Nima Adibpour, Deklan Chung, Abdel Kader Gaye, Christophe Rezk
Group 42

Montreal Urban Apparel Database Application

Requirement Analysis

Introduction

Purpose

Our database application will store the relevant information of an online apparel retail company to effectively capture and manage a company's resources. With a retail company, there are various forms of liabilities and revenues that must be accounted for in order to optimize net income. Thus, the purpose of our database will be to minimize the costs associated with disorganization and improper tracking of resources by storing the information in an easily queryable way. Our database provides convenient means of tracking an order back to its customer, the units involved, and the employee who handled the order. We are able to keep track of the stock that the store currently has for each model, as well as categorize the clothes such that browsing units, adding them to a shopping cart, and ordering through different payment methods can all be possible.

Database Description

Entities and attributes

*The primary key for each entity set is underlined in the descriptions below.

Customer: A customer is someone who is looking to purchase or has purchased items from the online store.
The attributes associated with customers are "first" and "last name", "email", and "password". Every customer must have a unique email which serves as a primary key.
Clothing Model: A clothing model is the representation of the different types of garments that are sold by the store.
Clothing Models have attributes "price", "model discount"*, "name", "sex", and "age range". Each model must have a unique name such as *"Fall2016BomberJacket",* hence why there are no artificial keys associate with it.
Clothing Unit: A clothing unit is an item that can be purchased. Clothing unit is a weak entity of Clothing Model and has a "color" attribute .
A clothing unit can only be identified by a unique combination of "model name" and its partial key, "color". For example, a red *"Fall2016BomberJacket"* and a blue *"Fall2016BomberJacket"* are two distinct items.
Clothing Category: This entity set contains the different classes a clothing model can be a part of. Category is-a **topwear**, is-a **legwear**, is-a **footwear**, and is-an **accessory**.
Each and every clothing category must belong to one of the sub-entity sets (Covering Constraint required).
Each clothing category has a unique "category name" attribute, as well as a "category discount"*.
Topwear: Topwear is a subcategory of the clothing category entity. Topwear has the attribute "chest size"**.
Legwear: Legwear is a subcategory of the clothing category entity. Legwear has the attribute "waist size"**.

Footwear: Footwear is a subcategory of the clothing category entity. Footwear has the attribute “shoe size”**.
Accessory: Accessories is a subcategory of the clothing category entity. Accessories include ties, scarves, gloves, jewellery and other miscellaneous items. Accessories do not require a “size” attribute since such items are one-size-fits-all.
Shopping Cart: A shopping cart contains units that a customer adds to it. Once a customer decides to buy (pay for) the units, an order entity is created for that shopping cart.
Shopping carts have the attribute “date created” as well as a unique artificial “cart ID”.
Order: An order contains information about the “method of payment” chosen and the shopping cart made by the customer. Order entities are linked to shopping carts, and are created when a customer decides to buy the items in a shopping cart.
Orders have attributes “paid for”, “order date”, “payment method”, “final amount,” and “billing address”, and a unique artificial “orderID”.
Shipping Info: Shipping Info contains the information required for a store to prepare a package for delivery to a customer upon receiving an order.
This entity set has the attributes “shipping address”, “shipping date”, “shipping method”, and a unique “tracking number”.
Warehouse: Warehouse holds information concerning the merchandise available for sale or distribution. The warehouse entity set also stores information on when the next shipment of an item will arrive (from factories to the warehouse).
The attributes associated with warehouse are “maximum stock”, “restock date”, and a unique “location” which serves as a primary key.
Employee: The employee holds the information of the employee responsible for packaging and putting together the order. The attributes for employee are name, “employment date”, “employment rank” (which can be normal staff, manager, etc.), and a unique artificial “employee ID” (eID).

NOTE*: The attributes “model discount” and “category discount” might be applied simultaneously for a specific item: if it happens that two conflicting discounts are applied (e.g. 25% off black round neck t-shirts as well as 30% off all topwear), then the higher discount rate shall be applied. Furthermore, the two discount attributes allow us to put a specific clothing model on sale, or put an entire category of clothing on sale. For example, often times stores have a specific t-shirt line “X” on sale, but sometimes they will put all t-shirt lines in store on sale.
NOTE**:The reason why there are different attribute sizes for categories is because topwear, legwear, footwear, and accessories all have different metrics for measuring size. For example, footwear will be stored with datatype Integer (7, 8, 9, …), topwear will be stored with datatype String (XS, S, M, L, XL, …).

