STSCI 5060 Homework 3

Pete Rigas (pbr43 cornell.edu)

November 27, 2020

Please see the end of the problem for all relevant diagrams!

1 Problem: Vendors Supplying Parts

1.1 Part A: Convert Table to the First Normal Form

In order to convert the given table into the normal form, we will recall that with the normal form, we will normalize the given relations and entities. Because the sample parts from different vendors are divided into the logic chip and the memory chip, these individual parts each have the attribute of the vendor name, the vendor address, and the cost of each unit. From this basic information, we also know that, from flexible characteristics of the database, that any changes to quantities relating to the vendor name, vendor address, and the cost per unit will be reflected as the vendor name for a certain part could change, in addition to the rates per unit that the vendor charges the company. Furthermore, we know that redundant data is not beneficial to have in the data base, as this redundant data consumes store space, and also does not make the most efficient use of memory in the system. As mentioned several times in lecture already, depending on the business rules that are enforced, it is possible that such wasted space and null values could present difficulties with regards to data integrity. Therefore, it is important to optimally design the database so as to maximize the storage capacity, and in doing so it is probable that normalization will be used.

For these diagrams, we also conclude that the attributes that are underlined in the diagram represent the primary key values. Specifically, the primary key values are important because they signal how the values in the database will be ordered. Namely, depending on the attributes of an entity that is represented in the database, we know that the values with higher significance will have primary key values in them. With this information, we can convert to the first normal form by observing that the model would be in the first normal form if it does not contain any redundant or replicate information, which includes information in the fields or group of fields. Additionally, we can begin to represent the Part Supplier in the first normal form by listing its attributes, which include the Part Number, the part description, the Vendor name that sells the unit to the company at some fixed cost, and also the address and even the unit cost of the part that is sold. Next, we know that the attribute of the Part Number is explicitly detailed by the vendor because each part fits specifications relating to the way that it is produced, ie how it is manufactured, which is also influential on the cost per unit. With these attributes of the relation Part Supplier in mind, we know that the table below appropriately represents the quantity in the first normal form because, again, all of the quantities that are represented contain specific information about the part number, which include its ID number, the description of the part, the name of the Vendor that sold the part, and finally the address and unit cost.

1.2 Part B: Functional Dependency, Candidate Key

To illustrate the functional dependencies of the vendor supplying parts, we first observe that between two attributes, or two set of attributes, constitute the same relational database table constitutes a functional dependency, given some constraint. Besides the definition of the functional dependency, we also know that the arrows belows specify the functional dependencies from the Part Number onto the description, as the Part Number of a part that is sold to some vendor at a fixed cost is functionally dependent on the description of the part itself, as the vendor reads a description provided by the company to determine if it will purchase the part at the advertised unit cost. Similarly, the vendor name is functionally dependent on the address, as the name of the company that manufactures the parts that it ships to the companies absolutely depend on the address.

Finally, from the Part Supplier Relation, the Part Number and Vendor Name are functionally dependent on the cost of the unit because the cost of the unit is specified by the Part Number and the Vendor Name. The diagram **below** demonstrates the directionality of the arrows that we have described in the functional dependencies from different attributes in the given diagram from the problem statement. In particular, we know that the candidate key is an attribute, or a set of attributes, that we can select from the primary key. But from multiple candidates that are presented in the table or relation, the **only** key that can be referred to with the relations and entities described is the primary key. The Part Number and the Name of the vendor are candidate keys for the Part Supplier Relation.

1.3 Part C

To represent anomalies in the Part Supplier relation, we observe the following. To insert an anomaly, we observe that particular attributes cannot be inserted into our database table without other attributes from the entities that would cause an insert anomaly to occur. Besides encountering an anomaly, it is also possible to delete an anomaly, which would have the effect of causing loss of data in the database; if the user deletes an anomaly, and the part number information that refers to this anomaly, then the vendor information would also be deleted with the anomaly that was deleted. In return, the user deleting the part number information could have unintended consequences, not only for the database management system, but also for the information that is intended to be retained in the database. In particular, if we have a part number and delete this anomaly, we would not also want to accidentally delete the vendor name that is selling the part, as an anomaly is presumably a rare instance out of all data occurrences in the database, and should not impact other parts of the data structure that we want to retain, including the Vendor Name.

