**Функциональные требования:**

В Банке имеется собственная особая система категоризации кредитных договоров. Можно задать принадлежность кредита к стандартной категории, либо к какой-либо льготной категории, которая определяет скидку при начислении комиссий по кредиту.

Для этого в системе необходимо реализовать настраиваемый справочник категорий и заполнить его по требованиям банка первоначальными значениями.

Принадлежность кредитного договора к той или иной категории указывается на карточке КД (кредитного договора). В течении жизни договора категория КД может изменяться, при этом необходимо сохранять историю изменения категории КД.

В зависимости от категории КД сумма комиссии за обслуживание кредита (в рамках данной задачи – периодическая комиссия, взимаемая ежемесячно) начисляется либо в полной сумме, либо в размере коэффициента от полной суммы, задаваемого на категории КД.

**Примечание**: везде, где указано S## – подразумевается S – константа, ## - номер из логина студента.

Префикс студента следует использовать:

* В коротких и полных наименованиях новых ТБП
* В коротких и полных наименованиях новых операций, создаваемых в дистрибутивных ТБП
* В коротких и полных наименованиях новых представлений, создаваемых в дистрибутивных ТБП

**Задание 1.** *Пп. 1, 2 освещаются в лекции Типы базовых понятий, п.3 - Синтаксис языка. Выборки и курсоры и Синтаксис языка.(Типы данных, выражения, методы, условный оператор, циклы), п. 4 - Представления*

* 1. Создать ТБП-справочник **«S##. Кредиты. Категория ссуды»** с реквизитами:
* Код - строка
* Наименование - строка
* вид кредита - ссылка на вид кредита
* Коэффициент - число

*@dictionary*

*@entity*('TYPE')

*@name*('S226. Кредиты. Категория ссуды')

**class** S226\_LOAN\_CTGR {

*@name*('Код') CODE [STRING];

*@name*('Наименование') NAME [STRING];

*@name*('Вид кредита') CREDIT\_TYPE [KIND\_CREDITS\_REF];

*@name*('Коэффициент') COEFFICIENT [NUMBER];

* 1. Автоматически сгенерировать операции **Добавить/Изменить/Удалить**
  2. В операциях **Добавить/Изменить** реализовать следующие проверки:
* Проверка уникальности кода (нельзя сохранить запись, если есть запись с таким же кодом)
* Проверка обязательности полей - Код, наименование, коэффициент (нельзя сохранить запись, если не заполнены обязательные поля)

**class** S226\_LOAN\_CTGR;

*@name*('Добавить')

*@tag*('CREATED')

**constructor** NEW\_AUTO **is**

NEW\_AUTO (

*@name*('Код') P\_CODE **in** [STRING],

*@name*('Наименование') P\_NAME **in** [STRING],

*@name*('Вид кредита') P\_CREDIT\_TYPE **in** [KIND\_CREDITS\_REF],

*@name*('Коэффициент') P\_COEFFICIENT **in** [NUMBER]

) **return** **ref** [S226\_LOAN\_CTGR] **is**

**validate** **is**

**begin**

**if** p\_message = 'DEFAULT' **then**

P\_CODE := this.*[CODE]*;

P\_NAME := this.*[NAME]*;

P\_CREDIT\_TYPE := this.*[CREDIT\_TYPE]*;

P\_COEFFICIENT := this.*[COEFFICIENT]*;

**end** **if**;

**end**;

**execute** **is**

**begin**

**if** P\_CODE **is** **null** **then**

**pragma** error('Не заполнен реквизит "Код"');

**end** **if**;

**if** P\_NAME **is** **null** **then**

**pragma** error('Не заполнен реквизит "Наименование"');

**end** **if**;

**if** P\_COEFFICIENT **is** **null** **then**

**pragma** error('Не заполнен реквизит "Коэффициент"');

**end** **if**;

**for** (

**select** a(1:a)

**in** ::[S226\_LOAN\_CTGR]**all**

**where** a.*[CODE]* = P\_CODE

)

