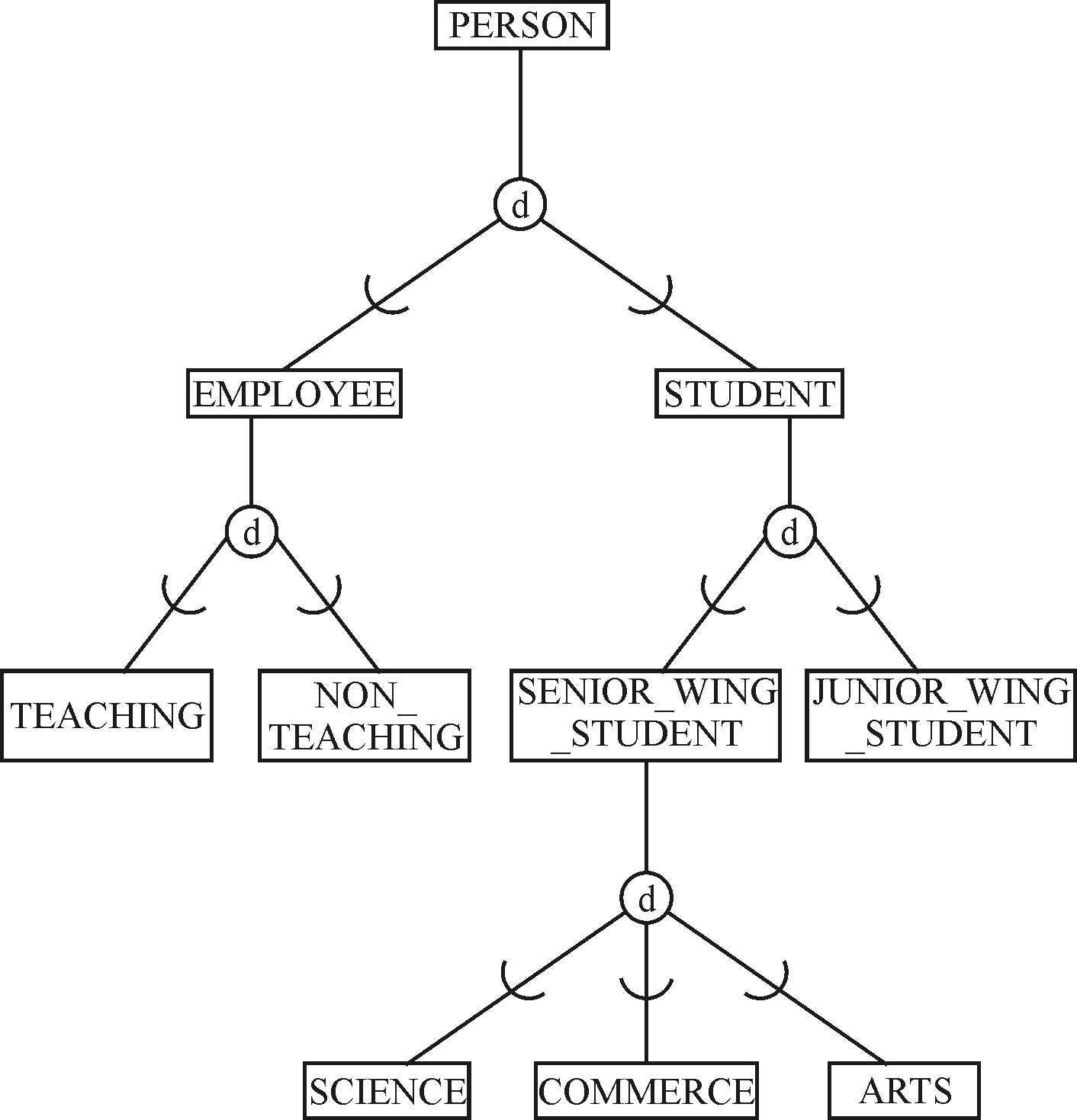
Please provide your thoughts related to EER modeling:

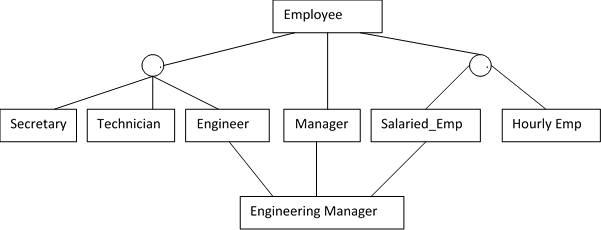
* Discuss the two main types of constraints on specializations and generalizations.
* What is the difference between a specialization hierarchy and a specialization lattice?
* How does a category differ from a regular shared subclass?  What is the category used for?
* Illustrate your answer with examples.
* If you want to create EERD, do you find a database design tool to support EERD?

