Graph Data Modelling Exercise

Learning Management System

Graph Modelling – An Outline

1. Understand the domain.
2. Create high-level sample data.
3. Define specific questions for the application.
4. Identify entities.
5. Identify connections between entities.
6. Test the questions against the model.
7. Test scalability.

Graph Modelling – The domain

- There are many courses in the LMS, each of which contains a number of lessons that must be completed in a specific order.
- Every course grants a certificate upon completion.
- This certificate has a term of validity. When it expires, students must take the course again.
- Students can enroll in as many simultaneous courses as they want to.
- When a student logs in and chooses a course, the LMS must send them to their latest unfinished lesson.

Graph Modelling – Sample Data

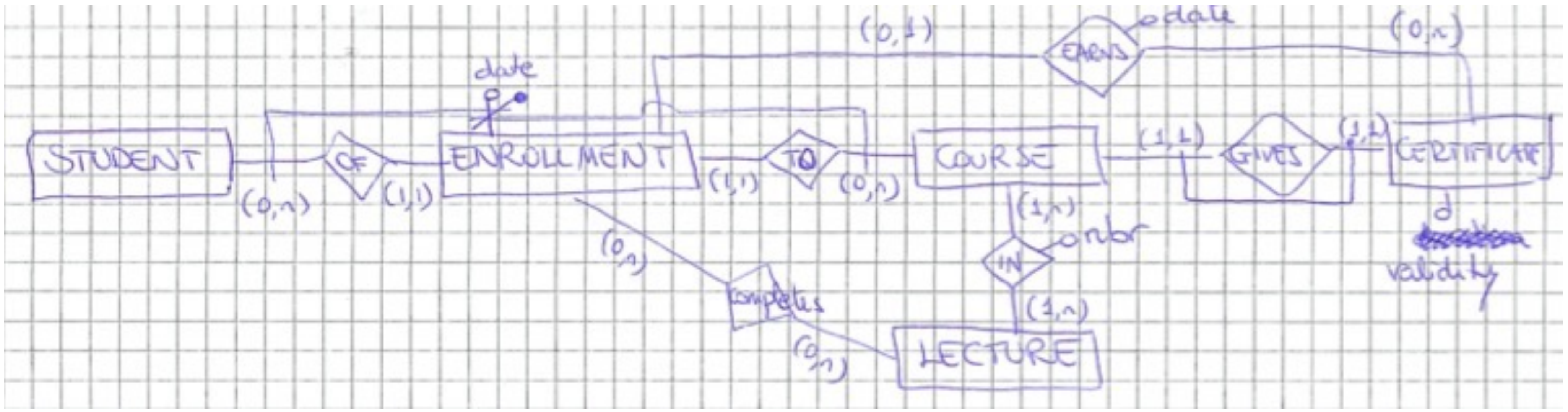
Courses	Lessons	Certificate
Introduction to Neo4j	Graph Theory, Graph Databases, Basic Cypher	2-year validity
Neo4j for Developers	Graph Theory, Property Graph, Graph Databases	6-month validity

Students	Completed Courses	In-Progress Courses
Alice	Introduction to Neo4j (2016), Introduction to Neo4j (2018)	Introduction to Neo4j (lesson 1)
Dan		Introduction to Neo4j (lesson 3), Neo4j for Developers (lesson 2)

Graph Modelling – Application Questions

1. Which lesson(s) is Dan currently working on?
2. What are Alice's current certifications?
3. Which lessons are in the Neo4j for Developers course?
4. What is the last lesson in the Introduction to Neo4j course?
5. Which lesson follows Graph Theory in the Neo4j for Developers course?
6. Who has completed Introduction to Neo4j?