**loop**

**pragma** error('Уже существует запись с таким кодом');

**end** **loop**;

-- Установка значения реквизита "Код"

*[CODE]* := P\_CODE;

-- Установка значения реквизита "Наименование"

*[NAME]* := P\_NAME;

-- Установка значения реквизита "Вид кредита"

*[CREDIT\_TYPE]* := P\_CREDIT\_TYPE;

-- Установка значения реквизита "Коэффициент"

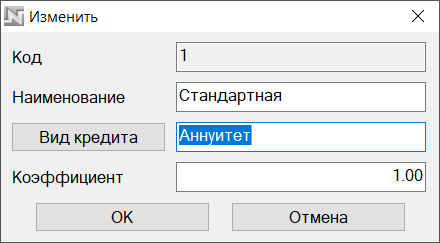
*[COEFFICIENT]* := P\_COEFFICIENT;

**end**;

**end**;

**end**;

**ЭФ операции:**

****

* 1. Создать представление для просмотра, в которое вывести колонки
* Код
* Наименование
* Вид кредита (вывести наименование)
* Коэффициент

**class** S226\_LOAN\_CTGR;

**@name**('S226. Список всех кредитов')

**view** VW\_CRIT\_S226\_LOAN\_CTGR {

**type** main **is**

**select** a(

a.*[CODE]* : C\_CODE,

a.*[NAME]* : C\_NAME,

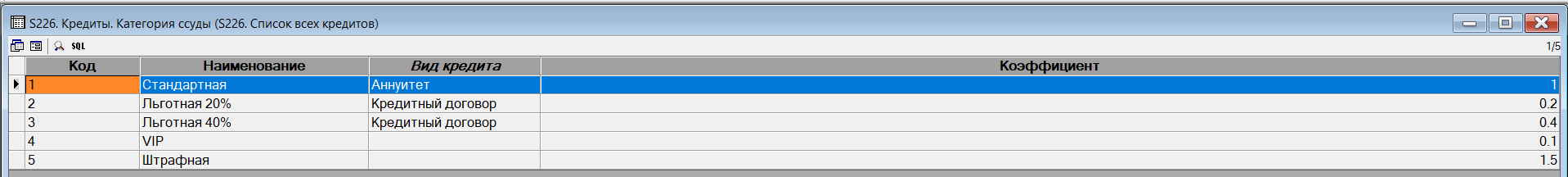
a.*[CREDIT\_TYPE]*.*[NAME]* : C\_CRED\_TYPE,

a.*[COEFFICIENT]* : C\_COEFF

)

**in** ::[S226\_LOAN\_CTGR];

}



**Задание 2.** *Пп. 1, 3 освещаются в лекции Типы базовых понятий, пп.4, 5 - Синтаксис языка. Выборки и курсоры и Синтаксис языка.(Типы данных, выражения, методы, условный оператор, циклы), п. 2 - Представления*

* 1. Создать ТБП-справочник **«S##. Кредиты. Принадлежность к категории»** с реквизитами:
* Кредит – ссылка на кредитный договор
* Дата начала – Дата
* Дата окончания – Дата
* Категория – ссылка на ТБП **«S##. Кредиты. Категория ссуды»**

*@dictionary*

*@entity*('TYPE')

*@name*('S226. Кредиты. Принадлежность к категории')

**class** S226\_CRED\_CATEG {

*@name*('Кредит') CREDIT [PR\_CRED\_REF];

*@name*('Дата начала') DATE\_BEG [DATE];

*@name*('Дата окончания') DATE\_END [DATE];

*@name*('Категория') CRED\_CAT [S226\_LOAN\_CT\_REF];

}

* 1. Создать представление для просмотра, в которое вывести колонки
* Кредит (вывести номер договора)
* Дата начала
* Дата окончания
* Категория (вывести наименование)
* Коэффициент

