Entity Relationship Diagram Mapping Honors Midterm Report

by

Cheryl Dunn

100963953

Prof. Louis D. Nel

School of Computer Science

Carleton University

Ottawa, Ontario

# Intro

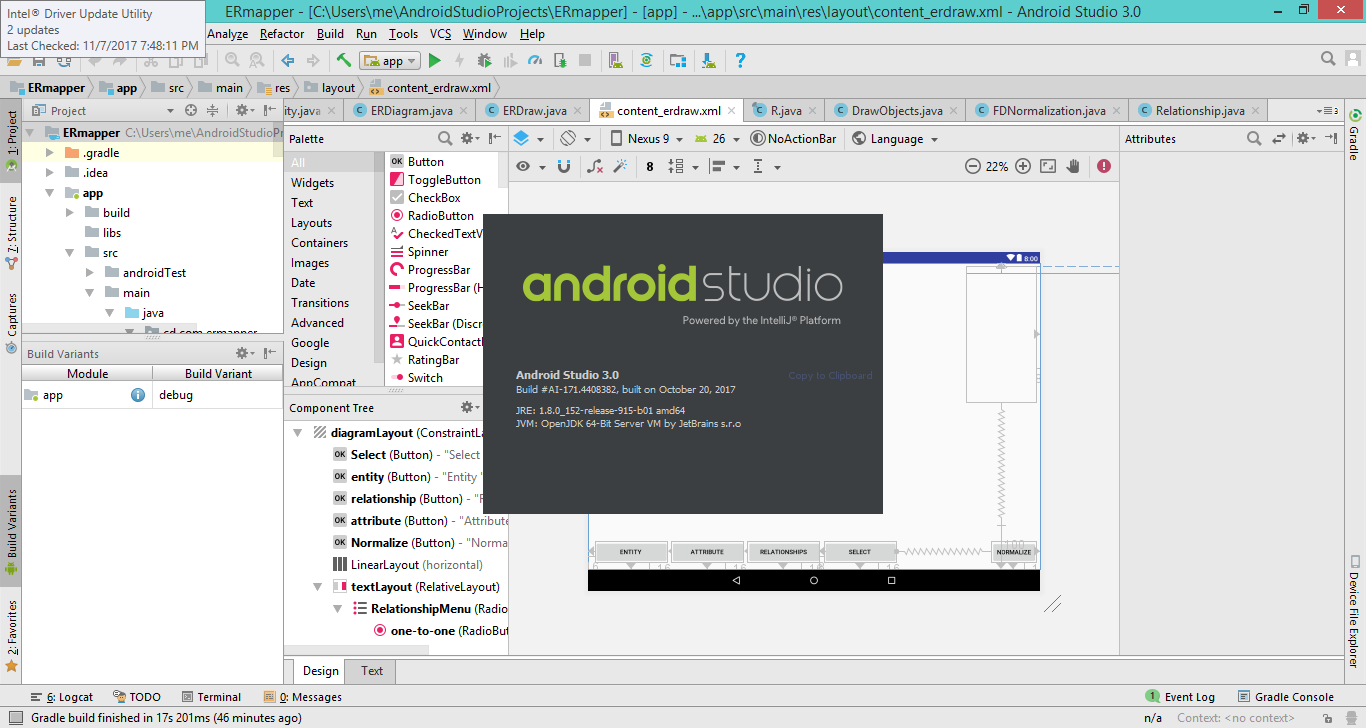
Computers have countless applications in the world ranging from text editors to games to security and much more. An important part of each of these application is storing data and information. In some scenarios, the best way to do so is by using a database. Databases organize a collection of data as schemas, tables, queries, reports and objects. When designing a database, it is important to organize the data so that it models the data in a realistic way to support the information being collected. For designers to organize data they may first make a visual diagram which will allow them to identify entities, attributes and their relationships. An Entity Relationship (ER) Diagram is a widely used method for conceptualizing and visualizing the logical structure of a database. By Creating an ER Diagram and normalizing it to its functional dependencies Database Designers can easily create an accurate database for whatever data needs to be stored.

