Web Store Database

Daniel Medina (CS-Traditional), Justin Ramirez(CS-Traditional)

CS342, Fall 2015
Dr. Wang

Table of Contents

Phase I

1.1 - Fact-Finding Techniques and Information Gathering

In phase I section 1.1 we will introduce our enterprise and discuss different topics related to the business. In section 1.1.1 we will introduce the concept of our enterprise and describe what our business does and the activities that it participates in. Then in section 1.1.2 we discuss fact-finding and data reports. Section 1.1.3 discusses what part of our enterprise the database represents as well as a brief description of the major entity and relationship sets. Finally in section 1.1.4 we give a brief itemized description of the entity and relationship sets within the database.

1.1.1 - Introduction to Enterprise/Organization

Electronic retail or Web stores are business's that make available either general or specialized product and then deliver those product to customers or in cases where a physical store is involved, make those products available to the customer if not in the physical store. Those hired by a web store usually consist of a smaller amount of employees, dependent on the business's size, that process and ship customer orders. Certain job's can be added, like accounting or other aspects of the business unrelated to the website specific database, or retracted given the size of the business; our database is being made for a moderately sized web store.

Structure of A Web Store

The structure of most web stores usually consists of a database that houses all data associated with products, employees, and customers. The database also keeps track and reports information like top viewed categories, best sellers, and most searched items. Most merchants purchase either free or licensed Frontend software or software from a hosted

service provider. For the requirements of our database project we are developing both the Backend and the Frontend software.

1.1.2 - Fact-Finding Techniques

Each online retailer has its own quirks and conduct of business, just like most businesses. With the wealth of information on the Internet, however, we can use Wikipedia and other sources to draw information from.

Frontend and GUI of a many web stores to gather much information and to peak into how, both generally and specifically, most web store's conduct their business and run their websites. The merchant in charge of the website and the employees who process orders are the only people who can change or add data to the database and only the merchant can change or add things like items and categories. Reports generated by the database will generally consist of: top selling item, most searched category, most viewed item, most viewed item from a certain sex and other queries similar to that. We can study these reports to see how well certain items are doing and what direction the business should take in a given economic climate. We could also interview the merchant and employs of the web store to find out the normal conduct of business and to help us form our conceptual database design. We could also send out surveys to the company to interview a larger amount of people related to the business at once.

1.1.3 - Structure of Enterprise

The conceptual database we are developing covers: transaction between customer and merchant, employee order processing and fulfillment, and full inventory of items purchased from distributors. The major entity sets of our database consist of customer, order and item. The customer entity consists of information to verify customer's access to their associated accounts and to ensure successful fulfillment of order; additional customer information is provided by the credit card entity. The order entity contains items purchased by the customer as well as information regarding shipment of the item. The employee entity provides more information regarding who and when processed the order. Finally the item entity contains all item products within the database and information associated with all

items. The item entity has three subclasses: music movies and games. These subclasses specialize each item giving them a more in depth description. The distributor entity gives us information on where items are purchased as well as information regarding the distributor itself. The major relationships of our database are between our primary entity types: customer places order and order contains items.

1.1.4 - Major entities and relationships

Our database contains the entities: employee, order, item, customer, credit card, and distributor. All entities within the database are strong entities. The relationships between these entities are: employee processes order, customer purchase order, customer owns credit card, order charges credit card, order contains items, and distributor supplies items. The item entity is divided into the movie, music and games subclass. In section 1.2 we take a much more in-depth look into each entity and relationship set as well as their related attributes.

1.2 – Conceptual Database Design

The following sections give detailed information on the entities, and their relationships with one another. A name, description, domain/type, value-range, default value, null-ability, uniqueness, single/multiple value information and simple or composite information are provided for each entity. Relationship sets are defined by a name, description, entity set involved, mapping cardinality, the descriptive field and participation constraint. Lastly, this section explains the specialization/generalization process between the entities.

1.2.1 – Entity Set Description

Customer

Description: This entity stores information such as customer ID, name, address, email address and password information for customers and will only be created once they have created an account on the website. Name, address, email address and phone number will be updated as needed. Customers will also be associated with one or more credit cards, which will be used to pay for the orders they make.

This is a strong entity.

Candidate Keys: Customer ID (primary key), Name

Indexed Fields: Customer ID, Address

Attribute Name	Description	Domain / Type	Value / Range	Default Value	Null-able	Unique	Single/ Multiple	Single/ Composite
cID	ID number, for internal use	Integer	0 to Max Integer	Max custID +1	No	Yes	Single	Single
Name	First, Middle & Last names	3 Strings	Any	None	No	No	Single	Composite
ShippingAddress	Street address, City, State, Zip code	5 Strings	Any	None	No	No	Single	Composite
Phone	10 digit phone number (3 area code 7 phone number)	2 Integers	AreaCode 3 Digits Number 7 Digits	None	No	No	Single	Composite
Gender	Male or Female	String	Male or Female	None	No	No	Single	Single
BirthDate	Date of Birth	Date	Date-Type Range	None	No	No	Single	Composite
EmailAddress	Contact Email/ Login	String	Any	None	No	Yes	Single	Single
Password	Login password	String	Any	None	No	No	Single	Single

Employee

Description: This entity contains information on the employees working for the enterprise such as: Name, start date, end date, address and phone number. An instance will be created whenever an employee is hired and would not be deleted, in order to keep track of past employees.

Strong entity

Candidate Keys: eID (primary Key), name

Indexed Fields: eID, name

Attribute Name	Description	Domain/ Type	Value / Range	Default Value	Nullable	Unique	Single/ Multiple	Single/ Composite
EmployeeId	ID number, for internal use	Integer	0 to Max Integer	Max empID +1	No	Yes	Single	Single
Name	First, Middle & Last names	3 Strings	Any	None	No	No	Single	Composite
Address	Street address, City, State, Zip code	5 Strings	Any	None	No	No	Single	Composite
PhoneNumber	10 digit phone number (3 area code 7 phone number)	2 Integers	AreaCode 3 Digits Number 7 Digits	None	No	No	Single	Composite
BirthDate	Date of Birth	Date	Date-Type Range	None	No	No	Single	Composite
StartDate	Date of Hire	Date	Date-Type Range	None	No	No	Single	Composite
EndDate	Date of Fire	Date	Date-Type Range	None	Yes	No	Single	Composite

Item

Description: This entity contains information on the items that will be sold on the enterprise website such as: the item number, name of the item, release/preorder date, genre and whether or not it can be returned. Items will be added as new inventory becomes available and will be removed when they become out of stock or out of print. When a customer places an order, it will form a relationship between one or more items and the order entity. The employee will then process the items and ship them to the customer. Items also belong to one of the three subclasses Music, Movie and Game.

Strong entity.

Candidate Keys: ItemID

Indexed Fields: ItemID, Title

Attribute Name	Description	Domain / Type	Value / Range	Default Value	Null-able	Unique	Single/ Multiple	Single/ Composite
ItemID	ID number, for internal use	Integer	0 to Max Integer	Max itmID +1	No	Yes	Single	Single
Title	Name of item	String	Any	None	No	No	Single	Single
Price	Cost of Item	Double	Max Double; only 2 decimals	None	No	No	Single	Single
Genre	Genre of Item	String	Any	None	No	No	Multiple	Single
Format	DVD, Blu-ray, PS4, XBOX ONE or Wii U	String	Any	None	No	No	Multiple	Single
ReturnStatus	Item can/cannot be returned	Boolean	True or False	None	No	No	Single	Single
PreorderDate	When item is available for preorder	Date	Date-Type Range	None	No	No	Single	Composite
ReleaseDate	When item is available for purchase	Date	Date-Type Range	None	No	No	Single	Composite

Subclasses of Item

Music

Description: Audio sold on either vinyl or CD format.

Weak entity.

Candidate Keys: None

Indexed Fields: None

Attribute Name	Description	Domain/ Type	Value / Range	Default Value	Null-able	Unique	Single/ Multiple	Single/ Composite
Artist	Creator of Music	String	Any	None	No	No	Single	Single
RecordLabel	Owner of Music	String	Any	None	No	No	Single	Single

Movies

Description: Video sold on either DVD or Blu-ray disk format.

Weak entity.

Candidate Keys: None

Indexed Fields: None

Attribute Name	Description	Domain/ Type	Value / Range	Default Value	Null-able	Unique	Single/ Multiple	Single/ Composite
Director	Creator of Movie	String	Any	None	No	No	Single	Single
Production Company	Owner of Movie	String	Any	None	No	No	Single	Single

Games

Description: Games sold on Xbox 360, PS3, Wii, Xbox One, PS4, Wii U or Xbox One.

Weak entity.

Candidate Keys: None

Indexed Fields: None

Attribute Name	Description	Domain/ Type	Value / Range	Default Value	Null-able	Unique	Single/ Multiple	Single/ Composite
Artist	Creator of Game	String	Any	None	No	No	Single	Single
RecordLabel	Owner of Music	String	Any	None	No	No	Single	Single

Order

Description: This entity contains information on orders placed by customers and contains information on where the order needs to be shipped and when it has been shipped. Orders are created as the customers place them. This entity would only be updated or deleted if the customer accidentally placed the order or ordered the wrong items or wrong item quantity.

This is a Strong Entity

Candidate Keys: OrderID (primary key), Address

Indexed Fields: OrderID, Address

Attribute Name	Description	Domain/ Type	Value / Range	Default Value	Null-able	Unique	Single/ Multiple	Single/ Composite
OrderID	ID number, for internal use	Integer	0 to Max Integer	Max ordID +1	No	Yes	Single	Single
Status	Order Status (shipped/ processing)	String	Shipped or Processing	Processing	No	No	Single	Single
Tracking Number	Tracking number from delivery service	String	Shipper's Format	Null	Yes	Yes	Single	Single
Shipping Method	Which delivery service was used	String	UPS, USPS, FEDEX, Etc.	None	No	No	Single	Single
DateShipped	Date of item shipment	Date	Date-Type Range	None	No	No	Single	Composite

Credit Card

Description: This entity contains customers' credit card information such as: billing name, billing address, credit card number, expiration date, security code and credit card company. This information will be used to bill the customer for the items purchased in their order. This information is very sensitive and must be protected. Information will be periodically updated at the customers will.

Strong Entity

Candidate Keys: CardNumber (primary key), CardHolder

Indexed Fields: CardNumber, CardHolder

Attribute Name	Description	Domain/ Type	Value / Range	Default Value	Null-able	Unique	Single/ Multiple	Single/ Composite
CardHolder	Card Holder Name	String	Any	None	No	Yes	Single	Composite
CardNumber	Credit Card Number	String	Credit Card format	None	No	No	Single	Single
ExpDate	Expiration Date	Date	Month and year	None	No	No	Single	Composite
SecurityCode	3-digit Security Code	Integer	3 digit integer	None	No	No	Single	Single
Type	Type of Credit Card	String	Visa, MasterCard, Discover, Amex	None	No	No	Single	Single

Distributor

Description: This entity contains contact information on the various distributors that will be supplying the store with items. Over time different distributors will be added and existing distributors will be updated. Distributors would only be deleted if they were to go out of business.

Strong Entity

Candidate Keys: DistributorID, Name

Indexed Fields: DistributorID, Name

Attribute Name	Description	Domain/ Type	Value / Range	Default Value	Null-able	Unique	Single/ Multiple	Single/ Composite
DistributorID	ID number, for internal use	Integer	0 to Max Integer	Max distID +1	No	Yes	Single	Single
Name	Distributor Name	String	Any	None	No	No	Single	Single
Address	Street address, City, State, Zip code	5 Strings	Any	None	No	No	Single	Composite
PhoneNumber	10 digit phone number (3 area code 7 phone number)	2 Integers	AreaCode 3 Digits Number 7 Digits	None	No	No	Single	Composite

1.2.2 - Relationship Set Description

Customer Owns Credit Card

Description: In order for a customer to place an order they must own a credit card that they can purchase with. Each customer can have as many credit cards associated with him or her as they want but each credit card can only be related to one customer.

Entity Sets Involved: Customer, Credit Card

Mapping Cardinality: 1…m

Descriptive Field: none

Participation Constraint: Customer has partial participation since we can have information regarding a customer without their credit card information. Credit Card has total participation since it must be associated with a customer in order to be valid.

Customer Places Order

Description: Each customer will be able to add and remove items to the cart until they wish to place their order. A customer can place as many orders as they wish but an order can only be associated with one customer. The date ordered attribute records the date the order is purchased on for customer and historical record keeping.

Entities Involved: Customer, Order

Mapping Cardinality: m…1

Descriptive Field: Date Ordered

Participation Constraint: A customer has partial participation since we can have information regarding a customer but they may not have placed an order. Order has total participation since an order cannot exist unless a customer has placed it.

Order Charges Credit Card

Description: to be able to successful bill an order there must be a credit card to charge. An order can only charge one credit card but a credit card can be charged by many different orders.

Entity Sets Involved: Order, Credit Card

Mapping Cardinality: 1…m

Descriptive Field: None

Participation Constraint: Both order and credit card have total participation. An order can only be placed when there is a credit card to charge. Likewise, credit card information will only be requested when placing at the least one order.

Employee Processes Order

Description: Employees prepare orders for shipment by reviewing at the invoice generated by the customers' orders, then gathers the items and packages them in order to be shipped.

Entities Involved: Employee, Order

Mapping Cardinality: 1…m

Descriptive Field: StartDate, EndDate

Participation Constraint: Both Employee and Order have total participation, there must be an order for the employee to fill and there must be an employee to fill the order.

Order Contains Item

Description: Orders placed by customer contain several items of their choosing; the employee will use this information to ship the order.

Entities Involved: Order, Item

Mapping Cardinality: 1…m

Descriptive Field: None.

Participation Constraint: Order requires total participation but Item requires no participation, there can be no order without an item, but there can be items with no orders.

Distributor Supplies Item

Description: The distributor supplies the store with items that will be sold to customers.

Entities Involved: Distributor, Item

Mapping Cardinality: 1…m

Descriptive Field: None.

Participation Constraint: Both Distributor and Item require total participation there can be no Items with out a distribution and with out items there is no need for a distributor.

