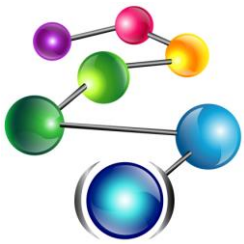
Graph & Modern Databases

COMP1835

1

1

AllegroGraph



https://allegrograph.com/

2

2

Objectives

- Introduce AllegroGraph database
- Review Resource Description Framework (RDF)
- Discuss triples
- Introduce AllegroGraph WebView and Gruff
- Discuss SPARQL commands

3

3

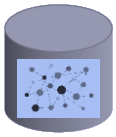
Part 1

https://allegrograph.com/

4

4

Types of Graph Databases



- As we discussed before, graph data stores could be divided into 2 subcategories, based on their data model:
  - Labeled Property Graph (or just Property Graph)
    - Neo4J is one of LPG databases
  - RDF – based
    - Native RDF databases first were known as triplestores, next they were called quad stores, then RDF stores, and most recently they call themselves “semantic graph databases”:
    - Examples: AllegroGraph, AWS Neptune, StarDog, Oracle Spatial…

5

5

AllegroGraph Overview

- AllegroGraph is **a database and application framework** for building Semantic Web applications
- It can
  - store data and meta-data as triples
  - query these triples through various query APIs like SPARQL and Prolog
  - apply RDFS++ reasoning with its built-in reasoner
- AllegroGraph includes support for Federation (grouping multiple stores within a single virtual store), Social Network Analysis, Geospatial capabilities and Temporal reasoning

6

6

AllegroGraph Facts

- Official Online Resources - https://allegrograph.com/
- History - Created at Franz Inc. in 2004. (Just one year after Neo4J)
- Technologies and Language - Java, Python, Common Lisp
- Access Methods
  - AllegroGraph WebView Browser
  - Gruff- visual navigation tool
  - Client interfaces for Java, Python, Ruby, Perl, C#, Clojure, and Common Lisp
- Query Language
  - Supports SPARQL, RDFS++, and Prolog
- Open-Source License – None, proprietary commercial software, (but it is free for 5 Million of triplets or less)
- Who Uses It
  - Bioinformatics, Healthcare, and Pharma
  - Defence and Intelligence (US Army and US Air Force, Australian Gov. Dept of Defence …)
  - Financial (Bank of America, CITI bank, JPMorgan …)
  - Manufacturing (Boeing, Ford, Siemens, Xerox …)
  - Telecoms (At&T, Cisco, KDDI…)

7

7

Benefits

AllegroGraph provides:

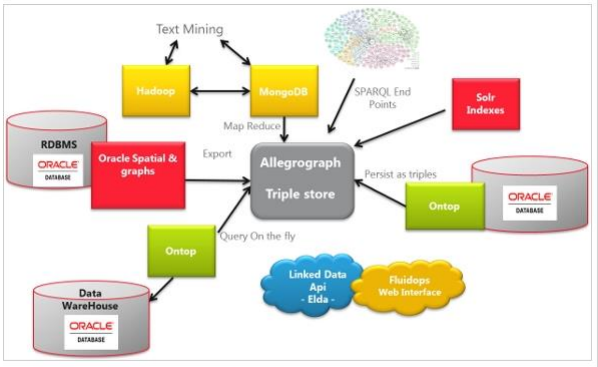
- All essential enterprise capabilities of a major relational database:
  - ACID transactions,
  - backup/restore,
  - point in time recovery,
  - security,
  - replication,
  - warm fail over,
  - clustering,
  - triple level security.
- Geospatial reasoning, temporal reasoning, and social network analysis.
  - These features are all directly accessible in SPARQL
- Business rules with an ISO compatible Prolog compiler
- Server side JavaScript stored procedures
- Gruff - a powerful visualization tool which allows user friendly navigation of triples.
  - Gruff's graphical query editor allows easy composition of SPARQL queries.
  - The ability to automatically discover patterns by highlighting nodes and turning them into SPARQL queries

8

8

Life Example

- Trivadis AG, a pharmaceutical company Switzerland



Overview of federated data implementation.

https://www.americanlaboratory.com/913-Technical-Articles/167956-Semantic-Web-Technologies-as-a-Key-to-Successful-Federation-of-Data-in-Life-Science-Organizations/

9

9

Resource Description Framework (RDF)