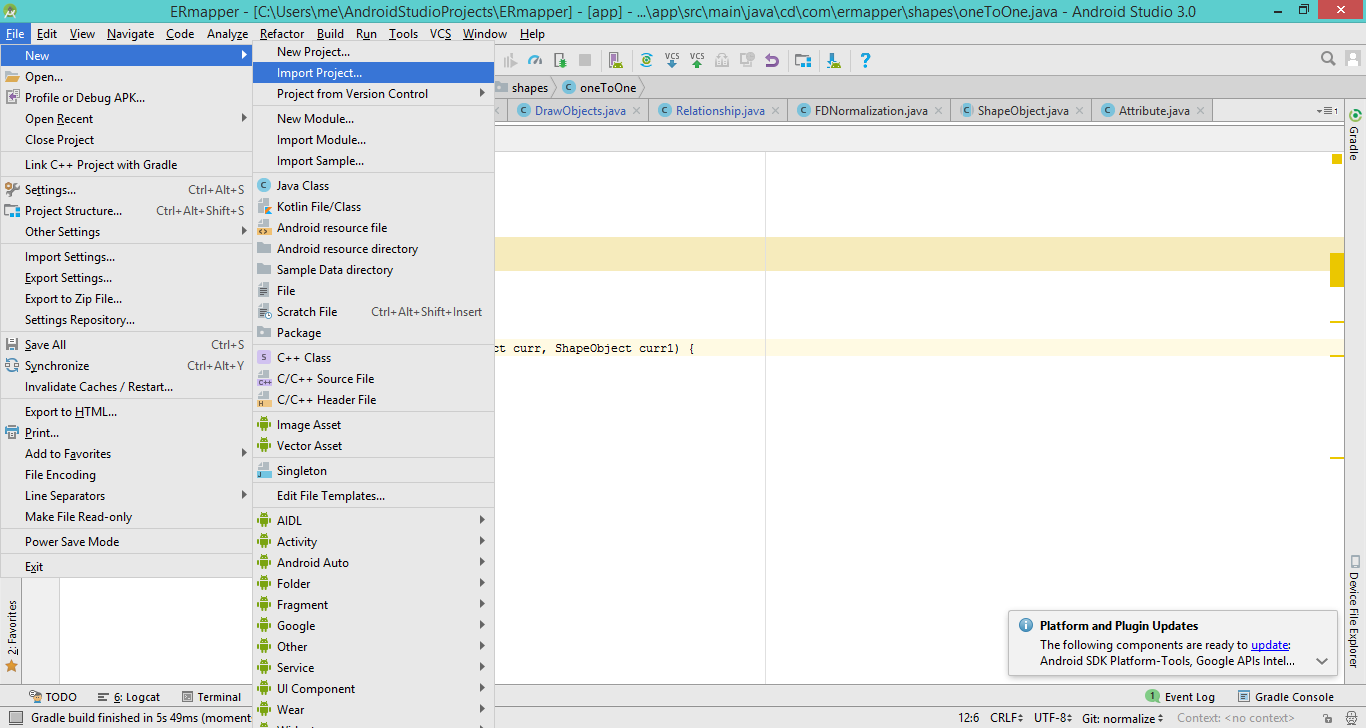
# Thesis

ER Diagrams are largely used in the database development process, creating a program that allows users to create an ER diagram and then converts it to a database would help save time, ensure consistency and accuracy.

# Setup

The following are instructions of how to install and set up the software required to run the ERMapper application in Android studio.

1. Install Android studio 3.0
   1. JRE: 1.8.0\_152-release-9159b01
   2. JVMOpenJDK 64-bit Server VM by JetBrains s.r.o
2. In Android studio go to File -> new -> import Project. A dialog will pop up, select the directory where you saved ERMapper.zip.



# Approach

The ERMapper creates a blank canvas that allows users to draw an ER diagram. In the drawing phase users can create Entities, Attributes and Relationships between each. Users simple click a button to select what they want to draw and then they can move them around the screen to place them where they would like, as shown in Figure 1 below.

