University of British Columbia, Department of Computer Science

CPSC 304

2015 Summer Term 1

Project Part _3

Group Name:

_____Phocas_____

Group Members:

Name	Student Number	Unix ID	Email Address
Rock Luo	30841134	k0a9	rockthebesr@gmail.com
Ryan Quong	42183137	g4w9a	ryan_q73@hotmail.com
David Cai	32109134	n1a0b	davidcai@live.ca

By typing our names and student numbers in the above table, we certify that the work in the attached assignment was performed solely by those whose names and student IDs are included above.

In addition, we indicate that we are fully aware of the rules and consequences of plagiarism, as set forth by the Department of Computer Science and the University of British Columbia.

Description of Project

We modelled a fast food chain mostly using data that we have made up. Our project has two user classes that include employees and customers.

A customer can create an online order for himself/herself by providing the necessary information.

An employee can login using his/her name and employee ID. An employee can then modify our database by adding any of the following: order, item, store, menu or item. An employee can also delete a menu, delete an employee or modify the status of an order at any time. In addition an employee can search the database for any of the following: instances of any of the entities in our database, list the employee IDs working at a store, find a store that has all the menus, sales of each store, sales of each province, find the most expensive item and find the least expensive item.

Final Schema

The one change we made was we allowed empID in allOrder to be able to be set to null, this was done because an order should still exist when the employee no longer works there. In order for the creation of the queries to work, we had moved the order of the queries but the structure had stayed the same.

item(itemName:String, stock:integer, price:Decimal(19,4))

- Represents: the entity set item
- Primary key: itemName
- Price and stock cannot be negative.

delivery(deliveryID :integer, deliveryDate: date, deliveryStatus: String)

- Represents: the entity set delivery
- Primary key: deliveryID
- Constraints: Each delivery must be associated with one online order
- Delivery status must be one of {“out on delivery” or “delivered”}
- Delivery needs to have at least one item
- Delivery needs to associate with an online-order

deliveryHasItems(**deliveryID**: integer, **itemName**: String)

- Represents: the relationship has between delivery and item
- Primary key: deliveryID, itemName

- Foreign key:
    - deliveryID references delivery(deliveryID)
    - itemName references item(itemName)

orders(**orderID**:integer, **itemName**:String)

- Represents: the relationship orders
- Primary key: orderID, itemName
- Foreign key:
    - orderID references allOrder(OrderID)
    - itemName references item(itemName) cannot be null

serves(**menuID**:integer, **itemName**:String)

- Represents: relationship serves
- Primary key: menuID, itemName
- Foreign key:
    - menuID references Menu(menuID)
    - itemName references item(itemName) cannot be null

store(storeID:integer, city: String, province: String, location: String, **empID**: int)

- Represents: the entity set store
- Primary key: storeID
- Foreign key:
    - empID references Manager(empID), empID cannot be null
- Store needs to have a set of menus
- Store needs to have a manager

storeHasMenus(**storeID**: integer, **menuID**: int)

- Represents: the relationship between store and menu
- Primary key: storeID, menuID
- Foreign key:
    - storeID references store(storeID)
    - menuID references menu(menuID)

menu(menuID: integer, serveStartTime: time, serveEndTime: time)

- Represents: the entity set menu
- Primary key: menuID
- Menu needs to serve a set of items

dayTimeMenu(**menuID**: int)

- Represents: the entity set day time menu
- Primary key: menuID
- Foreign key: menuID references menu(menuID)

breakfastMenu(**menuID**: int)

- Represents: the entity set breakfast menu
- Primary key: menuID
- Foreign key: menuID references menu(menuID)

drinkMenu(**menuID**: int)

- Represents: the entity set drink menu
- Primary key: menuID
- Foreign key: menuID references menu(menuID)

employee(empID: integer, ename: String, gender: String)

- Represents: the entity set employee
- Primary key: empID
- Gender is one of {“male”, “female”}
- Employee has to be either a regular employee or a manager

regularEmployee(**empID**: integer, **storeID**: integer, **managerID**: int)

- Represents: the worksAt-regularEmployee-worksUnder relationship set
- Primary key: empID
- Foreign key: empID references employee(empID)
    - storeID references store(storeID) cannot be null
    - managerID references manager(empID) cannot be null
- Regular employee has to work at a store
- Regular employee has to work under a manager

manager(**empID**: int)

- Represents: entity set manager
- Primary key: empID
- Foreign key:
    - empID references employee(empID)
- Manager has to have employees working under him/her
- Manager has to manage one store

manages(**empID**: integer, **storeID**: int)

- Represents: the relationship set manges
- Primary key: empID
- Foreign key:
    - empID references manager(empID)
    - storeID references store(storeID) cannot be null

allOrder(orderID: integer, **storeID**: integer, orderDate: date, price:Decimal(19,4), orderStatus: String, **empID**: int)

- Represents: the fulfill-order relationship set
- Primary key: orderID
- Foreign key:
    - storeID references store(storeID) cannot be null
    - empID references employee(empID)
- Price cannot be negative
- Order has to associate with a store
- Order has to have a set of items
- Order has to be either in-store order or an online order
- orderStatus is one of {“in preparation”, “out on delivery”, “delivered”, “finished”, “cancelled”}
- order has to be fulfilled by an employee

inStoreOrder(**orderID**: int)

- Represents: the entity set inStoreOrder
- Primary key: orderID
- Foreign key:
    - orderID references allOrder(orderID)

onlineOrder(**orderID**: integer, address: String,customerName: String, phoneNumber: int)

- Represents: the onlineOrder entity set
- Primary key: orderID
- Foreign key:
    - orderID references allOrder(orderID)

delivers(**deliveryID**: integer, **orderID**: int)

- Represents: the delivers relationship set
- Primary key: deliveryID
- Foreign Key:
    - orderID references allOrder(orderID) cannot be null
    - deliveryID references delivery(deliveryID)

Queries Used

Arranged by method name

makeInStoreOrder
Select stock
From item
Where itemName=‘INPUT’

-to check if current stock is zero for such item

makeOnlineOrder
Select stock
From item
Where itemName='INPUT'
-to check if current stock is zero for such item

fulfillDelivery
Select orderID
From deliver
Where deliveryID=INPUT
-a trigger to also update orderStatus in allOrder

updateOrderStatus
Select orderStatus
From allOrder
Where orderID=INPUT
-to check if such order is already finished

checkOrderStatus
Select orderStatus
From allOrders
Where orderID=INPUT

addRegularEmployee
Select storeID
from store
where storeID =INPUT

Select Manager
From Select empID
from manager
where empID=INPUT
-check if manager and store id exist

existEmployee
SELECT *
FROM employee
WHERE ename = 'INPUT'' AND empID = INPUT
-check if employe already exist

isManager

SELECT *
FROM manager
WHERE empID = INPUT
-check if employee is a manager

getStoreIdForEmp
SELECT storeID FROM manages WHERE empID =INPUT
SELECT storeID FROM regularEmployee WHERE empID =INPUT
-get storeID

SaleOfEachStore
Select storeID, SUM(price) From allOrder Group By storeID

SaleOfEachProvince
Select Province, SUM(price) From allOrder a, store s where a.storeID = s.storeID Group By Province

MaxMinPrice
Select i1.itemName, i1.price From item i1 where price = (select max(i2.price) from item i2)
Select i1.itemName, i1.price From item i1 where price = (select min(i2.price) from item i2)

findEmpIDForStore
Select e.empID from regularemployee e, store s where e.storeID = s.storeID and e.storeID =INPUT