- RDF is a data model in which the basic unit of information is known as triple
- A triple consists
  - of **subject**, a **predicate** and an **object**.
  - Or, of resource identifier, an attribute (or property name) and property value
- Each triple has unique identifier, the Uniform Resource Identifier (URI)
- Example:

Subject	Predicate	Object
:John	:hasAge	:28
:Anita	:hasPet	:Sunny

10

10

A Triple

- A triple consists
  - of **subject**, a **predicate** and an **object**.
- Example:

http://gre.ac.uk/students#student | http://hea.org/Level7#hasCompleted | http://gre.ac.uk/CMS#module1707

subject	predicate	object
resource identifier	attribute name	attribute value
URI: http://gre.ac.uk/students/	URI: http://hea.org/Level7/	Could be URI: http://gre.ac.uk/CMS/

11

11

URI and Prefixes

- In RDF the names of the subject and predicate parts (and sometimes objects) must belong to specific namespace so that no person or process confuses those names with similar ones especially if that data gets combined with other data
- Like in XML, it is good practice to define a prefix to represent a namespace URI so that your data doesn't get too verbose
- The **PREFIX** keyword describes prefix declarations for abbreviating URIs
- Example:

```
prefix st:<http://gre.ac.uk/students#>
prefix lv:<http://hea.org/Level7#>
prefix md:<http://gre.ac.uk/CMS#>
```

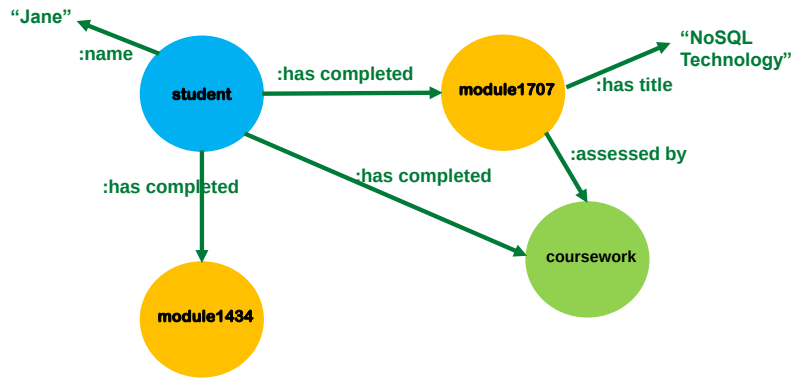
- When you use the abbreviation (st:student), it appends the string after the colon (:) to the URI that is referenced by the prefix string

st:student | lv:hasCompleted | md:module1707

12

12

Graph of Triples



13

13

RDFS and OWL

- RDFS is a schema for RDF
- RDFS defines
  - Classes and properties
  - Hierarchies of classes and properties
- RDFS specification identifies
  - Resource, Literal, Class, Datatype, Domain, range and relationships, for example, subPropertyOf, subClassOf
- An ontology is a formal description of knowledge as a set of concepts within a domain and the relationships that hold between them
  - Unlike relational database schemas, ontologies express relationships and enable users to link multiple concepts to other concepts in a variety of ways
- **OWL** (or **W**eb **O**ntology **L**anguage) is the ontology (think "schema") language of the Semantic Web
- It is one of the core Semantic Web standards, along with RDF and SPARQL

14

14

Quiz

- **Question 1:** AllegroGraph is a
  - **A.** Label Property Graph database
  - **B.** RDF Graph database
  - **C.** Document Graph database
- **Question 2:** A triple consist of :
  - **A.** Substance, premise, entity
  - **B.** Subject, predicate, object
  - **C.** Object, predicate, property
  - **D.** Subject, property, object
- **Question 3:** RDF stands for:
  - **A.** Reports Description Framework
  - **B.** Resource Definition Framework
  - **C.** Resource Description Framework

