- Vocabularies and ontologies to help with data modeling
- Focus on *connections*
- Ths is a *network* graph!!

Why a Graph Database?

- Models the thought process
- Each node can contain just what it needs and no more
- Ability to adapt and change to the data schema on the fly
- Already deigned to facilitate networks and network connections

Why a Graph Database?

- Outputs / inputs can be in different models
  - Tables are very popular
  - Can write scripts to transform lists / json / other structures
  - Programs like Gephi can be used as inputs with the right software
- Extensive support in python, quarto, etc for manipulation and presentation

Issues

- No standard language (although it is being worked on)
- Still not well-known in some fields of study
- Need to think of your data as a network
- Could be a lack of experience with digital / computational studies

Neo4j and Cypher

- Cypher is a graph query language that is used to query the Neo4j Database.
- It is proprietary, but there are open source movements
- Also heavily influencing the development of GQL, which will hopefully standardize things

Neo4j Deployment Options

- Neo4j Desktop: Ideal for local research and development
- Neo4j Aura (Cloud): Free and paid hosted options
- Docker: Advanced users and integration
- APOC & Graph Data Science: Libraries for extended functionality

Hands-On Neo4j SandBox

We are using the movies database

- Click on the database icon
- Look at the sections; we have:
  - node labels
  - relationship types
  - property keys
- Click on the `Person` button

Hands-On Neo4j SandBox

- In writing a Cypher query, a node is enclosed between a parenthesis
  - `(p:Person)`
  - p is a variable (what we are calling it) and *Person* is the node
  - Also needs a `RETURN` statement
  - We are using `MATCH` that specifies the patterns we will use in the data
    - Often with WHERE clause

Write something!

Can you think of a way to write the following query:

- Load movies into variable `m`
- Where the released property > 1999
- Now just return the *count*

Relationships

- Enclosed in square brackets
  - `[w:WORKS_FOR]`
    - w is a variable
    - WORKS_FOR is the type of relationship

What do you think this does?

```
MATCH (p:Person)-[d:DIRECTED]-(m:Movie)
WHERE m.released > 2010
RETURN p,d,m
```

Your Turn

Query to get all the people who acted in a movie that was released after 2010.

Another question

What is the difference between these two queries?

```
MATCH (p:Person)
RETURN p
LIMIT 20
```

```
MATCH (n)
RETURN n
LIMIT 20
```

MATCH vs MERGE

- MATCH assumes that a node with the specified query exsists
- MERGE adds the node if it does not

Match

```
MATCH (p:Person)
WHERE p.name = "John Doe"
SET p.personstatus = 'found'
RETURN p
```

Merge

```
MERGE (p:Person {name: "John Doe"})
ON MATCH SET p.personstatus = 'found'
RETURN p
```

Create a Relationship Between Nodes

```
MATCH (p:Person), (m:Movie)
WHERE p.name = "Tom Hanks" AND m.title = "Cloud Atlas"

CREATE (p)-[w:WATCHED]->(m)
RETURN type(w)
```

Wait, what is this?

- Did you see the “->” above?
- What is going on?

Directed vs. Undirected Graph

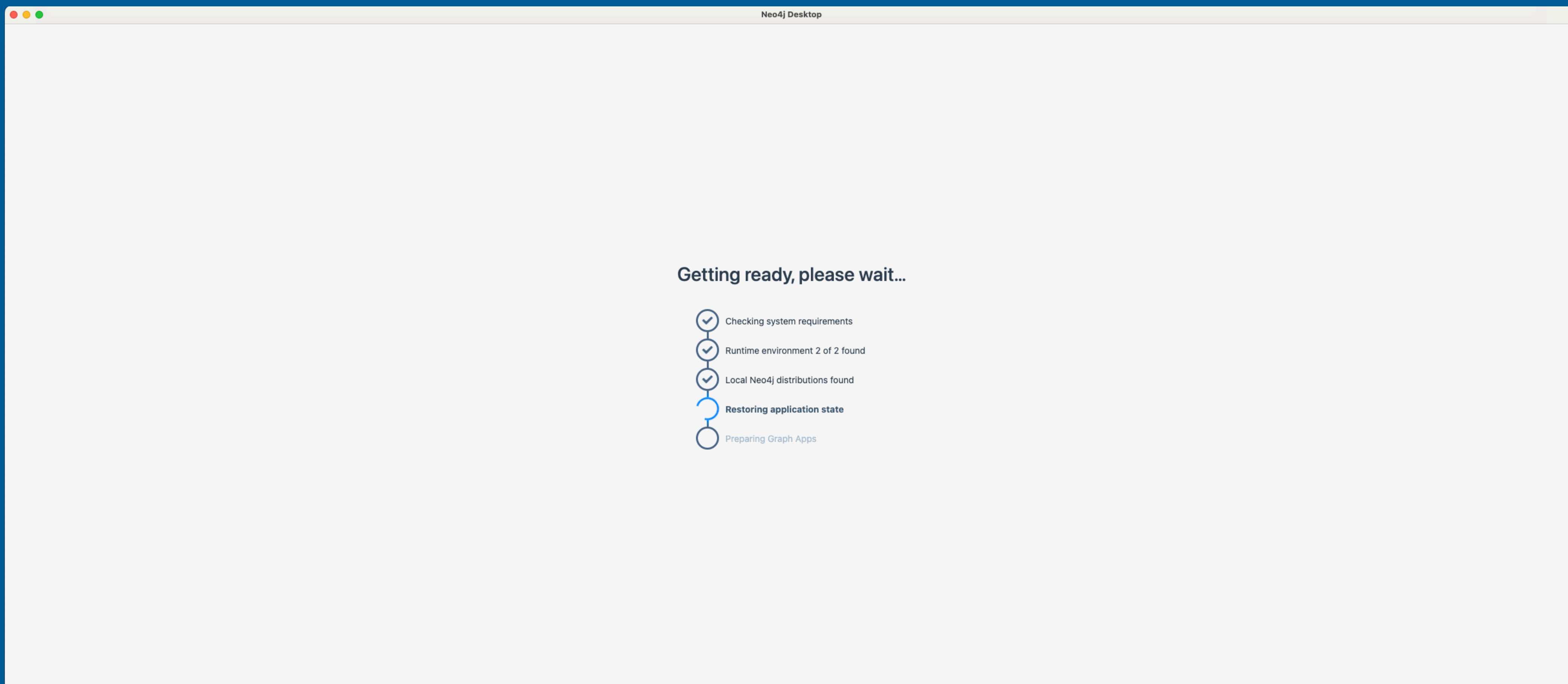
- We can specify the *direction* of the relationship
  - or just leave out the arrow for undirected connections
- Why might this be important?
- When do we not care (or that direction does not make sense?)