1.2.3 - Related Entity Set

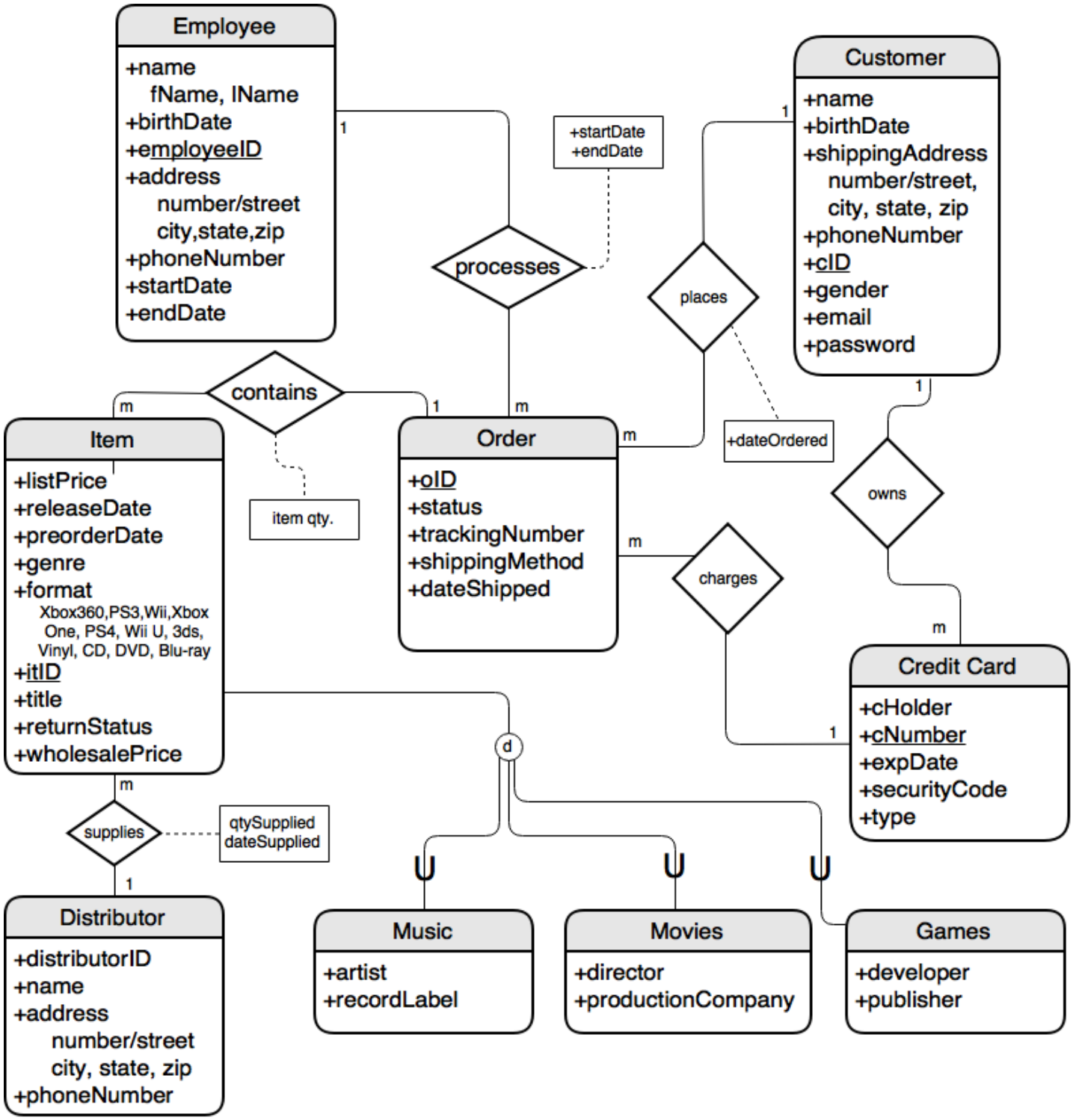
Is-a relationships

This web store contains three subclass entities, Music, Movies, and Games, that are derived from the superclass entity, Item. Item is-a Movie, Item is-a Music or Item is-a Game. The union between subclasses is a disjointed one

Has-a relationships

This web store contains two has-a relationships. Customer has-a credit card and Order has-a Item.

1.2.4 - E-R Diagram



Phase II

2.1 - E-R Model and Relational Model

In the following sections we will discuss the process of converting from the conceptual, E-R Model to the logical, Relational Model. Section 2.1.1 will describe the E-R Model and Relational Model, their histories, features, and purposes. In section 2.1.2 we will compare the two models' strengths, weaknesses, advantages and disadvantages along with their similarities and differences.

2.1.1 - Description of E-R model and Relational Model

The E-R Model, or Entity Relation Model is a high-level, conceptual data-modeling tool that is used to organize the data collected and generated by a business in order to create a database. The E-R uses the concept of Entities and Relations in order link physical goods and/or services together. Entities and relations contain descriptive attributes necessary for the business' function and information storage as well as information that may be useful to the client. E-R diagrams are often used to visualize the data exchange that takes place in the business. American computer scientist Peter Chen introduced the E-R model in 1976.

The Relational Model, which was introduced by E.F. Codd in the year 1969, employs the idea of utilizing rows and columns to form a grid-like matrix of information, organized by the attributes which business function or operation. Codd's model turns these rows and columns into tables, or relations, in which the rows form a unique value called a tuple. Another aspect of the Relational Model pertains to its use of keys; primary keys are used to order the tuples and keep uniqueness across the entire table, the primary keys from one relation can be also be used as a foreign key in another relation in order to create relationship between the two relations. The model can also be used to enforce integrity constraints on the data entered into the tables in order to structure the data properly. The Relational Model, which is based upon the fundamentals of relational mathematics, is used to create a Relational Database and can be queried using relational algebra and calculus.

2.1.2 – Comparison of E-R model and Relational Model

The E-R and Relational Models both have their strengths and weaknesses, and should be used in tandem to thoroughly map the functions and data flow within a business when creating a database. The E-R model is a higher-level modeling tool and can be easily understood by any member of the organization and models the actual, real life operations of a business more accurately and plainly than the relational model. The Relational Model, on the other hand, does not describe the data in the easy-to-understand fashion like the E-R Model; this model is generally of little use to most of the members of an organization and is only used by database administrators and programmers. Unlike the E-R Model, the Relational Model can be manipulated and queried using query languages, making it possible for implementation in a computer-based application.

2.2 Conceptual/Logical Database and Conversion to Relational Database

In phase II section 2.2 we will discuss the conceptual database and the logical database, and the conversion from the E-R model to the relational model. A conceptual database design is a high-level description of a business's information needs. A logical database design is essentially the mapping of the E-R model entities, relationships and attributes into tables using the relational model or other kinds of data models. Currently there is no method to directly implement the E-R model into a physical database so we must map the E-R model into an implantation model like the relational model. In section 2.2.1 we will discuss the conversion of different types of entities: weak/strong entity types, simple/composite entity attributes, and single/multi-valued entity attributes. In section 2.2.2 we will discuss the conversion of entity relationship types into relations such as: relationship types that are one to one, relationship types that are 1 to many, relationship types that are many to many, specialization and generalization relationship types, recursive relationship types, Category or union relationship types, and ternary and N-ary relationship types. In section 2.2.3 we will discuss database constraints like entity constraints, primary and unique key constraints, referential key restraints, check constraint, and business rules.

2.2.1 – Converting Entity Types to Relations

To convert strong entities we create a relation that includes all simple attributes of the entity. When an entity contains composite attributes we break it down into its simple attributes and include those in the relation. One or more simple attributes form a key that is used as the relation's primary key. Candidate key data can be kept for indexing purposes.

To convert weak entities we create a relation that includes all simple attributes and all simple composite attributes of the entity. The primary key of the owner entity is included in the relation as a foreign key and part of the primary key. If the relation contains a partial key then that is included as part of the primary key.

Multi-valued attributes must be converted into their own relations. To do this we will include the primary key of the entity that contained the multi-valued attribute as a foreign key in the multi-valued attribute relation. The relation will contain the multi- valued attribute as a single tuple for every value. Component will be split into simple attributes.

2.2.2 – Converting Relationship Types to Relations

Binary one to one relationship types can be converted into relations in three different ways.

a) Include the primary key of one of the two relations as a foreign key in the other. This method is good to use if one of the relations has total participation. The foreign key should be included in the relation with the highest level of participation to avoid null values that would appear in the foreign key for tuples that did not appear in the relationship. Relationship attributes will be implemented into the relation that contains the foreign key.
b) If the relations both have total participation then those relations can be merged into a single relation.
c) Both relations can be cross-referenced by creating a third relation. The new relation will include the primary keys of the relations that it cross-references as foreign keys. The combination of these foreign keys will combine to form the primary key of the new relation. The new relationship will contain any relationship attributes. This is a good method to use if both relations have low participation.

Binary one-to-many and many-to-one relationship types can be converted to relations in two different ways.

a) Implement the primary key of the one side as a foreign key in the many side. Relationship attributes will be implemented into the many side.
b) If the many side of the relationship has low participation then the relations can be cross-referenced like in a one-to-one relationship.

Binary many to many relationship types must be converted using the cross-reference method used for both many-to-one and one-to-one relationship types. The cascade option should be set on the foreign keys on update and one delete since each instance has an existence dependency on each entity it relates.

Ternary and N-ary relationship types must be cross-referenced and the primary key of each relationship included will be implement as foreign keys in the cross-referenced table.

Specialization and generalization can be converted into relations in four different ways.

a) Create a relation for each superclass and each subclass. The primary key of each superclass will be included as a foreign key into each subclass it corresponds to. This method will work for any specialization.

b) Create a relation for each subclass. Implement all the attributes of the superclass as attributes in each of the subclasses. The superclass must have total participation in each of the subclasses. This method may produce duplicates if the specialization is overlapping.
c) Create a relation with a single attribute type. This will merge each superclass and subclass into a single relation. Each subclass must be disjoint; also this method may create many Null values if the subclasses contain many attributes.
d) Create a relation with multiple attribute types. This is very similar to the previous method except each attribute is a Boolean type attribute that indicates which subclass it belongs to. This is used for specializations that have subclasses that are overlapping.

Recursive relationships can be converted into relations be including a foreign key that references the primary key of the same relation.

Category and union relationship types can be converted into relations in two ways.

a) Create a relation that contains all the attributes of every category and a new attribute called a surrogate key that is used as the primary key of the new relation. The surrogate key is included as a foreign key in each of the superclasses.
b) If each of the superclasses has the same primary key then there is no need for the surrogate key since we can use the primary key of all the superclasses. The rest is the same as the implementation above.

2.2.3 – Database Constraints

Constraints within the database define properties that data within the database must comply with. Constraints implemented in the E-R model like attribute data types and primary key uniqueness are implemented into the relational model. We need these constraints because without them attributes can contain data types that do not fit the data type of the attribute and their could be no way to ensure the uniqueness of each primary key.

The entity integrity constraint ensures that each primary key of all relations within the database is unique a not null. The uniqueness constraint ensures that a given attribute for a relation will be unique for each tuple in the relation. The referential integrity constraint ensures that any defined foreign key attribute will be null or will have a corresponding primary key attribute in another relation. All of these constraints can be implemented via corresponding settings in the database schema of the DBMS.

Business rules are another type of constraint and are very specific and defined by the business itself like, “age must be over 18”. You can implement check constraints that will check to make sure a set of conditions is satisfied in order to commit any changes to the relation.

2.3 – E-R Database to Relational Database Conversion

The following two sections detail the relation schema created by converting the data from our E-R Model into a Relational Model. Section 2.3.1 outlines each attribute, constraint, primary key and candidate key of each relation in the relational database. Section 2.3.2 contains sample instances of each relation within the logical database.

2.3.1 – Relation Schema for Local Database

Employee

Employee (eID, fname, lname, bDate, street, city, state, zip, phone, sDate, eDate)

1. eID – Domain: integer
    - Default Value: (Max ID +1)
    - Primary Key (Unique / Not NULL)
2. fName – Domain: Varchar2(25)
    - Not NULL
3. lName – Domain: Varchar2(25)
    - Not NULL
4. bDate – Domain: Date
    - Not NULL
5. street – Domain: Varchar2(45)
    - Not NULL
6. city – Domain: Varchar2(25)
    - Not NULL
7. state – Domain: Varchar2(2)
    - Not NULL
8. zip – Domain: Varchar2(5)
    - Not NULL
9. phone – Domain: Varchar2(10)
    - Not NULL
10. sDate – Domain: Date
    - Not NULL
11. eDate – Domain: Date
    - can be NULL

Customer

Customer (cID, fname, lname, bDate, street, city, state, zip, phone, gender, email, password)

1. cID – Domain: integer
    - Default Value: (Max ID +1)
    - Primary Key
    - Unique / Not NULL
2. fName – Domain: Varchar2(25)
    - Not NULL
3. lName – Domain: Varchar2(25)
    - Not NULL
4. bDate – Domain: Date
    - Not NULL
5. street – Domain: Varchar2(45)
    - Not NULL
6. city – Domain: Varchar2(25)
    - Not NULL
7. state – Domain: Varchar2(2)
    - Not NULL
8. zip – Domain: Varchar2(5)
    - Not NULL
9. phone – Domain: Varchar2(10)
    - Not NULL
10. gender – Domain: Varchar2(1)
    - Must be 'M' or 'F'
11. email – Domain: Varchar2(45)
    - Not NULL / Unique
12. password– Domain: Varchar2(12)
    - Not NULL

Candidate Keys:

1. cID (primary key)

2. email

Credit Card

CreditCard (cNumber, cHolder, expDate, securityCode, type)

1. cNumber – Domain: integer
    - Primary Key
    - Unique / Not NULL
2. cHolder– Domain: Varchar2(50)
    - Not NULL
3. expDate – Domain: Date
    - Not NULL
4. securityCode – Domain: Varchar2(3)
    - Not NULL
5. type – Domain: Varchar2(10)
    - Not NULL

Candidate Keys:

1. cNumber (primary key)

Distributor

Distributor (dID, name, street, city, state, zip, phone)

1. dID – Domain: integer
    - Default Value: (Max ID +1)
    - Primary Key
    - Unique / Not NULL
2. name – Domain: Varchar2(25)
    - Not NULL
3. street – Domain: Varchar2(45)
    - Not NULL
4. city – Domain: Varchar2(25)
    - Not NULL
5. state – Domain: Varchar2(2)
    - Not NULL
6. zip – Domain: Varchar2(5)
    - Not NULL
7. phone – Domain: Varchar2(10)
    - Not NULL

Candidate Keys:

1. dID (primary key)

Item

Item (iID, dID, title, listPrice, releaseDate, preDate, genre, format, returnStatus, wholesalePrice, imgURL)

1. itID – Domain: integer
    - Default Value: (Max ID +1)
    - Primary Key
    - Unique / Not NULL
2. dID – Domain: integer
    - Foreign Key – Distributor.dID
    - Not NULL
3. title – Domain: Varchar2(45)
    - Not NULL
4. listPrice – Domain: Number(2,5)
    - Not NULL
5. releaseDate – Domain: Date
    - Not NULL
6. preDate – Domain: Date
    - Not NULL
7. genre – Domain: Varchar2(15)
    - Not NULL
8. format – Domain: Varchar2(15)
    - Not NULL
9. ReturnStatus – Domain: Varchar2(1)
    - Not NULL
10. wholesalePrice – Domain: Number(2,5)
    - Not NULL
11. imgURL – Domain: Varchar2(25)
    - Not NULL

Candidate Keys:

1. iID (primary key)

Music

Music (iId, artist, recordLabel)

1. iID – Domain: integer
    - Foreign Key –Item.iID
    - Not NULL
2. artist – Domain: Varchar2(40)
    - Not NULL
3. recordLabel – Domain: Varchar2(25)
    - Not NULL

Movie

Movie (iId, director, productionCo)

1. iID – Domain: integer
    - Foreign Key –Item.iID
    - Not NULL
2. director– Domain: Varchar2(40)
    - Not NULL
3. productionCo – Domain: Varchar2(25)
    - Not NULL

Game

Game (iId, developer, publisher)

1. iID – Domain: integer
    - Foreign Key –Item.iID
    - Not NULL
2. developer– Domain: Varchar2(40)
    - Not NULL
3. publisher – Domain: Varchar2(25)
    - Not NULL

Order

Order(oID, eID, cID, cNo, sDate, eDate, dateOrdered, status, trackNo, shipMethod, shipDate)

1. oID – Domain: integer
    - Foreign Key – Order.oID
    - Not NULL
2. eID – Domain: integer
    - Foreign Key – Employee.eID
    - Not NULL
3. cID – Domain: integer
    - Foreign Key – Customer.cID
    - Not NULL
4. cNo – Domain: integer
    - Foreign Key – CreditCard.cNo
    - Not NULL
5. sDate – Domain: Date
    - Can Be NULL
6. eDate – Domain: Date
    - Can be NULL
7. status – Domain: Varchar2(10)
    - Not NULL
8. trackingNo – Domain: Number(25)
    - Can be NULL
9. shipMethod– Domain: Varchar2(10)
    - Not NULL
10. shipDate – Domain: Date
    - Can be NULL

Candidate Keys:

1. oID (primary key)

2. trackingNumber

Order Contains Item

OrderContainsItem (iID, oID, itemQty)

1. iID – Domain: integer
    - Foreign Key – Item.iID
    - Not NULL
2. oID – Domain: integer
    - Foreign Key – Order.oID
    - Not NULL
3. itemQty – Domain: integer
    - Not NULL

Candidate Keys:

1. iID, oID, itemQty (primary key)

