**1. Requirement Analysis**

**1.1 Database Description**

*Project Purpose*

The designed application has for objective the data management of a Montreal Book and Toy Store called Kidlink. The owners of the store have agreed to guide us in creating an adequate database, which will be designed to replace an obsolete information storage system that is currently used within the store.

The purpose of this document is to present a detailed and complete project description of the database that will be designed. This includes database entities such as the books and toys, and the relationships linking them. The completed database will assist the store’s owners by allowing them to review all logged purchases. Information relevant to sales, such as the most popular books and toys, is also a focus of the project.

*Scope*

The software will be designed to replace the existing MSDOS framework currently being used by store administration. A database framework will be used as a replacement. The database will be implemented using the DB2/PostgreSQL system. It will be populated with all required tables and relational logic within an infrastructure that is well suited for managing purchases, sales and inventory. A user friendly interface will be implemented to streamline interaction between the database and the end user. Due to the store containing no employees other than the store owners, no login system is necessary, therefore only one end user exists, and with full privileges.

**1.2 Database Description**

***Entities and Attributes***

All entities stored within the database are described below, as well as their attributes.

**Transaction:** A transaction is an entity that contains Items. There are two types of transactions: sales and orders. A sale will reduce item stock while orders will increase item stocks. It has the attributes date, and unique primary key, tID.

**Sale:** A sale is a type of transaction and is used to model all the items that are being sold to a particular customer. When a sale occurs, each item’s stock attribute decreases. The sale entity has attributes price and payment type. It also inherits all the attributes of the Transaction entity.

**Card:** A card transaction happens when a customer pays for items using a credit or debit card. This entity has attributes type, number, expiration, name, and a unique primary attribute, cID.

**Order:** An order is a type of transaction and is used to request items that need to be ordered from a supplier. When a supplier fulfills an order, each item’s stock attribute increases. It has the attributes cost and status. It also inherits all the attributes of the Transaction entity.

**Items:** An Item is an entity which is contained in a transaction. An Item entity is used to store the attributes name, price, section, and stock amount of the item. It can be identified by a unique primary key called iID.

**Supplier:** A supplier is an entity that fulfill an order by supplying Items. It has the attributes city, name, and address.It also has a unique primary key, sid.

**Contact:** A contact is used by the store owners to communicate with a supplier. It has the attributes name and unique primary key, name.

**Book:** A book is a type of Item that can be sold. It has the attributes ISBN, Publisher, and author. It also inherits all the attributes of the Item entity

**Toy:** A toy is a type of Item that can be sold. It has the attributes age and manufacturer. It also inherits all the attributes of the Item entity.

**Gift Certificate:** A gift certificate is a type of Item that can be sold and used later on by the customer to purchase another item. It has one attribute called the expiration date. It also inherits all the attributes of the Item entity.

***Relationships***

Relationships connect several entities in a specific way. Below are described all the relationships between the entities of the project.

**Fills an**: An *Supplier* fills an *Order*. This is a one-to-many relationship. Order are used to restock the store on items in the store, and are linked to a single Supplier. A Supplier can, however, have many Orders currently in progress.

**Supplies**: An *Order* supplies an *Item*. This is a many-to-many relationship, because a single Order can bring to the store a variety of different Items. Likewise, a specific Item can be supplied from more than one Order.

**Includes**: A *Transaction* includes an *Item*. This is a many-to-many relationship. A Transaction such as a client buying from the store can include more than one type of Item. Similarly, a specific type of item can be part of more than one transaction. It is important to note what is dealt with is the *type* of items, not the number of items. The number is stored separately as an attribute for this relationship.

**Paid by**: A *Transaction* is paid by a *Card*. This is a one-to-many relationship. Any Transaction can be paid by card, but only one card is used. However, a Card can be used to pay for as many Transactions as the client wishes.

**Reduces Price of**: A *Gift Certificate* reduces the price of a *Sale*. This is a one-to-many relationship, because once used, the Gift Certificate only applies to one Sale. However, said Sale can be affected by many Gift Certificates.