Find all People With any Relation to a Node

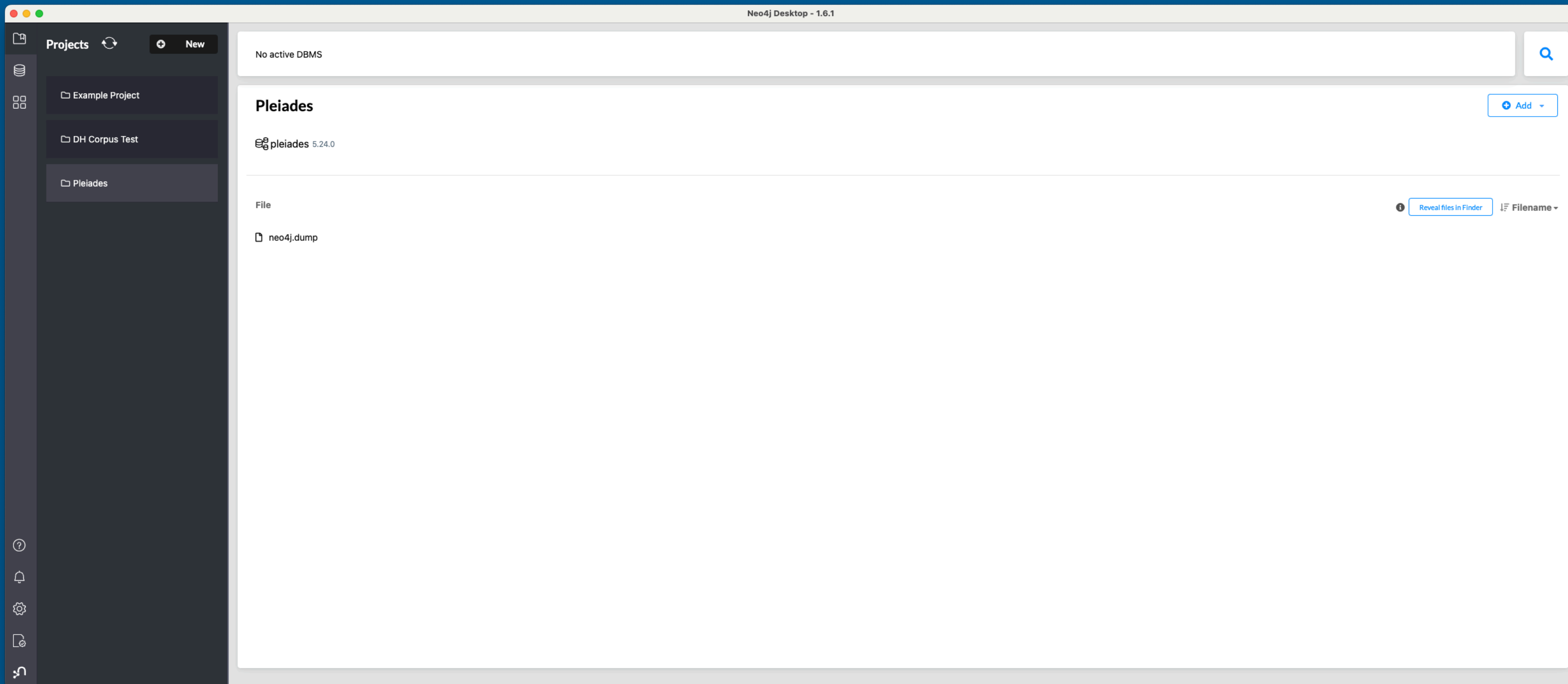
```

MATCH (p:Person)-[relatedTo]-(m:Movie {title: "Cloud Atlas"})
RETURN p.name, type(relatedTo)

```

Desktop Neo4j

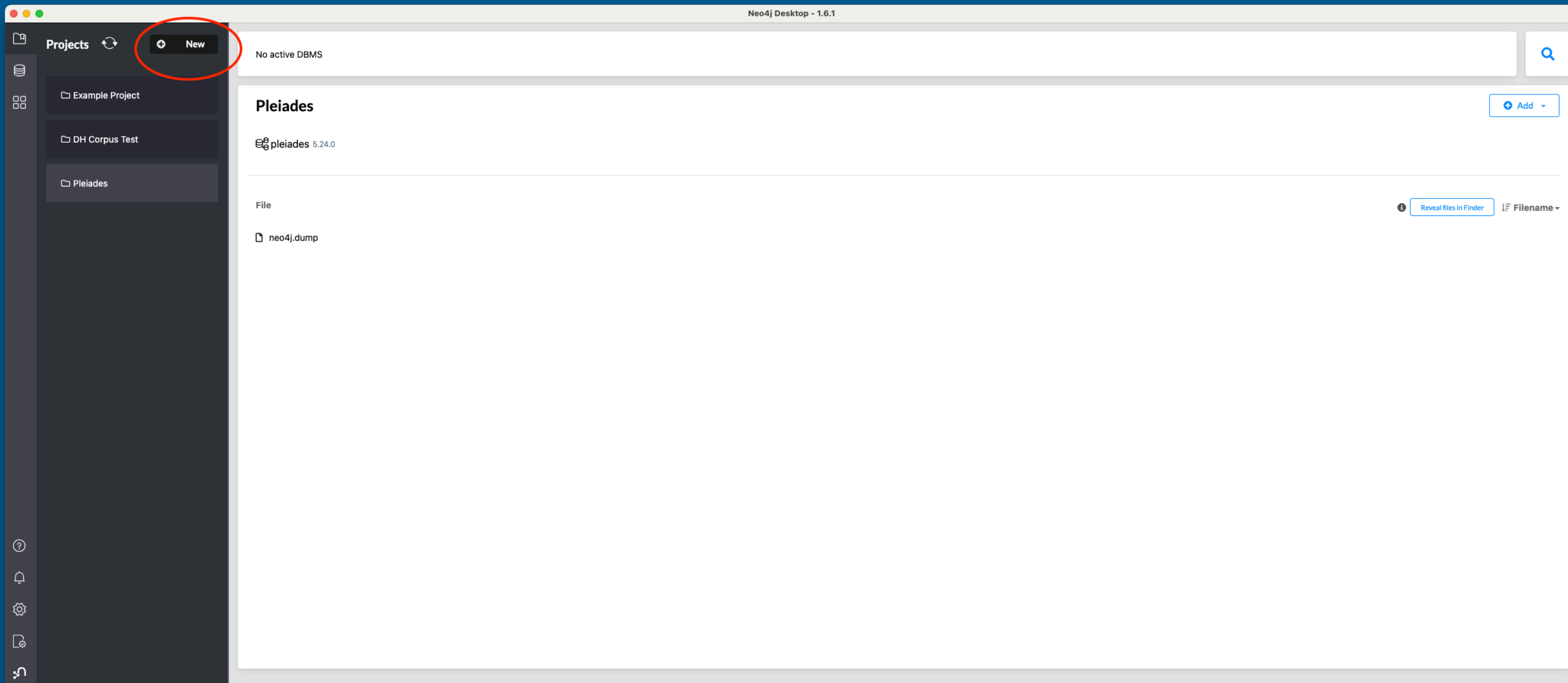
After we install the application, we should see something like this:



