Project Deliverables:

October 8, 2018: Groups are formed and Project Proposal **COMPLETED**
By: Xinyu Huang & Shi Li
October 24, 2018: Systems Analysis and E-R Model **COMPLETED**
By: Xinyu Huang, Shi Li
November 7, 2018: Logical and Physical Modeling **COMPLETED**
By: Chanhee Kang, Jessie Han
November 21, 2018: Physical Database Implementation (using SQL) **COMPLETED**
By: Jessie Han, Donna
December 19, 2018: Final Project Report
By: Donna, Chanhee Kang

Members:

Xinyu Huang	xinyu.huang@baruchmail.cuny.edu
Shi Li	shi.li2@baruchmail.cuny.edu
Donna Tang	donna.tang@baruchmail.cuny.edu
Jessie Han	jiahui.han@baruchmail.cuny.edu
Chanhee Kang	chanhee.kang@baruchmail.cuny.edu

Resources:

Group Project Class Website
Sample Database Project
Group Meeting Log Sheet

Proposal

First, groups should submit for approval of their project ideas. This proposal should include:

- A separate cover page indicating the title of your project, the full names of the group members (with e-mail), the course number and course section.
- A narrative description of the business used for the project or application being created. This should also include a description of the problem or opportunity being addressed.
- Identification of the information needs - what information would help solve the problem or allow one to take advantage of the opportunity.
- The initial list of entities (tables) that have been identified. This should come naturally from the above discussions.
- Distribution of duties for the project. List the names of each group member and what their primary role will be (e.g., systems analyst, application developer, documentation writer).

ABC company is known for its production of various air conditioners. However, because of their lack of management the company would often misplace or lose raw materials and ebem finished products. This resulting in the CEO to have to reorder many raw materials and the cost of the company increases. The company is currently having trouble breaking even due to these extra costs. Therefore, the CEO wishes to use a database to better manage their items thus cutting costs and waste.

Each of their AC **product** consists of different **parts.** These parts are made of different **raw materials**. Raw materials were purchased from **suppliers** and every **order list** containing raw materials, date, purchase price, etc., were recorded by the company.

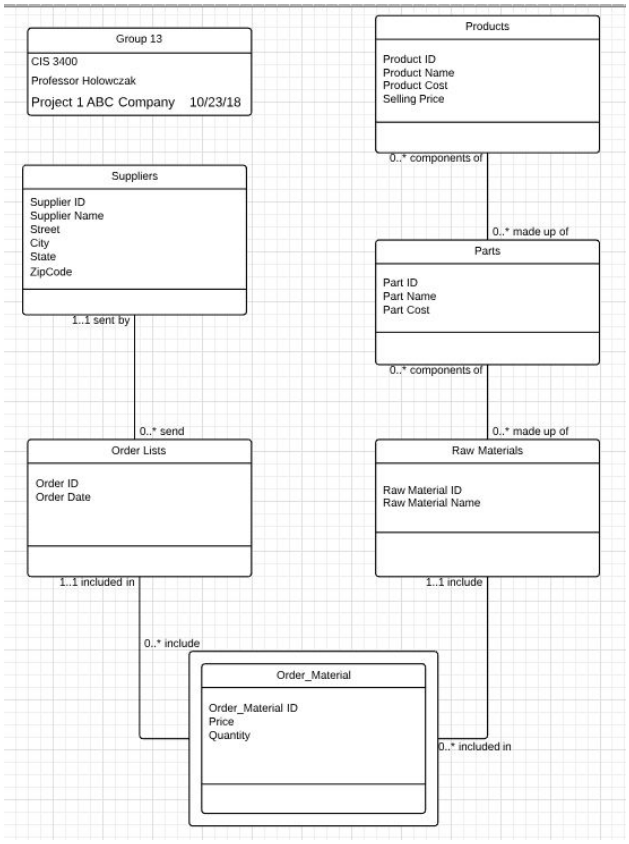
We are provided with all the different products, parts, raw materials, and supplier information. We would like to know the exact number of each raw material and parts used, sold, and ordered. Also, the number of each item's waste/lost every month to better solve the problem.

