Изисквания :

1/ обща Oracle база за проекта, ще се предостави на всеки екип

2/ ПроС програмите са разработват на линукс.

Alter table customers add external\_reference varchar2(60);

update customers set external\_reference = 'EXT' || customer\_id where external\_reference  is null ;

//Да се добави unique not null constraint към полето customers.external\_reference.

//За стари съществуващи данни да е във формат EXT3427.

Alter table payments add original\_payment\_id varchar2(60); -> key kym payment\_id

Alter table invoices add original\_invoice\_id varchar2(60); -> key kym inv\_id

// original\_invoice\_id и original\_payment\_id трябва да съдържат съответно само стойности от полетата invoice\_id и payment\_id или стойност NULL.  
 Да се създаде такава релация в базата от данни.

Още една таблица :  
create table over\_payments (

invoice\_no varchar2(50),

payment\_id varchar2(60),

payment\_dt date,

amount number(10,2),

payment\_method varchar2(10),

currency varchar2(3),

fg\_processed varchar2(1),

iban varchar2(60),

bank\_name varchar2(100),

created\_by varchar2(50),

free\_text varchar2(1000),

original\_payment\_id varchar2(60),

constraint op\_paym\_no\_pk primary key (payment\_id) ,

CONSTRAINT op\_fk\_inv FOREIGN KEY (invoice\_no) REFERENCES invoices(inv\_no)

);

alter table over\_payments

add constraint fk\_over\_pay\_id\_orig\_paym\_id

foreign key (original\_payment\_id) references payments(payment\_id)

deferrable initially deferred;

# Задание

1. Създайте входен интерфейс(програма, която чете данни от файлове, обработва данните и ги слага в базата). Интерфейсът чете файлове, които се намират в предварително дефинирана (от вас) директория и имат име във формат IMP\_CUSTOMER\_DATA\_YYYYMMDDHH24MISS.txt, където YYYYMMDDHH24MISS е датата с часове, минути и секунди, например 20230214183755. Ако в директорията има повече от един файл се обработват по ред като се започва от най-стария спрямо датата в името на файла. Всеки файл се чете линия по линия. Отварят се два файла в друга директория със същото име и разширение .OK и .KO . Всяка успешно обработена линия се записва във файла с разширение .OK, а всяка грешна се записва във файла .KO. Веднъж обработен, файлът се мести от основната директория в друга директория с разширение .back   
   Във всеки файл може да има 3 вида линии – клиент, фактура, плащане. Първото поле от линията определя нейния тип (01 – клиент, 02 - фактура, 03 - плащане). Всяко поле е разделено със символ **;** ( символът **;** не може да се съдържа в данните).

Примерен файл:

01;;EXT3427;VALENTIN;MILANOV;m;SOFIA;BULGARIA;998;v-milanov@jjj.gg

01;79;EXT1473;IVELINA;SIMOVA;f;;BULGARIA;5000;ivvvv45@jjj.gg

02;EXT3427;15022023;60.00;BGN

02;EXT3427;17022023;20.00;BGN

03;EXT3427;16022023;70;BankTransf;BGN;

Формат на линиите :

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | Тип на линията | 2 символа, цифри, задължително поле | 01 – клиент |
| 2 | Вътрешна референция на клиента | Макс. 20 символа, цифри,не задължително поле | Ех. 456 |
| 3 | Външна референция на клиента | Макс. 60 символа, цифри и букви, задължително поле | Ех. EXT3427 |
| 4 | Първо име | Макс. 60 символа, задължително поле | Ех. IVELINA |
| 5 | Фамилия | Макс. 60 символа, задължително поле | Ех. SIMOVA |
| 6 | Пол | 1 символ, задължително поле | Стойност m или f |
| 7 | град | Макс. 50 символа, не задължително поле | Еx. SOFIA |
| 8 | Държава | Макс. 30 символа, задължително поле | Ex. BULGARIA |
| 9 | Лимит | Макс. 10 символа, цифри, задължително поле | Ex. 5000 |
| 10 | Email | Макс. 50 символа, задължително поле | Ex. ivvvv45@jjj.gg |

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | Тип на линията | 2 символа, цифри, задължително поле | 02 – фактура |
| 2 | Външна референция на клиента | Макс. 60 символа, цифри и букви, задължително поле | Ех. EXT3427 |
| 3 | Дата на фактурата | 8 символа, задължително поле, формат DDMMYYYY | Ех. 15022023 |
| 4 | Сума на фактурата | До 10 цифри преди точката и до 2 след точката , задължително поле | Ех. 60.00 |
| 5 | Валута на фактурата | 3 символа, задължително поле | Ех. BGN |

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | Тип на линията | 2 символа, цифри, задължително поле | 03 – плащане |
| 2 | Външна референция на клиента | Макс. 60 символа, цифри и букви, задължително поле | Ех. EXT3427 |
| 3 | Дата на плащането | 8 символа, задължително поле, формат DDMMYYYY | Ех. 16022023 |
| 4 | Сума на плащането | До 10 цифри преди точката и до 2 след точката , задължително поле | Ех. 60.00 |
| 5 | Начин на плащене | До 10 символа, задължително поле | Ех. BankTransf |
| 6 | Валута на плащането | 3 символа, задължително поле | Ех. BGN |

Обработка на линиите :

01 – Клиент

Проверява се дали има стойност в полето “Вътрешна референция на клиента”