An ER Conceptual Model for the Domain



Graph Modelling – List entities = labels for homogeneous sets of nodes

Graph Modelling – List entities = labels for momogeneous sets of nodes

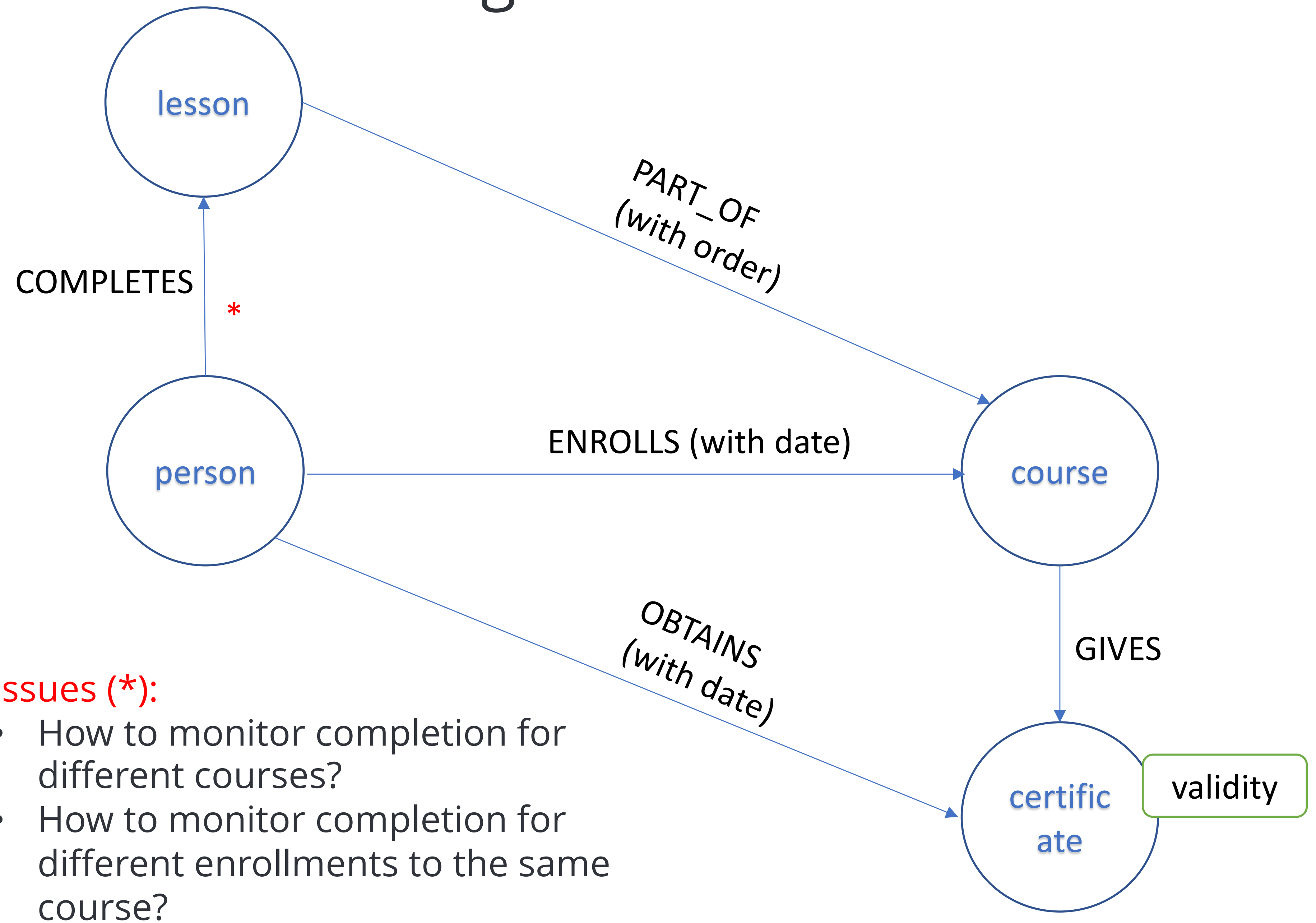
1. www.wooclap.com/GRAPHMODEL

STUDENT
COURSE
LESSON
CERTIFICATE

TIME? -> just properties with domain date could be enough?