15

15

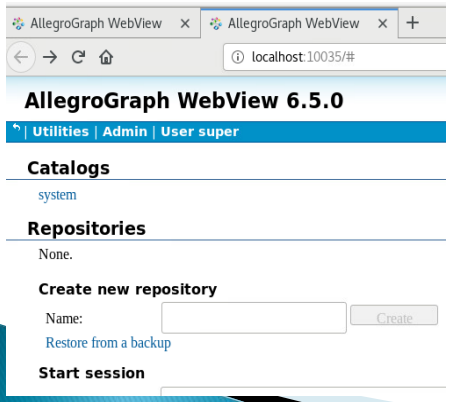
Part 2

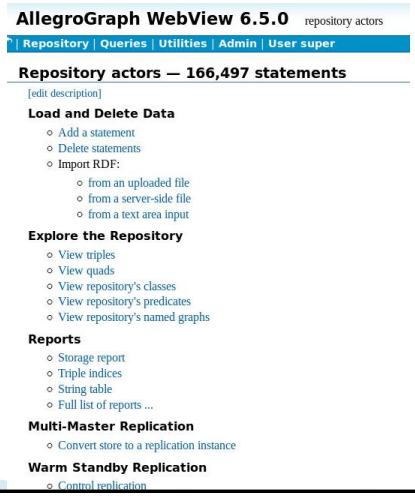
16

16

AllegroGraph WebView

- AllegroGraph WebView (AGWebView) is a graphical user interface for exploring, querying, and managing AllegroGraph repositories



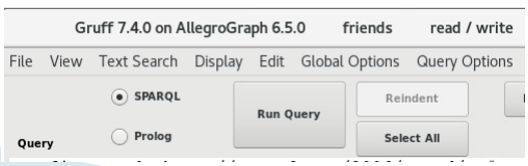


17

17

AllegroGraph Gruff

- AllegroGraph Gruff is powerful visual navigation tool
- It allows you to
  - create a new triple store,
  - download triple files to populate the store,
  - query triples or display triples on the screen.
- Gruff comes in two forms:
  - a standalone version that includes a basic version of AllegroGraph,
  - and the server edition.
- You will need the server edition if you are working with hundreds of millions to billions of triples
- We are going to use the standalone version of Gruff



18

18

Turtle

- Terse RDF Triple Language (Turtle)
  - is a syntax and file format for expressing data in the RDF data model
  - It is a common data format for storing RDF data, along with N-Triples, JSON-LD and RDF/XML
  - Example:

```
@prefix vcard:<http://www.w3.org/2006/vcard/ns#> .
@prefix st:<http://gre.ac.uk/student#> .

st:student1 vcard:given-name "Anita" .
st:student1 vcard:family-Name "Patel" .
st:student1 st:level "PG" .
st:student1 st:startDate "2018-09-01" .
st:student1 st:completeDate "2019-09-30" .

st:student2 vcard:given-name "Rajesh" .
st:student2 vcard:family-Name "Patel" .
st:student2 st:level "PG" .
st:student2 st:startDate "2018-01-01" .
st:student2 st:completeDate "2020-01-31" .

st:student3 vcard:given-name "Francis" .
st:student3 vcard:family-Name "Smith" .
st:student3 st:level "UG" .
st:student3 st:startDate "2018-09-01" .

st:student4 vcard:given-name "Jane" .
st:student4 vcard:family-Name "Ford" .
st:student4 st:level "UG" .
st:student4 st:startDate "2018-09-01" .
```

student Num	given-name	family-name	level	startDate	completeDate
student1	Anita	Patel	PG	2018-09-01	2019-09-30
student2	Rajesh	Patel	PG	2018-01-01	2020-01-31
student3	Francis	Smith	UG	2018-09-01	
student4	Jane	Ford	UG	2018-09-01	

Filename extension:
.ttl

https://www.vectorstock.com/royalty-free-vector/cute-turtle-cartoon-vector-16947950

SPARQL

- **S**PARQL
- **P**rotocol
- **A**nd
- **R**DF*
- **Q**uery
- **L**anguage
  - is a query language for RDF, (pronounced "sparkle")
  - can be used to express queries across diverse data sources, whether the data is stored natively as RDF or viewed as RDF via middleware.
  - also supports aggregation, subqueries, negation, creating values by expressions, extensible value testing, and constraining queries by source RDF graph.
- The results of SPARQL queries can be result sets or RDF graphs

Adding triples

- AllegroGraph can load data in the following RDF formats:
- JSON-LD
- N-Quads
- N-Triples
- Extended N-Quads
- RDF/XML
- TriG
- TriX
- Turtle
- as well as in several non-RDF formats, like JSON, JSONlines, and CSV.

