This issue is designed to break down and simplify the process of creating, reordering, and redesigning our Raider Planner system to support a relational database. The primary goal is to move our system away from serializing data objects and storing them in the StudyPlanner.dat file. Currently, our application is rigidly designed to exclusively utilize serialization; so the process of moving to a relational database will be complex and time-consuming.

Therefore, it is imperative that we have a well thought out, designed, and modular strategy for completing this migration process. In order to do this we will break down this migration process into four main 4 steps. These steps can be completed in parallel manner by multiple developers and once each one of these steps has been completed we will then be able to easily and subsequently merge all the developer’s reposts successfully into our system. Additionally, each one of these four steps has sub-steps that need to be completed concurrently. The 4 primary steps are as follows:

- [x] \*\*Bug 2: The ".project"\*\*

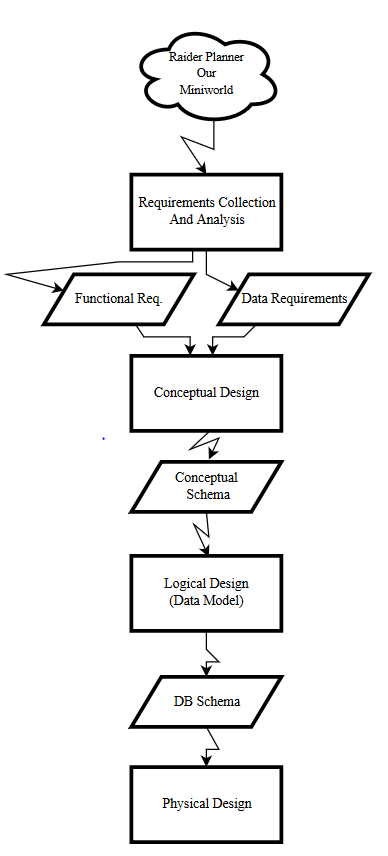
1. Install Apache Derby and configure it to run inside of Gradle
2. Database Design
   1. Conceptual Data Model
      1. ER diagram
      2. Cardinality Constraints
      3. Total and Partial Participation
   2. Relational Data Model
   3. Normalization of Relational Model
3. Create Relational Database Schema SQL and Data Access Layer with JDBC
   1. Basic Implementation of Relational Schema
   2. Implement Complex Queries, Triggers, Views, and Schema Modification
4. Integration, Regression testing, and unit test

For more information about each one of these four steps please visit their wiki pages.

Database Design

Before explaining the steps that need to happen in order to create the database design documentation and diagrams we are going to cover some basic concepts of databases themselves.

A **Database** is simply a collection of interconnected and related data; whereas, **Data** is simply just facts that can be recorded and have meaning to our enclosed universe. A Database Management System **DBMS** is generally recognized as the system that helps users create, maintain, and query the database. In essence, a DBMS is a general-purpose software system that facilitates the completion of specific processes. For our application we are going to use Apache Derby’s DBMS and through this DBMS we are going to create a relational schema and queries to manipulate our data.In order to do this we are going to have to **define** our database**. Defining** a database means specifying the data types, relationships, structures, and constraints on our data that must be catalogued and stored in our database. This process has many important steps that must be followed in order to create a functional database system. And for the process of creating our raiderplanner database we are going to go through certain important phases. Figure 1 demonstrates the four phases that the database designers will need to follow.



Phase 1 **Requirements Collection and Analysis**:

Generally this phase entails interviewing prospective users of the system in order to gain an understanding of how the data flows in their mini-world. The goal of this phase is to understand how data relates to each other and how users interact with it. For the purpose of our application database designers will thoroughly research through the raiderplanner application to understand what data is stored in serialized objects, and how this data interacts with the entire system.

The result of this research is going to be a concisely written set of requirements for how data moves through our raiderplanner system. This document will contain written and diagrammatic information regarding the system. For example, the developers may consider creating a level 3 data flow diagram along with a sequence diagram for the entire raiderplanner system. Note: eclipse will actually generate, with the proper plug-ins, a sequence diagram or a UMLdiagram, of the entire system.

Moreover, during this phase the database designers will create a document of functional requirements that contain the operations and transactions that the database will need to include. For example, retrieving and updating the database.

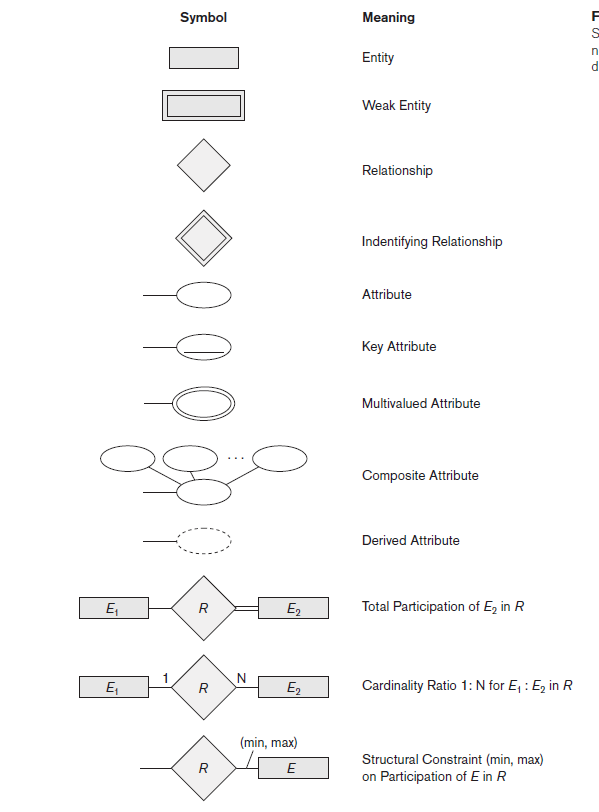
In summation, the outcome of this phase is the creation of two documents (I.E. the functional requirements and data requirements documents).