Latest/unfinished -> via properties/relationships

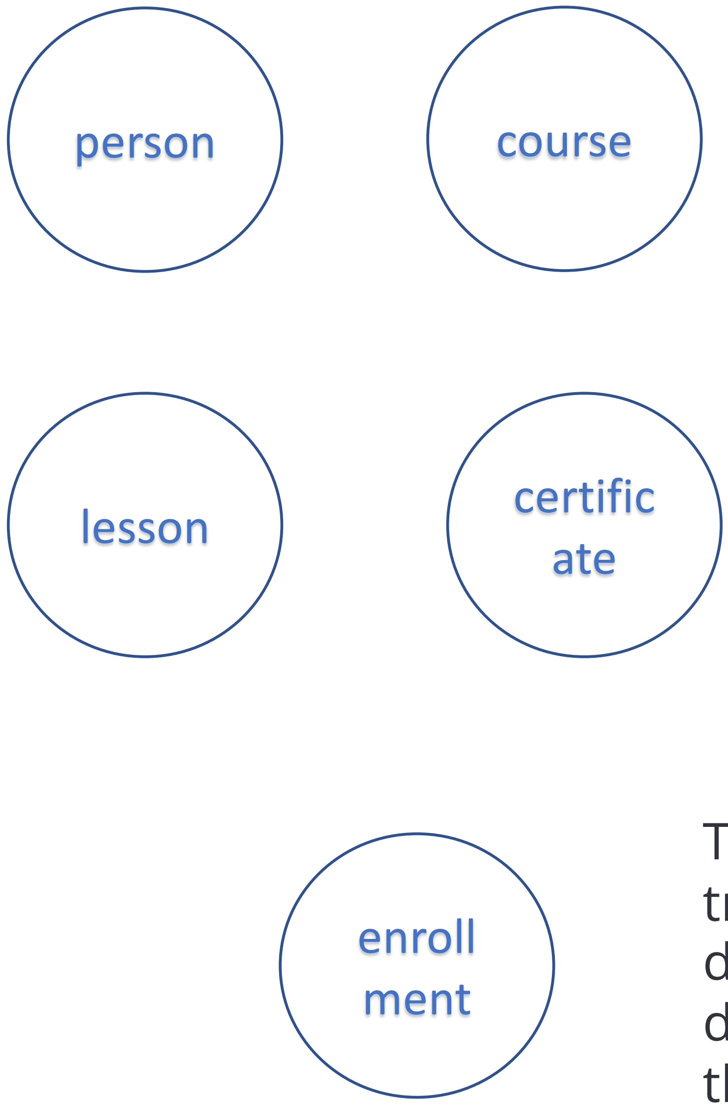
Graph Modelling – List entities = labels for homogeneous sets of nodes



Issues (*):