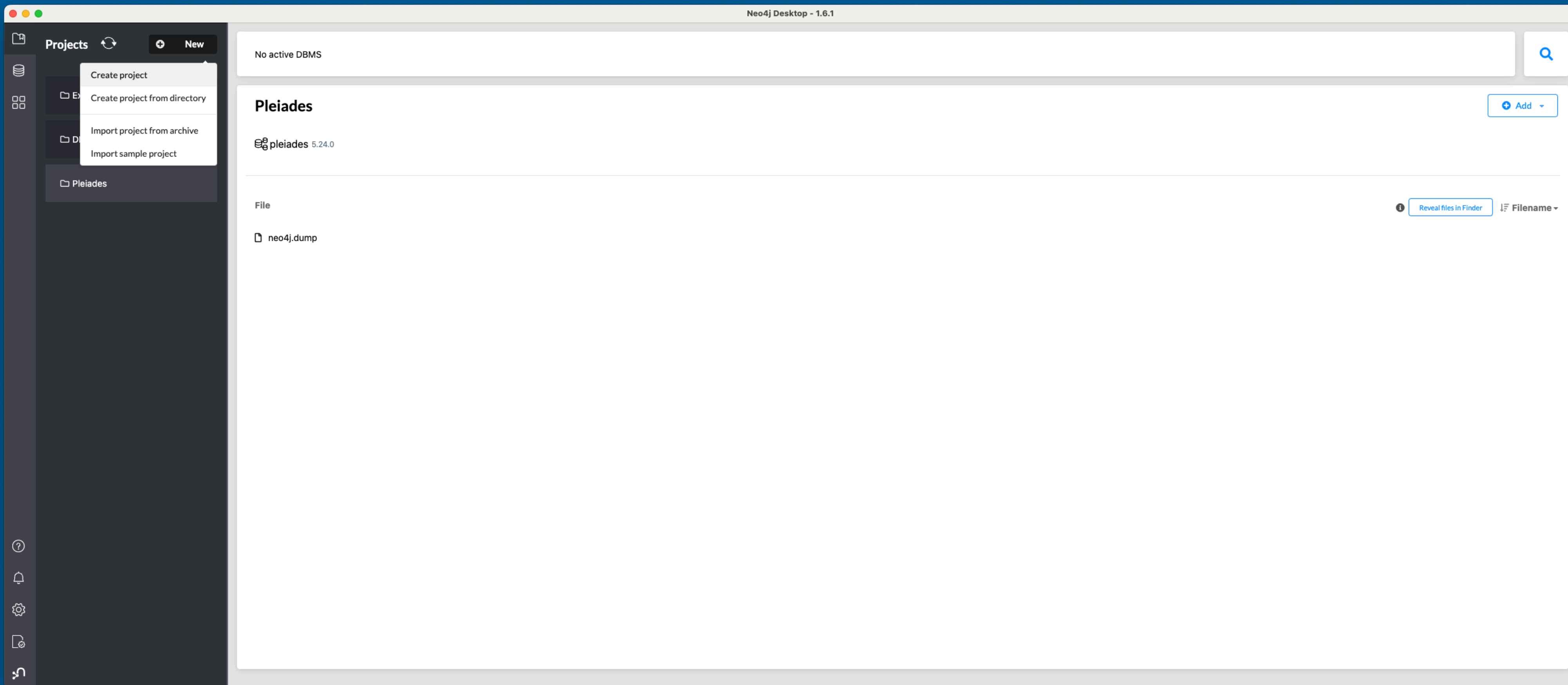
Basic Screen

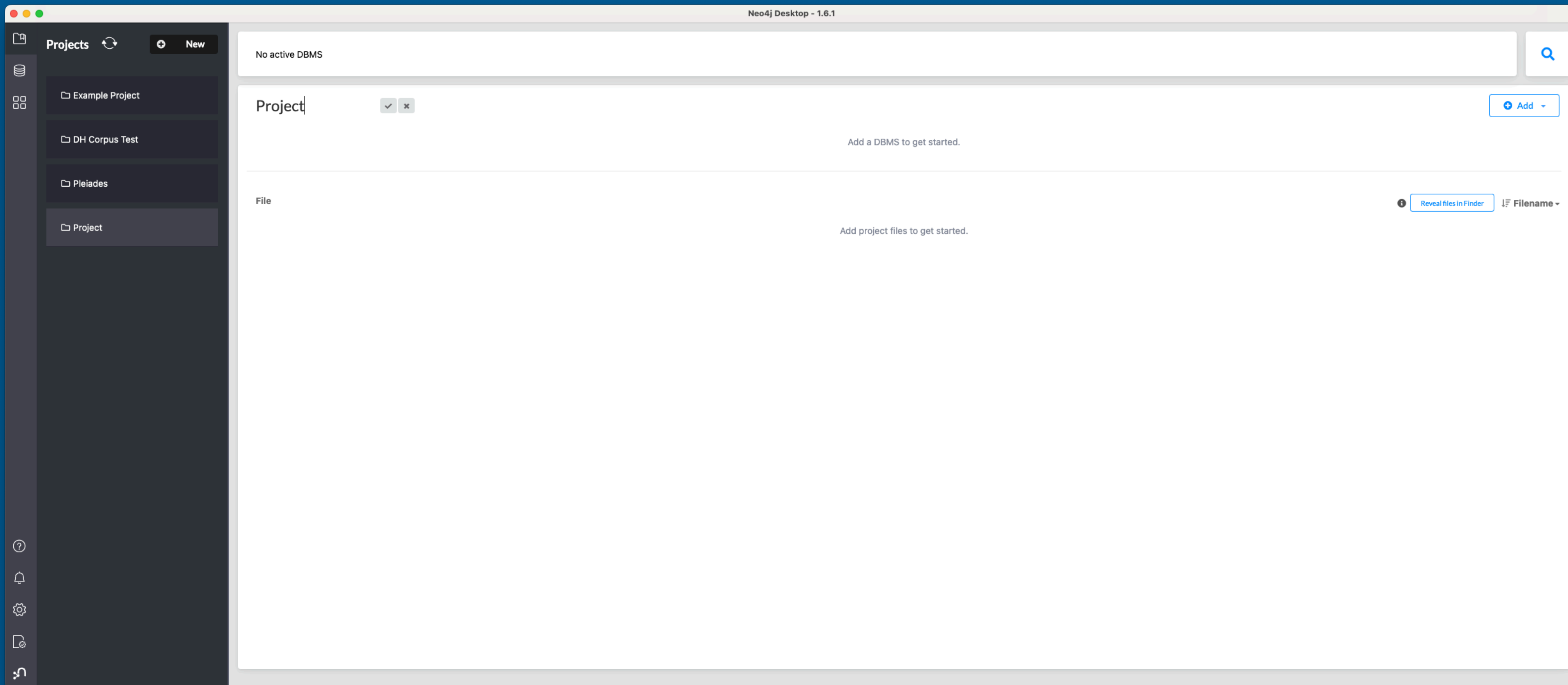


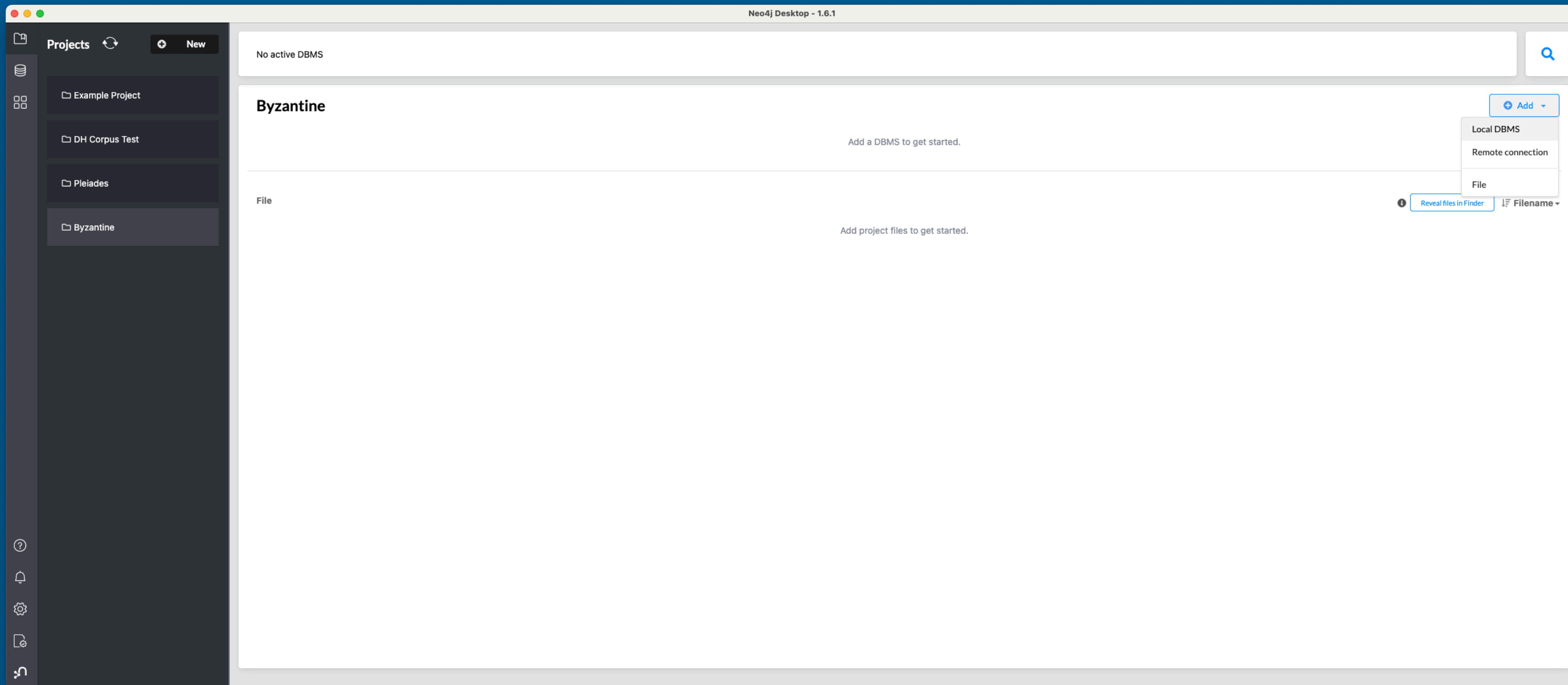