Relationships

addsTo: This is a ternary relationship between *customer*, *clothing unit* and *shopping cart*. A *customer* adds *a clothing unit* to a *shopping cart*. There is both a key constraint and a participation constraint on *shopping cart* because each *shopping cart* can only be linked to a unique combination of *clothing unit* and *customer*. Each *customer* can create many *shopping carts* and can choose many *clothing units*. Each *clothing unit* can be added to many *shopping carts* and can be chosen by many *customers*. The addsTo relationship has an attribute indicating the “quantity” of clothing units added to a *shopping cart* by a *customer*.

belongsTo: A *clothing unit* belongs to a *cloth model*. This is a one-to-many relationship with a participation constraint and a key constraint on *clothing unit* because each unit of clothing must belong to a single *cloth model*, while each *cloth model* can have many units. The belongsTo relationship has an attribute indicating the quantity of each *clothing model* available.
describedAs: A *cloth model* is described as a *clothing category*. This is a one-to-many relationship with both a participation constraint and a key constraint on *clothing category* because each *cloth model* must be described as a *category* of clothes, whereas each *category* can contain many *cloth models*.
goesTo: An *order* goes to *shipping information*. This is a one-to-one relationship with participation constraints and key constraints on both *order* and *shipping info* because each *shipping info* must belong to one *order* while each order must contain one shipping information.
handles: An *employee* handles an *order*. This is a many-to-one relationship with both a participation constraint and a key constraint on *order* because each *employee* can handle many *orders* whereas each *order* must be handled by a single *employee*. The handles relationship has an attribute indicating the “(handle) date” where the employee handles an order.
linkedTo: An *order* is linked to a *shopping cart*. This is a one-to-one relationship with key constraints on both *order* and *shopping cart* and a participation constraint on *order* because each *order* must be linked to at most one *shopping cart* while each *shopping cart* can only be linked to at most one *order*. Indeed, an *order* is placed only when a *shopping cart* is finalized.
places: A *customer* places an *order*. This is a many-to-one relationship with both a participation constraint and a key constraint on *order* because each customer can place many *orders* while each *order* must be placed by at most one *customer*. The places relationship has an attribute indicating the “type” of *order* a *customer* places i.e. whether it is a purchase or a refund.
stockedIn: A *clothing model* is stocked in a *warehouse*. This is a many-to-many relationship because each *clothing model* can be stocked in many *warehouses* and each *warehouse* can stock many *clothing models*.