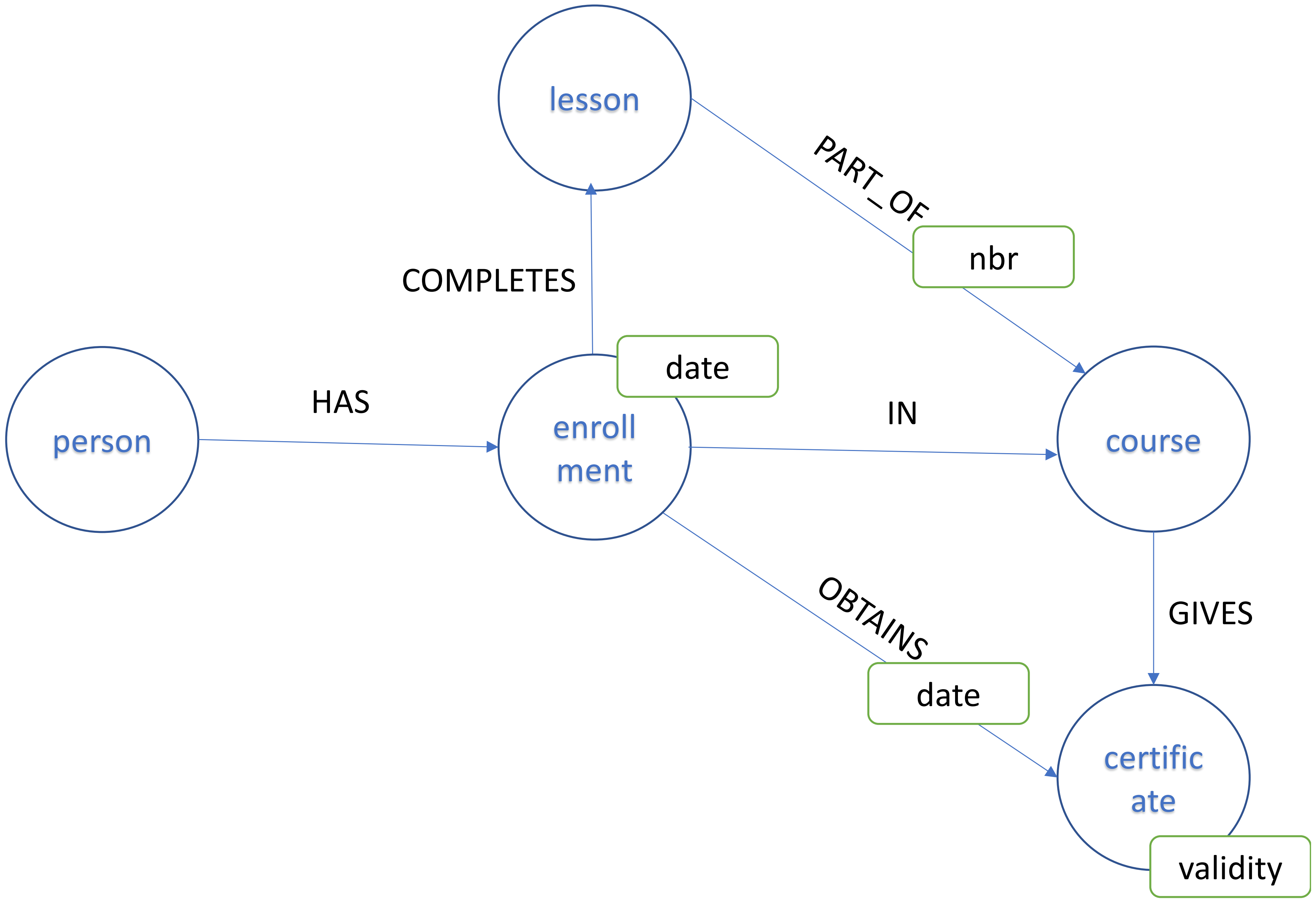
- How to monitor completion for different courses?
- How to monitor completion for different enrollments to the same course?

