Relational Model: Formative Assessment 1

Feedback: Overall, I received well-descriptive relational schemata with well-defined PKs and FKs that materialize the relationships between relations! Many of you captured the underlying semantics for the one-to-many and many-to-many relationships like the academic position relation and the co-authorships. Moreover, many solutions dealt with associating relations using only the correct foreign and primary keys, which is the core component in the relational database design (Guideline No. 1 in our recent lecture). There were also some submissions including instances/tuples to provide more clarification on their models!

Hereinafter, I provide some comments/feedback over certain issues raised from some solutions:

There were some relational schemata without FKs (or FKs pointing to non-PKs). Recall, that the FKs are **mandatory** in every relational schema design since they establish the referential integrity constraints and provide the meaning of the 'relational modelling'. Hence, we should avoid having relations without any reference (participating either as referencing or as referenced relation) in the provided schema. The PKs are **mandatory** to every relation 'by definition of' the relational modelling, i.e., we need to define a specific subset of attributes to uniquely identify tuples.

When drawing a FK, the arrow points towards the PK of the referenced relation departing from the referencing relation, e.g., in the provided schema below, ProfessorID in the HighDegree relation points to the ID in the Professor relation demonstrating that the former is the FK and the latter the PK.

For the co-authorship concept, one can associate an author (researcher) with a paper (publication) in a 'relationship' relation defining a composite primary key: research-id authors a paper-id (like in our example Company schema: employee-id works on a project-id in the WORKS_ON relation). Then, based on this association relation, one can retrieve the co-authors if we search for those authors/researchers having authored the same paper (paper-id); or in our example Company schema, we can find our colleagues/co-workers in a project by selecting those employees having the same project-id in the WORKS_ON relation. This can be achieved by a 'recursive' selection query, which we will be discussing in our SQL lecture this week. Hence, we do not need to 'constraint' our schema by putting a fixed number of co-authors to denote this relationship.

Finally, in the relational model, there is no notation on the cardinality ratio (e.g., 1:N, N:M, 1:1) between relations. When we need to model e.g., an **one-to-many** relationship between relations A and B, then, we associate the PK of the relation A with the FK of the relation B denoting that **one** tuple from A is associated with **many** tuples from relation B and vice versa. Consider in our example Company schema that an employee (relation A is EMPLOYEE) has many dependents (relation B is DEPENDENT). For a **many-to-many** relationship, consider the WORKS_ON relation associating an employee with many projects, and on the other side where a project is having more than one employee. In this case, such type of relationship is modelled via a new association relation having two FKs: one for each associated relation (look at the Guideline No. 1 in our recent lecture).

Possible Schema: I provide a possible schema that derives from the provided textual description (some of your colleagues come up with very similar model). Note that this solution is a normalized

relational schema dealing with all the requirements for efficient relational modelling, which is the objective of the last lecture being in the BCNF.

Note: The provided schema is **not** unique; you can definitely derive different schemata. The relations in green correspond to basic relations, i.e., relations that do not reference to any other relations.