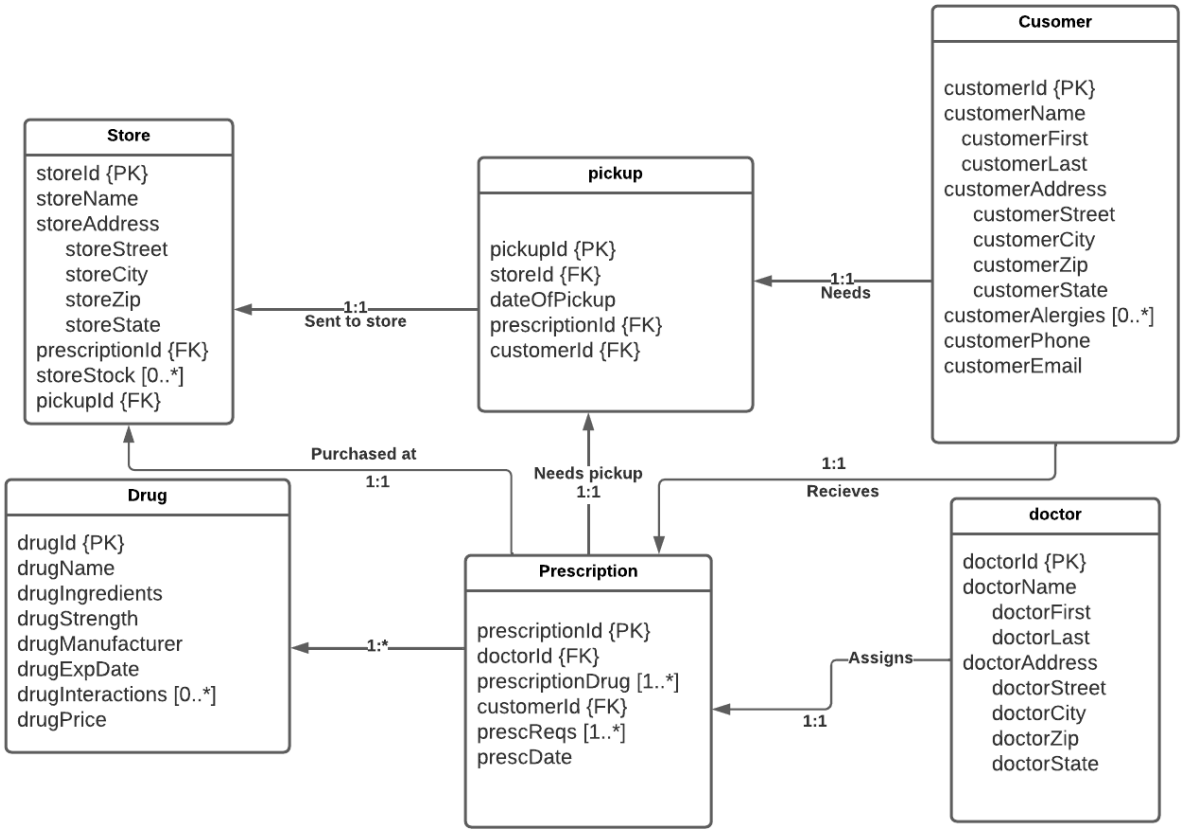
Alif Rahi
CSCI 331 / Section 32
Project 1

UML for CVS dataBase

Created by: Alif Rahi
Last 4 digits: 2468



Alif Rahi
CSCI 331 / Section 32
Project 1

Convert UML to relations

Key

- Primary keys are <u>underlined</u>
- Foreign keys are *italic*
- Multivalued relations are written like this *[relation]*

<u>Relations</u>**:**

Drug (<u>drugId</u>*,* drugName, drugIngredients, drugStrength, drugManufacturer, drugExpDate, drugPrice)

- This relation doesn't include the drugInteractions. I made another relation for it.

drugInteractions (<u>drugInteractionId</u>, *drugId*, drugInteractionName)