Otherwise, we can also modify an anomaly by partially updating the rows in our database records that cause the inconsistency within the database. That is, if we encounter some anomaly, we can fix it by merely changing the rows of the data which holds information in the database that would be affected after modifying the anomaly. In this case, the modification anomaly, as do the insertion and deletion anomaly, allow us to accommodate changes in database inconsistencies that we can have when storing the data itself, in addition to how anomalies should be treated in relation to other data components.

1.4 Part D

For the relational schema, we can illustrate the functional dependencies with the diagram **below**. In particular, we know that the relational schema is helpful because it, broadly, describes the data that is relevant to the overall structure of the data that we are retaining in the database. What's more, we know that the unique name can be provided to an individual that is represented in the database. From the logical data model, we can easily translate notions that we store in the database, from the vendor that sells a part to specific characteristics about the part, that are significant for relationally determining what attribute of the part is the most significant to retain in the database. In this case, again from the diagram below, we observe, from the Part Supplier relation, that the description of the part is functionally dependent on the part number; that the address of the vendor that is selling the part that is dependent on the name of the vendor; and finally, that the cost of the part being sold by a vendor is also functionally dependent on the part number and the name of the vendor that is selling the part.

1.5 Part E

From previous comments and description of the relations and entities that we have described, we know that the normal form for the Part Supplier Relation is in the first normal form. We know that this holds because it gives the rational dependency of the anomaly, provides precise relationships that do **not** contain any data redundancies and do not present ambiguity in the attributes of each entity that is involved, and finally, because it does not contain any attributes that are repeated, the Part Supplier relation is therefore in the First Normal Form.

1.6 Part F

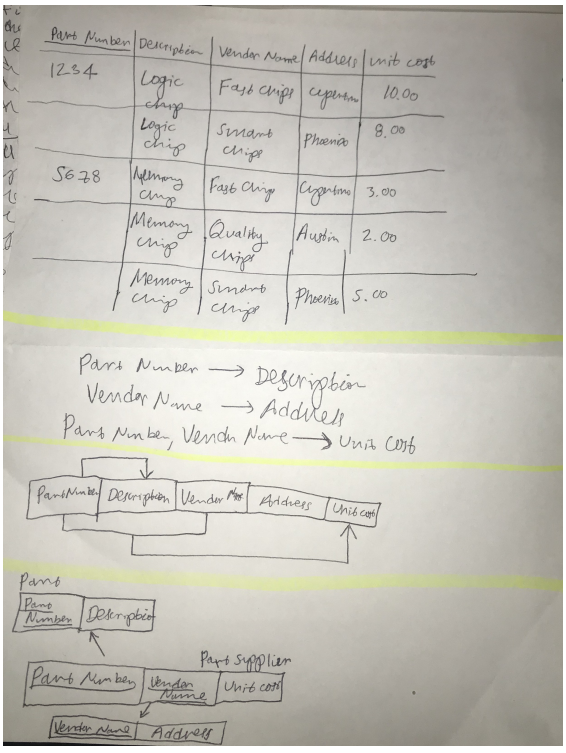
On the other hand, the Part Supplier relation, unlike the comments that we described in **E**, can be developed in third normal forms relations with the following. First of all, if we want a relation to be in the third normal form, we know that, unlike relations in the first normal form, that the third normal form requires that the

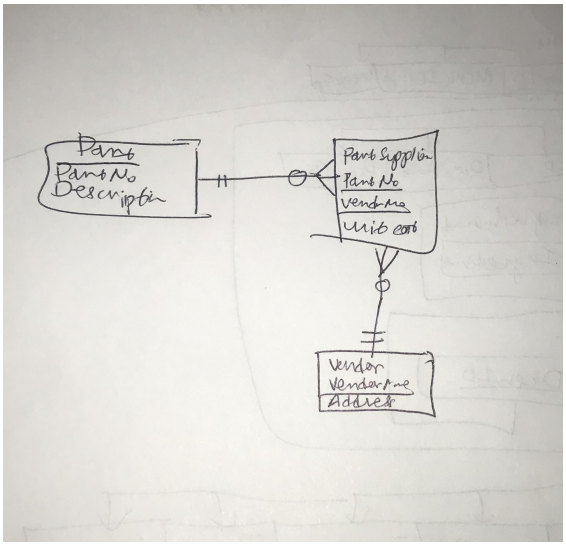
model should always be in the first and second normal form. Beyond these requirements, we also know that the 'structure' of the data will be preserved in the sense that attributes that are non-primary will not dependent on any other non-primary key attributes. Although it is possible that attributes from the non-primary key could depend on other non-primary key attributes, in this case the attributes, for a third normal form set of relations for the Part Supplier to be developed, would be either moved or deleted. With this 'transitive dependency,' from the first normal form relation we know that an attribute, such as the unit cost for a part that is sold by the vendor, would be dependent on the primary key attribute of the Part Number. In general, from this diagram and the set of entities and relations present, the part number for a part that is sold is a primary key attribute of the part entity, in which we assign the attribute of a part number that is related to the vendor name. In this case, because the part number relates to an attribute, namely through the Vendor name, that is non-key attribute, we must delete or move this value from the database.