Create a New Project

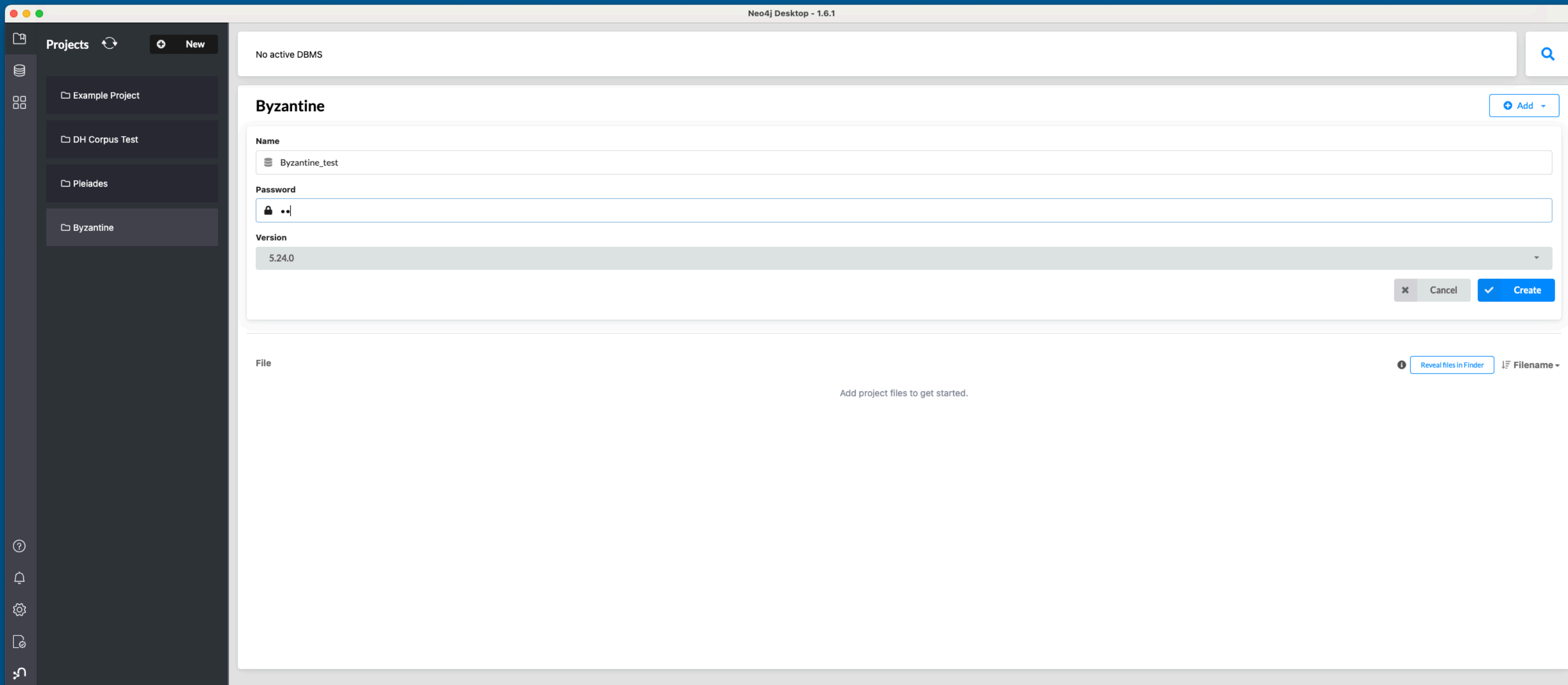


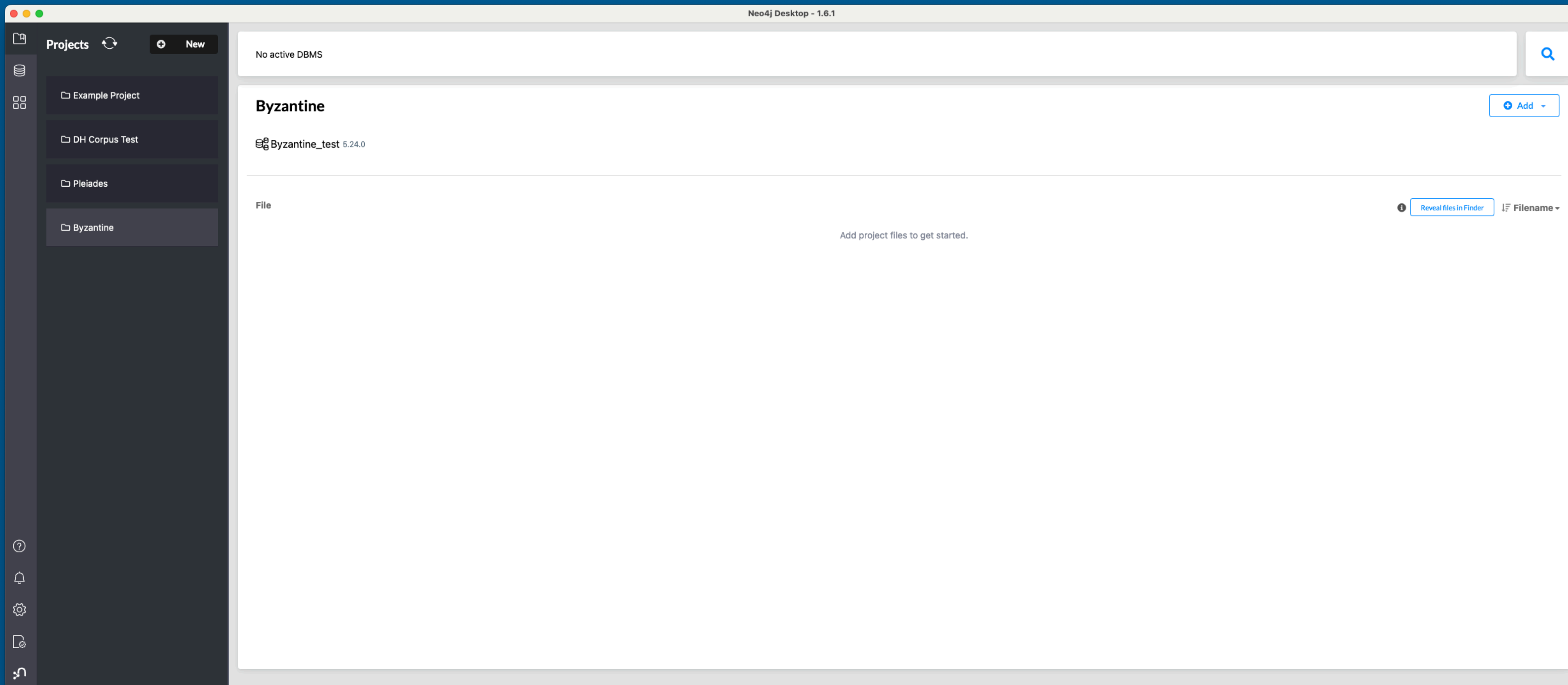
Create a New Project