Graph Modelling – List entities = labels for homogeneous sets of nodes

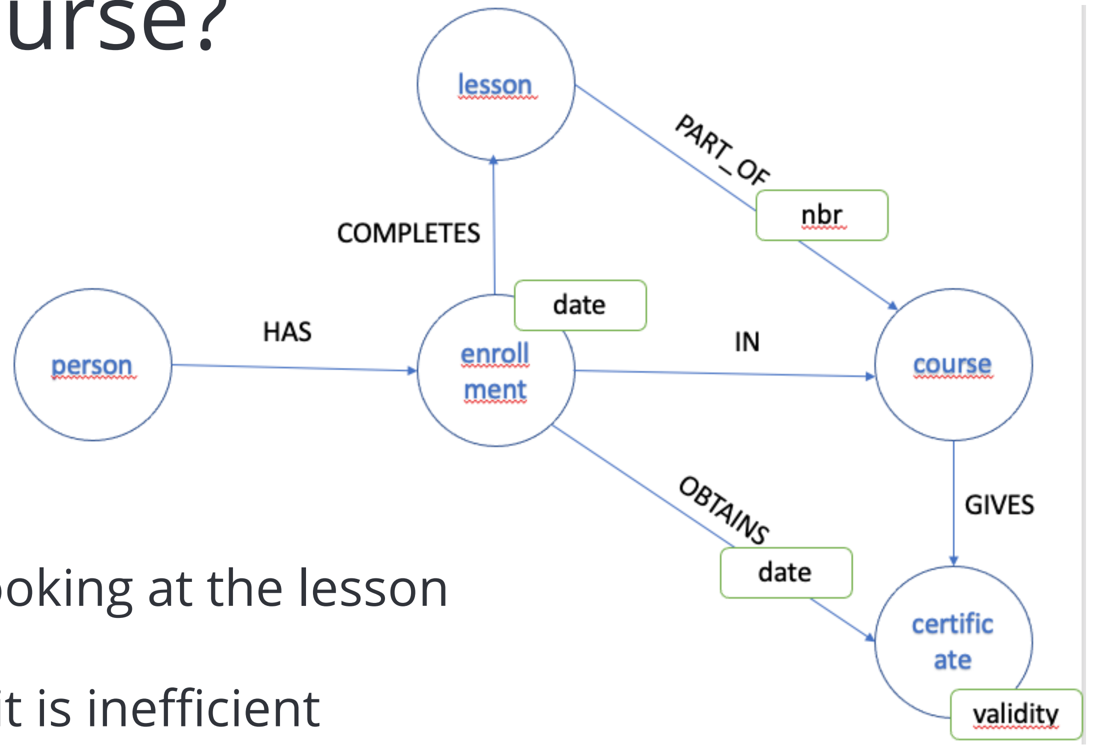


This intermediate node allows us to keep track of students' progress through different concurrent courses, and to differentiate multiple subsequent passes through the same course.

Graph Modelling – List entities = labels for homogeneous sets of nodes



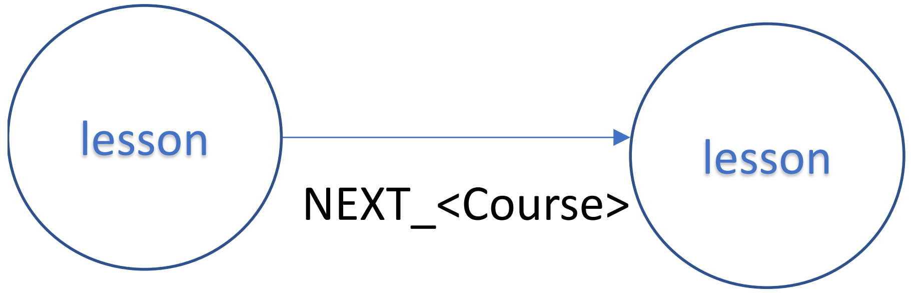
How to manage the order of lessons in a course?



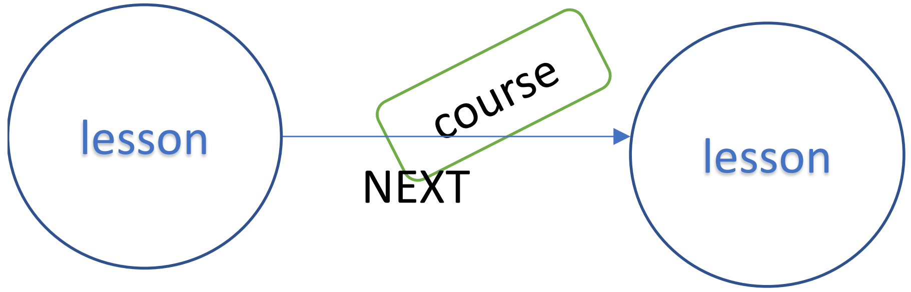
- The order can be reconstructed by looking at the lesson numbering
- But it is a property of a relationship, it is inefficient
  - E.g., given a lesson L PART_OF a course C with number n, to find the next lesson (frequent request) we need to look in all the relationships with type PART_OF, those for course C with the minimum number greater than n and find the corresponding lesson L_{next}

- To make this operation more efficient we can explicit model the next relationship between lessons inside a course
  - This can in turn be obtained through two different modelling

How to manage the order of lessons in a course?