2.3.2 – Sample Data of Relation

Customer

Customer (cID, fname, lname, bDate, street, city, state, zip, phone, gender, email, password)

cID	gender	fName	lName	bDate	street	city	state	zip	phoneNo.	email	password
1	F	Christine	Mccoy	6/4/1988	1840 Wayridge Court	Santa Barbara	California	93111	1-(805)884-3047	cmccoy0@jigsy.com	vtqumhb7
2	F	Mary	Lynch	3/28/1989	4377 Eagle Crest Junction	Cincinnati	Ohio	45254	1-(513)491-6055	mlynch1@illinois.edu	RKRUmp0
3	F	Nancy	Lopez	3/11/1989	8 Corry Court	Portland	Oregon	97201	1-(971)927-6466	nlopez2@qq.com	Hu1VcvO
4	F	Julia	Woods	12/24/1991	9421 Haas Center	San Jose	California	95123	1-(408)528-8993	jwoods3@rakuten.co.jp	Dhe4BgLc
5	M	Bruce	Young	8/11/1989	30646 Reindahl Street	New Orleans	Louisiana	70124	1-(985)384-2952	byoung4@behance.net	ugLCjkmmOK
6	M	Harold	Long	3/10/1991	0 Ridge Oak Pass	Houston	Texas	77299	1-(713)395-8253	hlong5@loc.gov	MiuUwQK
7	M	Shawn	Harris	7/3/1991	66 North Lane	New York City	New York	10184	1-(212)382-6964	sharris6@gizmodo.com	l2J1mEx80tAf
8	M	Andrew	Payne	5/7/1988	2792 Morning Trail	Boise	Idaho	83711	1-(208)675-1653	apayne7@bizjournals.com	6BFEPsx5Ma
9	F	Amy	Reid	12/10/1986	1 Anderson Alley	Kansas City	Kansas	66160	1-(913)253-9400	areid8@umn.edu	APZJUQnLtZ
10	M	Clarence	Griffin	12/23/1986	018 Butterfield Pass	Charlotte	North Carolina	28263	1-(704)581-7640	cgriffin9@nba.com	Pd6ezChhEVG

Credit Card

CreditCard (cNumber, cHolder, expDate, securityCode, type)

cardNo	customerID	fName	lName	expDate	securityCode	cardType
4844542949956840	1	Christine	Mccoy	12/1/2018	904	visa-electron
3550506265672170	2	Mary	Lynch	1/1/2021	234	jcb
3578745162625680	3	Nancy	Lopez	4/8/2018	372	jcb
490524148153588000	4	Julia	Woods	8/19/2019	681	switch
3533257510000310	5	Bruce	Young	7/6/2021	857	jcb
5602251631942650	6	Harold	Long	2/26/2019	507	bankcard
4017954072617	7	Shawn	Harris	9/6/2018	707	visa
5602252585509700000	8	Andrew	Payne	2/7/2021	865	china-unionpay
3579813360873430	9	Amy	Reid	8/5/2018	993	jcb
6706271357771240000	10	Clarence	Griffin	12/30/2019	428	laser

Employee

Employee (eID, fname, lname, bDate, street, city, state, zip, phone, sDate, eDate)

eID	fName	lName	birth Date	street	city	state	zip	phoneNo.	sDate	eDate
1	George	Carpenter	3/8/1992	798 Stone Corner Pass	Columbus	Ohio	43240	1-(740)177-4729	5/23/2010	8/28/2014
2	Kathy	Bishop	10/20/1991	2 Brown Place	Dayton	Ohio	45470	1-(937)383-1518	4/4/2010	9/27/2010
3	Ernest	Carter	7/24/1990	3 Karstens Hill	Raleigh	North Carolina	27605	1-(919)307-9809	10/1/2009	NULL
4	Beverly	Brown	5/28/1990	53 Melvin Terrace	Orlando	Florida	32808	1-(321)817-9978	5/4/2010	NULL
5	Alan	Greene	7/11/1991	23 Erie Point	Salt Lake City	Utah	84145	1-(801)551-9643	12/14/2008	NULL
6	Doris	Nichols	1/30/1989	99563 Scoville Drive	Sparks	Nevada	89436	1-(775)540-3055	10/11/2009	2/11/2013
7	Janice	Turner	11/20/1985	80090 5th Circle	Raleigh	North Carolina	27626	1-(919)455-6692	5/9/2010	5/7/2014
8	Martin	Cruz	1/15/1988	02557 Loftsgordon Lane	Akron	Ohio	44310	1-(330)213-9124	5/26/2010	NULL
9	Virginia	Price	6/17/1992	8741 Caliangt Park	Saint Louis	Missouri	63116	1-(314)819-9469	2/5/2008	NULL
10	Anne	Kim	9/25/1988	9726 Onsgard Center	Norfolk	Virginia	23520	1-(757)679-5201	5/22/2009	6/25/2012

Distributor

Distributor (dID, name, street, city, state, zip, phone)

dID	name	street	city	state	zip	phoneNo
1	Voonyx	475 Muir Crossing	Washington	District of Columbia	20099	1-(202)690-4435
2	Riffwire	36 Hermina Avenue	Springfield	Massachusetts	1105	1-(413)842-5648
3	Ooba	5 Oakridge Crossing	Garden Grove	California	92645	1-(714)309-4845
4	Jetwire	50224 Cambridge Pass	Austin	Texas	78783	1-(512)931-0539
5	Voonte	97012 Cascade Avenue	Houston	Texas	77090	1-(281)480-7381
6	Riffwire	63742 Dennis Hill	Sacramento	California	95813	1-(916)209-6888
7	Mita	9567 Grayhawk Plaza	Gadsden	Alabama	35905	1-(256)736-8862
8	Vimbo	229 Orin Trail	Denver	Colorado	80204	1-(303)650-6838
9	Roombo	6 Logan Crossing	New Orleans	Louisiana	70124	1-(985)791-1394
10	Aibox	7 Pierstorff Plaza	Minneapolis	Minnesota	55470	1-(612)721-1216

Item

Item (iID, dID, title, listPrice, releaseDate, preDate, genre, format, returnStatus, wholesalePrice, imgURL)

iID	dID	title	format	genre	imgURL	listPrice	wholesalePrice	releaseDate	preDate	returnStatus
15	1	The Matrix 5	Blu-Ray	Action	The-Matrix-5.jpeg	$19.99	$10.00	1/18/2018	10/24/2017	Y
27	2	Kirby: Right Back at Ya!	Blu-Ray	Adventure	Kirby-Right-Back-at-Ya!.jpeg	$19.99	$10.00	12/4/2014	NULL	Y
36	2	Madden 2015	Physical-Disc	Sports	Madden-2015.jpeg	$59.99	$40.00	6/30/2017	9/8/2016	N
38	3	The Power of Color	CD	Gospel	The-Power-of-Color.jpeg	$9.99	$6.00	9/29/2013	NULL	Y
50	2	Uncharted 7	Physical-Disc	Action	Uncharted-7.jpeg	$59.99	$40.00	7/26/2018	4/19/2016	Y
56	4	Space Adventure	Physical-Disc	Adventure	Space-Adventure.jpeg	$59.99	$40.00	12/16/2015	3/30/2014	N
57	4	Final Fantasy &	Phycical-Disc	Role-Playing Game	Final-Fantasy-&.jpeg	$59.99	$40.00	8/15/2017	5/17/2016	N
58	5	Jak and Baxter	Physical-Disc	Adventure	Jak-and-Baxter.jpeg	$49.99	$30.00	10/27/2015	NULL	Y
59	5	Dog Bones	Physical-Disc	Kids	Dog-Bones.jpeg	$49.99	$30.00	6/7/2016	1/20/2016	Y
61	1	Little Nicki	DVD	Comedy	Little-Nicki.jpeg	$9.99	$6.00	7/28/2012	NULL	Y

Music

Music (iId, artist, recordLabel)

itemID	artist	recordLabel
14	The Willies	MGM
23	Gilfoil	Sony
35	Bronson	Metal Blade
38	The Jackson 4	Metal Blade
48	The Looking Glass	Sony
56	Christopher	Grass Blue
77	Marbile	MGM
81	The Spooky Door	Grass Blue
82	Numb Brughs	Grass Blue
89	Weaboo	Orange Place

Movie

Movie (iId, director, productionCo)

itemID	director	productionCompany
15	Johnny Griffin	The Movie Place
27	Sarah Day	The Movie Place
28	James Howell	Holo
30	Lois Grant	Holo
34	Roger Ford	Holo
39	Gregory Wilson	Gassy Studios
46	Martha Holmes	Breakfast Club Productions
53	Ruth Fields	Grape Sauce
60	Katherine Moreno	Steak Parfait Studios
61	Howard Richards	Orange Soda Productions

Game

Game (iId, developer, publisher)

itemID	developer	publisher
1	BioWare	Bethesda Softworks
4	BioWare	Bethesda Softworks
36	The Looking Glass	Electronic Arts
37	Frictional Games	TriHeart
50	Frictional Games	Bethesda Softworks
56	Frictional Games	SilverSoft
57	Ubisoft	Deep Silver
58	Obsidian	Deep Silver
59	Obsidian	SilverSoft
101	Rocksockom	Bethesda Softworks

Order

Order(oID, eID, cID, cNo, sDate, eDate, dateOrdered, status, trackNo, shipMethod, shipDate)

oID	eID	cID	cNo	oDate	sDate	eDate	status	trackNo	shipMethod	shipDate
1	3	1	4844542949956840	1/9/2014	1/15/2014	3/21/2015	shipped	243020715	USPS	3/16/2015
2	3	2	3550506265672170	10/9/2015	10/13/2015	NULL	processing	NULL	NULL	NULL
3	4	3	3578745162625680	1/24/2014	2/4/2014	6/13/2015	shipped	251288355	UPS	6/7/2015
4	4	3	3578745162625680	2/1/2014	2/6/2014	6/18/2015	shipped	656235882	USPS	6/8/2014
5	5	5	3533257510000310	10/12/2014	10/18/2014	8/21/2015	shipped	55810113	FEDEX	8/4/2015
6	5	6	5602251631942650	9/23/2015	10/1/2015	NULL	processing	NULL	NULL	NULL
7	5	6	5602251631942650	7/17/2014	7/24/2014	2/25/2015	shipped	472097423	FEDEX	2/20/2015
8	8	8	5602252585509700000	7/1/2014	7/5/2014	11/21/2014	shipped	297755506	USPS	11/8/2014
9	9	9	3579813360873430	5/1/2015	5/9/2015	5/14/2015	shipped	750199078	FEDEX	5/10/2015
10	9	9	3579813360873430	10/5/2015	10/14/2015	NULL	processing	NULL	NULL	NULL

Order Contains Item

OrderContainsItem (iID, oID, itemQty)

oID	iID	itemQty
1	224	1
1	133	6
1	205	4
1	99	1
1	79	5
1	193	1
1	24	6
1	100	2
1	183	1
1	210	4
1	16	4
2	4	6
2	76	2
2	204	2
2	177	6
2	117	4
2	156	6
2	28	6
2	230	5
2	24	1
3	157	4
3	16	2
3	65	3
3	18	1
3	224	1
3	90	1
3	165	3
3	34	6
4	24	3
4	165	4
4	8	4
4	99	2
4	140	3
4	148	3
4	86	2

4	99	3
4	211	4
4	18	2
4	74	2
5	103	2
5	107	2
5	224	4
5	30	2
6	56	1
6	92	3
6	71	6
6	22	4
7	130	4
7	168	5
7	26	5
7	26	5
7	223	6
8	102	2
8	49	6
9	113	4
9	202	1
9	114	5
10	217	2
10	133	4
10	159	1
10	93	6
10	20	2
10	124	3
10	202	5
10	192	4
10	34	6
10	192	1
11	183	2
11	189	3
11	187	3
11	204	5
11	164	1
12	166	6
12	187	5
12	208	3

12	87	5
13	44	2
13	147	1
14	134	1
14	20	2
14	106	3
15	102	4
15	29	6
15	175	3
15	108	2
16	173	3
16	3	6
16	160	5
16	158	4
17	164	2
17	193	3
17	52	4
18	233	4
18	155	1
18	11	3
19	79	1
19	157	1
19	177	5
19	178	1

2.4 – Sample Queries

In Section 2.4.1 we will list all of the queries that we will be using on our database. In section 2.4.2 we will briefly define relational algebra and list all of our queries in relation algebra. In section 2.4.3 we will briefly tuple relational calculus and list all of our queries in tuple relational calculus. Finally in section 2.4.4 we will briefly define domain relational calculus and list our queries in domain relational calculus.

2.4.1 – Design of Queries

1. Select employees born in 1986 who have processed orders placed by customers who were born in 1986.

2. Select orders that contain the most expensive item.

3. Select employees who have processed all orders placed by Mary Lynch.

4. Select all customers born after 1991 named Julia Woods.

5. Select items of music with an artist who is not The Jackson 4.

6. Select items that have been supplied by the distributor Riffware and who is located in Springfield.

7. Select all employees and the orders they are currently processing.

8. Select customers who have only purchased items greater than 30 dollars.

9. Select all employees who have processed orders placed before 2008 or after 2010.

10. Select customers who have purchased every item.

2.4.2 – Relational Algebra Expressions

1. Select employees born in 1986 who have processed the order of a customer also born in 1986

$$\sigma_{e.bDate \geq 1-1-1986 \wedge e.bDate < 12-31-1986}(Employee^{e}) * \pi_{eID}(\pi_{cID}(\sigma_{c.bDate > 1-1-1986 \wedge c.bDate < 12-31-1986} Customer^{c}) * Order)$$

2. Select orders that contain the most expensive item

$$I \leftarrow (\pi_{iID} Item - \left(\pi_{i1.iID}\left(\sigma_{i1.listPrice < i2.listPrice}\left(Item^{i1} \times Item^{i2}\right)\right)\right) * Contains\ )$$

$$\pi_{oID}(I) * Orders$$

3. Select employees who have processed all orders placed by Mary Lynch

$$Employee * \pi_{eID}(\pi_{cID,eID} Order \div \pi_{cID}(Order * \pi_{cID}(\sigma_{fname="Mary" \wedge lname="Lynch"} Customer)))$$

4. Select all customers born after 1991 named Julia Woods

$$\sigma_{bdate > 12-31-1991 \wedge fname="Julia" \wedge lname="Woods"} Customer$$

5. Select items of music with an artist who is not The Jackson 4.

$$Item * (\ Music - \ \sigma_{artist="The\ Jackson\ 4"}(Music))$$

6. Select items who have been supplied by the distributor Riffware and who is located in Springfield.

$$Items * \pi_{dID}(\sigma_{name="Riffware" \wedge city="Springfield"} Distributor)$$

7. Select all employees and the orders they are currently processing.

$$(\sigma_{e.eDate=NULL}(Employee^{e})) \bowtie_{e.eid=o.eid} (\ \sigma_{o.edate=NULL}(\text{orders}^{o}))$$

8. Select customers who have only purchased items greater that $30.

$$Customer - (Customer * \pi_{cID}\left(\sigma_{price<30}(Order * Contains * Items)\right))$$

9. Select all employees who have processed orders placed before 2008 or after 2010.

$$Employee * (\pi_{cID}(\sigma_{o.dateOrdered<1-1-2008} Order) \cup \pi_{eID}(\sigma_{dateOrdered>12-31-2010} Order))$$

10. Select customers who have purchased all items.

$$Customer * \pi_{cID}(\pi_{cID, iID}(Order * Contains) \div \pi_{iID}(Item))$$

2.4.3 – Tuple Relational Calculus Expressions

1. Select employees born in 1986 who have processed the order of a customer also born in 1986

$$\{e|Employee(e) \ \wedge (\exists c)(\exists o)\ (Customer(c)\ \wedge Order(o) \wedge\ o.cID = c.cID \wedge o.eID = \ e.eID\ \wedge (e.bDate \geq \text{"1/1/1986"}\ \wedge e.bDate \leq \text{"12/31/1986"}\ ) \wedge (c.bDate \geq \text{"1/1/1986"}\ \ \wedge\ c.bDate \leq \text{"12/31/1986"}\ )\ )\}$$

2. Select orders that contain the most expensive item

$$\{o|Orders(o) \wedge\ (\exists i1)(\exists co)(Item(i1) \wedge Contains(co) \wedge i1.iID = co.iID\ \wedge o.oID = co.oID \wedge\ \neg(\exists i2)(Item(i2) \wedge i2.listPrice > i1.listPrice))\}$$