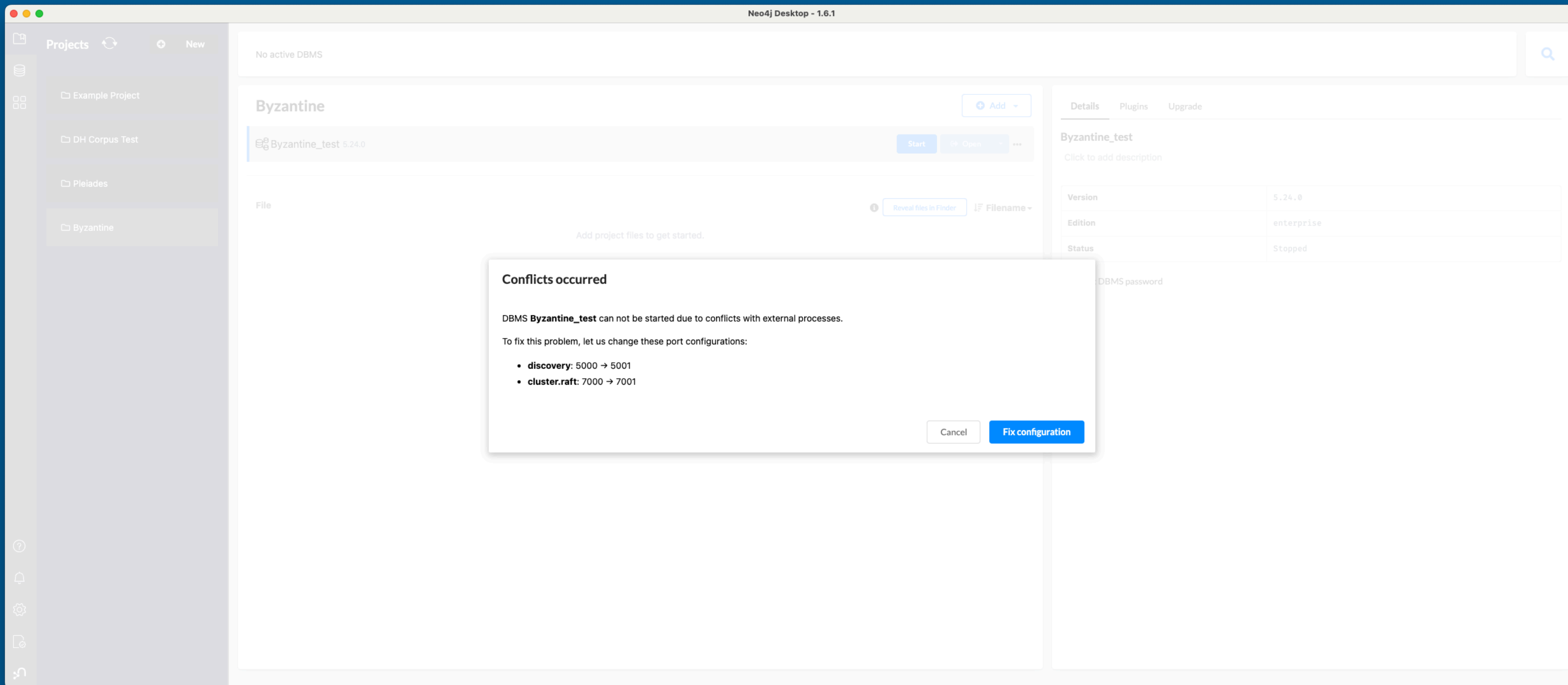


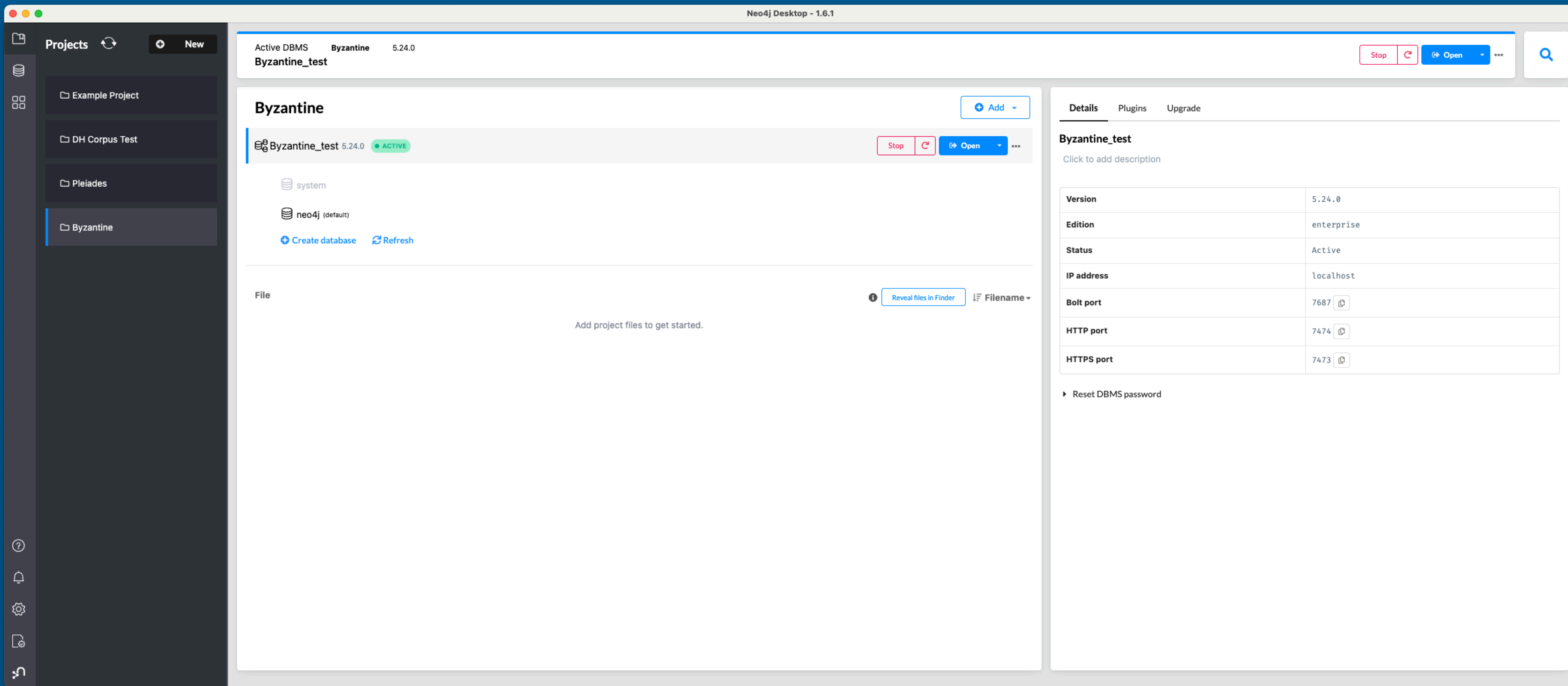

Neo4j Desktop - 1.6.1
Projects
New
Example Project
DH Corpus Test
Pleiades
Project
No active DBMS
Project
Add
Add a DBMS to get started.
File
Reveal files in Finder
Filename
Add project files to get started.