- This relation consists of all types of drug reactions to particular substances.
- Eg. drugAllergy, drugFood, drugHerb, drugDisease, drugGene..
- This relation is not included in the UML chart.

prescription (<u>prescriptionId</u>, *doctorId, customerId*, prescDate)

- This relation does not include the multivalued attribute prescReqs.
- Each prescription holds multiple drugs thus, I made a prescriptionDrug relation for that.

prescriptionDrug (<u>*drugId*</u>, <u>*prescriptionId*</u>)

- This relation has a composite key that maps multiple drugs per prescriptions.
- This relation is also not included in the UML chart.

prescReqs (<u>*prescriptionId*</u>, <u>*drugId*</u>, refills, dosage, frequency)

- This relation has a composite key consisting of 2 foriegn keys.
- This relation is needed to check refills, dosage and frequency of drug taken for each individual drug in prescriptions.
- This relation is not included in the UML chart.

Customer (<u>customerId</u>, customerName, customerFirst, customerLast, customerAddress, customerStreet, customerCity, customerZip, customerState)

- This relation is for all the customers in the CVS dataDase.

Alif Rahi
CSCI 331 / Section 32
Project 1

customerAllergies (*customerId, drugInteractionId)*
- This relation has 2 primary foreign keys as the composite primary key.
- This relation is used to keep track of each customer's drug allergies.
- Every customer doesn't have to be in this relation.

Store (storeId, storeName, storeAddress, storeStreet, storeCity, storeZip, storeState)
- This relation is to

storeStock (storeStockId, *storeId*, *drugId*)
- This relation has two foreign keys.
- This will keep a track of the stock available in a CVS store.
- This relation is also not included in the UML chart.

pickup (pickupId, *storeId*, dateOfPickup, *prescriptionId*, *customerId*)
- This relation is used to determine dateOfPickup for specific prescriptions.

Doctor (doctorId, doctorName, doctorFirst, doctorLast, doctorAddress, doctorStreet, doctorCity, doctorZip, doctorState)
- This relation links to every prescription because every prescription has to be prescribed by a doctor.

1. View the 12 month prescription history for Joseph song. Display the customer name, drug name, date of pickup and strength.

$(Person) \leftarrow \sigma_{customerFirst='Joseph' \wedge customerLast='song'}(Customer)$

- Filters out the cutsomer relation for only customers who have the name 'jospeh song'

$(Last12months) \leftarrow \sigma_{prescription.prescDate \geq 3-2-21 \wedge Person.customerId = prescription.customerId} (prescription \times Person)$

- Filters out any prescriptions older then a year under Joseph songs records.

$(A) \leftarrow \sigma_{Last12months.prescriptionId = prescriptionDrug.prescriptionId} (prescriptionDrug \times Last12months)$

- Makes a new relation with the multivalued 'prescriptionDrug' relation.

$(B) \leftarrow \sigma_{Drug.drugId = A.drugId}(A \times drug)$

- Cartesian product to add drug attributes/columns that correspond to josephs prescriptions.

$(pickupDate) \leftarrow \sigma_{pickup.customerId = B.customerId}(B \times pickup)$

- Another cartesian product to add DateOfPickup column to the final answer.

$(Answer) \leftarrow \pi_{customerFirst,\ drugName,\ DateOfPickup,\ drugStrength}(pickupDate)$

---

2. Identify prescriptions available for pickup now at the CVS in 'Francis Lewis Blvd, Queens, NY 11357' for Sandra lee. Display the customer name, drug and strength.

$(Person) \leftarrow \sigma_{customerFirst = 'Sandra' \wedge customerLast = 'lee'} (Customer)$

- Creates a new relation for only Sandra lee.

$(Today) \leftarrow \sigma_{Person.customerId = prescription.customerId} (prescription \times Person)$

- Filters all prescriptions that belong to Sandra.