3. Select employees who have processed all orders placed by Mary Lynch

$\{e|Employee(e) \wedge (\exists o)(Order(o) \wedge o.eID = e.eID \wedge\ \neg(\exists c)(Customer(c) \wedge c.cID = o.cID \wedge c.fName\ != \text{"}Mary\text{"}\ \ \wedge c.lName\ != \text{"}Lynch\text{"}\ ))\}$

4. Select all customers born after 1991 named Julia Woods

$$\{c|Customer(c) \wedge c.fName = \text{Julia} \wedge c.lName = \text{Woods} \wedge c.bDate > 12/31/1991\}$$

5. Select items of music with an artist who is not The Jackson 4.

$$\{i|Item(i) \wedge (\exists m)(Music(m) \wedge m.iID = i.iID \wedge\ m.artist \neq \text{The Jackson 4})\}$$

6. Select items who have been supplied by the distributor Riffware and who is located in Springfield.

$$\{i|Item(i) \wedge (\exists d)(Distributor(d) \wedge i.dID = d.dID \wedge d.name = \text{RiffWare} \wedge d.city = \text{Springfield})\}$$

7. Select all employees and the orders they are currently processing.

$$\{e.fName, e.lName, o.oID|Employee(e) \wedge (\exists o)(Order(o) \wedge e.eID = o.eID \wedge e.eDate = \text{"}NULL\text{"}\ \wedge o.eDate = \text{"}NULL\text{"}\ )\}$$

8. Select customers who have only purchased items greater that $30.

$$\{c|Customer(c) \wedge (\exists o)(\exists co)(Order(o) \wedge Contains(o) \wedge o.oID = co.oID \wedge co.cID = o.cID \wedge \neg(\exists i)(Item(i) \wedge i.iID = co.iID \wedge i.listPrice \leq \$30))\}$$

9. Select all employees who have processed orders placed before 2008 or after 2010.

$$(e|Employee(e) \wedge (\exists o)(Order(o) \wedge e.eID = o.eID \wedge (o.dateOrdered < "1/1/2008" \vee o.dateOrdered > "12/31/2010"))\}$$

10. Select customers who have purchased all items.

$$\{c|Customer(c) \wedge (\exists co)(\exists o)(\forall i)(Item(i) \rightarrow Contains(co) \wedge Order(o) \wedge i.iID = co.iID \wedge co.oID = o.oID \wedge c.cID = o.cID)\}$$

2.4.4 – Domain Relational Calculus Expressions

1. Select employees born in 1986 who have processed the order of a customer also born in 1986.

$$\{f, l \mid employee\ (\_, f, l, \geq 1/1/1986\ \wedge \leq 12/31/1986, \_, \_, \_, \_, \_, \_)\ \wedge\ (\exists c)(\exists o)Customer(c, \_, \_ \geq 1/1/1986\ \wedge \leq 12/31/1986, \_, \_, \_, \_, \_, \_, \_)\ \wedge\ Order(\_, e, c, \_, \_, \_, \_, \_, \_, \_, \_)$$

2. Select orders that contain the most expensive item

$$\{o \mid Order(o, \_, \_, \_, \_, \_, \_, \_, \_, \_, \_)\ \wedge\ (\exists i2)(\exists p)(Item(i1, \_, \_, p, \_, \_, \_, \_, \_)\ \wedge$$

$$Contains(i1, o, \_)\ \wedge\ \neg(\exists i2)(Item(\_, \_, \_, > p, \_, \_, \_, \_, \_, \_, \_)$$

3. Select employees who have processed all orders placed by Mary Lynch

$$\{f, l \mid (\exists e)Employee(e, f, l, \_, \_, \_, \_, \_, \_, \_, \_)\ \wedge\ (\exists c)Order(\_, e, c, \_, \_, \_, \_, \_, \_, \_, \_)\ \wedge\ \neg(Customer\ (c, \neq mary, \neq lynch, \_, \_, \_, \_, \_, \_, \_, \_)$$

4. Select all customers born after 1991 named Julia Woods

$$\{f, l \mid customer(\_, f, l, > 12/31/1991, \_, \_, \_, \_, \_, \_, \_)\ \wedge\ f = "Julia"\ \wedge l = "Woods"$$

5. Select items of music with an artist who is not The Jackson 4.

$$\{i, t, p, w \mid Item(i, \_, t, p, \_, \_, \_, \_, w, \_) \wedge \neg Music(i, "Jackson\ 4", \_)\}$$

6. Select items who have been supplied by the distributor Riffware and who is located in Springfield.

$$\{i, t, p, w \mid (\exists d)Item(i, d, t, p, \_, \_, \_, \_, \_, w, \_) \wedge (Distributor(d, "Riffware", \_, "Springfield", \_, \_, \_)$$

7. Select all employees and the orders they are currently processing.

$$\{f, l, o \mid (\exists e)Employee(e, f, l, \_, \_, \_, \_, \_, \_, = NULL)\ \wedge\ order(o, e, \_, \_, \_, = NULL, \_, \_, \_, \_)$$

8. Select customers who have only purchased items greater that $30.

$$\{c, f, l \mid customer(c, f, l, \_, \_, \_, \_, \_, \_, \_, \_) \wedge (\exists i)(\exists o)(contains(o, i, \_) \wedge\ order(o, \_, c, \_, \_, \_, \_, \_, \_, \_, \_)$$

$$\wedge\ \neg Item(i, < 30, \_, \_, \_, \_, \_, \_, \_, \_)\}$$

9. Select all employees who have processed orders placed before 2008 or after 2010.

$$\{c, f, l \mid Employee(c, f, l, \_, \_, \_, \_, \_, \_, \_) \wedge Order(\_, \_, c, \_, \_, \_, \leq 1/1/2008 \vee \geq 12/31/2010, \_, \_, \_, \_)$$

10. Select customers who have purchased all items.

$$\{ c, f, l \mid Customer(c, f, l, \_, \_, \_, \_, \_, \_, \_, \_)(\exists Co)(\exists o)(\forall i)(Item(i, \_, \_, \_, \_, \_, \_, \_, \_, \_, \_)$$

$$\rightarrow Contains(o,i,\_) \wedge Order(o,\_,c,\_,\_,\_,\_,\_,\_,\_)$$

Phase III

In phase 2 section 4.1.1 we will define and describe the normalization of relations in 1st, 2nd, 3rd, and Boyd-Codd Normal form and certain update anomalies that can occur and how they relate to the normal forms. In section 4.1.2 we will check and list each of every relation that was not in 3rd or Boyd-Codd normal form. In section 4.2 we will describe the main purpose of SQL*PLUS and the functionalities that SQL*PLUS provides. In section 4.3 we will describe and define the schema objects allowed in the Oracle DBMS and list the schema objects within our project. In section 4.4 we will list all of our relations schema and contents using SQL*PLUS. In section 4.5 we will run our queries from phase 2 as SQL statements on our relations and display the reports that it generates. Finally in section 4.6 we will describe the java data loader and its functionality.

3.1 Normalization of Relations

3.1.1 – Normalization

First Normal Form

First normal form (or 1NF) is a relation property, which requires a relation to only contain attributes with single-valued, atomic values and is considered a minimal requirement in the process of database normalization. First normal form helps to keep duplicate tuples out of a table by creating new tables for each related set of data, and identifies each set of data by using a primary key.

Second Normal Form

Second normal form (or 2NF) requires that a relation be in First Normal Form; beyond that a second normal form relation must only contain non-key attributes that are dependent on the primary-key. This only applies when the primary-key is composted of more than one attribute and is said to be composite.

Third Normal Form

Third normal form (or 3NF) as you may have guessed, requires that a relation also be in second normal form. Third normal form then requires that all non-key attributes be independent of all other non-key attributes in the table. Essentially third normal form ensures that no attributes that don't belong in a table are in a table.

Boyce-Codd Normal Form

Boyce-Codd normal form is a slightly stricter version of third normal form in which all functional dependency based redundancy has been removed. In other words for all dependencies in a relation ($X \rightarrow Y$), the dependency must be trivial or X must be the super key of the relation.

Anomalies

Anomalies can occur upon deletion, update, etc., in relational databases that have not been normalized. If an attribute that is functionally dependent on a primary key is changed, all other instances of that value must also be changed. The normalization process seeks to eliminate all occurrences of update anomalies. For example, if our Order relation contained the customer's address and an order was to be deleted, the customer's address would be lost, likewise, if a customer were to make more than one order, the address would be duplicated.

3.1.2 Checking for Normalization

Most of our relations were converted to third normal form during the process of converting from the ER-model to the Relational Model. Much though was put into ensuring that no data could be duplicated and that all redundancy had been eliminated during that process; primary keys are used in all of our relations and all m:m relations employ the foreign key method. One detail that we did overlook resides in the Item relation. The attribute format is a multivalued attribute that must be split into its own relation; an item may exist on several formats, for example a new video game is likely to be released on all three major consoles and a movie is likely to be released on both DVD and Blu-Ray, music may exist on several formats aswell.

Item (iID, dID, title, listPrice, releaseDate, preDate, genre, format, returnStatus, wholesalePrice, imgURL)

Item (iID, dID, title, listPrice, releaseDate, preDate, genre, returnStatus, wholesalePrice, imgURL)

Format (iID, formatName)

3.2 – SQL*PLUS

SQL*PLUS is an interactive command line tool installed with every Oracle database installation. SQL*PLUS allows you to write and execute SQL statements in standard SQL and PL/SQL. SQL*PLUS also has internal commands and scripts that you may run. SQL*PLUS is generally used to create and manage schema objects, manipulate current data and query data in existing relations. SQL*PLUS allows you to write your own scripts for all of your database schema objects in a hierarchal fashion so you can quickly create an entire database.

3.3 – Schema Objects

Several schema objects exist in the Oracle database such as tables, indexes, views, sequences, synonyms, clusters, procedures, functions, packages, database links and snapshots. Each schema is stored within a tablespace and each schema object may be stored in one or multiple data files.

TABLES

Tables are your relational schemas, the place where all the information and data of a relation is stored. Columns represent attributes and tuples represent rows. Each table has associated metadata that lets you define constraints such as entity integrity constraint, referential key constraint, domain constraints and null constraints. While for the most part the tables of your database represent your relations there are a couple differences. For instance a relation is an unordered set of values but a table in most practicality is ordered.

Some syntax for the creation of a table would be:

CREAT TABLE *name* (
columnName1 *type1* *defaultExpresion1* *constraint1,*
columnName2 *type2* *defaultExpresion2* *constraint2,*
columnName3 *type3* *defaultExpresion3* *constraint3*
tableConstraint
)

Webstore tables:
JRDM_employee
JRDM_customer
JRDM_creditcard
JRDM_distributor
JRDM_order
JRDM_item
JRDM_contains
JRDM_format
JRDM_music
JRDM_movies
JRDM_games

INDEXES

An index is an optional structure associated with a table or table cluster, which can optimize the speed of data access in the Oracle database. Creating an index on one or more columns on a table creates an ordered result set, which can be used as a lookup table for the original table. Indexes can be unique or can contain duplicate values. A clustered index can physically order the contents of a table, where as a non-clustered index can logically order the contents of a table even if the physical contents are not ordered.

Syntax for creating a relation can be used as followed:

```
CREATE INDEX indexName ON table TABLESPACE tablespace;
```

```
Webstore indexes:
pk_JRDM_employee
pk_JRDM_customer
pk_JRDM_creditcard
pk_JRDM_distributor
pk_JRDM_order
pk_JRDM_item
pk_JRDM_contains
pk_JRDM_format
pk_JRDM_music
pk_JRDM_movies
pk_JRDM_games
ix_JRDM_distriName
ix_JRDM_employeeName
ix_JRDM_customerName
ix_JRDM_format
```

VIEW

A view is a stored query that is treated like a virtual table. Views are vary useful in that they let you as they let you create relations that are very important to have but don't necessarily need to create a table for. Views simplify join statements to a single virtual table. Views are not stored directly; instead they are stored in underlying tables or are computed at query time.

Syntax for a view would be as follows:

```
CREATE OR REPLACE VIEW viewName AS selectStatement;
```

SEQUENCES

In the Oracle database a sequence is used to generate a sequential series of numbers automatically. Sequences are mainly used to generate the primary key attribute for a given relation. Sequences are not tied to a specific table, so a sequence can be used across many tables.

Syntax for creating a sequence is as follows:

```
CREATE SEQUENCE sequenceName
     MINVALUE minValue
     MAXVALUE maxValue
     START WITH initialValue
     INCREMENT BY incrementValue
     CACHE cacheNumberValue;
```

SYNONYMS

In the Oracle database a synonym is an alternate way to name a schema object. You can use a synonym when you are granting access to an object from another schema but you don't want any users to worry about knowing which schema owns the object.

Syntax for creating a synonym is as follows:

```
CREATE OR REPLACE PUBLIC SYNONYM synonymName FOR schemaObject;
```

CLUSTERS

In an Oracle database a cluster stores data from one or more tables. The tables in a cluster have to have one or more columns in common. Oracle stores all the rows from all the tables together that share the same cluster key. An example would be a table that

contained “birth_date and “fname” where all the other tables involved in the cluster would also need to contain “birth_date” and “fname”.

Syntax for a cluster would be as follows:

```
CREATE CLUSTER clusterName (column1, column2, column3)
SIZE dataSize
STORAGE (INTIAL initialClusterSize NEXT sizeToAllocate)
```

PROCEDURES

In an Oracle database a procedure is a group of PL/SQL statements complied and stored on a given database. Stored procedures can be used to implement business rules during certain tasks that you normally wouldn’t be able to accomplish. Stored procedures can be executed by name.

Syntax for a stored procedure would be as follows:

```
CREATE OR REPLACE PROCEDURE procedureName (paramaterList) AS
     declarationSection
BEGIN
     PL/SQL_statementsSection
EXCEPTION
     exceptionHandling
END;
```

FUNCTIONS

In an Oracle database a function is very similar to the procedure schema object but there are some differences. Procedures can return a value, result set or nothing but a function must always return a value. A function can be used in a select statement where a procedure must always be executed. Lastly where a stored procedure input and output parameters a function can only have input parameters.

Syntax for a function would be as follows:

```
CREATE OR REPLACE FUNCTION functionName(parameterList)
RETURN returnVariable
IS
     DeclarationSection
BEGIN
     Pl/sqlStatement
EXCEPTION
     exceptionHandling
END;
```

PACKAGES

In an Oracle database a package is a group of related procedures and functions stored together in the database for continued use. Like procedures and functions, applications and users can call packaged procedures and functions explicitly. An advantage of packages is certain schema objects like procedures can be overloaded. When a package subprogram is called for the first time it is loaded into memory, allowing for better performance.

Partial syntax for a package is shown as follows:

```
CREATE OR REPLACE PACKAGE packageName AS
     proceduresList
     functionsList
     exceptionsList
END packageName;
```

DATABASE LINKS

In an Oracle database a database link is a pointer that defines a one-way communication path from the Oracle database server to another database server. A client connected to database A can use a link stored in database A to access information in the remote database B, but database B cannot access stored information in database A unless it establishes a link with database A.

Syntax for a database link is as follows:

```
CREATE DATABASE LINK linkName
CONNECT TO schema BY password
USING connectString;
```

SNAPSHOTS

In an Oracle database a snapshot is copy of a target table from a single point in time. Snapshots in newer versions of Oracle are called a materialized view. Snapshots can be used to replicate data non-master sites in a replicated environment and to store the result set of expensive queries.

Syntax for a snapshot would be:

CREATE SNAPSHOT *snapshotName* AS SELECT * FROM *tableName*;

3.4 – Relational Instances

The following section contains sample sqlplus reports of each relation in the database.