**Phase 2 Conceptual Schema or Design:**

Only once the **Requirements Collection and Analysis** phase has been completed and reviewed can the database designer’s move on to the creation of the conceptual schema for the database. This process entails creating a high-level conceptual data model. This conceptual schema will contain the precise and concise description of the data requirements of our application and will include a description of the entity types, constraints, and relationships. In order to create this description of the entities the database designers will need to decide what data modeling diagram tool to utilize.

My recommendation for the database designers is to use the Enhanced-Entity-Relationship (EER) model. The EER model describes data as entities, relationships, and attributes. The primary reason we create a high-level conceptual schema is to make sure that the requirements are met and that there are no conflicts between them and how our system works.

By doing this approach the database designers will be able to better understand how data interacts in our system, along with the requirements for our application to function properly. This approach allows us to not worry about low-level concepts. Figure 2 illustrates some of the diagrammatic shapes and components that comprise the EER diagram. Note: This is not a comprehensive list.



Things that will need to be considered while creating this data model are as follows:

■ Simple or atomic attributes

■ Composite attributes

■ Multivalued attributes

■ Entity types and their corresponding entity sets

■ Key attributes of entity types

■ Value sets (domains) of attributes

■ Relationship types and their corresponding relationship sets

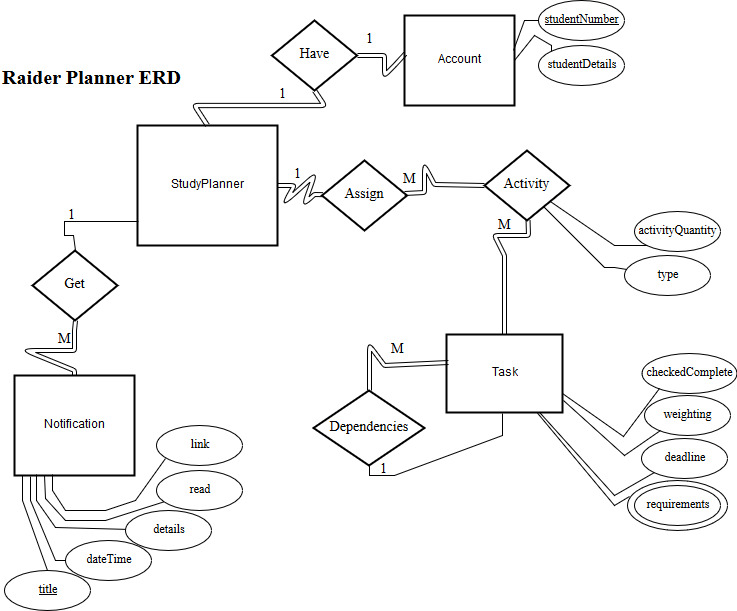
■ Participation roles of entity types in relationship types

■ Cardinality ratios (1:1, 1:N, M:N for binary relationships)

■ Participation constraints (total, partial)

■ All constraints upon our system

Below is an example of an EER diagram that depicts key classes and the data that needs to be stored for our application to be able to function properly. This diagram illustrates what should be done by the data base designers, and it contains the cardinality, participation, and other constraints that our system must uphold.

9

In summation, the outcome of this phase will be an EER diagram, or some other data modeling diagram, that completely encapsulates all the requirements of our system.

**Phase 3 Logical Design:**

Only once phase 2 has been completed can the database designer’s move on to Phase 3. In this phase the goal is to take the EER diagram or the other data modeling diagram potentially used and migrate it to the relational model. This process is called the logical design or data modeling mapping.

To clarify, the goal of the third phase is to create a relational schema that will be implemented in our DBMS. In this process the designers will not only create a relational model, but will also have to normalize the relational model they create up to BCNF. If certain relations cannot be normalized up to be BCNF then 3NF will be accepted. The goal of normalization is to eradicate data redundancy and to prevent anomalies. This wiki page will not dive into the process of normalizing a database; however, here is a link to a course that covers normalization.

<http://www.techtud.com/computer-science-and-information-technology/database-management-system/database-design>

In summation, the outcome of this phase will be a relational schema normalized up to BCNF. Additionally, the database designers will create a relational diagram and document the normalization process. This means that the database designers will justify why each relation is normalized and how they went about this process. For example, each relation should have a list of all functional dependencies, prime and nonprime attributes, F closure, minimal closure, and all candidate keys. All of these above mentioned things are needed in order to normalize a relational schema properly. The above mentioned links explains these concepts.

**Phase 4 Physical Design:**

The last phase can only be started once all other phases have been completed successfully. Its goal is to create the actual relational schema inside of the DBMS. Additionally, the database designers will have to implement all necessary indexes, access paths, and physical design parameters for the database itself. For example, during this final phase the database designers will use SQL to create the relations, set all domain restrictions for each attribute, and create all necessary queries. Obviously, all previously defined functional requirements and data requirements have to be applied to the relational database.

In summation, this is the final phase of creating our relational database and all the previous steps lead to this final step. The outcome of this phase is a functioning relational database with all requirements and queries created. This means once this phase is done we can start writing JDBC code to get, insert, and update our relations in our database.