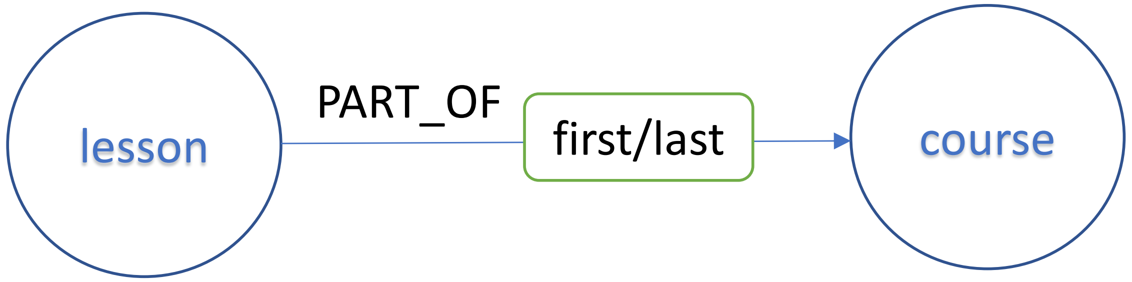


- Better for traversing from one lesson to the next one inside a given course
- A traversal to find the lessons that follows one lesson in any course is less efficient (and more difficult to express)
  - Still reasonable since a lesson likely is not part of too many courses
  - Query rewriting needed if we add a new course …

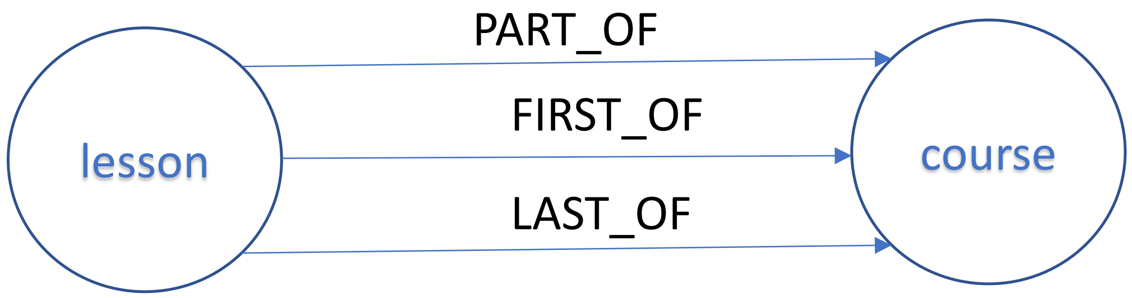


- Better for traversing from one lesson to the ones that follows it in any course
- Worse for traversing from one lesson to the next one inside a given course