Customer

DESC jrdm_customer;

```
 Name                                      Null?    Type
 ----------------------------------------------------------------
 C_ID                                      NOT NULL NUMBER(6)
 GENDER                                    NOT NULL VARCHAR2(1)
 FNAME                                     NOT NULL VARCHAR2(40)
 LNAME                                     NOT NULL VARCHAR2(40)
 BIRTH_DATE                                NOT NULL DATE
 STREET                                    NOT NULL VARCHAR2(40)
 CITY                                      NOT NULL VARCHAR2(40)
 STATE                                     NOT NULL VARCHAR2(40)
 ZIP                                       NOT NULL NUMBER(5)
 PHONE_NUM                                 NOT NULL NUMBER(11)
 EMAIL_ADDRESS                             NOT NULL VARCHAR2(40)
 PASSWORD                                  NOT NULL VARCHAR2(40)
```

SELECT * FROM jrdm_customer ORDER BY c_id DESC

```
  C_ID G FNAME      LNAME     BIRTH_DATE STREET                   CITY          STATE          ZIP   PHONE_NUM  EMAIL_ADDRESS            PASSWORD
------ - ---------- --------- ---------- ------------------------ ------------- -------------- ------ ---------- ------------------------ -------------
    10 M Clarence   Griffin   23-DEC-86  018 Butterfield Pass     Charlotte     North Carolina 28263  1.7046E+10 cgriffin9@nba.com        Pd6ezChhEVG
     9 F Amy        Reid      10-DEC-86  1 Anderson Alley         Kansas City   Kansas         66160  1.9133E+10 areid8@umn.edu           APZJUQnLtZ
     8 M Andrew     Payne     07-MAY-88  2792 Morning Trail       Boise         Idaho          83711  1.2087E+10 apayne7@bizjournals.com  6BFEPsx5Ma
     7 M Shawn      Harris    03-JUL-91  66 North Lane            New York City New York       10184  1.2124E+10 sharris6@gizmodo.com     l2J1mEx80tAf
     6 M Harold     Long      10-MAR-91  0 Ridge Oak Pass         Houston       Texas          77299  1.7134E+10 hlong5@loc.gov           MiuUwQK
     5 M Bruce      Young     11-AUG-89  30646 Reindahl Street    New Orleans   Louisiana      70124  1.9854E+10 byoung4@behance.net      ugLCjkmmOK
     4 F Julia      Woods     24-DEC-92  9421 Haas Center         San Jose      California     95123  1.4085E+10 jwoods3@rakuten.co.jp    Dhe4BgLc
     3 F Nancy      Lopez     11-MAR-89  8 Corry Court            Portland      Oregon         97201  1.9719E+10 nlopez2@qq.com           Hu1VcvO
     2 F Mary       Lynch     28-MAR-86  4377 Eagle Crest Junction Cincinnati   Ohio           45254  1.5135E+10 mlynch1@illinois.edu     RKRUmp0
     1 F Christine  Mccoy     04-JUN-88  1840 Wayridge Court      Santa Barbara California     93111  1.8059E+10 cmccoy0@jigsy.com        vtqumhb7
```

Employee

DESC jrdm_Employee;

```
 Name                                                  Null?    Type
 ----------------------------------------------------------------------
 E_ID                                                  NOT NULL NUMBER(6)
 FNAME                                                 NOT NULL VARCHAR2(40)
 LNAME                                                 NOT NULL VARCHAR2(40)
 BIRTH_DATE                                            NOT NULL DATE
 STREET                                                 NOT NULL VARCHAR2(40)
 CITY                                                  NOT NULL VARCHAR2(40)
 STATE                                                 NOT NULL VARCHAR2(40)
 ZIP                                                    NOT NULL NUMBER(5)
 PHONE_NUM                                             NOT NULL NUMBER(11)
 START_DATE                                            NOT NULL DATE
 END_DATE                                              DATE
```

SELECT * FROM jrdm_Employee ORDER BY e_id DESC

E_ID	FNAME	LNAME	BIRTH_DAT	STREET	CITY	STATE	ZIP	PHONE_NUM	START_DAT	END_DATE
10	Anne	Kim	25-SEP-88	9726 Onsgard Center	Norfolk	Virginia	23520	1.7577E+10	22-MAY-09	25-JUN-12
9	Virginia	Price	17-JUN-92	8741 Caliangt Park	Saint Louis	Missouri	63116	1.3148E+10	05-FEB-08	
8	Martin	Cruz	15-JAN-88	02557 Loftsgordon Lane	Akron	Ohio	44310	1.3302E+10	26-MAY-10	
7	Janice	Turner	20-NOV-85	80090 5th Circle	Raleigh	North Carolina	27626	1.9195E+10	09-MAY-10	07-MAY-14
6	Doris	Nichols	30-OCT-89	99563 Scoville Drive	Sparks	Nevada	89436	1.7755E+10	11-OCT-09	11-FEB-13
5	Alan	Greene	11-JUL-91	23 Erie Point	Salt Lake City	Utah	84145	1.8016E+10	14-DEC-08	
4	Beverly	Brown	28-MAY-90	53 Melvin Terrace	Orlando	Florida	32808	1.3218E+10	04-MAY-10	
3	Ernest	Carter	24-JUL-86	3 Karstens Hill	Raleigh	North Carolina	27605	1.9193E+10	01-OCT-09	
2	Kathy	Bishop	20-OCT-91	2 Brown Place	Dayton	Ohio	45470	1.9374E+10	04-APR-10	27-SEP-10
1	George	Carpenter	08-MAR-92	798 Stone Corner Pass	Columbus	Ohio	43240	1.7402E+10	23-MAY-10	28-AUG-14

Credit Card

DESC jrdm_creditcard;

```
----------------------------------------------------------------
 CARD_NO                                   NOT NULL NUMBER(20)
 C_ID                                      NOT NULL NUMBER(6)
 FNAME                                      NOT NULL VARCHAR2(40)
 LNAME                                     NOT NULL VARCHAR2(40)
 EXP_DATE                                  NOT NULL DATE
 SECURITY_CODE                                      NOT NULL NUMBER(3)
 CARD_TYPE                                      NOT NULL VARCHAR2(40)
```

SELECT * FROM jrdm_creditcard;

CARD_NO	C_ID	FNAME	LNAME	EXP_DATE	SECURITY_CODE	CARD_TYPE
3.5787E+15	3	Nancy	Lopez	08-APR-18	372	jcb
4.9052E+17	4	Julia	Woods	19-AUG-19	681	switch
3.5333E+15	5	Bruce	Young	06-JUL-21	857	jcb
5.6023E+15	6	Harold	Long	26-FEB-19	507	bankcard
4.0180E+12	7	Shawn	Harris	06-SEP-18	707	visa
5.6023E+18	8	Andrew	Payne	07-FEB-21	865	china-unionpay
3.5798E+15	9	Amy	Reid	05-AUG-18	993	jcb
6.7063E+18	10	Clarence	Griffin	30-DEC-19	428	laser
4.8445E+15	1	Christine	McCoy	01-DEC-18	904	visa-electron
3.5505E+15	2	Mary	Lynch	01-JAN-21	234	jcb

Distributor

DESC jrdm_Distributor;

```
 Name                                                  Null?    Type
 ----------------------------------------------------------------------
 D_ID                                                  NOT NULL NUMBER(6)
 NAME                                                  NOT NULL VARCHAR2(40)
 STREET                                                NOT NULL VARCHAR2(40)
 CITY                                                  NOT NULL VARCHAR2(40)
 STATE                                                 NOT NULL VARCHAR2(40)
 ZIP                                                    NOT NULL NUMBER(5)
 PHONE_NO                                               NOT NULL NUMBER(11)
```

SELECT * FROM jrdm_Distributor ORDER BY d_id DESC;

D_ID	NAME	STREET	CITY	STATE	ZIP	PHONE_NO
10	Aibox	7 Pierstorff Plaza	Minneapolis	Minnesota	75470	1.6127E+10
9	Roombo	6 Logan Crossing	New Orleans	Louisiana	70124	1.9858E+10
8	Vimbo	229 Orin Trail	Denver	Colorado	80204	1.3037E+10
7	Mita	9567 Grayhawk Plaza	Gadsden	Alabama	35905	1.2567E+10
6	Riffwire	63742 Dennis Hill	Sacramento	California	95813	1.9162E+10
5	Voonte	97012 Cascade Avenue	Houston	Texas	77090	1.2815E+10
4	Jetwire	50224 Cambridge Pass	Austin	Texas	78783	1.5129E+10
3	Ooba	5 Oakridge Crossing	Garden Grove	California	92645	1.7143E+10
2	Riffwire	36 Hermina Avenue	Springfield	Massachusetts	11105	1.4138E+10
1	Voonyx	475 Muir Crossing	Washington	District of Columbia	20099	1.2027E+10

Item

DESC jrdm_Item;

```
Name                                      Null?    Type
 ----------------------------------------------------------------
 I_ID                                     NOT NULL NUMBER(6)
 D_ID                                     NOT NULL NUMBER(6)
 TITLE                                    NOT NULL VARCHAR2(40)
 GENRE                                    NOT NULL VARCHAR2(40)
 IMG_FILE                                 NOT NULL VARCHAR2(40)
 LIST_PRICE                               NOT NULL NUMBER(6)
 WHOLE_SALE_PRICE                         NOT NULL NUMBER(6)
 RELEASE_DATE                             NOT NULL DATE
 PRE_ORDER_DATE                                    DATE
 RETURN_STATUS                            NOT NULL NUMBER(1)
```

```
SELECT * FROM jrdm_Item ORDER BY i_id DESC;
```

I_ID	D_ID	TITLE	GENRE	IMG_FILE	LIST_PRICE	WHOLE_SALE_PRICE	RELEASE_D	PRE_ORDER	RETURN_STATUS
101	9	Wow What an Adventure	Adventure	101.jpeg	50	30	27-OCT-15		0
89	8	Wow Now Thats What I Call Anime	Pop	89.jpeg	10	6	29-SEP-13		0
82	8	The Numbness	Pop	82.jpeg	10	6	29-SEP-13		0
81	7	The Spookining	Dark Gothic	81.jpeg	10	6	29-SEP-13		0
77	7	Kids on the playground	Pop	77.jpeg	10	6	29-SEP-13		0
62	7	Little Richard	Comedy	62.jpeg	10	6	28-JUL-12		0
61	6	Little Nicki 2	Comedy	61.jpeg	10	6	28-JUL-12		0
60	6	Little Nicki	Comedy	60.jpeg	10	6	28-JUL-12		0
59	5	Dog Bones	Kids	59.jpeg	50	30	07-JUN-16	20-JAN-16	0
58	5	Jak and Baxter	Adventure	58.jpeg	50	30	27-OCT-15		0
57	4	Final Fantasy &	Role-Playing Game	57.jpeg	60	40	15-AUG-17	17-MAY-16	1
56	4	Space Adventure	Adventure	56.jpeg	60	40	16-DEC-15	30-MAR-14	1
53	4	Jeffrey Dahmer	Historical	53.jpeg	20	10	04-DEC-14		0
52	4	Jenayfer	Soul	52.jpeg	10	6	29-SEP-13		0
50	4	Uncharted 7	Action	50.jpeg	60	40	26-JUL-18	19-APR-16	0
48	4	MMMM what Godd Cak	Meme Spout	48.jpeg	10	6	29-SEP-13		0
46	4	The Showrunner	Drama	46.jpeg	20	10	04-DEC-14		0
39	4	Grandmas Pa	Comedy	39.jpeg	20	10	04-DEC-14		0
38	3	The Power of Color	Gospel	38.jpeg	10	6	29-SEP-13		0
37	3	Bloodhound	Third-Person Shooter	37.jpeg	60	40	30-JUN-17	08-SEP-16	1
36	3	Madden 2015	Sports	36.jpeg	60	40	30-JUN-17	08-SEP-16	1
35	2	Death Wish	Power Metal	35.jpeg	10	6	29-SEP-13		0
34	2	Blues Poos	Children	34.jpeg	20	10	04-DEC-14		0
30	2	Jamba Juice And Friends	Children	30.jpeg	20	10	04-DEC-14		0
28	2	Kirby: Right Back at Ya! - The Reckoning	Horror	28.jpeg	20	10	04-DEC-14		0
27	1	Kirby: Right Back at Ya!	Adventure	27.jpeg	20	10	04-DEC-14		0
23	1	Scream Canadian	Rock n Roll	23.jpeg	10	6	29-SEP-13		0
15	1	The Matrix 5	Action	15.jpeg	20	10	18-JAN-18	24-OCT-17	0
14	1	Eloi Eloi Lama Shabachtani	True Worship	14.jpeg	10	6	29-SEP-13		0

Music

DESC jrdm_Music;

```
Name                                      Null?    Type
----------------------------------------------------------------
 I_ID                                     NOT NULL NUMBER(6)
 ARTIST                                   NOT NULL VARCHAR2(40)
 RECORD_LABEL                             NOT NULL VARCHAR2(40)
```

SELECT * FROM jrdm_Music ORDER BY i_id DESC;

```
     I_ID ARTIST                                 RECORD_LABEL
--------- ---------------------------------------- ----------------------------------------
        35 Bronson                                 Metal Blade
        38 The Jackson 4                           Metal Blade
        48 The Looking Glass                       Sony
        52 Christopher                             Grass Blue
        77 Marbile                                 MGM
        81 The Spooky Door                         Grass Blue
        82 Numb Brughs                             Grass Blue
        89 Weaboo                                  Orange Place
        14 The Willies                             MGM
        23 Gilfoil                                 Sony
```

Movie

DESC jrdm_Movies;

```
Name                                      Null?    Type
 ----------------------------------------------------------------------
 I_ID                                     NOT NULL NUMBER(6)
 DIRECTOR                                 NOT NULL VARCHAR2(40)
 PRODUCTION_COMPANY                       NOT NULL VARCHAR2(40)
```

SELECT * FROM jrdm_Movies ORDER BY i_id DESC;

```
      I_ID DIRECTOR                               PRODUCTION_COMPANY
---------- ---------------------------------------- ----------------------------------------
        61 Howard Richards                          Orange Soda Productions
        60 Katherine Moreno                         Steak Parfait Studios
        53 Ruth Fields                              Grape Sauce
        46 Martha Holmes                            Breakfast Club Productions
        39 Gregory Wilson                           Gassy Studios
        34 Roger Ford                               Holo
        30 Lois Grant                               Holo
        28 James Howell                             Holo
        27 Sarah Day                                The Movie Place
        15 Johnny Griffin                           The Movie Place
```

Game

DESC jrdm_Games;

```
Name                                      Null?    Type
----------------------------------------------------------------------
I_ID                                      NOT NULL NUMBER(6)
DEVELOPER                                 NOT NULL VARCHAR2(40)
PUBLISHER                                 NOT NULL VARCHAR2(40)
```

SELECT * FROM jrdm_Games ORDER BY i_id DESC;

```
     I_ID DEVELOPER                              PUBLISHER

---------- ---------------------------------------- ----------------------------------------

      101 Rocksockom                              Bethesda Softworks

       59 Obsidian                                SilverSoft

       58 Obsidian                                Deep Silver

       57 Ubisoft                                 Deep Silver

       56 Frictional Games                        SilverSoft

       50 Frictional Games                        Bethesda Softworks

       37 Frictional Games                        TriHeart

       36 The Looking Glass                       Electronic Arts

        4 BioWare                                 Bethesda Softworks
```

Order

DESC jrdm_Order;

```
 Name                                                  Null?    Type
 ----------------------------------------------------------------------
 O_ID                                                  NOT NULL NUMBER(6)
 E_ID                                                  NOT NULL NUMBER(6)
 C_ID                                                  NOT NULL NUMBER(6)
 CARD_NO                                           NOT NULL NUMBER(20)
 DATE_ORDERED                                          NOT NULL DATE
 START_DATE                                            NOT NULL DATE
 END_DATE                                               DATE
 STATUS                                            NOT NULL NUMBER(1)
 SHIPPING_METHOD                                            VARCHAR2(40)
 TRACKING_NO                                                VARCHAR2(40)
 DATE_SHIPPED                                                            DATE
```

SELECT * FROM jrdm_Games ORDER BY i_id DESC;