Entities:
PRODUCTS
PARTS
ORDER LIST
SUPPLIERS
RAW MATERIALS

System Analysis

The next step is for the group to take the requirements from the "users" and draw an Entity Relationship diagram using UML notation.
The E-R Diagram should then be submitted to the professor for approval before proceeding.
Students may use a specific E-R modeling tool such as MS Visio, LucidChart, drawing tools available in MS Word or MS Powerpoint, or in very neat pen or pencil.
Note: The "Relationship View" in MS Access *is not an E-R modeling tool*

Our E-R Diagram



E-R Model

---------- Logical & Physical Modeling ----------

Given the E-R diagram and sets of attributes for each entity, the next step is to convert the E-R model into a relational model and go through the process of normalization. This step will require the group to list all of the functional dependencies.
The normalized relations should be approved by the professor before proceeding.

Our Logical Model

---------- Physical Database Implementation (using SQL) ----------

I. Database Implementation

Groups should then implement the database tables from the normalized set of relations created in the previous step.

- For each normalized relation, write a SQL CREATE TABLE statement. Write separate ALTER TABLE statements to add PRIMARY KEY and FOREIGN KEY constraints to the tables.
- Data should be supplied for each table by writing SQL INSERT statements. The amount of data should be such that the need for a database is clear. In other words, provide enough examples to demonstrate why a database was required in the first place.

```
CREATE TABLE Suppliers
(  SupplierID     VARCHAR(5)     NOT NULL,
   SupplierName   VARCHAR(50),
   Street         VARCHAR(35),
   City           VARCHAR(30),
   State          VARCHAR(2),
   ZipCode        VARCHAR(12)
)

CREATE TABLE OrderList
(  OrderID        VARCHAR(5)    NOT NULL,
   OrderDate      DATE,
   SupplierID     VARCHAR(5)    NOT NULL
)

CREATE TABLE RawMaterials
(  RawMaterialID     VARCHAR(5)    NOT NULL,
   RawMaterialName   VARCHAR(50)
)

CREATE TABLE Order_Material
(  Order_MaterialID    VARCHAR(5)    NOT NULL,
   OrderID             VARCHAR(5)    NOT NULL,
   Price               NUMBER,
   Quantity            INTEGER       NOT NULL,
   RawMaterialID       VARCHAR(5)    NOT NULL
)
```

```
CREATE TABLE Parts
(  PartID         VARCHAR(5)     NOT NULL,
   PartName       VARCHAR(20),
   PartCost       NUMBER         NOT NULL
)

CREATE TABLE Parts_RawMaterials
(  PartID          VARCHAR(5)     NOT NULL,
   RawMaterialID   VARCHAR(5)     NOT NULL
)

CREATE TABLE Products
(  ProductID       VARCHAR(5)     NOT NULL,
   ProductName     VARCHAR(30),
   ProductCost     NUMBER         NOT NULL,
   SellingPrice    NUMBER         NOT NULL
)

CREATE TABLE Products_Parts
(  ProductID       VARCHAR(5)     NOT NULL,
   PartID          VARCHAR(5)     NOT NULL
)
```

-------ADD PRIMARY KEY
ALTER TABLE order_material
ADD PRIMARY KEY (order_materialID,order_id,raw_material_id);

ALTER TABLE orderlist
ADD PRIMARY KEY (orderID);

ALTER TABLE parts
ADD PRIMARY KEY (partID);

ALTER TABLE products
ADD PRIMARY KEY (productID);

ALTER TABLE parts_RawMaterials
ADD PRIMARY KEY (partID, rawMaterialID);

ALTER TABLE products_parts
ADD PRIMARY KEY (partID, productID);

ALTER TABLE rawMaterials
ADD PRIMARY KEY (rawMaterialID);

ALTER TABLE suppliers
ADD PRIMARY KEY (supplierID);

--------ADD FOREIGN KEY
ALTER TABLE order_material
ADD FOREIGN KEY (orderID) REFERENCES orderlist(orderID);

ALTER TABLE order_material
ADD FOREIGN KEY (rawMaterialID) REFERENCES rawMaterials(rawMaterialID);

ALTER TABLE orderlist
ADD FOREIGN KEY (supplierID) REFERENCES suppliers(supplierID);

ALTER TABLE parts_rawMaterials
ADD FOREIGN KEY (partID) REFERENCES parts(partID);

ALTER TABLE parts_rawMaterials
ADD FOREIGN KEY (rawMaterialID) REFERENCES rawMaterials(rawMaterialID);

ALTER TABLE products_parts
ADD FOREIGN KEY (productID) REFERENCES products(productID);

ALTER TABLE products_parts
ADD FOREIGN KEY (partID) REFERENCES parts(partID);

Link to Sample Data excel file

II. Application Implementation

The application (forms, reports, queries, menus or navigation form) can then be created on top of the tables. In general, a simple data entry form should be created for each table.