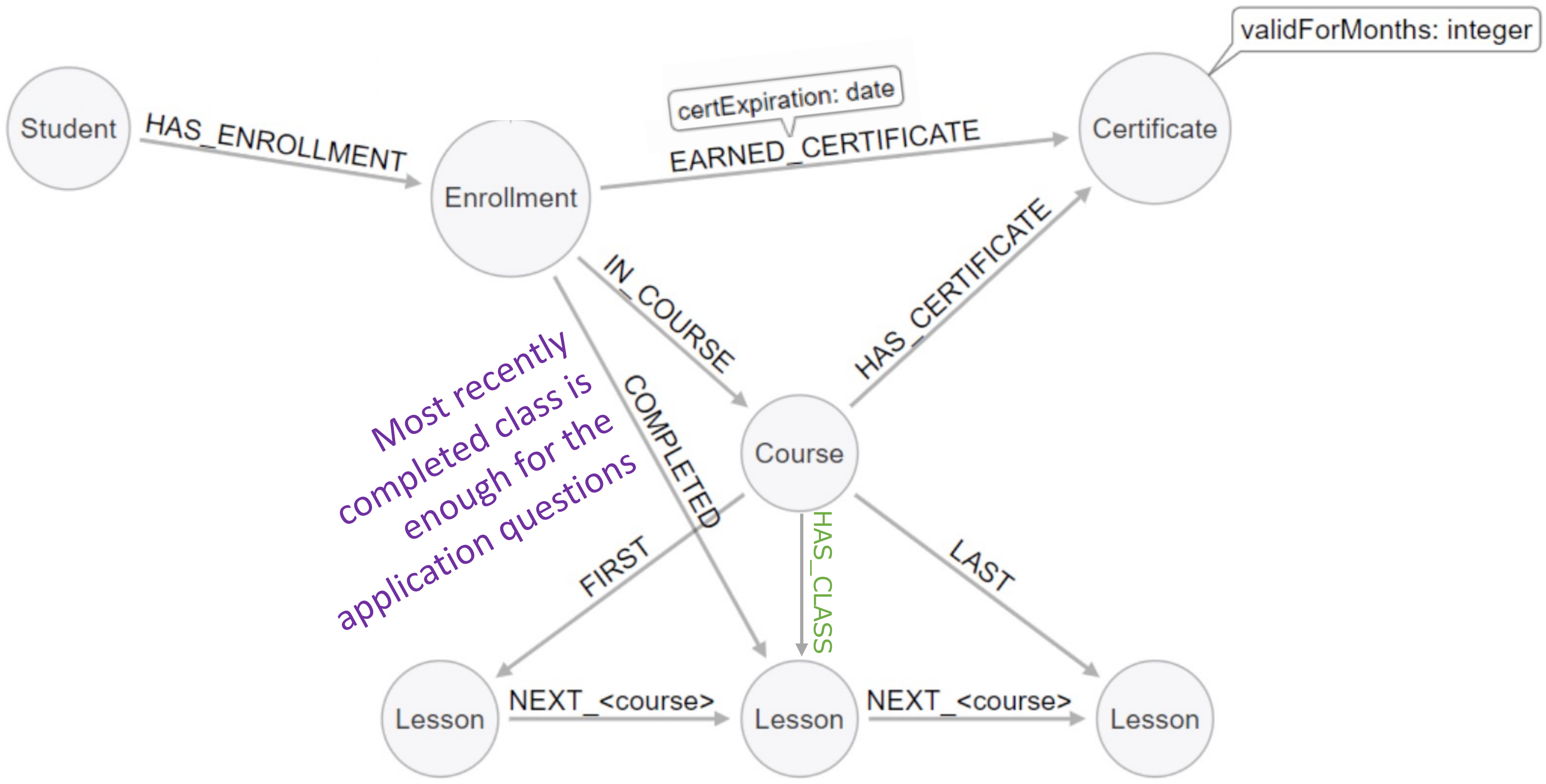
How to model the first (and last) lessons in a course?



- It works, but still we need to access a property of a relationship to solve a frequent request
  - better with last as boolean rather than with nbr (require to check whether nbr is max(nbr) for a course