O_ID	E_ID	C_ID	CARD_NO	DATE_ORDE	START_DAT	END_DATE	STATUS	SHIPPING_METHOD	TRACKING_NO	DATE_SHIP
10	9	10	3.5798E+15	05-OCT-15	14-OCT-15		1	NULL	NULL	
9	9	9	3.5798E+15	01-MAY-15	09-MAY-15	14-MAY-15	0	FEDEX	750199078	10-MAY-15
8	8	8	5.6023E+18	01-JUL-14	05-JUL-14	21-NOV-14	0	USPS	297755506	08-NOV-14
7	5	7	5.6023E+15	17-JUL-14	24-JUL-14	25-FEB-15	0	FEDEX	472097423	20-FEB-15
6	5	6	5.6023E+15	23-SEP-15	01-OCT-15		1	NULL	NULL	
5	5	5	3.5333E+15	12-OCT-14	18-OCT-14	21-AUG-15	0	FEDEX	55810113	04-AUG-15
4	4	4	3.5787E+15	01-FEB-14	06-FEB-14	18-JUN-15	0	USPS	656235882	08-JUN-14
3	4	3	3.5787E+15	24-JAN-14	04-FEB-14	13-JUN-15	0	UPS	251288355	07-JUN-15
2	3	2	3.5505E+15	09-OCT-15	13-OCT-15		1	NULL	NULL	

Contains

DESC jrdm_Contains;

```
Name                                      Null?    Type
 ----------------------------------------------------------------
 O_ID                                     NOT NULL NUMBER(6)
 I_ID                                     NOT NULL NUMBER(6)
 QUANTITY                                 NOT NULL NUMBER(2)
```

SELECT * FROM jrdm_Contains ORDER BY o_id;

O_ID	I_ID	QUANTITY
10	62	5
10	61	3
10	59	4
10	58	2
10	57	1
10	56	6
10	38	6
10	36	1
10	27	4
10	15	2
9	36	5
9	27	1
9	15	4
8	56	6
8	50	2
7	62	6
7	61	5
7	38	5

7	27	5
7	15	4
6	58	4
6	57	6
6	56	3
6	50	1
5	38	2
5	36	4
5	27	2
5	15	2
4	62	2
4	61	2
4	59	4
4	58	3
4	57	2
4	56	3
4	50	3
4	38	2
4	36	4
4	27	4
4	15	3
3	58	6
3	57	3
3	56	1
3	50	1
3	38	1
3	36	3

3	27	2
3	15	4
2	59	1
2	58	5
2	57	6
2	56	6
2	50	4
2	38	6
2	36	2
2	27	2
2	15	6
1	101	1
1	89	1
1	82	1
1	81	1
1	77	1
1	62	1
1	61	1
1	60	1
1	59	1
1	58	1
1	57	1
1	56	1
1	53	1
1	52	1
1	50	1
1	48	1

1	46	1
1	39	1
1	38	1
1	37	1
1	36	4
1	35	4
1	34	1
1	30	2
1	28	6
1	27	1
1	23	5
1	15	1
1	14	4
1	4	6
1	1	1

3.5 – Queries in SQL

1. Select employees born in 1986 who have processed the order of a customer also born in 1986.

```
SELECT *
FROM JRDM_employee NATURAL JOIN (  SELECT e_ID
                                    FROM (  SELECT c_ID
                                            FROM JRDM_customer
                                            WHERE birth_date >= to_date('1986/1/1','yyyy/mm/dd')
                                              AND birth_date <= to_date('1986/12/31','yyyy/mm/dd')
                                            ) NATURAL JOIN JRDM_order
                                   )
WHERE birth_date >= to_date('1986/1/1','yyyy/mm/dd') AND birth_date <= to_date('1986/12/31','yyyy/mm/dd');
```

2. Select orders that contain the most expensive item

```
  SELECT UNIQUE *
  FROM JRDM_order NATURAL JOIN (SELECT o_ID
                                FROM JRDM_contains NATURAL JOIN (SELECT i_ID
                                                                 FROM JRDM_item
                                                                 MINUS
                                                                 SELECT i1.i_ID
                                                                 FROM JRDM_item i1, JRDM_item i2
                                                                 WHERE i2.list_price > i1.list_price)
                               );
```

3. Select employees who have processed all orders placed by Mary Lynch

```
SELECT *
FROM JRDM_employee e
WHERE NOT EXISTS (SELECT *
                  FROM JRDM_customer c INNER JOIN JRDM_order o2 ON c.c_ID = o2.o_ID
                  WHERE c.fname = 'Mary' AND c.lname = 'Lynch' AND NOT EXISTS (SELECT *
                                                                              FROM JRDM_order o
                                                                              WHERE o.e_ID = e.e_ID AND o.o_ID = o2.o_ID)
                 );
```

4. Select all customers born after 1991 named Julia Woods

```
SELECT *
FROM JRDM_customer
WHERE birth_date > TO_DATE('1991/12/31','yyyy/mm/dd') AND fname = 'Julia'
AND lname = 'Woods';
```

5. Select items of music with an artist who is not The Jackson 4.

```
SElECT *
FROM JRDM_item NATURAL JOIN JRDM_music
WHERE artist NOT IN ('The Jackson 4');
```

6. Select items who have been supplied by the distributor Riffware and who is located in Springfield.

```
SELECT *
FROM JRDM_item NATURAL JOIN (SELECT d_ID
                             FROM JRDM_distributor
                             WHERE name = 'Riffwire' AND city = 'Springfield');
```

7. Select all employees and the orders they are currently processing.

```
SELECT *
FROM JRDM_employee e LEFT OUTER JOIN JRDM_order o ON e.e_ID = o.e_ID AND e.end_date
                                                  IS NULL AND o.end_date IS NULL;
```

8. Select customers who have only purchased items greater that $30.

```
SELECT *
FROM JRDM_customer
WHERE EXISTS (SELECT *
              FROM JRDM_customer NATURAL JOIN JRDM_order)
MINUS
SELECT *
FROM JRDM_customer NATURAL JOIN (SELECT c_ID
                                 FROM JRDM_order NATURAL JOIN (SELECT o_ID
                                                               FROM JRDM_contains NATURAL JOIN (SELECT i_ID
                                                                                                FROM JRDM_item
                                                                                                WHERE list_price < 30)
                                                               )
                                 );
```

9. Select all employees who have processed orders placed before 2008 or after 2010.

```
SELECT UNIQUE *
FROM JRDM_employee NATURAL JOIN (SELECT e_ID
                                 FROM JRDM_order
                                 WHERE date_ordered < TO_DATE('2008/01/01','yyyy/mm/dd') OR
                                       date_ordered > TO_DATE('2010/12/31','yyyy/mm/dd'));
```

10. Select customers who have purchased all items.

```
SELECT *
FROM JRDM_customer c
WHERE NOT EXISTS (SELECT *
                  FROM JRDM_item i
                  WHERE NOT EXISTS (SELECT *
                                    FROM JRDM_order o INNER JOIN JRDM_contains co ON o.o_ID = co.o_ID
                                    WHERE i.i_ID = co.i_ID and o.c_ID = c.c_ID)
                 );
```

3.6 – Data Loader

There are many methods that can be used to take all the data for your relational database and insert into the Oracle database. The SQL INSERT INTO … VALUES(…) statement allows you to insert information into a table one at a time. Multiple records can be added to a table using INSERT INTO … *queryStatement*. There are also multiple Dataloading applications for free and for purchase online including Oracles SQL Developer software application.

Dr. Huaqing Wang provided a dataloader for the class. The dataloader is written in java and provides a functionality that allows the dataloader to connect to the Oracle database and parse a text file to import the data into the tables. The program also allows the user to choose the delimiting character to separate columns. A text file could be formatted thusly:

```
TABLENAME| tableName| numberOfColumns
attribute1| attribute2| attribute3
attribute1| attribute2| attribute3
attribute1| attribute2| attribute3
```

All the data for the Webstore database was imported using the dataloader. The only problem encountered was that NULL values for dates could not be added, but Dr. Wang added functionality to the dataloader that allowed the functionality to insert NULL values into columns with data type dates.

Phase IV

In phase 4 section 4.1 we will discuss PL/SQL, its program structure and how it handles stored procedures/functions, packages and triggers. In section 4.2 we will also demonstrate some of the stored subprograms written using PL/SQL for our webstore database. In Section 4.3 we will compare and contrast MySQL and MS SQL Server and how each one handles stored procedures/functions.

4.1 – PL/SQL

4.1.1 what is PL/SQL?

PL/SQL is Oracles procedural extension SQL and the Oracle relational database. PL/SQL allows for procedural statements like conditions and loops. It also allows for the declaration of variables, constants, procedures, functions, and triggers. PL/SQL can also handle exceptions. Procedures, like discussed in phase 3, are a PL/SQL blocks formed to be the unit that is the procedure. Functions are the same as procedures in that they are composed of blocks that are the unit we call a function and is dissimilar to procedures in the same way as discussed in phase 3. PL/SQL block are also significantly more efficient than single SQL statement calls to the server. Each statement you send to the Oracle server must be finished before the next one can begin but if you compose the statement in a single PL/SQL block then all the statements together in the block is one call to the server, significantly reducing the network traffic. Also stored subprograms are precompiled making it much more efficient then constantly calling and compiling single SQL statements.

4.1.2 – PL/SQL program structure

The basic unit of PL/SQL program is the block; grouping together related declarations and statements. A PL/SQL block is defined by the keywords DECLARE, BEGIN, EXCEPTION, and END, dividing the block into three main parts, the declarative part, the executable part and the exception part. The syntax for a basic PL/SQL block would be:

DECLARE

declarations

BEGIN

statements

EXCEPTION

WHEN *exception* THEN *exceptionHandling*

END;

Control structures in PL/SQL are very similar to the essence of control statements in any other programing language. In PL/SQL there are conditional, iterative and sequential control structures. There are if statements and while/for loops in PL/SQL. Basic syntax for some control structures would be:

IF *condition* THEN

statements;

ELSEIF *condition* THEN

statements;

ELSE

statements;

END IF;

FOR *counter* IN *lowerbound .. upperbound* LOOP

statements;

END LOOP;

Cursors in PL/SQL are a SQL construct that allows access to one or multiple rows from a result set. Oracle creates cursors implicitly for any SELECT statement that returns n rows, but for multiple rows a cursor must be declared explicitly to access each row one at time. Syntax for a cursor would be as follows:

CURSOR *cursorName* [(*parameterName* TYPE [, *parameterName* TYPE])]

IS *selectStatement*;

4.1.3 Procedures

Procedures and functions PL/SQL are the named subprograms in PL/SQL that can be called by name once defined. Stored procedure in PL/SQL is declared very similarly to a normal PL/SQL block with some minor differences. The DECLARE section of normal PL/SQL block is replaced with the name of the procedure and any input parameters you define. A parameter can be declared is IN (input), OUT (output), and IN OUT (input/output). Parameters can also be declared with default values. Stored procedures are also able to return values. Syntax for declaring a procedure would be:

CREATE OR REPLACE PROCEDURE *procedureName* (*paramaterList*) AS

declarationSection

BEGIN

PL/SQL_statementsSection

EXCEPTION

exceptionHandling

END;

4.1.4 Functions

Functions are almost identical to procedures in their declaration except that a return type has to be specified.

Syntax for a function would be as follows:

CREATE OR REPLACE FUNCTION *functionName*(*parameterList*)

RETURN *returnVariable*

IS

DeclarationSection

BEGIN

Pl/sqlStatement

EXCEPTION

exceptionHandling

END;

4.1.5 Package

A package in PL/SQL is a collection of PL/SQL units, including procedures, functions and triggers. Packages have a section for declaring the package specification and a separate part for declaring the body. The declaration section of the package declares constants, types, variables, exceptions, and cursors and can also declare prototypes for procedures and functions; all these declarations are declared as public. The package body is used for defining implementations of stored procedures and functions and for making any of the declarations in the declaration part private. Syntax for a package would be:

CREATE OR REPLACE PACKAGE *packageName* AS

publicDeclarations

END *packageName*;

CREATE PACKAGE BODY *packageName* AS

privateDeclarations

implementations

END;

4.1.6 Triggers

Triggers in Oracle are code blocks that are executed automatically occur given certain events. Triggers are good for reacting to certain events that occur in inserting, updating and deleting, like creating logs for and making sure business rules are always implemented. Trigger can be specified to trigger before an update, after an insert or delete, and instead of an update. Triggers can be executed for each row and an only be executed for certain columns. Syntax or a trigger is as follows:

CREATE OR REPLACE TRIGGER *triggerName*

specifiedExecution

OF *columns*

ON *tableName*

FOR EACH ROW

DECLARE

    declarations

BEGIN

    statements

EXCEPTION

    WHEN *exception* THEN *exceptionHandling*

END;

4.3- SQL Procedures Functions and Triggers

We have used the procedures, functions and triggers Dr. Huaqing Wang wanted us to practice for the coding requirements in this phase and applied them to our Webstore database.

JRDM_insCustomer:

This procedure inserts a new row into customer and uses a sequence to automatically update the primary key of the new entry ensuring uniqueness of each new entry.

```
CREATE OR REPLACE PROCEDURE JRDM_insCustomer (
    gender IN JRDM_customer.gender%TYPE,
    fname  IN JRDM_customer.fname%TYPE,
    lname IN JRDM_customer.lname%TYPE,
    birth_date IN JRDM_customer.birth_date%TYPE,
    street IN JRDM_customer.street%TYPE,
    city IN JRDM_customer.city%TYPE,
    state IN JRDM_customer.state%TYPE,
    zip IN JRDM_customer.zip%TYPE,
    phone_num IN JRDM_customer.phone_num%TYPE,
    email_address IN JRDM_customer.email_address%TYPE,
    password IN JRDM_customer.password%TYPE
)
iS
BEGIN
    INSERT INTO JRDM_customer(
        c_ID,
        gender,
        fname,
        lname,
        birth_date,
        street,
        city,
        state,
        zip,
        phone_num,
        email_address,
        password
    ) VALUES (
        c_idSequence.nextval,
        TRIM(gender),
        TRIM(fname),
        TRIM(lname),
        TRIM(birth_date),
        TRIM(street),
        TRIM(city),
        TRIM(state),
        TRIM(zip),
        TRIM(phone_num),
        TRIM(email_address),
        TRIM(password)
    );

    COMMIT;

    EXCEPTION
        WHEN OTHERS THEN
            ROLLBACK;
            DBMS_OUTPUT.PUT_LINE(SQLCODE || ', ' || SQLERRM);
            COMMIT;
END;
/
```

JRDM_delCustomer:

This procedure deletes a customer from the customer table and instead of cascade deletion of foreign keys, the procedure will delete from any tables that reference its key first then delete from its own table.

```
/ CREATE OR REPLACE PROCEDURE JRDM_delCustomer (
    delete_id IN JRDM_customer.c_ID%TYPE
)
IS
BEGIN
    DELETE
    FROM JRDM_creditcard
    WHERE c_ID = delete_id;

    DELETE
    FROM JRDM_order
    WHERE c_ID = delete_id;

    DELETE
    FROM JRDM_customer
    WHERE c_ID = delete_id;

    COMMIT;

    EXCEPTION
        WHEN OTHERS THEN
            ROLLBACK;
            DBMS_OUTPUT.PUT_LINE(SQLCODE || ', ' || SQLERRM);
            COMMIT;
END;
/
```

JRDM_youngAverage:

This function computes the average of the top N youngest customer and returns that value.

```
CREATE OR REPLACE FUNCTION JRDM_youngAverage (
    N NUMBER DEFAULT 1
)
RETURN NUMBER
IS
    avg_age NUMBER
BEGIN

    WITH orderByAge AS (
        SELECT FLOOR(MONTHS_BETWEEN(SYSDATE, birth_date) / 12) AS age
        FROM JRDM_customer
        ORDER BY birth_date DESC
    )
    SELECT AVG(age)
    INTO avg_age
    FROM orderByAge
    WHERE ROWNUM <= N;
```

```
    RETURN avg_age;

    EXCEPTION
        WHEN OTHERS THEN
            ROLLBACK;
            DBMS_OUTPUT.PUT_LINE(SQLCODE || ', ' || SQLERRM);
            COMMIT;
END;
/
```

