**Group Project - Online Sales**

For this group project, we have selected a real-world application: a database system that manages the information associated with an online business. Just like any online business, this one offers products to its customers electronically. Customers can purchase and pay for a product through an online storefront at which point the order is fulfilled and shipped to the customer. The database serves as a tool that will organize all business information and automate many processes. In designing the structure of the database, we strived to implement separation of concerns to provide a simple yet complete picture of all relevant information.

Let’s begin examining the structure of the database from an intuitive starting point: the customer’s point of view. A “Customer” is represented as a single entity in the database. All relevant customer information is stored in the customer table: name, phone number, birthday, and email. While one customer can refer multiple new customers to our company, a customer can only be referred by one other customer. In order to document this relationship, the “Referral” entity was created wherein each entry maps a referring customer to the referred customer. A customer can have zero or one entries where they are the referred customer, but zero or many where they are the referring customer.

When a customer completes one or more order(s) on the online storefront, an invoice for the order(s) is/are generated. Thus, a customer can have multiple invoices where an “Invoice” is represented as an entity in our database. Each table entry of an invoice contains a foreign key to the customer who placed the order, the date of purchase, total cost, and the customer address. Each invoice has one or more items sold where “Item Sold” is represented as an entity in the database. Each entry of an item sold contains the quantity of the item sold, the total cost of one or more item(s) sold, and a foreign key: product\_id. A “Product” is represented as an entity in the database with product\_id as its primary key. Each entry in the product table contains the name, description, cost, and inventory of the product.

“Purchase Order” is represented as an entity in the database. Each purchase order entry contains a foreign key: supplier\_ID, purchase order cost, and customer name/address. Each purchase order contains multiple items purchased where “Items Purchased” is an entity in the database. Each entry of items purchased references its associated purchase order and the supply it came from with foreign keys and contains fields for the quantity and cost of the item purchased.

“Supplier” is represented as an entity in the database for which supplier\_ID is the primary key. Each entry in the supplier table contains supplier information (name, phone, company name, and email). “Supply” is represented as an entity in the database for which supply\_ID is the primary key. Each entry in the supply table contains supply information (name, description, cost, and inventory) and a preferred supplier in the form of a foreign key to the supplier table. In order to keep track of the additional suppliers who will produce a product, a database specific entity “Source” is used. Each source entry simply maps one supplier to a supply. This table is used to find an alternate supplier when the preferred supplier cannot satisfy our demand.

Both customers and suppliers can have one or more instances of an address where “Address” is an entity in the database. Each entry in the address table contains relevant address information.

“Note” is represented as an entity in the database. Any entity whose primary key consists of a single integer attribute can have an attached note, as long as it’s type is documented in the entity type list. The entity type list is a map of the numbers 0-255 to a table with that property. For example the invoice, customer, and supplier types are referenced by 0, 1, and 2. This system allows for notes to be attached to more types of entities in the future, as the need arises. For example, if the shipping department wants to be able to add delivery instructions to an address, no changes need to be made to the database structure. The address entity would be assigned a type ID of 3 and employee software would be able to attach notes using that new type. Each entity type can have one or more notes (up to 255) attached as the primary key of a note consists of the entity type, the entity id (the primary key of the referenced entity), and a note number (an enumerator).