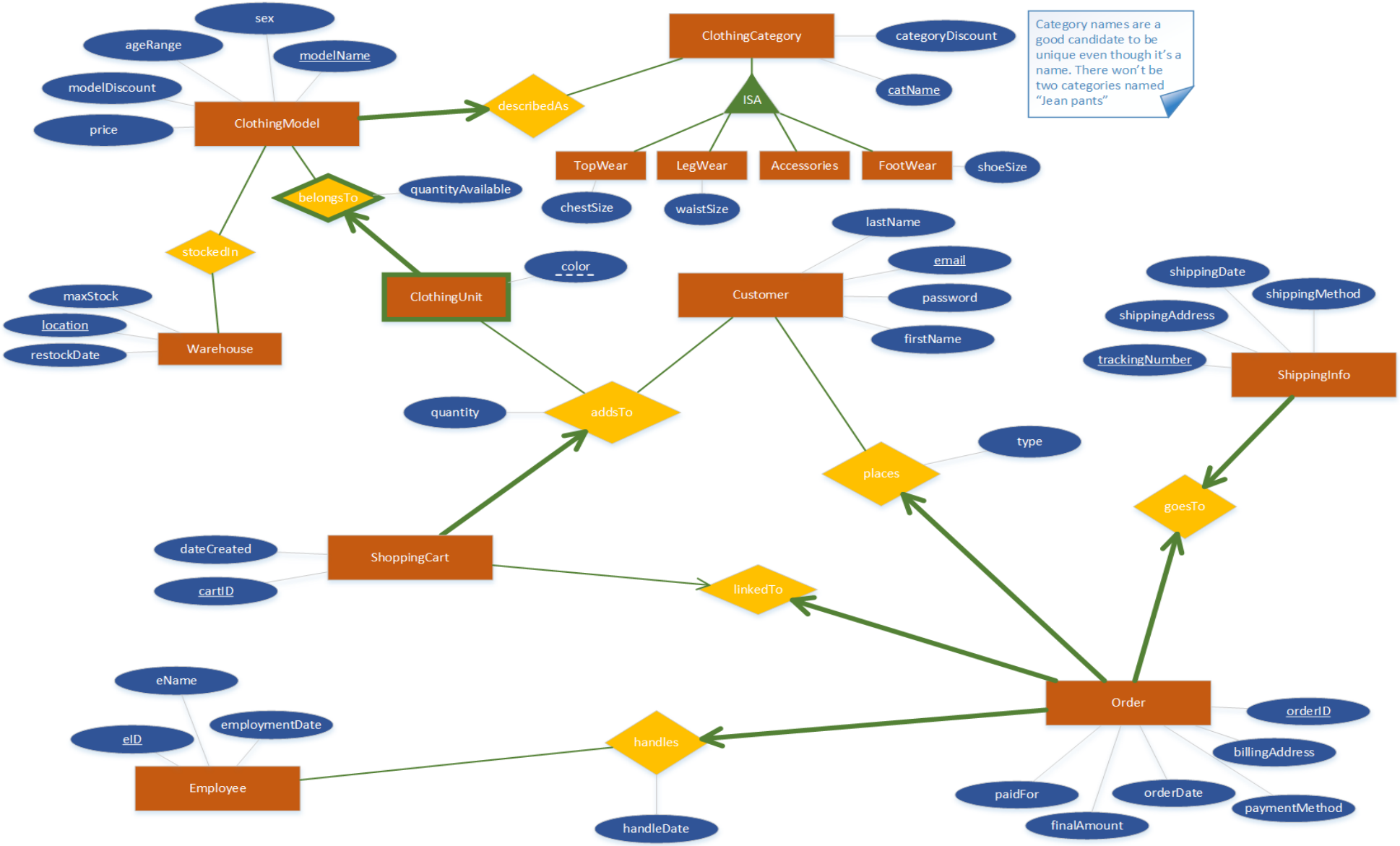
Application description

Overview
The application seeks to efficiently track consumer purchasing patterns and company allocation costs, and store the information in an easily queryable way to avoid improper tracking of resources, minimize costs, and determine the optimal methods for maximizing profit.

Storing calculations in the database would lead to redundancy, and instead, we will run our calculation extraction on a weekly and monthly basis to analyze the information.

Nima Adibpour, Deklan Chung, Abdel Kader Gaye, Christophe Rezk
Group 42

Entity Relation Diagram



Relational Model

ClothingCategory(catName, categoryDiscount)

TopWear(catName, chestSize) *(catName ref ClothingCategory)*

LegWear(catName,waistSize) *(catName ref ClothingCategory)*

Accessories(catName) *(catName ref ClothingCategory)*

FootWear(catName, shoeSize) *(catName ref ClothingCategory)*

ClothingModel(modelName, catName, sex, ageRange, modelDiscount, price) *(catName ref ClothingCategory)*

ClothingUnit(modelName, color)*(modelName ref ClothingModel)*

ModelUnits(modelName, color, quantityAvailable)*(modelName ref ClothingModel) (color ref ClothingUnit)*

NOTE: *Despite the participation and key constraint, it was necessary to create a table for the relation between ClothingModel and ClothingUnit to capture the attribute quantityAvailable.*

ModelsStock(modelName, location) *(modelName ref ClothingModel) (location ref Warehouse)*

Warehouse(location, maxStock, restockDate)

ShoppingCart(cartID, dateCreated)

OrderPlacement(email, orderID, type, trackingNumber) *(email ref Customer) (orderID ref Order) (trackingNumber ref ShippingInfo)*

NOTE: *Despite the participation and key constraint, it was necessary to create a table for the relation between Customer and Order to capture the attribute type.*

Order(orderID, cartID, paidFor, finalAmount, orderDate, paymentMethod, billingAddress, trackingNumber, eID) *(trackingNumber ref ShippingInfo) (cartID ref ShoppingCart) (eID ref Employee)*

AddsTo(color, modelName, email, cartID, quantity) (color, modelName ref ClothingUnit) (email ref Customer) (cartID ref ShoppingCart)

NOTE: *Participation and key constraint; a shopping cart is linked to a unique combination of customer and unit, meaning a shopping cart is linked to exactly one customer and one or more different units.*

*eaning*ShippingInfo(trackingNumber, shippingMethod, shippingDate, shippingAddress, orderID) *(orderID ref Order)*

Customer(firstName, lastName, email, password)

Employee(eID, employmentDate, ename)

OrderHandling(eid, orderID, handleDate) *(eid ref Customer, orderID ref Order)*

NOTE: *Despite the participation and key constraint, it was necessary to create a table for the relation between Employee and Order to capture the attribute handleDate.*