$(pickupToday) \leftarrow \sigma_{Today.customerId = pickup.customerId \wedge pickup.dateOfPickup = 3-2-22}(Today \times pickup)$

- Creates new columns with pickup attributes and filters all prescriptions that need to be picked up on 3-2-22.

$(whichStore) \leftarrow \sigma_{store.storeId = pickupToday.storeId \wedge store.storeZip = '11357' \wedge store.storeStreet = 'Francis\ Lewis\ Blvd'} (store \times pickupToday)$

- Creates new columns with store attributes for Sandras prescriptions available at the designated CVS

$(Presc) \leftarrow \sigma_{prescriptionDrug.prescriptionId = whichStore.prescriptionId} (prescriptionDrug \times whichStore)$

- Creates a new cartesian product with the multivalued 'prescriptionDrug' relation.

$(DrugsPickup) \leftarrow \sigma_{Presc.drugId = drug.drugId} (Presc \times drug)$

- Creates a new cartesian product to add drug attributes/columns that correspond to each prescription.

$(Answer) \leftarrow \pi_{customerFirst,\ customerLast,\ drugName,\ drugStength}(DrugsPickup)$

---

3. Identify prescriptions that require refilling for Andrew keblish. Display the drug name, number of remaining refills and doctor.

$(Person) \leftarrow \sigma_{customerFirst = 'Andrew' \wedge customerLast = 'keblish'} (Customer)$

- Creates a new relation for Andrew keblish.

$(A) \leftarrow \sigma_{Person.customerId = prescription.customerId} (Person \times prescription)$

- Selects all prescriptions for Andrew keblish.

$(allPresc) \leftarrow \sigma_{A.prescriptionId = prescriptionDrug.prescriptionId} (A \times prescriptionDrug)$

- Creates new relation with the multivalued 'prescriptionDrug' relation.

$(refills) \leftarrow \sigma_{allPresc.prescriptionId = prescReqs.prescriptionId \wedge prescReqs.refills > 0} (allPresc \times prescReqs)$

- Creates new relation with the multivalued 'prescReqs' relation. We disclude any prescription that doesnt need refills.

$(DrugsPickup) \leftarrow \sigma_{refills.drugId = drug.drugId}(refills \times drug)$

- Makes a new cartesian product of all the drugs that each prescription prescribes.

$(customerDoc) \leftarrow \sigma_{doctor.doctorId = DrugsPickup.doctorId}(DrugsPickup \times doctor)$

- Cross products the doctor relation to add the doctor columns for the final answer.

$(Answer) \leftarrow \pi_{drugName,\ drugRefills,\ doctorFirst,\ doctorLast}(customerDoc)$

---

4. Identify drugs not sold in the last month at CVS location flushing 11354. Display the drug name, quantity.

$(theStore) \leftarrow \sigma_{store.storeZip = '11354' \ \wedge \ store.storeCity = 'flushing'}(store)$

- Locates the correct address of the store to narrow down searching.

$(Sold) \leftarrow \sigma_{theStore.prescriptionId = prescription.prescriptionId \ \wedge \ prescription.prescDate \geq 2-2-22}(theStore \times prescription)$

- Finds the prescriptions in the store from the last month.

$(thePrescriptions) \leftarrow \sigma_{Sold.prescriptionId = prescriptionDrug.prescriptionId}(Sold \times prescriptionDrug)$

- Since each prescription can include multiple drugs, we join this relation with the prescrptionDrug relation.

$(theDrugs) \leftarrow \sigma_{thePrescriptions.drugId = drug.drugId}(thePrescriptions \times drug)$

- Adds all the drug attributes.

$(notSold) \leftarrow \pi_{drugId}(drug) - \pi_{drugId}(theDrugs)$

- Subtracts all the drugs sold this month from the (available) drugs.

$(display) \leftarrow \sigma_{drug.drugId = notSold.drugId}(drug \times notSold)$

- This will add drug attributes

$(amount) \leftarrow \sigma_{display.storeId = storeStock.storeId}(display \times storeStock)$

- This will allow us to count the quantity with an aggrigate function.