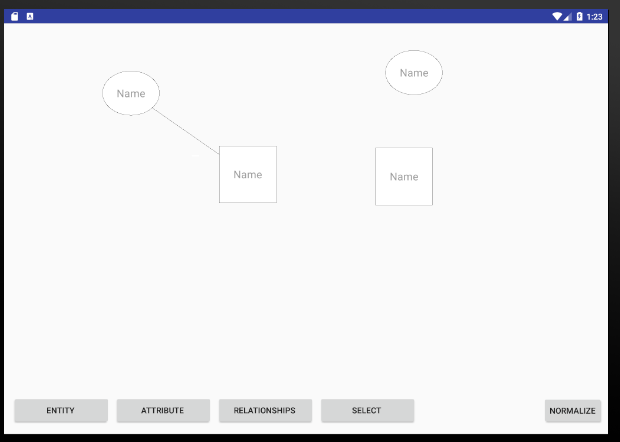


Figure 1 ERMapper Drawin

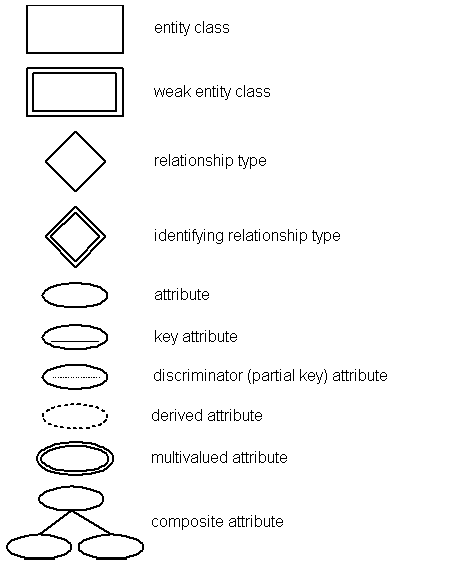
ER Diagrams are a visualization of Relations in a database. Entities, modeled as squares, represent a real world concept that a Relation is, where as attributes, modelled as ovals, represent property of its corresponding relation. In a Database the Entity or Relation is the table itself, and the attributes are the columns in that table. Every entity can have multiple attributes, however it requires at least one attribute to be a key (which uniquely identifies each row in the table). In an ER diagram the key attribute(s) are identified with an underline underneath their name. A relationship is a line that connects attributes and entities, they may also indicate the cardinality of the relationship, though at this point the program only deals with binary 1:1 relationships. Figure 2 below shows the Symbols that may appear in an ER Diagram. Currently the program can handle regular and weak entities, regular, key and composite attributes, and regular relationships.

Figure 2 ER Diagram Symbols

## Drawing the Diagram

The program contains a class called ER Diagram which stores an arraylist of shapeobjects which can be entities, attributes or relationships, with their coordinates to be drawn to the screen. The main page of the application seen in figure 1 shows the canvas of the page. The user can select to create an entity, attribute or relationship. Upon doing so the object is create along with an edittext object to store its name and its set of coordinates and it is added to the object list. Unless the object is a relationship, in which case the system creates a line that follows the mouse to connect to objects. If the relationship is valid then it gets added to the object list. The program search its list of objects and draws the correct shape based on the object type onto the screen at its correct coordinates. The canvas also has a motion event listener which activates when the user clicks the screen. It checks if the user has click a coordinate that is inside a shape and then allows the user to move that object around the screen, if the object has any relationships it will also update those coordinates. To create key attributes a user can double click on the attribute they want to make key, and an underline will appear. To make a weak entity the user can double click an entity. Once the user has completed has completed their drawing the they can click normalize to normalize the objects and create a relation diagram. Figure 3 below is an image of a complete ER Diagram.

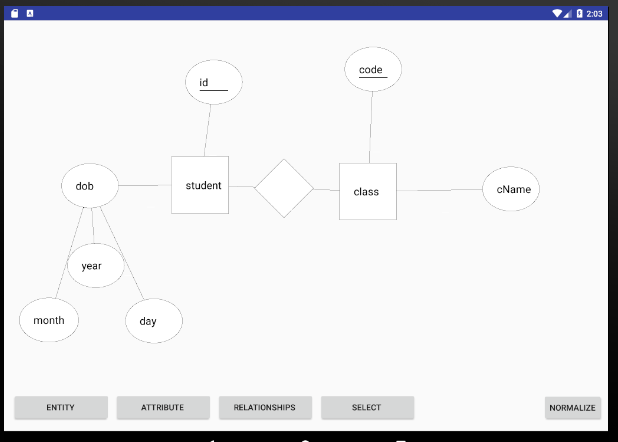


Figure 3 Completed ERDiagram

## Normalizing the Drawing

Once the user clicks normalize the drawing. The erDiagram passes all of its entity objects, which contains a list of all of their attributes to the FDNormalize class. The FDNormalize class then takes the entities and converts them to functional dependencies. The program is currently only set up to work with binary 1:1 relationships and therefore only does steps 1, 2 and 6 in the algorithm mapping ER diagrams from the book *Fundamentals of Database Systems* by Ramex Elmasri & Shamkant B. Navathe.