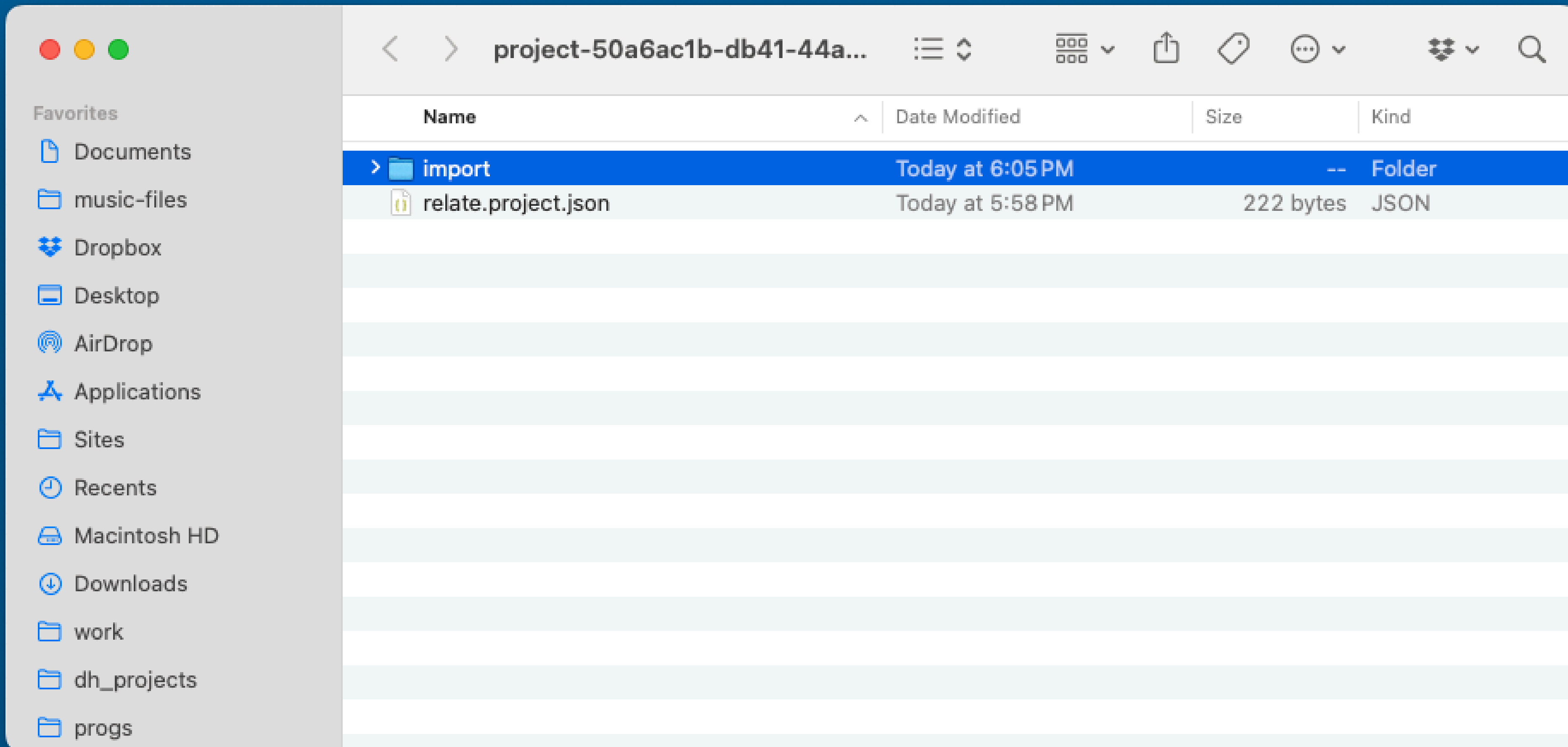

Neo4j Desktop - 1.6.1
Projects
New
Example Project
DH Corpus Test
Pleiades
Byzantine
No active DBMS
Byzantine
Add
Local DBMS
Remote connection
File
Add a DBMS to get started.
File
Reveal files in Finder
Filename
Add project files to get started.


Neo4j Desktop - 1.6.1
Projects
New
Example Project
DH Corpus Test
Pleiades
Byzantine
No active DBMS
Byzantine
Add
Name
Byzantine_test
Password
Version
5.24.0
Cancel
Create
File
Reveal files in Finder
Filename
Add project files to get started.

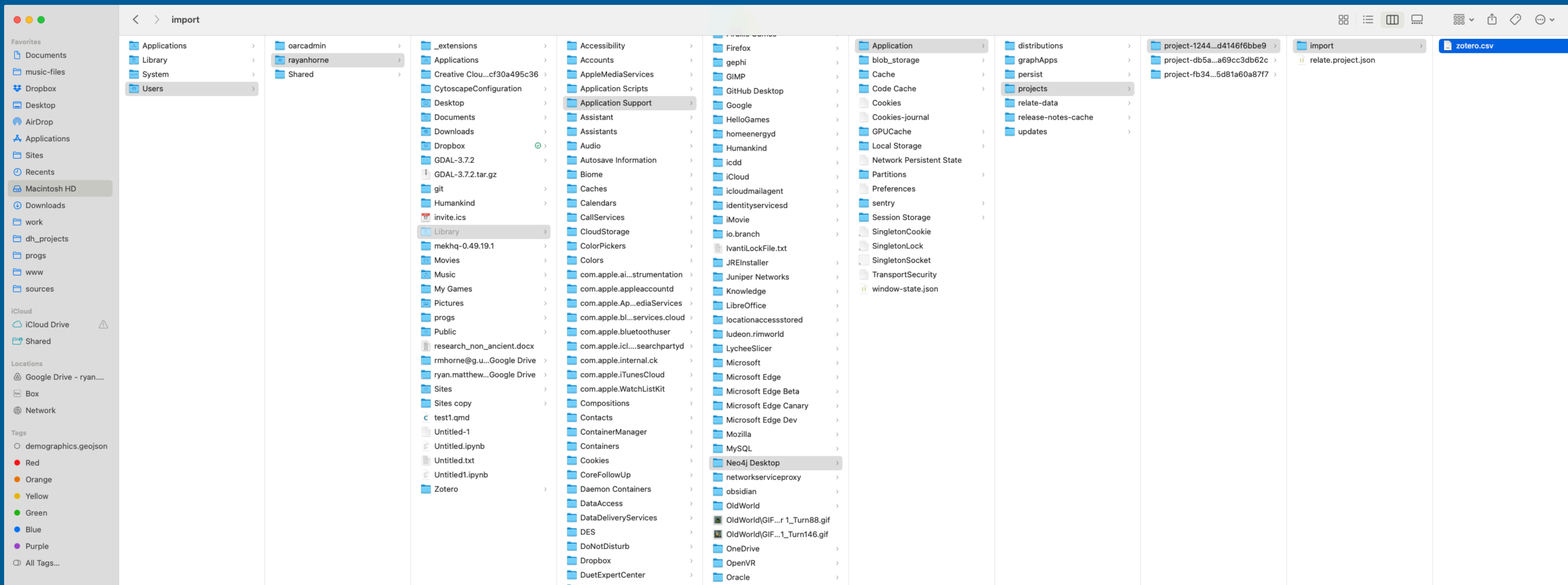

Neo4j Desktop - 1.6.1
Projects
New
Example Project
DH Corpus Test
Pleiades
Byzantine
No active DBMS
Byzantine
Add
Byzantine_test 5.24.0
File
Reveal files in Finder
Filename
Add project files to get started.