After the removal of this problematic data, we also know that creating a third normal form (3NF) can be accomplished by removing transitive dependencies. That is, if we are given some relations and want to construct the normal form to reduce a given set of relations to simpler form, the new relations would include the part, part supplier and the vendor. From these set of relations, we know that these new relations are constructed to achieve 3NF, with the Part relation having the part number of the part that was sold, as well as the description of the part, as attributes. Moreover, with the Part Number as the primary key attribute, we know that the Part supplier relation is the second normal form. As for the normal forms above and below the Part supplier relation, we observe that the vendor relation has the attributes of the vendor name and address, with the vendor name serving as the primary key attribute.

1.7 Part G

For the third normal form (3NF), we know that from previous comments that the model should be presented in the first and second normal forms. On a similar note, we also will impose the condition that any non-primary key attributes will not depend on any other non-primary key attributes, and in which case these values are detected, they ought to be removed or deleted from the database. Now, we again observe from **F** that the set of 3NF that we developed from the Part Supplier relation is in the desired third normal form, in which from the Part, Part supplier and vendor relations, we know that the primary and non-primary attributes appropriately align with representing the primary key, and non-primary key attributes, insofar as to ensure that each individual relation represented in the third normal form reflect the attributes of the part that are sold from different vendors.





2 Problem: University Dining Organization EER

Please see the end for all relevant diagrams!

2.1 Part A: Developing Relational Schema after transforming the EER Diagram

In order to transform the given EER diagram from the given figure, we must first think of the important relationships that we would want to have in the organization. In particular, we know that the resultant set of relations that we want is illustrated below, in which we have a menu entity, which has the obvious attributes of Menu ID, Menu Description and Menu Type, in addition to the dish, dish ingredients, event, staff and staff skill. Importantly, we observe that we can create the types of relationships that we want because from the regular entities in the EER diagram that we are given, namely menu, event, dish and staff, we can create the relations for each regular entity that not only contain all of its simple attributes, but also that all of the entities have clearly defined normal forms that we can classify. In particular, we know that the dish entity has multi-valued attributes assigned to it, as any dish that can be ordered from the university dining organization can order a dish which is composed of specified ingredients, and has a specified identification.

Similarly, from the staff entity, we know that within the dining organization there are staff that work at the location, in addition to skills that individual staff have, all of which is specified by the ID of the employee and the skill that he or she has. But from these preliminary observations about the EER and the relational schema that we are trying to develop, we know that arrows must be drawn between the work schedule entity, and the staff and event entities, so that we can account for how temporal events would impact the function of the organization for different events that it chooses to hold. In short, for the weak entities, we can transform them to relations because from the relation work schedule with the attributes that we have discussed, there exists a primary key of event and staff, which is also the primary key in the menu dish relation.

In order to map the binary relations, we now observe that the cardinality of the relationships that we have described depends on, first, clarifying the type of the relationship that exists between the menu and event entities, and second, determining the cardinality of the relationship between the menu and dish. In the case of the first aforementioned relationship, we can expect that there would exist a binary $1 : N$ relationship between the menu and event entities, precisely because, as indicated in our diagram, we know that the menu entity can describe many events. Also, we know that we must include the Menu as a primary key in the relation event, and from the cardinality of this relationship, it is clear that the Menu primary key, as a result of the Event relation, has a one to many cardinality. AS for the menu and dish entities, there exists a binary M:N relationship, precisely because different menus can contain more than one dish, which constitutes a many to many relationship. Besides the many to many cardinality the results because several menus can have a large variety of dishes, we also observe that the relation menu dish is created, as a result of the primary key Menu, and the Dish foreign key that are present in the Menu Dish relation.