Step 1. For every regular entity create a Functional dependency, where the primary key is the left hand side, and the attributes are the right hand side

* If the primary key is composite, all of the values will be part of the primary key

Step 2. For all weak entities, create a new functional dependency, but add the primary key from its identifying entity to its primary key.

Step 6. For composite values, choose one attribute to be the primary key and add the others as attributes.

The program itself looks at each entity and checks if it is regular, if its is it performs step1, by looping through each attribute and checking if that attribute is primary or not, and then adding the attributes to the correct side of the functional dependency. It also searches if the entity has an relationship with a weak entity. If it does it creates a functional dependency for the weak entity and adds its primary key to the weak entities primary key, as proposed by step 2. Next the program looks at each attribute from the entity and checks if the attribute is composite, if so it performs step 6.

Once the functional dependencies are produced, the normalize function that was provided by Prof. Louis Nel is performed and a new screen is produced that lists the results of the normalization as shown in figure 4 below of the ERDiagram from figure 3.



Figure 4 Normalized ERDiagram

### Converting Ternary Relationships to Binary

Replace *R* between entity sets A, B and Cby an entity set *E*, and three relationship sets:

1. *RA*, relating *E* and *A*

2.*RB*, relating *E* and *B*

3. *RC*, relating *E* and *C*

Create a special identifying attribute for *E*

Add any attributes of *R* to *E*

For each relationship (*ai , bi , ci*) in *R,* create

1. a new entity *ei* in the entity set *E*

2. add (*ei , ai* ) to *RA*

3. add (*ei , bi* ) to *RB*

4. add (*ei , ci* ) to *RC*

Mapping of N-ary Relationship Types.

* For each n-ary relationship type R, where n>2, create a new relationship S to represent R.
* Include as foreign key attributes in S the primary keys of the relations that represent the participating entity types.
* Also include any simple attributes of the n-ary relationship type (or simple components of composite attributes) as attributes of S.

## Work to do

Next to be done in the system is to complete steps 3,4,5 and 7 of the normalization process which includes handling more complex relationships. To do so I need to create relationships that show cardinality. I will also have to create the connection to a sql database take the relations and create the database. Testing and other debugging will also be required. Other things that I would like to include if time permits is adding a way to remove objects so that you do not have to restart, refactoring the code to simply and remove any duplication as well as add more error checking.

## Known issues

Some issues with the system include that you order matters when creating a complex attribute. Foreign keys are not represented with a dashed line. Along with this sometimes double clicking is ends up moving an object.

## Work Schedule

The table below is an updated version of expected time to finish the project.

|  |  |  |  |
| --- | --- | --- | --- |
| Objective | Estimated Time | Due Date | final date |
| Research | 1 week | Sept 3 | Sept 2 |
| Identify all functional Requirements | 2 days | Sept 3 | Sept 2 |
| cREATE sTRUCTURE FOR er dIAGRAM including a user interface | 3 weeks | Sept 30 | Sept 20 |
| Create structure + objects for Functional dependencies | 1 week | Oct 10 | Oct 5 |
| write + Submit Mid-term Report | 1 weeks | Oct 30 | Oct 30 |
| Add complex RElationships | 1 week | Nov 10 |  |
| Apply rules to convert ER to FD for complex relationships  (this already works for simple relatiosnhips) | 3 weeks | Nov 10 |  |
| Implement provided program for FD to DB | 1 week | Dec 1 |  |
| Write + Submit First Draft report | 2 weeks | Dec 1 |  |
| Testing and review | On going | Dec 14 |  |
| Submit Final Report | 2 weeks | Dec 15 |  |