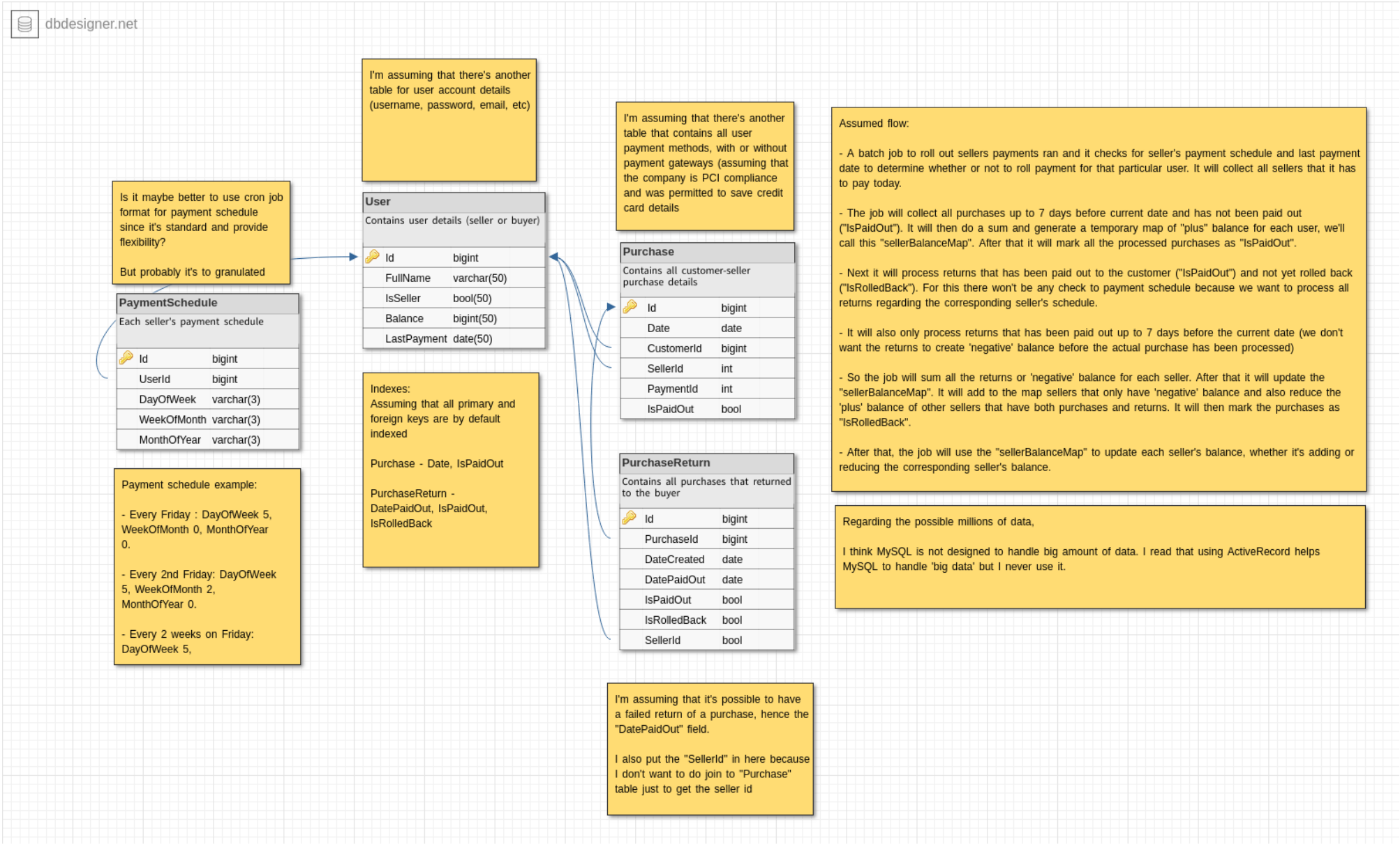
dbdesigner.net
I'm assuming that there's another table for user account details (username, password, email, etc)
Is it maybe better to use cron job format for payment schedule since it's standard and provide flexibility?
But probably it's to granulated
User
Contains user details (seller or buyer)
Id bigint
FullName varchar(50)
IsSeller bool(50)
Balance bigint(50)
LastPayment date(50)
I'm assuming that there's another table that contains all user payment methods, with or without payment gateways (assuming that the company is PCI compliance and was permitted to save credit card details
Assumed flow:
- A batch job to roll out sellers payments ran and it checks for seller's payment schedule and last payment date to determine whether or not to roll payment for that particular user. It will collect all sellers that it has to pay today.
- The job will collect all purchases up to 7 days before current date and has not been paid out ("IsPaidOut"). It will then do a sum and generate a temporary map of "plus" balance for each user, we'll call this "sellerBalanceMap". After that it will mark all the processed purchases as "IsPaidOut".
- Next it will process returns that has been paid out to the customer ("IsPaidOut") and not yet rolled back ("IsRolledBack"). For this there won't be any check to payment schedule because we want to process all returns regarding the corresponding seller's schedule.
- It will also only process returns that has been paid out up to 7 days before the current date (we don't want the returns to create 'negative' balance before the actual purchase has been processed)
- So the job will sum all the returns or 'negative' balance for each seller. After that it will update the "sellerBalanceMap". It will add to the map sellers that only have 'negative' balance and also reduce the 'plus' balance of other sellers that have both purchases and returns. It will then mark the purchases as "IsRolledBack".
- After that, the job will use the "sellerBalanceMap" to update each seller's balance, whether it's adding or reducing the corresponding seller's balance.
PaymentSchedule
Each seller's payment schedule
Id bigint
UserId bigint
DayOfWeek varchar(3)
WeekOfMonth varchar(3)
MonthOfYear varchar(3)
Indexes:
Assuming that all primary and foreign keys are by default indexed
Purchase - Date, IsPaidOut
PurchaseReturn - DatePaidOut, IsPaidOut, IsRolledBack
Purchase
Contains all customer-seller purchase details
Id bigint
Date date
CustomerId bigint
SellerId int
PaymentId int
IsPaidOut bool
PurchaseReturn
Contains all purchases that returned to the buyer
Id bigint
PurchaseId bigint
DateCreated date
DatePaidOut date
IsPaidOut bool
IsRolledBack bool
SellerId bool
Payment schedule example:
- Every Friday : DayOfWeek 5, WeekOfMonth 0, MonthOfYear 0.
- Every 2nd Friday: DayOfWeek 5, WeekOfMonth 2, MonthOfYear 0.
- Every 2 weeks on Friday: DayOfWeek 5,
Regarding the possible millions of data,
I think MySQL is not designed to handle big amount of data. I read that using ActiveRecord helps MySQL to handle 'big data' but I never use it.
I'm assuming that it's possible to have a failed return of a purchase, hence the "DatePaidOut" field.
I also put the "SellerId" in here because I don't want to do join to "Purchase" table just to get the seller id