21

21

INSERT data

- In addition to various mechanisms for loading data, you can use SPARQL's **INSERT DATA** command (usually for small amount of test data)
  - SPARQL commands are not case-sensitive but usually the best practice is to use uppercase to make it more readable
- Example:

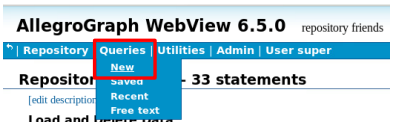
```
prefix vcard:<http://www.w3.org/2006/vcard/ns#>
prefix st:http://gre.ac.uk/student/
INSERT DATA {
st:student1 vcard:given-name "Anita" .
st:student1 vcard:family-name "Patel" .
st:student1 st:level "PG" .
st:student1 st:startDate "2018-01-01" .
st:student1 st:completeDate "2019-09-30" .
}
```

22

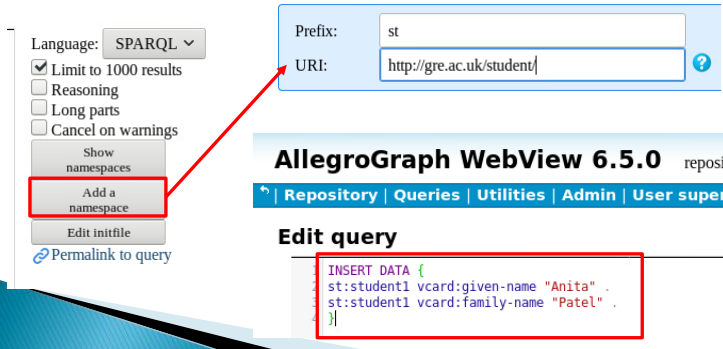
22

INSERT DATA in AGWebView

- You can execute SPARQL commands using AllegroGraph WebView interface



  - It is useful to define prefixes and namespaces first

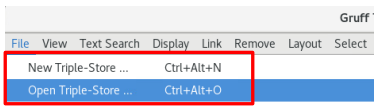


23

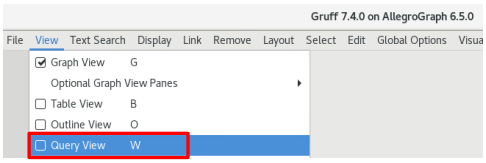
23

INSERT DATA using Gruff

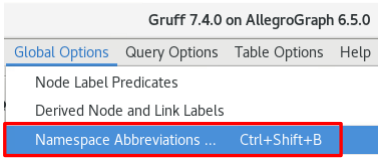
- First, open Triple-Store or create a new one



- Then, open Query view



- Define prefixes and namespaces



- Run **INSERT DATA** command

24

24

SPARQL query

- A SPARQL query typically says:
  - "I want these pieces of information from the subset of the data that meets these conditions"
- You describe the conditions with triple patterns, which are similar to RDF triples but may include variables to add flexibility
  - Every triple pattern is denoted by curly brackets
  - Variable are denoted by a **?** or **$** before a string
  - Every time a triple pattern matches against a triple in a store, it produces a *binding* for each variable
  - For example, pattern `st:student1 vcard:given-name ?y .`
  - produces one binding for **?y**, value **Anita**.
  - SPARQL query includes **SELECT** clause with **WHERE** clause followed by the triple pattern in curly brackets

```
SELECT nodes, variables
WHERE {
triple pattern
}
```

Example:

```
SELECT ?s ?p ?o
WHERE {
?s ?p ?o
}
LIMIT 10
```

```
SELECT *
FROM
database
```

25

25

Query – Example

- Example: get all students with last name "Patel"