* Ако да (попълнено e), тогава се търси линия в базата в таблицата CUSTOMERS със същото customer\_id.
  + Ако не съществува клиент с това ID, отхвърляме линията
  + Ако съществува клинте със същото ID, тогава проверяваме дали някое от следващите полета в линия е различно от данните в базата. Ако да, ги променяме
* Ако не (полето “Вътрешна референция на клиента” е празно), тогава търсим в базата клиент (таблица CUSTOMERS), който external\_reference = “Външна референция на клиента” (трето поле от линията).
  + Ако не съществува, тогава инсъртаваме нов клиент с всички данни от линията (ред в таблицата CUSTOMERS)
  + Ако съществува клинте със същия external\_reference, тогава проверяваме дали някое от следващите полета в линия е различно от данните в базата. Ако да, ги променяме

02 – фактура

Търсим линия в CUSTOMERS, където customers.external\_reference е равна на стойността в полето “Външна референция на клиента”. Ако намерим, създаваме фактура за този клиент. Ако не намерим, отхвърляме линията с грешка “Фактура по несъществуващ клиент”.

03 – плащане

Търсим линия в CUSTOMERS, където customers.external\_reference е равна на стойността в полето “Външна референция на клиента”.

* Ако не намерим, отхвърляме линията с грешка “Плащане от несъществуващ клиент”.
* Ако намерим клиента, слагаме плащането по последната фактура на този клиент. Ако клиента няма нито една фактура, отхвърляме линията.

*// като дойде плащане винаги ще отива по последната фактура – виж долу тригера трябва да го хване и оправи, дори фактурата да е платена, ще трябва да му смени сумата на 0 и да сложи друго вързано към това с цялата сума и NULL за Invoice\_no*

1. Напишете процедура – която да попълни полето invoice.due\_date, за всички фактури където това поле е NULL по правилата : ако за същия клиент полето credit\_limit е по-малко от 5000, тогава 10 дни след датата на издаване на фактурата; ако credit\_limit е между 5000 и 8000, тогава 15 дни след датата на издаване на фактурата и ако е по-голямо от 8000, тогава 20 дни.
2. Напишете тригер, който при инсърт в таблицата PAYMENTS ~~или ъпдейт на полето PAYMENTS.invoice\_no от NULL на стойност различна от NULL, тогава~~ ако сумата е по-голяма от дължимата сума по фактурата да направи сумата на плащането да е колкото се дължи по фактурата. С разликата от сумата се инсъртва ~~друго плащане~~ линия в таблицата over\_payments, където original\_payment\_id = payment\_id на главния запис и invoice\_no is NULL. Ако се промени дължимата стойноста на фактура се променя и в базата.
3. Да се напише бач (ПроС програма, която обработва данни), който за всички ~~плащания, където invoice\_no is NULL~~ надплатени суми (линии в таблицата over\_payments) търси фактура, коята ~~това плащане може да покрие~~ е надплатена за същия клиент и ако намери ~~­­­­­­­­­­­­­променя invoice\_no да е номера на фактурата~~ мести (копира данните и след това изтрива) линията от таблицата over\_payments в таблицата payments със попълнен номер на фактура.
4. Да се напише бач, който да слагага лихви – линия в таблицата Invoices по правилото – за всички фактури, които не са напълно платени и са минали 5 дни от датата на която се дължат, се слага фактура за същия клиент, която е на стойност 5% от дължимата сума по фактурата. original\_invoice\_id на новата фактура е = invoice\_id на главния запис. Да не слагаме лихва върху лихвата (където original\_invoice\_id е попълнено не се взимат тези линии)­

//4. и 5. Да се дообави входен параметър customer на бачовете (т.е. customers.external\_reference). Ако е попълнен, бачовете работят само по данни свързани с входния параметър customer, в противен случай обработват всички customers.

1. Да се напише репорт (ПроС програма, която извежда данни от базата по определени критерии и ги извежда във файл), който може да получи първи параметър client или period. Ако първия параметър е client, тогава следващия трябва да е ID на клиента (customer\_id). При първи параметър period следват 2 параметъра с две дати date\_from и date\_to.   
   Когато репортът е пуснат с параметър client се извеждат всички фактури, които има клиента с подаденото ИД, заедно с информация дали са платени или не.  
   Когато репортът е пуснат с параметър period да се извеждат имената и двете референции на клиентите, които не са направили нито едно плащане в посочения период.

Още изисквания:

Всички ПроС програми – интерфейси, бачове, репорти да имат лог файлове(всяка програма свои), но да използват общи функции за принтиране на съобщение и на грешка log\_msg и log\_err

log\_msg принтира първо датата с минути в секунди в скоби [] след това съобщението

log\_err притрира във формат [ERROR][дата с минути и секунти] съобщение

Да се започне с файл “Project Assignments”, който да е в репото на проекта. Пример за такъв файл от друг проект:

------ 1 ------ - Иво

1.Обработването на файла

2.Валидидране на формата на файла + връщане на броя редове + попълване на датата в структурата

3.Популиране на масивите за потребителско ИД, ПродуктИД и количество на продукти

4.Процедура 1 и Репорт 1

------ 2 ------ - Боби

1.Валидиране на данните от поръчката

2.Тригер за промяна на статус на поръчка

3.Добавяне на поръчката в БД

4.Местенето на файловете и в нужда апендването на файла с грешката + лог файл

------ 3 ------ - Адриан

1.Измисляне на данни в БД

2.Бач

3.Репорт 2 и Процедура 2

4.Меню