The final ER diagram closely resembles the initial diagram, but with a few changes. There are now two relationship sets between the Customer and Party entity sets: “Owns” and “Are in”. While developing the schema, I realized that the participation constraints could not be enforced on both ends, so instead I created an “owner” column in the customer table. This would hold the id of the customer that created the party. Any additional participants would be held in the Party\_Customers junction table, which captures the “Are in” relationship. The Party table also has an “agentID” column to capture the relationship between Party and Agent. The Party Table has relationships with the Event, Attraction, Destination, Lodging, and Transportation. To capture these, I created junction tables which refer to the party ID in one column and the ID of the item in the other. However, I could not capture the participation constraint - the junction table refers to the party’s ID, so the party has to be inserted first. If I had added a check constraint to party that checks if its ID is in the junction table, the tuple in the junction table would have to be inserted first - an impossible condition. Each Event is associated with exactly one Attraction, each Attraction has exactly one Destination, and each Lodging has exactly one Destination. Therefore, I added a foreign key to each of these tables which refers to its Attraction/Destination ID. Finally, the Review table holds reviews for Events, Attractions, Lodgings, or Transportations. Without its associated object, the review is meaningless, so it is a weak entity. The Review table has four columns, each of which refer to the ID of the four different entities it can refer to. A specific review will have one of these columns populated with the ID of the object the review is associated with, while the other three columns are null. This way, the review table can be joined with a tuple for one of these objects to get all of the reviews for that object.