$\rho_{answer}(Drug\ name,\ quantity)_{drugName}\ \Im_{count\ drugId}(amount)$

---

5. Identify customers without a drug purchase in the last year. Display the customer name, address, email and phone.

$(purchased) \leftarrow \sigma_{prescription.prescDate \geq 3-2-21}(prescription)$

- This will filter all the prescription in all of CVS's databases who have purchased something in the last year.

$(notPurchased) \leftarrow \pi_{customerId}(customer) \ - \ \pi_{customerId}(purchased)$

- This will find the complement by subtracting the customerIds.

$(display) \leftarrow \sigma_{notPurchased.customerId = customer.customerId}(notPurchased \times customer)$

- This will add customer columns to appropriate Ids.

$(Answer) \leftarrow \pi_{customerFirst,\ customerLast,\ customerStreet,\ customerCity,\ customerZip,\ customerEmail,\ customerPhone}(display)$

---

6. Identify drugs with low inventory now at CVS 886, flushing. Display the drug name, manufacturer, drug id, store and quantity.

$(theStore) \leftarrow \sigma_{store.storeName = 'CVS\ 886' \ \wedge \ store.storeCity = 'flushing'}(store)$

- Locates the correct address of the store to narrow down searching.

$(inventory) \leftarrow \sigma_{theStore.storeId = storeStock.storeId}(theStore \times storeStock)$

- We find the available stock at the CVS 886, flushing.

$(atts) \leftarrow \sigma_{inventory.drugId = drug.drugId}(inventory \times drug)$

- We join the drug relation with our stores stock.

$(Answer) \leftarrow \rho_{answer}(drugName,\ drugManufacturer,\ drugId,\ storeName,\ quantity)_{drugName,\ drugManufacturer,\ drugId,\ storeName}\Im_{count\ drugId}(atts$

- We count all the drugs in our specific store.

$(finalAns) \leftarrow \pi_{drugName,\ drugManufacturer,\ drugId,\ storeName,\ quantity}(\sigma_{\ quantity < 5}\ (Answer))$

---

7. Identify drugs that will expire this week at CVS bayside, 11355. Display the drug name, manufacturer, drug id, store and quantity.

$(theStore) \leftarrow \sigma_{store.storeZip = '11355' \ \wedge\ store.storeCity = 'bayside'}\ (store)$

- Locates the correct address of the store to narrow down searching.

$(inventory) \leftarrow \sigma_{\ theStore.storeId = storeStock.storeId}\ (theStore \times storeStock)$

- We find the available stock at the CVS 886, flushing.

$(expiring) \leftarrow \sigma_{\ inventory.drugId = drug.drugId\ \wedge\ drug.drugExpDate \leq 3-9-22}\ (inventory \times drug)$

- We select all drugs expiring this week.

$\rho_{answer}(drugName,\ drugManufacturer,\ drugId,\ storeName,\ quantity)_{drugName,\ drugManufacturer,\ drugId,\ storeName}\Im_{\ count\ drugId}(expiring)$

---

8. Identify customers with allergies to current medication. Display the customer name, drug name, strength and quantity.

$(people) \leftarrow \sigma_{customerAllergies.customerId = customer.customerId}\ (customerAllergies \times customer)$

- Cartesian product with the customerAllergies relation to only find customers with allergies to current medication.

$(allergies) \leftarrow \sigma_{drugInteractions.drugInteractionId = people.drugInteractionId}\ (people \times drugInteractions)$

- Add the drug interactions relation with people who have allergies.

$(peopleDrugs) \leftarrow \sigma_{allergies.drugId = drug.drugId}\ (allergies \times drug)$

- Add the drug relation.

$(stock) \leftarrow \sigma_{storeStock.drugId = peopleDrugs.drugId}\ (peopleDrugs \times storeStock)$

- Find the stock of the drug in from stores available.