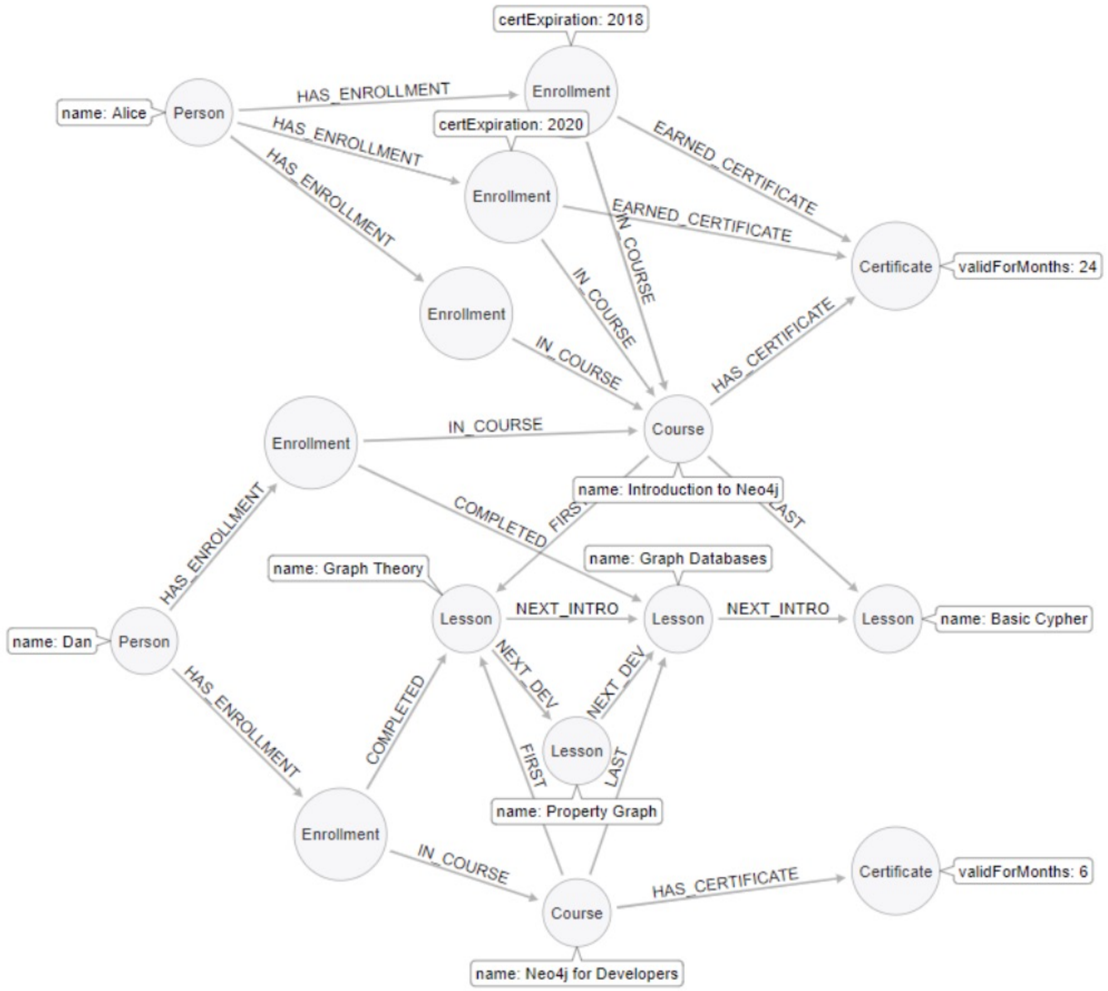


- More efficient to find first and last lesson in a course (frequent request)
- Some redundancy
  - Impact on space
  - Impact on insertion/update time



- No need to add it if we know we'll never ask for all the lessons of a course, but just traverse them from the first to the next one till the last one

Graph Modelling –
Test questions against model



1. Which lesson(s) is Dan currently working on?
2. What are Alice's current certifications?
3. Which lessons are in the **Neo4j for Developers** course?
4. What is the last lesson in the **Introduction to Neo4j** course?
5. Which lesson follows **Graph Theory** in the **Neo4j for Developers** course?
6. Who has completed **Introduction to Neo4j**?

- Choose your favourite modelling
- Construct the graph containing the sample data
- Formulate the queries
- Check whether they are reasonably fast to evaluate