In order to map the unary relations, we follow a similar procedure, in which we identify all unary relationships, which in this case are the supervise relation. The supervise relation invovles the Staff entity, and has the foreign key Supervisor, which in the case of the Staff relation referenced the Employee ID. Putting all of these observations together, we obtain the desired relational schema, in which the resultant set of relations contains the binary and unary relations.

2.1.1 Comments about Relational Schema

From the diagram above, we observe that the attributes for the Menu relation include the Menu identification, the menu type, and the menu description; the dish relation has the attributes which include the dish identification, the dish name and the preparation time as attributes, with the Dish ID being the primary key attribute of the dish relation. From the closely related dish ingredient entity, this entity has the attributes of Dish ID and ingredient ID that are used to make a dish, which in itself has the foreign key attributes that refer to the Dish and Ingredient relations that have been discussed, both of which provide specific information relevant for making different dishes that are displayed as different Menu options. From the Menu Dish relation, we observe that this relation has the attributes Menu ID and Dish ID, which serve as foreign key attributes referring to Menu and Dish relations. From these relations, we know that the menu and dish identification are important as they are foreign key attributes, which in themselves are the foreign key attributes for the menu and dish relations.

From the Event relation, we also know that this relation has attributes of the Event ID, Event Date, Event Location, Event Time and Menu ID. From the Staff relation, we know that this relation has the attributes Employee ID, Employee Name, Employee Salary, and Supervisor ID. From the Staff relation, the Employee ID is the primary key attribute, whereas the Staff relation, we know that this relation has the attributes Employee ID and Skill, and these attributes also serve as the primary key attributes. Finally, the Work Schedule relation has the attributes Event ID, Employee ID, Position, Start Time and End Time, and from these attributes, the Event ID and Employee ID are foreign key attributes. These foreign key attributes refer to the Staff and Menu relations that we have previously discussed to obtain any relevant information, whether it relates to the ID of an item that is included on the menu, or even the identification and other relevant information of the staff that is operating at the dining hall venue.

2.2 Part B: Diagramming the Functional Dependencies and Determining Normal Forms

From the relational schema that we have provided in the previous part, we know that the dependency illustrated below demonstrates contains between attributes. Moreover, from the relational schema we know that the functional dependency for the Menu relation can be represented with a series of arrows, starting from the Menu ID, which then goes onto the Menu Type, Menu Description. The functional dependency is directed from the Menu Relation onto the Menu Type, Menu Description because the Menu Type and Menu Description are not only functionally dependent on the Menu ID, but also because the Menu Relation is the Third Normal Form. From lectures, we know that the third normal form is a sufficient means to capture the key characteristics of the original relations that were depicted, in addition to how the relations functionally depend on each other. For the dish relation, we would expect a similar functional dependency from previous remarks and the relational schema, in which for the Dish ID, it is evident that the functional dependency would be directed away from the DishID onto the Dish Name, and preparation time of the dish. From the Menu which has the attributes of identifying specific objects that are offered from the dining service, we know that the Dish Name, and Preparation time of the dish are functionally dependent on the Dish ID. The Dish relation is also a third normal form.

For the Event relation, we know that this functional dependency can be represented by observing that for a given Event, the ID of that event has functionally dependent attributes including the Event ID, the Event Date, the Event Location, the Event Time, and the Menu ID. The Event relation is also in the third normal form. For the Work Schedule relation, we know that the Event ID and Employee ID has the functionally dependent attributes which include the Position, Starting Time, and Ending Time; the Work Schedule relation is also in the third normal form. Finally, for the Staff relation we observe that the functionally dependent attributes on the Employee ID include the Name, Salary and Supervisor ID, and that as with all previous relations, the Staff relation is in the third normal form.

2.3 Part C: Identifying the Normal Form

For the Third Normal Form, we know that a requirement for a third normal form includes that all nontrivial functional dependencies must be completely dependent on the primary key, as well as any non-key attributes. From our discussion in the previous part about the relations in the EER and their functional dependencies, all of which are neither partial nor transitive. Also, we know that these relations do not have any repeated attributes, and because the relations do not have any transitive dependencies either, each one of the relations that we have discussed is already in the third normal form.