$\rho_{answer}(customerFirst,\ customerLast,\ drugName,\ drugStrength,\ quantity)_{customerFirst,\ customerLast,\ drugName,\ drugStrength}\ \Im_{\ count\ drugId}(stock)$

---

9. Identify products sold today at CVS 285, NY, 11498. Display the customer name, drug name, strength, cost and store.

$(theStore) \leftarrow \sigma_{store.storeZip = '11498' \ \wedge\ store.storeState = 'NY' \ \wedge\ store.storeName = '285'}(store)$

- Locates the correct address of the store to narrow down searching.

$(presc) \leftarrow \sigma_{prescription.prescriptionId = theStore.prescriptionId\ \wedge\ prescription.prescDate = 3-2-22}(prescription \times theStore)$

- Finds prescriptions sold at the store today.

$(customerPresc) \leftarrow \sigma_{presc.customerId = customer.customerId}\ (presc \times customer)$

- Adds the customer attributes.

$(thePresc) \leftarrow \sigma_{customerPresc.prescriptionId = prescriptionDrug.prescriptionId}\ (customerPresc \times prescriptionDrug)$

- Finds all drugs sold by each prescription.

$(theDrugs) \leftarrow \sigma_{presc.drugId = drug.drugId}\ (thePresc \times drug)$

- Adds the drug columns/attributes.

$(Answer) \leftarrow \pi_{\ customerFirst,\ customerLast,\ drugName,\ drugStrength,\ drugPrice,\ storeName}\ (theDrugs)$

---

10. Identify total revenue and [drug] by neighborhood. Display 3 columns: neighborhood, total prescriptions and total sales.

$(allStores) \leftarrow \sigma_{store.prescriptionId = prescription.prescriptionId}\ (store \times prescription)$

- Find all prescriptions from all stores.

$(allPresc) \leftarrow \sigma_{allStores.prescriptionId = prescriptionDrug.prescriptionId}\ (allStores \times prescriptionDrug)$

- Find all drugs that match to each prescription.

$(allDrugs) \leftarrow \sigma_{allPresc.drugId = drug.drugId}\,(allPresc \times drug)$

- We need this step to obtain the drugPrice attribute which we will need to use an aggregate function on in the answer.

$(Answer) \leftarrow \rho_{answer}(Neighborhood,\ total\ prescriptions,\ total\ sales)\ {}_{storeZip}\,\Im_{\ count\ prescriptionId,\ sum\ drugPrice}(allDrugs)$

---

11. Identify the number of active customers in your neighborhood. Display 1 row with the number of active customers.

$(people) \leftarrow \sigma_{customerZip = '11354'}\,(customer)$

- My zipcode is 11354 and thats how I searched my area.

$(active) \leftarrow \sigma_{prescription.customerId = people.customerId\ \wedge\ prescription.prescDate \geq 3\text{-}2\text{-}21}(people \times prescription)$

- Kept only the people who have purchased a drug in the last year.

$(Answer) \leftarrow\ \Im_{\ count\ customerId}(active)$

---

12. Identify the number of prescriptions by doctor. Display 3 columns: doctor name, number of prescriptions and total sales.

$(presc) \leftarrow \sigma_{prescription.prescriptionId = prescriptionDrug.prescriptionId}\,(prescription \times prescriptionDrug)$

- Find all prescriptions and its prescripted drugs.

$(prescDrugs) \leftarrow \sigma_{presc.drugId = drug.drugId}\,(presc \times drug)$

- Join the drug relation so we can sum the drugPrice attribute in the answer.

$(prescDoc) \leftarrow \sigma_{prescDrugs.doctorId = doctor.doctorId}\,(prescDrugs \times doctor)$

- Join the doctor relation so we can group the aggregate function by the doctor name attributes.

$(Answer) \leftarrow \rho_{answer}(first\ name,\ last\ name,\ total\ prescriptions,\ total\ sales)\ {}_{doctorFirst,\ doctorLast}\,\Im_{\ count\ prescriptionId,\ sum\ drugPrice}(prescDoc)$