**Has**: A *Supplier* has a *Contact*. This is a one-to-one relationship: every Supplier has only one person that can be contacted in case of issues with the supplied items. Similarly, any contact is only affiliated with one Supplier.

**1.3 Application Description**

*Overview*

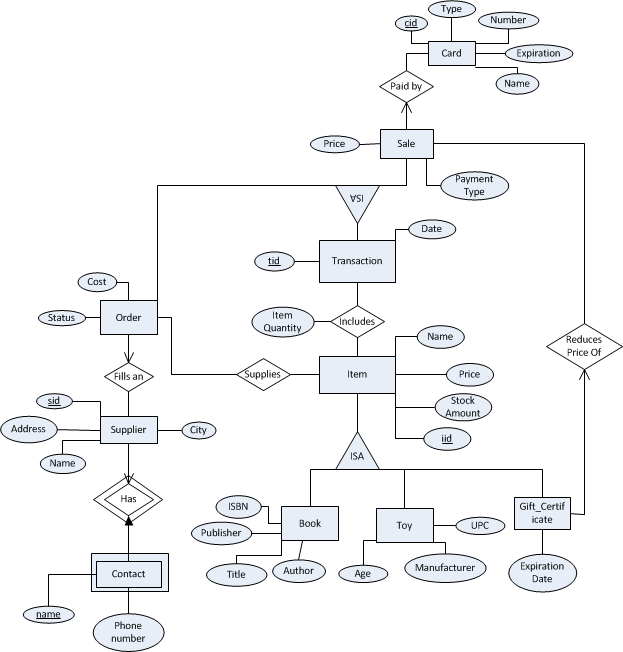
The application runs by compiling all the information currently stored in the database. In addition, certain temporary statistics are to be stored outside of the database, on the software side. One such example is the number of times a specific item was sold during the current month.

What is sought by the application is first and foremost the creation and updating of a log. Said log’s purpose is to store all incoming and outgoing purchases, ie. resupplies from suppliers and sales to clients. The store owners can refer to this log to observe the history of their store, but the log in itself is expected to be large and detailed; too difficult to use to get suggestions for managing the store. Therefore, the details behind the purchases, such as the involved items and their cost, is then used to generate basic sales data. Examples of this include computing and displaying the most popular books, toys, or other items, the most profitable items with respect to their current costs, and other such information. This data will be designed to be simple, intended to be periodically checked by the store owners in order to analyze possible management decisions.

*Computations Required*

The applications only requires basic computations in order to generate the sales data. The most important information that can be generated is the profits. For every item available in the store, the profit is defined as the price the client paid minus the cost paid to the supplier. The sum of this value for every recorded sold instance of an item defines the overall profit for that type of item. This value will be for a certain period of time, such as a month, in order to help the owners see the profits as time progresses. An important intricacy of this value is that the prices of buying/selling an item can change.

**2. Entity/Relation Diagram**



**3. Relations**

**Entities**

Transaction(tID, date)

Sale(tID, price, payment\_type)

Card(cid, type, number, expiration, name)

Order(tID, cost, status)

Supplier(sid, address, name, city)

Contact(name, phone\_number)

Item(iID, name, price, stock amount)

Book(iID, ISBN, publisher, author);

Toy(iID, age, manufacturer)

Gift\_Certificate(iID, expiration\_date)

**Relationships**

Fills\_an(tID, sID)

Supplies(tID, iID)

Contains(tID, iID, item\_quantity)

Paid\_by(tID, cID)

Reduces\_Price\_of(iID, tID)

We do not believe there is a way of combining entities without introducing ambiguity and redundancy. Combining Toys and Books, for example, may simplify the design somewhat, but it would introduce new difficulties because of several important attributes that are unique to either entity. The Age entity for Toys, for example, is important because it needs to be made obvious by law, something which is not applicable for books, and therefore these two entities are much better left separate.