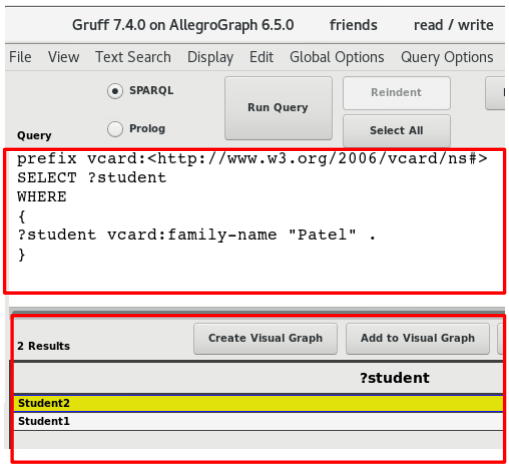
```
@prefix vcard:<http://www.w3.org/2006/vcard/ns#> .
@prefix st:<http://gre.ac.uk/student/> .

st:student1 vcard:given-name "Anita" .
st:student1 vcard:family-name "Patel" .
st:student1 st:level "PG" .
st:student1 st:startDate "2018-09-01" .
st:student1 st:completeDate "2019-09-30" .

st:student2 vcard:given-name "Rajesh" .
st:student2 vcard:family-name "Patel" .
st:student2 st:level "PG" .
st:student2 st:startDate "2018-01-01" .
st:student2 st:completeDate "2020-01-31" .

st:student3 vcard:given-name "Francis" .
st:student3 vcard:family-name "Smith" .
st:student3 st:level "UG" .
st:student3 st:startDate "2018-09-01" .

st:student4 vcard:given-name "Jane" .
st:student4 vcard:family-name "Ford" .
st:student4 st:level "UG" .
st:student4 st:startDate "2018-09-01" .
```

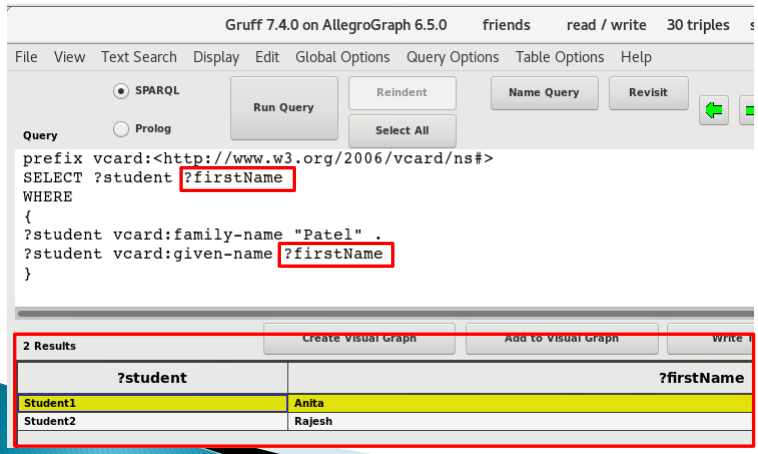


26

26

Returning More Attributes

- In order to return more information, you need to define variables:

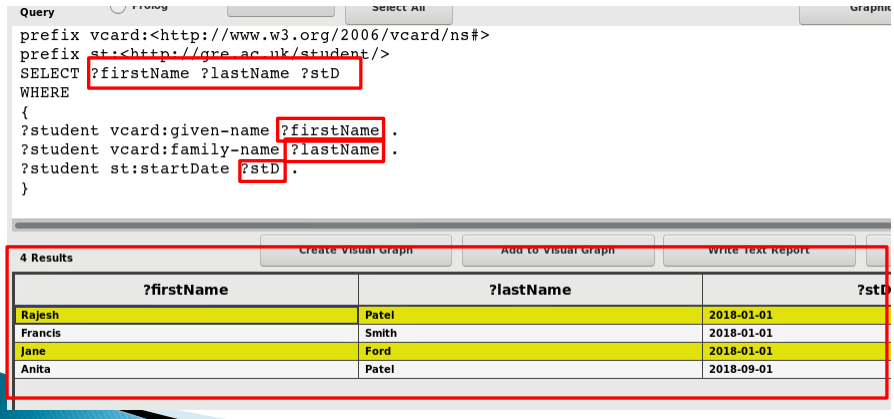


27

27

More details

- You want to see first names and last names and start date for all students:



28

28

Quiz

- **Question 1:** What is RDF Turtle?
  A. Large marine reptile
  B. Common data format for storing RDF data
  C. Common data format for storing XML data

- **Question 2:** Which command you need to use to insert triples into a triple-store?
  A. `INSERT`
  B. `POPULATE`
  C. `INSERT DATA`
  D. `LOAD`

- **Question 3:** Which of the following **are** correct SPARQL queries?
  A. `SELECT ?x WHERE { ?x ?y ?z .}`
  B. `SELECT ?x WHERE ?x = 1`
  C. `SELECT ?fn WHERE { ?fn vcard:given-name .}`
  D. `SELECT ?fn  WHERE { ?student vcard:given-name ?fn . }`

29

29

Part 3

30

30

Absence of attributes

- Now, you want to see first names, last names and complete date for all students:

```
prefix vcard:<http://www.w3.org/2006/vcard/ns#>
prefix st:<http://gre.ac.uk/student/>
SELECT ?firstName ?lastName ?cD
WHERE
{
?student vcard:given-name ?firstName .
?student vcard:family-name ?lastName .
?student st:completeDate ?cD .
}
```

2 Results

?firstName	?lastName	?cD
Rajesh	Patel	2020-01-31
Anita	Patel	2019-09-30

- But, the result has only 2 students?