JRDM_delOrderTrigger

This trigger will delete the foreign key in the contains relation before the deletion of the Order primary key can occur.

```
CREATE OR REPLACE TRIGGER JRDM_delOrderTrigger
BEFORE DELETE ON JRDM_order
FOR EACH ROW
BEGIN

    DELETE FROM JRDM_contains c
    WHERE c.o_ID = :old.o_ID;

    EXCEPTION
        WHEN OTHERS THEN
        ROLLBACK;
        DBMS_OUTPUT.PUT_LINE(SQLCODE || ', ' || SQLERRM);
        COMMIT;
END;
/
```

JRDM_updDistributorTrigger

This trigger makes sure that when the distributor id is updated in the distributor table the item tables distributor id is updated to reflect that.

```
CREATE OR REPLACE TRIGGER JRDM_updDistributorTrigger
BEFORE UPDATE ON JRDM_distributor
FOR EACH ROW
BEGIN

    UPDATE JRDM_item i
    SET i.d_ID = :new.d_ID
    WHERE i.d_ID = :old.d_ID;

    EXCEPTION
        WHEN OTHERS THEN
        ROLLBACK;
        DBMS_OUTPUT.PUT_LINE(SQLCODE || ', ' || SQLERRM);
        COMMIT;
END;
/
```

JRDM_viewTrigger

This trigger makes it possible to perform an update on the employeeProcessing view by executing the update in the trigger instead of the sql statement made at the command line updating the base tables and reflecting that in the view.

```
CREATE OR REPLACE VIEW employeeProcessing AS
SELECT e.e_ID, e.fname, e.lname, o.o_ID, o.status
FROM JRDM_employee e, JRDM_order o
WHERE e.e_ID = o.e_ID;

CREATE OR REPLACE TRIGGER JRDM_updEmpProcTrigger
INSTEAD OF UPDATE ON employeeProcessing
REFERENCING NEW AS n
FOR EACH ROW
BEGIN

    UPDATE JRDM_employee
    SET fname = :n.fname, lname = :n.lname
    WHERE JRDM_employee.e_ID = :n.e_ID;

    UPDATE JRDM_order
    SET status = :n.status
    WHERE JRDM_order.o_ID = :n.o_ID;

    EXCEPTION
        WHEN OTHERS THEN
        ROLLBACK;
        DBMS_OUTPUT.PUT_LINE(SQLCODE || ', ' || SQLERRM);
        COMMIT;
END;
/
```

The process of writing procedures, functions and triggers was very interesting and added more to the toolbox of our programming language knowledge. There was some slight difficulty in writing that but that was merely from the unfamiliarity of writing code in the SQL language after doing this we feel much more confident in our ability to apply the SQL language.

4.4 – Comparison to MS SQL Server & MYSQL

The constructs implemented by MS SQL Server, MYSQL and PL/SQL generally all very similar with MYSQL being the least developed of the three due to it's young age and the fact that it is open-source and free.

Selective Statements

Many of the selective statements in MYSQL and MS SQL Server are very similar to PL/SQL and in turn very similar to the selective statements in any programming language. There are however several syntax differences and some special cases. For instance, IF EXISTS statement does not appear within MS SQL Server but can be implemented using SELECT INTO WHERE EXISTS in conjunction with an IF statement. Many of the selective statements in MYSQL also function similar to the two other DBMS but may be more limited than the other two as MYSQL is still a burgeoning database management system.

Repetitive Statements

Repetitive statements generally work across the three DBMS in question the way they would in any programming language. The exception being, Oracle which does not have a continue statement and must make use of a goto statement in order to break from a while loop.

Sub Program Structure

While the syntax and some of the finer details may be different across MS SQL Server, MYSQL and PL/SQL, the overarching idea of a sub program remains the same. A piece of code is passed parameters, which are in turn used to call queries or create new rows in a table or anything else the database programmer can thing of, and the whole thing can be stored in another file or portion of code and called using a single statement elsewhere; This idea is a fundamental concept applied throughout all of computer science.

Parameter Passing

As expected, parameter passing among the three languages mentioned is very much similar, with only minor differences. All three make use of IN, OUT and IN OUT parameters, although some of the specifics have slight differences. In PL/SQL all parameters are specified as input parameters unless otherwise noted and thusly cannot be modified or used as an out value in another function. MS SQL Server also requires an @ sign to be placed in front of parameters (MYSQL also requires this) and also require the length/size to be specified. MYSQL

Phase V

The webstore frontend/GUI can be reached at http://delphi.cs.csubak.edu/~cs342/jrdm/ and is written primarily in php, html, css and javascript.

5.1 – Daily Activities of the User Group

Customer:

The main user of the database is the customer; customers can browse the items on the webstore frontend and make purchases. They can also create accounts, doing so inserts a new row into the customer relation. Customers build a cart, choosing each item and quantity, which is then placed into an order, which involves several relations. A customer's credit card information is stored to be used for future purchases. Customers can also view the information pertinent to there past orders. Future improvements could include special deals based on their previous orders

Administration:

The storeowner, and employees are the administration user group. The owner's use of the database would be to view daily, monthly and yearly sales reports and other information pertaining to the income of the store. The employees would be use this section of the frontend to view unshipped orders, and although not currently implemented the employee would also be able to update order-shipping statuses and add new items to the store.

5.2 – Relations, Views and Sub Programs

Relations:

All of the relations in our database are implemented somewhere in the front-end. The customer relation is used when the customer creates an account. The order, credit card, item and contains relations are all used when the customer places an order and also on the

reports page. Lastly, the item, movie, music, game, format, and distributor relations are all used to display the products, which the customer may order.

Views:

allItems

The allItems view is used to retrieve all the information required to create the product thumbnail and information/add to cart pop-up.

allGames

The allGames view produces all the same information as allItems but instead of all products, allGames only displays products that are in the category of game.

allMovies

The allMovies view produces all the same information as allItems but instead of all products, allMovies only displays products that are in the category of music.

allMusic

The allMusic view produces all the same information as allItems but instead of all products, allMusic only displays products that are in the category of music.

custOrder

The CustOrder view is used to gather and display the logged in customer's past orders

salesView

The salesView is used to generate json data, which is rendered into graphs on the reports page.

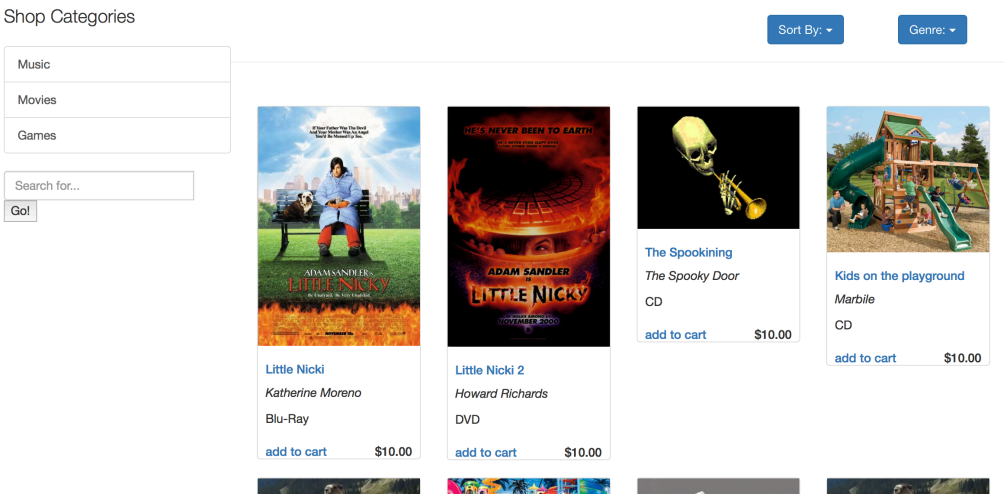
Procedures

JRDM_insCustomer

The insCustomer stored procedure is used when the customer creates an account and ensures that a unique customer id is given to each employee.

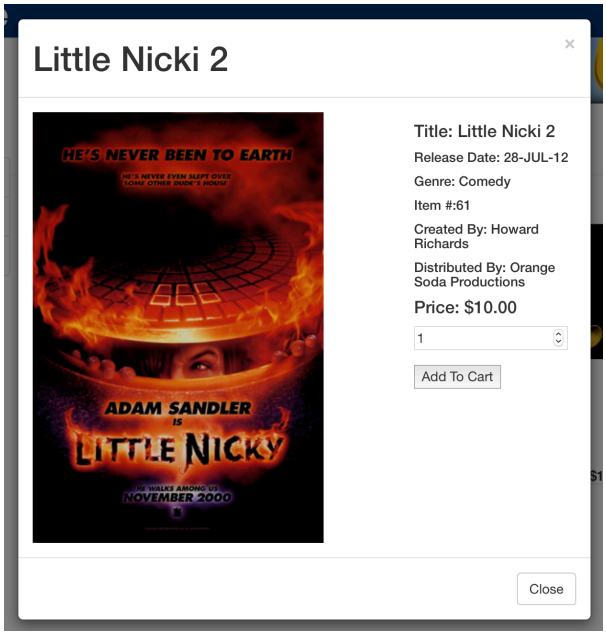
5.3 – Interface Screen Shots and Description

The following section describes the visual features of the frontend to our database.



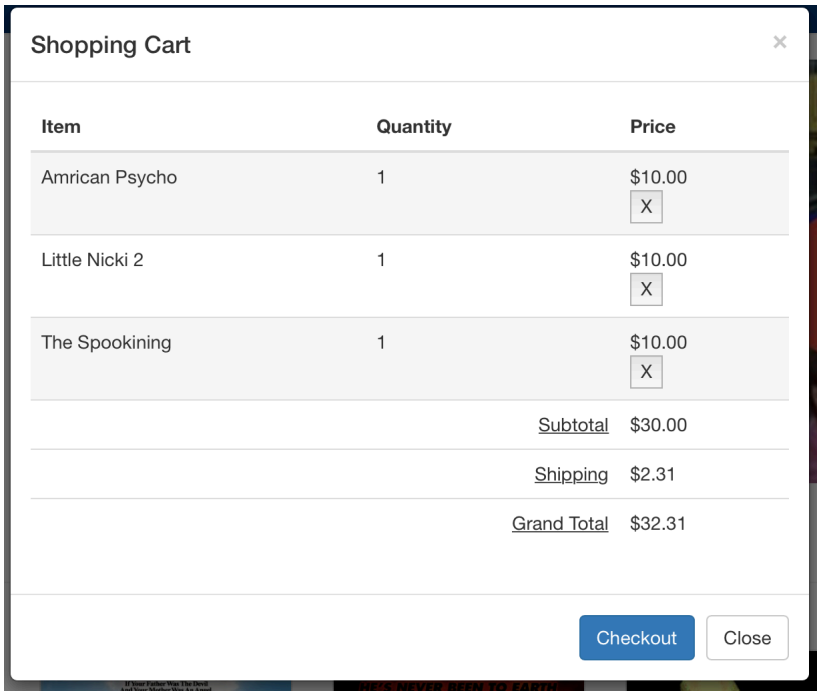
Items

Items are the most important aspect of the front end, if the user cannot view the product, they cannot purchase it either. On the left there are buttons, which filter the items by category and a search bar. Above the items there are two drop down buttons, which filter the items by genre and allows the customer to sort the items however they chooseItem Modal



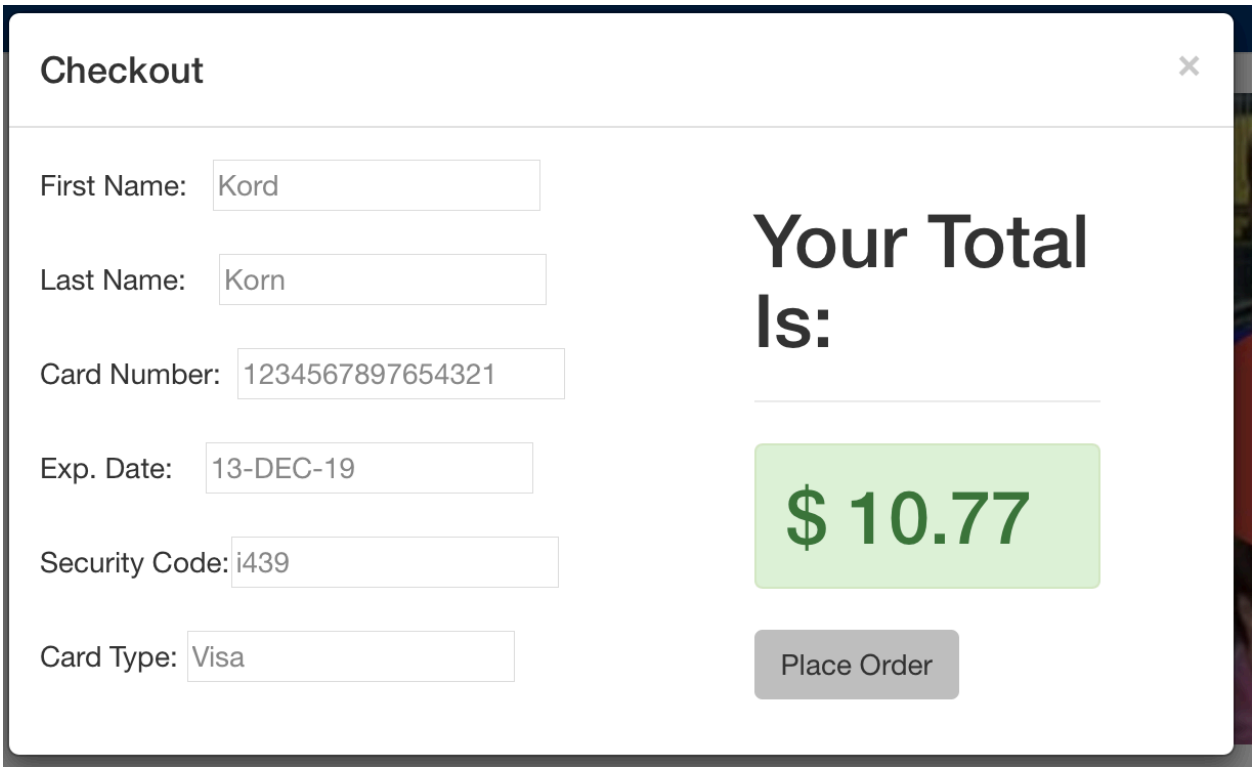
When an item is clicked, a popup modal is displayed containing all the information about the product along with a quantity selector and add to cart button.

Cart



The cart is used to display the items the customer currently wishes to purchase, items can be removed and when the customer is ready, pressing the checkout button will begin their order.

Checkout

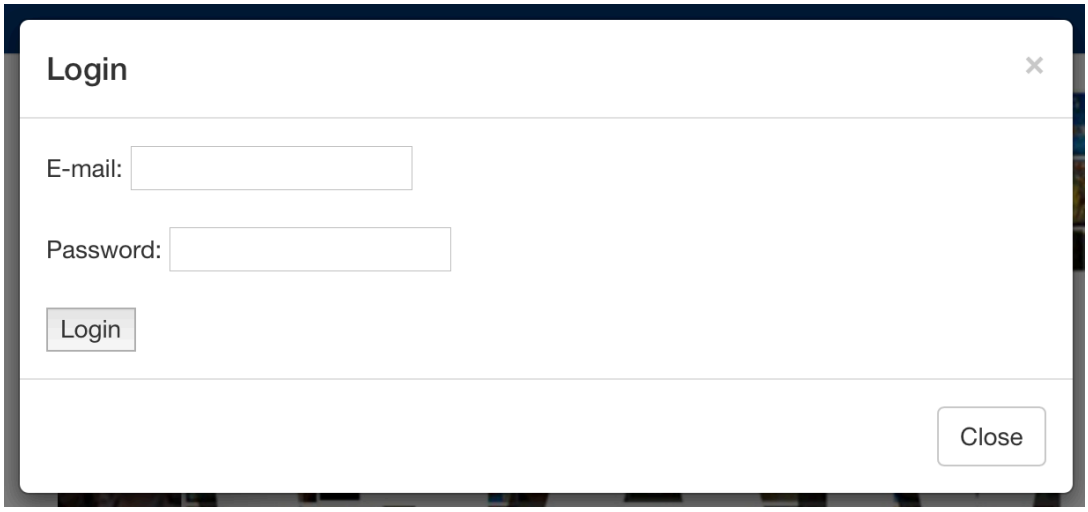


When the customer clicks the checkout button from their cart, the register modal is opened, from there they may enter their credit card information, or if there is a credit card already on file, the values will be automatically filled.

