Object Description document:

View 1:balance\_not\_paid\_customer

we will send the bill to customer at the first day of each month and text customer at the middle of the month if they have unpaid balance. So we need to select customer with unpaid balance each month.

View 2: deliver\_management

We will show the leadership relationship for the delivery manager and deliver man in a much clearer way in this view.

Trigger 1: customer\_pay\_balance

when the customer pay the balance, we will reduce the current\_balance, but we have to make sure they pay all required balance, not partial. This insert trigger will be triggered when the customer wants to pay partial balance and will prevent this invalid update.

Trigger 2: valid\_new\_contract

A customer can sign a new contract when he is active and have no unpaid balance. In other words, we will prevent inserting new role to contract table if the customer’s status is not active or he has unpaid balance.

Trigger 3: new\_item\_check

When the customer wants to have new order, they should satisfy two condition, the quantity of the item is smaller than the order quantity and the customer current balance is not locked. when they satisfy this two condition,we update item quantity. If not, we prevent inserting this new role to deliver item table.

Procedure 1:

The procedure input is the bill ID. The procedure is used to generate the customer ID and bill start date, which is going to be the input of procedure 2 bill.

Procedure 2: bill

The procedure input is the customer ID and bill start date, and bill id. It is used to generate the information in the bill. It includes Location, monthly fee, and also every items’ detail, such as item ID, quantity, description, price, discount and total fee, belong to certain location, certain month, and certain customer. Also, the procedure calculates two value as output, one is the total fee of all items the customer bought in the month. The other is the total monthly fee.

Procedure 3 : total\_bill

The procedure is used to print the data of bill, including Total charge, discount, tax, amount due, past due, late fee, total bil, and the due date of bill.

Procedure 4: call\_package

The procedure is used to call package 1.

Procedure 4: Add\_Bill

The procedure is used to add billing information for each customer every month. All billing information of the customers will be included in this bill table.

Function 1: total\_charge

The function input is each customer’s sum of monthly fee of each branch and the sum of all items each branch bought in this month. The output is the total charge.

Function 2: discount

The function will count the sum of customers referred by another customer. For each referral, we will give the customer $25 discount for the current bill.

Function 3: late\_fee

We will compute monthly bill in the first day of the next month, and the due date of last month’s bill is the last date of the next month. Thus, if the balance in customer is not 0 means they didn’t pay the bill on time, and they need to pay the penalty of 15% of the cumulative amount in the balance.

Package 1:

The package contains Procedure 1, Procedure 2, Procedure 3 and Function 1. And we use Procedure 4 to call the package.

Role 1: manager\_5

Role 2: driver\_6

These two roles have different level of permissions of viewing and modifying table of evaluation. Here, we grant SELECT, INSERT, UPDATE, DELETE privilege to manager\_5, while only grant SELECT to driver\_6. Therefore, manager\_5 can view and modify table evaluation, while driver\_6 can only view table evaluation but not evaluate it.

Job: PTEST

The job automatically adds billing information of customers in the table of bill. The attributes of END\_DATE and DATE\_DUE automatically change to last day of next month.

Also, you can use the following two lines of code to change interval from monthly to minutely to verify our job work properly.

exec dbms\_scheduler.set\_attribute('PTEST','repeat\_interval','FREQ=MINUTELY;INTERVAL=1');

exec dbms\_scheduler.run\_job('PTEST');

Alternate Index: index\_name

The index of customer\_name serves as alternate index. Together with index created automatically by primary key, this alternate index improves searching efficiency by applying dichotomy instead of searching one by one.

De-normalization

One deliver\_service report can have multiple pick up item and delivery item. We normalize them into three tables. Therefore, we need search three tables to get the information for just one service report. Therefore, we join three table together to provide a clearer representation for deliver service.