EER stands for enhanced entity relationship modeling. This allows for more complex portrayal of entity relationships, however the standard for EERD’s are less defined than for simple entity relationship, thus results may differ based on who creates the EERD. However, using an EER allows for superclass and subclass modeling, as well as some constraints of the superclass and subclass modeling. The two main constraints for specialization and generalizations are disjointness constraint and completeness constraint. Disjoint is a constraint that specifies that an entity can be a member of at most one of the subclasses of the specialization. For example, an employee can be only one of secretary, technician or engineer. On the other hand, completeness is a constraint that specifies that every entity in the superclass must be a member of at least one subclass in the specialization. For example, an employee has to be either an hourly\_employee or a salary\_employee. Thus, while disjoint specifies one or less, completeness specifies one or more.

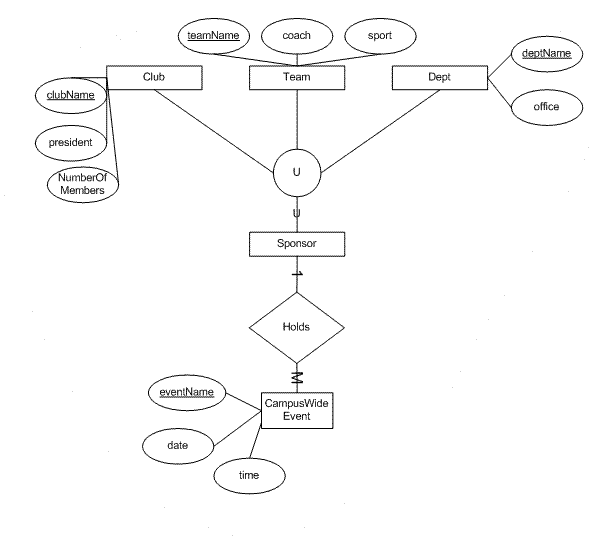
For a given specialization there may be a specialization hierarchy or specialization lattice. For the specialization hierarchy, the hierarchy has the constraint that each subclass has up to one parent and up to one child. This results in a tree structure for the hierarchy. One example of a specialization hierarchy for a person is shown below (note that each person can be only one of each of the subclasses):



The specialization lattice has the constraint that a subclass can have more than one class/subclass relationship. This results in the subclass inheriting all predecessor classes up to the root class. An example of a specialization lattice for an employee is shown below (note that an employee can be an engineer, manager, and salaried\_emp if they are an engineering manager):



Another thing that can be displayed in the EER is something called a category. A category is a union subclass which contain otherwise distinct entity types. For example, the owner of a property can be a person, company, or government, or a sponsor for a university event can be a club, team or department. An example of an EER showing the sponsor category is shown below:



Unfortunately, since the EERD does not have a standard format, there are no database design tools which will do that work for you. To create an EERD, it is best to use a more generic diagram creation tool, perhaps draw.io or powerpoint.

Please provide your thoughts related to ERD and EERD modeling:

* Share any experience in EER modeling used in the past (in school or at work). If you don’t have experience or you haven't seen others using EER modeling; you may elaborate the benefits of using EERD.
* Explain pros and cons for EER modeling. If you use EER,why and when you will use it.
* If you have seen an EERD based on requirements, is possible to come up with a similar ERD to meet the data requirements?
* If you have learned or used Unified Modeling Language (UML), you may create class diagram instead of ERD.  Compare and contrast class diagram and ERD.
* In the textbook, you know there are conceptual design, logical database design, and physical database design in sequence; if you use UML, can you come up with a class diagram to match the conceptual, logical, and physical database designs?

I do not have any personal experience with EER modeling through work or my personal time, so I will elaborate on the pros and cons for EER modeling. To start, some of the pros of EER modeling include creating more detailed relationship models, with specializations and generalizations, disjoint and completeness constraints, specialization hierarchy/lattice structures, and union classes like categories. This has the potential to communicate much more information than a standard ERD. On the flip side, some of the cons of EER modeling include the lack of a EER standard notation, as well as the added complexity potentially taking away from the message that would be more simply laid out with an ERD. Thus an EER could be used in lower-level technical meetings where you want to get into small details of a database design, while an ERD might be more suited to high-level, executive meetings where getting the general idea across is preferred.

From an EER, it is possible to create an ERD from the requirements. The level of detail will be of lower depth, and not all the constraints will be modeled, simply because the level of detail in an EER is higher than in an ERD. However, if you can get the point you are trying to make across with a similar ERD, sometimes you can get away without modeling all the information in the EER.

Alternatively to the ERD, you can also make a UML diagram which will convey some of the same information. One similarity between the ERD and the UML diagram is in the notation for an entity. Each entity is divided into its own area and each of the attributes for the entity are laid out. UML diagrams also specify some relationships between entities, such as the cardinality and subclasses/superclasses. However, the UML diagram will not specify things like foreign keys or foreign keys, and will also include the associated methods for the class/entity. Thus I would imagine that ERD’s are more useful for database design, while UML diagrams are more suited to planning object oriented programming type projects.

UML may be a good fit for some of the conceptual database design phases, but poor fits for other database design phases. For example, UML may suffice for the physical database design. However, when it comes to logical design, it may be challenging to display the conversion between data structure to physical memory. However, for the physical database design, UML may suffice. For the different database design phases, UML may suffice, but it less than ideal.