- However for the core business processes the group should create appropriate master/ detail/lookup forms that guide the user through carrying out a business process. For example, if the business takes orders from customers, I am expecting a form with Orders and Order items with lookups for customer and products (or services).
- At least two reports that reflect the core of the business should also be created.
- For Queries, provide the associated SQL statements and a description of what the queries are used for.
- For Access 2010 and later create a Navigation Form that provides a starting place with access to all of the forms and reports in logical groupings.

---------- Final Report ----------

The final step is to prepare a formal report and brief presentation. This report should include:

- A separate cover page indicating the GROUP Number, the title of your project, the full names of the group members (with e-mail), the course number and course section.
- An introduction section similar to the proposal.
- Entity Relationship Model diagram.

- The collection of normalized relations and functional dependencies, and a brief discussion as to the normal form(s) achieved, the methods used to achieve these normal forms, and reasons why any de-normalization was done.
- The SQL DDL used to create the tables and add primary key and foreign key constraints. You may either add the Primary Key constraint to the CREATE TABLE statement or add it using a separate ALTER TABLE statement. Foreign Keys should be added with a separate ALTER TABLE statement.
- An example printout of each of the forms, reports and queries accompanied by a description of the function of each. Along with each form, include any VBA code that was written to embellish the form.
- A picture of the Navigation form showing the organization of the different Forms and reports in the application.
- A narrative conclusion section that describes:
- a) the group's experience with the project (which steps were the most difficult? Which were the easiest ? what did you learn that you did not imagine you would have? if you had to do it all over again, what would you have done differently?)
- b) if the proposed benefits can be realized by the new system
- c) any final comments and conclusions

Narrative Conclusion:

The project overall was a learning experience for all of us. It helped us learn to apply the concepts that we learned in class. The steps that were the most difficult to us was the systems analysis step (when we had to create the E-R model and think about the relationships; whether it was one-to-many or many-to-many) and the application implementation step (writing the VBA code to automatically generate a new Supplier ID was the challenging). The easiest steps were when we did the logical and physical modeling and also the database implementation. What we learned that we didn't imagine that we would have was learning to create forms and using VBA code to customize and make it easier for users to use.

If we were to do it all over again, we would change the proposal so our E-R model would have more one-to-many relationships rather than many-to-many relationships because it would be less confusing. When we were going through the process of normalization, the many-to-many relationships required us to create several intersection relationships. This made it difficult for us to keep track of when we were creating the forms. For instance, we created a form that combined the products and their component parts together. We had to combine the products table, parts table, and raw materials table together.

Overall, this project was interesting and allowed to practice concepts that we learned throughout this course. Though the last step (application implementation) was quite challenging, we learned from it and we're glad that we got to experience this.

---

intro :

Our company named, ABC company, is known for its production of various air conditioners. We had trouble with misplace or lose raw materials and even finished because of our lack of management system. Meanwhile, the CEO to have to reorder many raw materials and the cost of the company increases.

Since, the company is currently having trouble breaking even due to these extra costs, the CEO wishes to use a database to better manage their items thus cutting costs and waste.

Each of
their AC **product** consists of different **parts**. These parts are made of different **raw materials**. Raw materials were purchased from **suppliers**
and every **order list** containing raw materials, date, purchase price,
etc., were recorded by the company.
Ultimately, we are provided with all the different products, parts, raw materials, and
supplier information. We would like to know the exact number of each raw material and parts used, sold, and ordered. Moreover, the number of each item's waste/lost every month to better solve the problem.