31

31

OPTIONAL clause

- Triple pattern specified inside curly braces must match exactly in order to be returned in the output
- So, in order for other two students who do not have complete date to be included, use **OPTIONAL** clause:

```
prefix vcard:<http://www.w3.org/2006/vcard/ns#>
prefix st:<http://gre.ac.uk/student/>
SELECT ?firstName ?lastName ?cD
WHERE
{
?student vcard:given-name ?firstName .
?student vcard:family-name ?lastName .
OPTIONAL {?student st:completeDate ?cD .}
}
```

4 Results

?firstName	?lastName	?cD
Rajesh	Patel	2020-01-31
Anita	Patel	2019-09-30
Francis	Smith	
Jane	Ford	

32

32

ORDER BY

- To impose a sorted order on a results set, add **ORDER BY** to a **SELECT** query
  - The results of a query can be ordered by any combination of variables in the results, in ascending (**ASC**, default) or descending order (**DESC**)

```
SELECT ?firstName ?lastName ?cD
WHERE
{
  ?student vcard:given-name ?firstName .
  ?student vcard:family-name ?lastName .
  OPTIONAL {?student st:completeDate ?cD .}
}
ORDER BY DESC(?firstName)
```

4 Results

?firstName	?lastName	?cD
Rajesh	Patel	2020-01-31
Jane	Ford	
Francis	Smith	
Anita	Patel	2019-09-30

33

33

FILTER

- To add additional conditions to the query use **FILTER** key word followed by an expression
  - The expression can be as complex as you want, as long as it returns a Boolean value

```
prefix vcard:<http://www.w3.org/2006/vcard/ns#>
prefix st:<http://gre.ac.uk/student/>
SELECT ?firstName ?lastName ?stD
WHERE
{
?student vcard:given-name ?firstName .
?student vcard:family-name ?lastName .
?student st:startDate ?stD .
FILTER(?stD<"2018-02-02")
}
```

3 Results

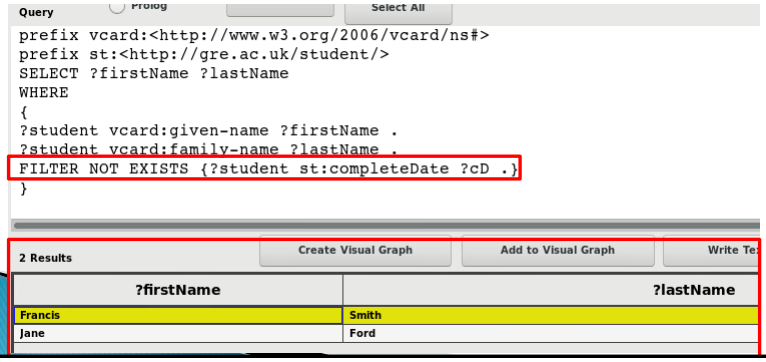
?firstName	?lastName	?stD
Rajesh	Patel	2018-01-01
Francis	Smith	2018-01-01
Jane	Ford	2018-01-01

34

34

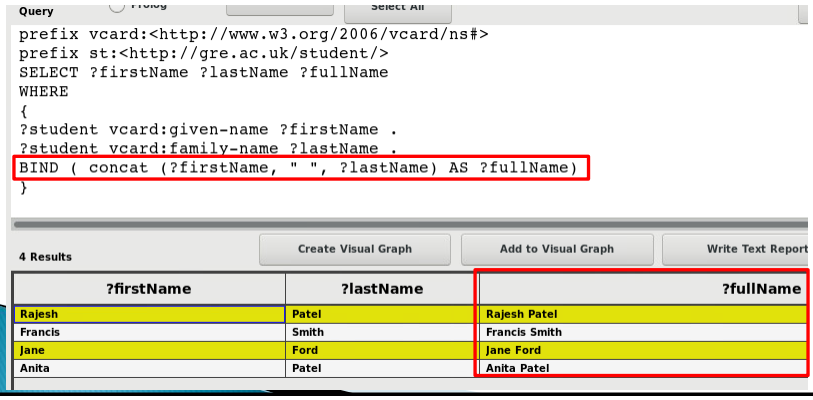
FILTER NOT EXISTS

- **FILTER NOT EXISTS** is a **FILTER** condition that returns a Boolean value *true* if the specified graph pattern doesn't exist
- Example: return all students who do not have complete date yet



