My research interests include creating systems to process natural language questions to dynamically query clinical databases of arbitrary data models. A contrived example of the use of a hypothetical system could begin with the request, “Find patients with a [past] diagnosis of asthma”, where in response the system is able to infer something akin to:

* Asthma *is a* physiological condition ->
* ICD9/10 codes *represent* physiological conditions ->
* <database> *has* <table><column> which *encodes* ICD9/10 ->
* A query *can be written* *using* <database> to *find humans* with Asthma *using* code(s) <x>

As such, I chose to attempt to model the metadata of the MIMIC-III database (specifically, an instance of it on my home computer). In order to create such a system, I sought to ensure that 1) linkage of the database schema to the clinical and biological concepts their data represent is done by OWL Object Properties such as *Encodes*, *HasColumn*, and *Represents*, and 2) database schema data (e.g., Column *DataType* and *Name*) are captured in sufficient detail to be able to construct a SQL query on MIMIC-III based on the ontology. I encountered a number of interesting questions and ideas during the project.

First, I struggled with the fact that my local copy of MIMIC-III is one *instance of many*, rather than a monolithic, universal instance*.* The MIMIC-III database schema (as maintained by MIT) changes over time, and even its representation on my local computer in *SQL* may differ elsewhere – the MIMIC-III data could be represented by RDF triples or CSV files as well, in which case their metadata would differ. Therefore, if I was to query multiple instances of MIMIC-III data, for example, I found it would be critical to model their *instance-specific* representations appropriately in the ontolo(gies).

I also wrestled with questions of how best to model database column metadata. For example, the Column [SUBJECT\_ID] appears on nearly all MIMIC-III Tables and represents unique patient identifiers (specific to MIMIC-III). Every instance of [SUBJECT\_ID] is an integer type. Initially, I concluded that it would therefore be best to simply have a single [SUBJECT\_ID] instance of class DatabaseColumn with each Table it was present in connected by an RDF triple, i.e., ?table :HasColumn :SUBJECT\_ID. While this representation was convenient and appeared to work for [SUBJECT\_ID], however, as a general ontology pattern it simply wouldn’t likely hold true for all other cases. For example, a column named [CATEGORY] appears in two Tables ([D\_ITEMS] and [D\_LABITEMS]), but carries a different semantic role and domain for each (i.e., categories of general clinical measurements versus categories of laboratory tests). Thus I ultimately chose to refactor all Columns into table-specific instances with the URI pattern <TABLE\_NAME.COLUMN\_NAME>, and each with an RDF *Name* Data Property of <COLUMN\_NAME>^^xsd:string.

In terms of