Convert to Relational Model

Here we'll convert the Entity Relationship diagram from the last step into a Relational Model. In this step, Identifiers in the Entities become Keys in the Relations. For each One-to-Many relationship, a foreign key is copied from the one side to the many side. For each Many-to-Many relationship, a new intersection relation is created to pertain the combination of these two entities.

Initial Set of Relations:

Suppliers (Supplier ID(key), Supplier Name, Street, City, State, ZipCode)

Order List (Order ID(key), Order Date, Supplier ID(fk))

Raw Materials (Raw Material ID(key), Raw Material Name)

Order_Material (Order_Material ID(key), Order ID(fk)(key), Price, Quantity, Raw Material ID(fk)
)

Parts (Part ID(key), Part Name, Part Cost)

Parts_Raw Materials (Part ID(key)(fk), Raw Materials ID (key)(fk))

Products (Product ID(key), Product Name, Product Cost, Selling Price)

Products_Parts(Product ID(key)(fk), Part ID(key)(fk))

Commentary

- Keys are shown with the (key) designation and foreign keys are shown with the (fk) designation.
- The **Order_Material** Entity is ID-Dependent. It needs an attribute in addition to the Order_Material ID to form a composite key.
- The **Product_Parts** and **Parts_Raw Materials** are intersection relationships newly created for many-to-many relationships.

Normalization

Suppliers (Supplier ID(key), Supplier Name, Street, City, State, ZipCode)

Key: Supplier ID

FD1: Supplier ID -> Supplier Name, Street, City, State, ZipCode

FD2: Zipcode -> City, State,

2NF: There is a transitive dependency in FD2

Solution

Split the relation Suppliers into two relations: ZipCodes and SuppliersInfo.

COPY ZipCode and REMOVE State, City.

New Relations

ZipCodes (ZipCode(key), City, State)

Key: ZipCode

FD1: ZipCode -> City, State

3NF: No transitive dependencies

SuppliersInfo (Supplier ID(key), Supplier Name, Street, ZipCode(fk))

Key: Supplier ID

FD1: Supplier ID -> Supplier Name, Street, ZipCode

3NF: No transitive dependencies

Order List (Order ID(key), Order Date, Supplier ID(fk))

Key: Order ID

FD1: Order ID -> Date, Supplier ID

3NF: No transitive dependencies

Raw Materials (Raw Material ID(key), Raw Material Name)

Key: Raw Material ID

FD1: Raw Material ID -> Raw Material Name

3NF: No transitive dependencies

Order_Material (Order_Material ID(key), Order ID(key)(fk), Price, Quantity, Raw Material ID(fk))

Key: Order_Material ID, Order ID

FD1: Order_Material ID, Order ID -> Price, Quantity, Raw Material ID
3NF: No transitive dependencies

Parts (Part ID(key), Part Name, Part Cost)
Key: Part ID
FD1: Part ID -> Part Name, Part Cost
3NF: No transitive dependencies

Parts_Raw Materials (Part ID(key)(fk), Raw Materials ID(key)(fk))
Key: Part ID, Raw Materials ID
Relation Parts_Raw Materials: ALL KEYS, therefore it meets all normal forms

Products (Product ID(key), Product Name, Product Cost, Selling Price)
Key: Product ID
FD1: Product ID -> Product Name, Product Cost, Selling Price
3NF: No transitive dependencies

Products_Parts(Product ID(key)(fk), Part ID(key)(fk))
Key: Product ID, Part ID
Relation Products_Parts: ALL KEYS, therefore it meets all normal forms

De-normalization

For this small database, we make the decision to de-normalize **ZipCodes** relation back into the original **Suppliers** relation to avoid data redundancy and achieve a performance improvement.

Final Set of Relations:

Suppliers (Supplier ID(key), Supplier Name, Street, City, State, ZipCode)
Order List (Order ID(key), Order Date, Supplier ID(fk))
Raw Materials (Raw Material ID(key), Raw Material Name)
Order_Material (Order_Material ID(key), Order ID(fk)(key), Price, Quantity, Raw Material ID(fk)
)

Parts (Part ID(key), Part Name, Part Cost)

Parts_Raw Materials (Part ID(key)(fk), Raw Materials ID (key)(fk))

Products (Product ID(key), Product Name, Product Cost, Selling Price)

Products_Parts(Product ID(key)(fk), Part ID(key)(fk))