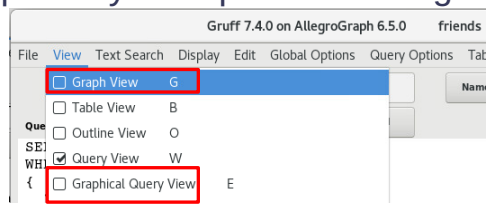
BIND clause and CONCAT function

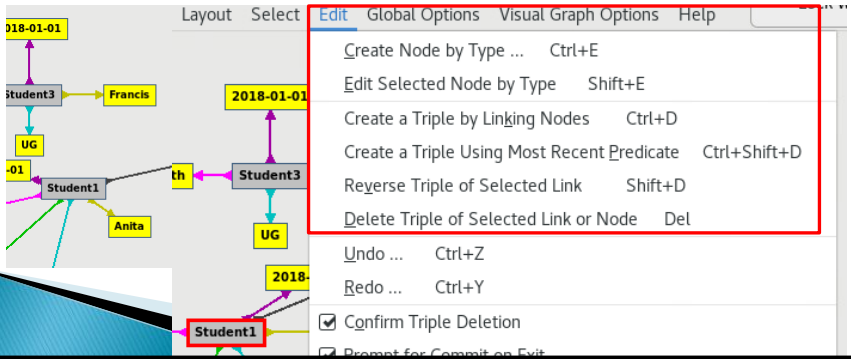
- If you need to return additional expressions or calculations, use BIND keyword and assign a new variable to it
- You can use any functions, for example **concat**



Using Graph View in Gruff

- You can manipulate your triple store using Graph View in Gruff





37

Quiz

Consider the following data ->

```
@prefix vcard:<http://gre.ac.uk/ns#> .
@prefix st:<http://gre.ac.uk/student/> .

st:student1 vcard:given-name "Anita" .
st:student1 vcard:family-name “Patel” .
st:student1 st:level “PG” .
st:student1 st:startDate “2018-09-01” .
st:student1 st:completeDate “2019-09-30” .

st:student2 vcard:given-name "Rajesh" .
st:student2 vcard:family-name “Patel” .
st:student2 st:startDate “2018-01-01” .
st:student2 st:completeDate “2020-01-31” .

st:student3 vcard:given-name "Francis" .
st:student3 vcard:family-name “Smith” .
st:student3 st:level “UG” .
st:student3 st:startDate “2018-09-01” .

st:student4 vcard:given-name "Jane" .
st:student4 vcard:family-name “Ford” .
st:student4 st:startDate “2018-09-01” .
```

Question. Which of the following SPARQL queries will show family names and levels for all students?

A.
```
SELECT ?fn ?lev
  WHERE { ?student vcard:family-name ?fn .
  ?student st:level ?lev .}
```

B.
```
SELECT ?fn ?lev
  WHERE { ?student vcard:family-name ?fn .
  OPTIONAL {?student st:level ?lev .} }
```

C.
```
SELECT ?fn
  WHERE { ?student vcard:family-name ?fn .
  OPTIONAL {?student st:level ?lev .} }
```

38

38

Further Reading

- The Semantic Web Made Easy:
https://www.w3.org/RDF/Metalog/docs/sw-easy
- Semantic Reasoning: The (Almost) Forgotten Half of AI:
https://aibusiness.com/semantic-reasoning-ai/
- What are Ontologies?
https://www.ontotext.com/knowledgehub/fundamentals/what-are-ontologies/

39

39

Essentials

- Introduced AllegroGraph database
- Reviewed Resource Description Framework (RDF)
- Discussed triples
- Introduced AllegroGraph WebView and Gruff
- Discussed SPARQL commands

40

40