Neo4j Desktop - 1.6.1
Projects
New
Example Project
DH Corpus Test
Pleiades
Byzantine
No active DBMS
Byzantine
Add
Byzantine_test 5.24.0
Start
File
Reveal files in Finder
Filename
Add project files to get started.
Details
Plugins
Upgrade
Byzantine_test
Click to add description
Version
5.24.0
Edition
enterprise
Status
Stopped
DBMS password
Conflicts occurred
DBMS Byzantine_test can not be started due to conflicts with external processes.
To fix this problem, let us change these port configurations:
discovery: 5000 → 5001
cluster.raft: 7000 → 7001
Cancel
Fix configuration


Neo4j Desktop - 1.6.1
Projects
New
Example Project
DH Corpus Test
Pleiades
Byzantine
Active DBMS
Byzantine
5.24.0
Byzantine_test
Stop
Open
Byzantine
Add
Byzantine_test 5.24.0
ACTIVE
Stop
Open
system
neo4j (default)
Create database
Refresh
File
Reveal files in Finder
Filename
Add project files to get started.
Details
Plugins
Upgrade
Byzantine_test
Click to add description
Version
5.24.0
Edition
enterprise
Status
Active
IP address
localhost
Bolt port
7687
HTTP port
7474
HTTPS port
7473
Reset DBMS password


project-50a6ac1b-db41-44a...
Favorites
Documents
music-files
Dropbox
Desktop
AirDrop
Applications
Sites
Recents
Macintosh HD
Downloads
work
dh_projects
progs
Name
Date Modified
Size
Kind
import
Today at 6:05 PM
--
Folder
relate.project.json
Today at 5:58 PM
222 bytes
JSON

Step 2: Path



import
Favorites
Documents
music-files
Dropbox
Desktop
AirDrop
Applications
Sites
Recents
Macintosh HD
Downloads
work
dh_projects
progs
www
sources
iCloud
iCloud Drive
Shared
Locations
Google Drive - ryan....
Box
Network
Tags
demographics.geojson
Red
Orange
Yellow
Green
Blue
Purple
All Tags...
Applications
Library
System
Users
oarcadmin
rayanhorne
Shared
_extensions
Applications
Creative Clou...cf30a495c36
CytoscapeConfiguration
Desktop
Documents
Downloads
Dropbox
GDAL-3.7.2
GDAL-3.7.2.tar.gz
git
Humankind
invite.ics
Library
mekhq-0.49.19.1
Movies
Music
My Games
Pictures
progs
Public
research_non_ancient.docx
rmhorne@g.u...Google Drive
ryan.matthew...Google Drive
Sites
Sites copy
test1.qmd
Untitled-1
Untitled.ipynb
Untitled.txt
Untitled1.ipynb
Zotero
Accessibility
Accounts
AppleMediaServices
Application Scripts
Application Support
Assistant
Assistants
Audio
Autosave Information
Biome
Caches
Calendars
CallServices
CloudStorage
ColorPickers
Colors
com.apple.ai...strumentation
com.apple.appleaccountd
com.apple.Ap...ediaServices
com.apple.bl...services.cloud
com.apple.bluetoothuser
com.apple.icl....searchpartyd
com.apple.internal.ck
com.apple.iTunesCloud
com.apple.WatchListKit
Compositions
Contacts
ContainerManager
Containers
Cookies
CoreFollowUp
Daemon Containers
DataAccess
DataDeliveryServices
DES
DoNotDisturb
Dropbox
DuetExpertCenter
Firefox
gephi
GIMP
GitHub Desktop
Google
HelloGames
homeenergyd
Humankind
icdd
iCloud
icloudmailagent
identityservicesd
iMovie
io.branch
IvantiLockFile.txt
JREInstaller
Juniper Networks
Knowledge
LibreOffice
locationaccessstored
ludeon.rimworld
LycheeSlicer
Microsoft
Microsoft Edge
Microsoft Edge Beta
Microsoft Edge Canary
Microsoft Edge Dev
Mozilla
MySQL
Neo4j Desktop
networkserviceproxy
obsidian
OldWorld
OldWorld\GIF...r 1_Turn88.gif
OldWorld\GIF...1_Turn146.gif
OneDrive
OpenVR
Oracle
Application
blob_storage
Cache
Code Cache
Cookies
Cookies-journal
GPUCache
Local Storage
Network Persistent State
Partitions
Preferences
sentry
Session Storage
SingletonCookie
SingletonLock
SingletonSocket
TransportSecurity
window-state.json
distributions
graphApps
persist
projects
relate-data
release-notes-cache
updates
project-1244...d4146f6bbe9
project-db5a...a69cc3db62c
project-fb34...5d81a60a87f7
import
relate.project.json
zotero.csv

Importing CSV Files into Neo4j

Step 1: Prepare Your Data

Structure your CSV with headers like name, id, birth, relation_id

Example: people.csv

```
id,name,birth
1,Anna Komnene,1083
2,John II Komnenos,1087
```

Step 2: Place CSV in import Folder

For Neo4j Desktop: use the built-in import directory

- File path: neo4j/import/people.csv

Step 3: Load Nodes with Cypher

```
LOAD CSV WITH HEADERS FROM 'file:///people.csv' AS row
CREATE (:Person {id: row.id, name: row.name, birth: toInteger(row.b
```

- `LOAD CSV WITH HEADERS` reads the file
- `CREATE` makes new nodes with properties

Step 4: Load Relationships

Assume a file `wrote.csv`:

```
author_id,text_title
1,Alexiad
2,History
```

```

LOAD CSV WITH HEADERS FROM 'file:///wrote.csv' AS row
MATCH (p:Person {id: row.author_id})
CREATE (p)-[:WROTE]->(:Text {title: row.text_title})
```

Step 5: Check Results

```

MATCH (p:Person)-[:WROTE]->(t:Text)
RETURN p.name, t.title
```

Thank You!

Any Questions?