Customer Information Bar

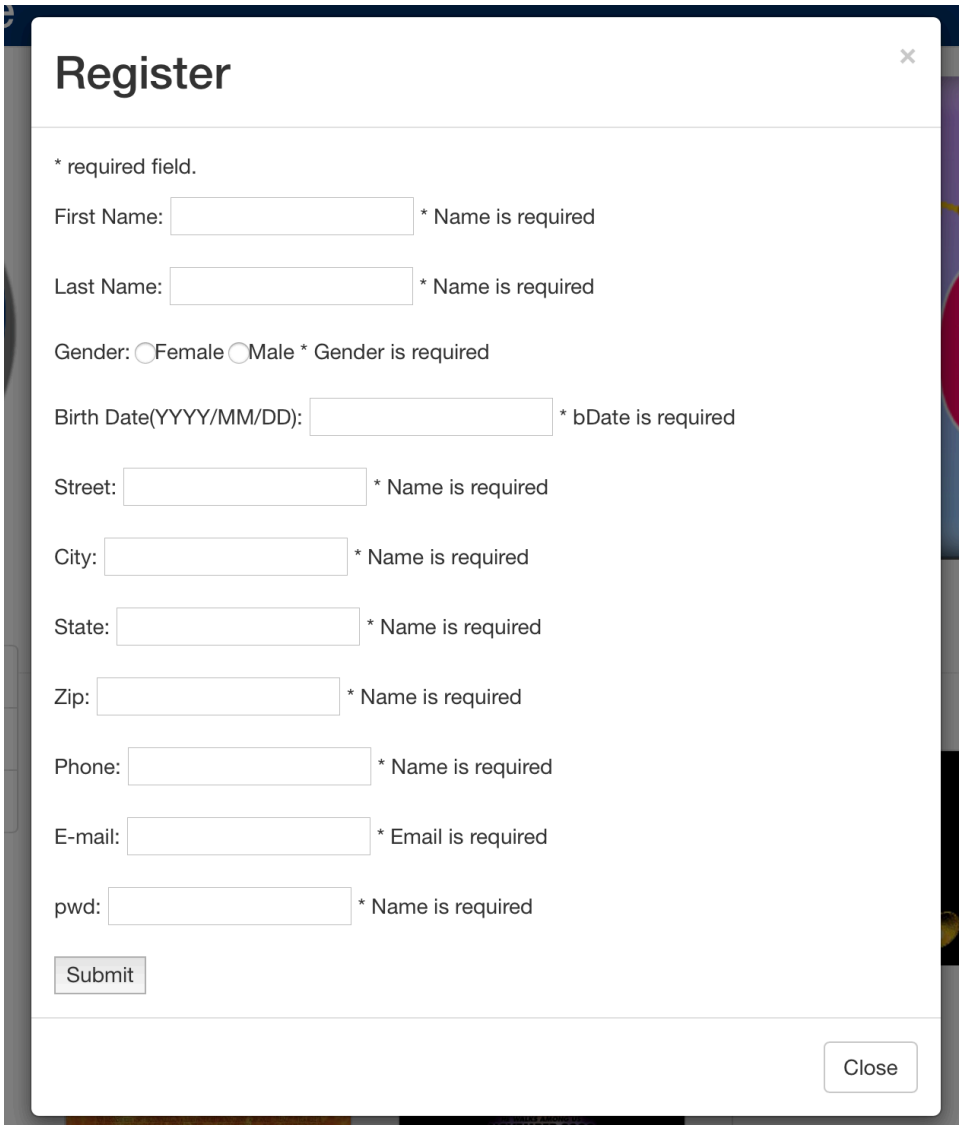
The customer information bar contains links that open modals to various features of the site including the login screen, registration form, customers' past orders and the current items in their cart. The cart button includes a badge displaying the number of items in the customers cart.

Login



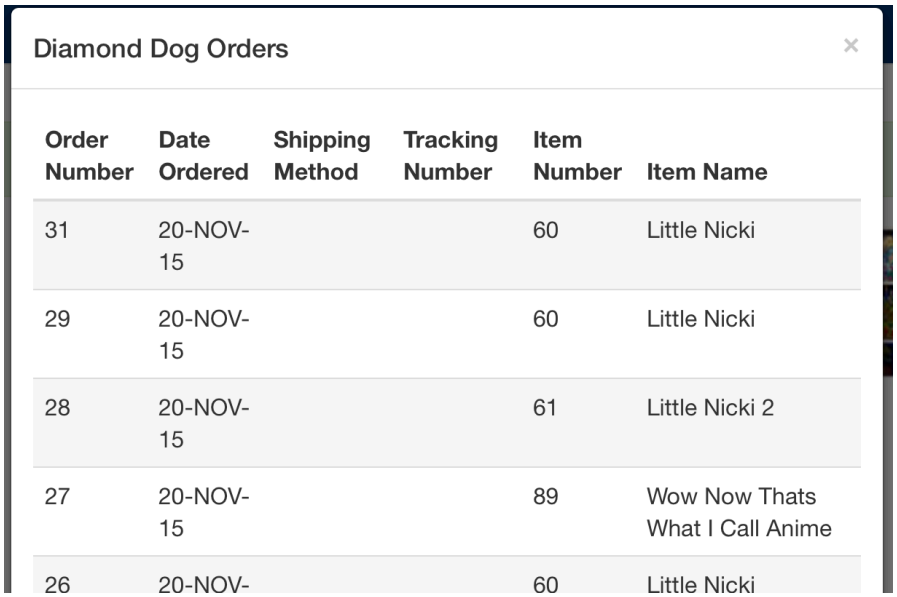
Clicking the login button opens the login modal. If a customer's email and password exist in the database, they may enter them into the login for shown above and login to the website.

Register



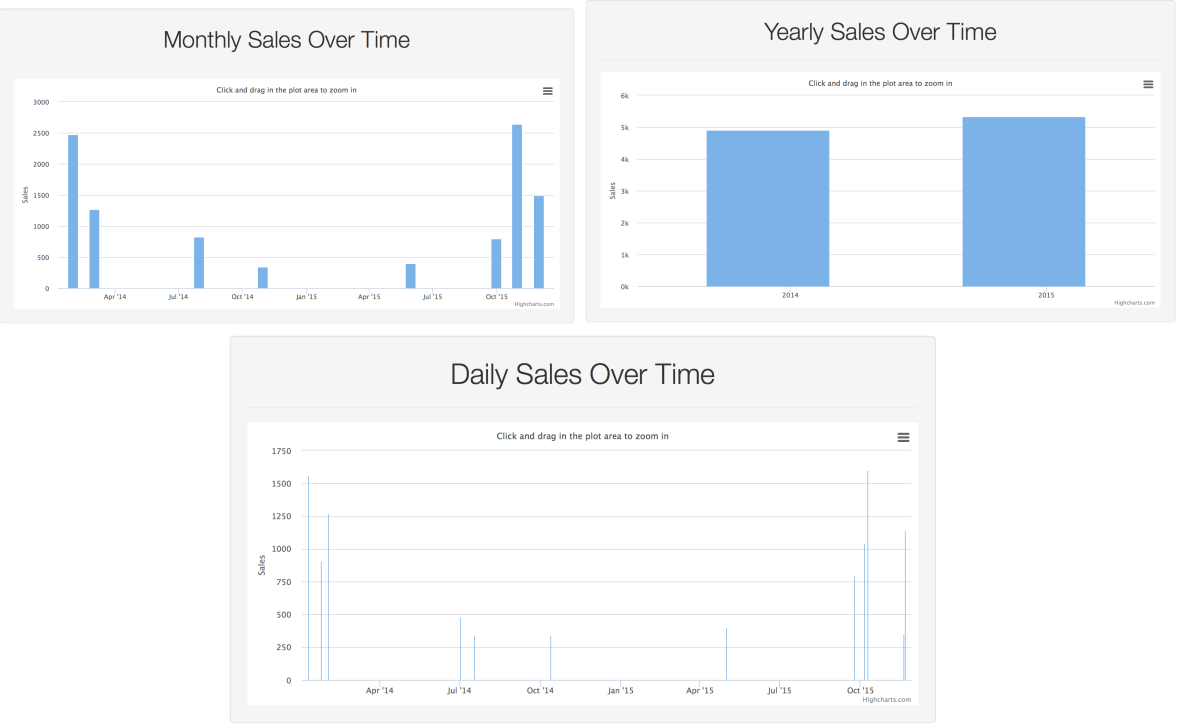
If the customer is not already logged in, they may create a new account by pressing the register button and inserting all of the required information. If the registration is successful or the user already exists an alert will be displayed and the custom

Orders



Here, the user can view past orders (if any) and other information relevant to their order. If the shipping method and tracking number are empty, the item hasn't shipped yet.

Sales Reports



The sales report page contains three interactive, zoom-able charts that display daily, monthly and yearly reports based on the total revenue created from the webstore's sales.

5.4 – Designing of an Interface

Our interface is written in HTML, PHP, CSS and JavaScript and being such, must be accessible to anyone who may be using the web, this is one of the primary concerns of any website. Mobile users, legacy users and the like should all be able to access the page.

When designing an interface, the user should be considered first and foremost. The interface designer needs to realize that all users may not be as technically inclined as he or she and the interface should reflect such. Ease of use is the ultimate end goal. Researching other websites and interfaces what a big part of the design of our interface, on the web there exists a myriad of webstores similar to ours which influenced the design of the webstore presented in this report. A full range of stress testing and bug fixes rounds out the process.

5.4.1 – PHP Database Access Packages

Accessing an Oracle database from a PHP application requires a set of features entitles OCI8, which includes several of the functions mentioned below.

OCI_Connect()

Perhaps the most important function of OCI 8, oci_connect establishes the connection to the database. After this function has been called it is possible to send queries and receive data from the database.

```
$conn = oci_connect('$user', '$pass','(DESCRIPTION=

(ADDRESS=(PROTOCOL=tcp)(HOST=$host)(PORT=$port))

(CONNECT_DATA=(SID=$db_sid)))');
```

OCI_Parse()

Once a connection has been established, a SQL statement may be sent to the database, but first the string containing said SQL statement must first be parsed, this prepares the string to be executed.

```
$stid = oci_parse($conn,
$sql);
```

OCI_Execute()

With an established connection, and parsed sql statement, the action can then be executed, this allows rows to be fetched from the parameter passed in.

```
oci_execute($stid);
```

OCI_Fetch_Array()

Upon execution, the data may then be extracted by row into an array. Each attribute in a row can be accessed using the attribute name as an index to the row array. Each time the function is called, the next row is extracted. The function can be placed into a while loop and all rows in a table can be accessed. This is how all the data used to create the webpage is accessed

```
$row = oci_fetch_array($stid,OCI_BOTH);
```

PHP SESSION Variables

These functions along with many other features of php are used to create the majority of the front end. A PHP session is created to store information containing login information, and an array containing information on the users cart. The following shows an example of how a session variable is when a user logs in. After setting the session variable, other parts of the application will be able to use its information.

```
$_SESSION['logged_in']=1;
```

5.4.2 – Bootstrap

Bootstrap is a web development framework created by the social networking company twitter. Written mostly html, css and javascript, bootstrap provides a lot of the parts of designing a website that would otherwise be very tedious. Bootstrap also provides attractive styling, icons, buttons and the like. Bootstrap works though the naming of html elements, for instance if a button is given the class “btn btn-primary” a nice blue button will be displayed, if a button is given the class “btn btn-danger” a nice red button will be displayed. With bootstrap this concept is applied to essentially every html element. This is how the majority of the visual aspect of out application is organized. Using php to manipulate bootstrap is the heart of our application.

5.5 – Main Features of the GUI

Browsing

Perhaps the most important aspect of our GUI is allowing customers to browse all items contained in the database, providing options to display certain categories and genres, search, and allowing the user to display the data in the order of their choice (by price, release date, etc.). Querying either the allItems, allGames, allMusic or allMovies views allows us to do so.

Cart/ Place order

After browsing for the item the customer wants, they must be able to store said item into a cart in order to later check out and place their order. The cart, which is stored as an array of item numbers, is then used along with customer information to create new rows in the order and contains relations.

Login/Register

Another important aspect is allowing a customer to register an account so that their information can be stored for viewing or use in placing new orders at a later time.

Reports

The final aspect of the GUI is to provide pertinent information in the form of reports. One report already mentioned is essentially a list of orders and the information including shipping method, tracking number, date ordered, etc. that pertains to each order the logged in customer has placed. The second report generated is for use by the owner of the webstore and displays daily, yearly, and monthly sales information provided by JSON data generated using the database contents using a simple php script. In order to display the chart in a friendly, easy to read format, the framework HighCharts was used. A chart created using HighCharts requires the data to be in JSON format.

5.5 Designing and Implementing a Database Application

One of the most important steps of designing and implementing a database, and really all software, is the initial design phase. The initial design needs to be strong and have purpose but must also be flexible enough to change and be altered to solve different kinds of problems. It is very much worth it to spend a lot of time on the initial design phase because you are spending on something that will pay off for several years. Without careful planning

changes can and will be made to stored procedures or tables that can have unknown and potentially disastrous changes to other parts of the application. Many changes of this nature will most likely be last minute and by consequence most likely undocumented making the process of fixing those changes very obfuscated.

The next step would be the implementation of the conceptual design and requirements document. This step has the possibility of ruining previously good design so the implementation of the conceptual to the relation must be carefully thought out. You must always consider every conversion technique; one that will work for someone else's conceptual model may not work elegantly for yours. You should always strive to apply the technique that will arrive to the most elegant solution.

After the implementation has been decided upon the next step is to build the physical database. You may write scripts to help create relations and the necessary constraints decided upon in the implementation. Careful consideration must be taken in storing the scripts because the can streamline the process of rebuilding the database. Also the capabilities of your DBMS must now be taken into consideration. Changes exist between each of every DBMS in how they handle sql statements and how schema objects are handled or whether certain ones even exist.

Another step that could exist after this is the application of business logic in the case where certain laws or policies might be needed to take into consideration. You might have business logic that requires you to check the age of your customer when selling them a product or requires you to make sure that someone should not be scheduled on certain days. The implementation of business logic is incredibly important considering that mistakes made during this step could lead to the destruction of your business or possibly jail time.

The final step for our database application was the implementation of our GUI application. Although this is usually the final step of the process it is not uncommon to consider this during the initial designs of the database application. In the real world there could be possibly a large and varied team of people working on the GUI, anywhere from graphic artists to programmers, so careful communication is needed in order to cleanly and successfully build the application. In our database application we used php to connect to our database and the biggest lesson learned from that is to choose you languages wisely. Some database might not like or appreciate certain server side scripting languages, so when choosing your scripting language it is best to try and match up with something you know will play nice with the database you are trying to connect to. Alternatively, you could consider this when physically building the database where instead of scripting language you would consider what DBMS would work well with the scripting language you choose.

At the end of the day what worked for us will probably not work for everyone and there will always need to be certain cases considered for certain problems leading to the only

conclusion, that in order to fully prepare yourself for any and all ways to create and implement your database, is for you to never stop learning and always try and learn all possible techniques that can serve any purpose and problem you have.

Outcome	Justin	Danny
(3b) An ability to analyze a problem, and identify and define the computing requirements and specifications appropriate to its solution.	10	10
(3e) An ability to design, implement and evaluate a computer-based system, process, component, or program to meet desired needs. An ability to understand the analysis, design, and implementation of a computerized solution to a real-life problem.	9	10
(3f) An ability to communicate effectively with a range of audiences. An ability to write a technical document such as a software specification white paper or a user manual.	10	10
(3j) An ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices.	8	9