**class** S226\_CRED\_CATEG;

**@name**('S226. Кредиты. Принадлежность к категории')

**view** VW\_CRIT\_S226\_CRED\_CATEG\_VIEW {

**type** main **is**

**select** a(

prod.*[NUM\_DOG]* : c\_num\_dog,

a.*[DATE\_BEG]* : c\_date\_begin,

a.*[DATE\_END]* : c\_date\_end,

a.*[CRED\_CAT]*.*[NAME]* : c\_category\_name,

a.*[CRED\_CAT]*.*[COEFFICIENT]* : c\_coeff

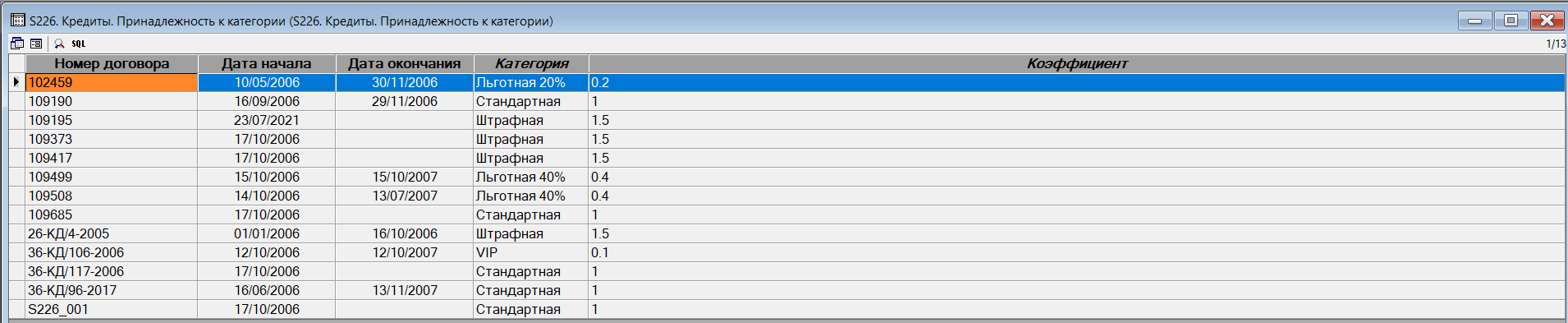
)

**in** ::[S226\_CRED\_CATEG],

(::[PRODUCT] : prod)

**where** a.*[CREDIT]***%id** = prod**%id**;

}



* 1. Автоматически сгенерировать операции **Добавить/Изменить/Удалить**
  2. В операциях **Добавить/Изменить** реализовать следующие проверки:
* Проверка обязательности полей – Кредит, Категория, дата начала
* Проверка пересечения дат – нельзя сохранить запись, если есть другая запись по данному кредиту, у которой «Дата начала» > «Даты начала» сохраняемой записи
* Проверка корректности дат – нельзя сохранить запись, у которой дата окончания < даты начала
  1. В операциях **Добавить/Изменить** необходимо реализовать следующее:
     + при сохранении записи, если по данному кредиту есть другая запись, у которой «Дата начала» < «Даты начала» сохраняемой записи и «Дата окончания» = Пусто, у этой записи проставляется «Дата окончания» = «Дате начала» сохраняемой записи минус 1 день
     + при сохранении записи, если по данному кредиту есть другая запись, у которой «Дата начала» = «Дате начала» сохраняемой записи, данная запись удаляется

**Задание 4 и 5:**

**class** S226\_CRED\_CATEG;

*@name*('Добавить')

*@tag*('CREATED')

**constructor** NEW\_AUTO **is**

NEW\_AUTO (

*@name*('Кредит') P\_CREDIT **in** [PR\_CRED\_REF],

*@name*('Дата начала') P\_DATE\_BEG **in** [DATE],

*@name*('Дата окончания') P\_DATE\_END **in** [DATE],

*@name*('Категория') P\_CRED\_CAT **in** [S226\_LOAN\_CT\_REF]

) **return** **ref** [S226\_CRED\_CATEG] **is**

**validate** **is**

**begin**

-- Start initialization of parameters and variables

**if** p\_message = 'DEFAULT' **then**

P\_CREDIT := this.*[CREDIT]*;

P\_DATE\_BEG := this.*[DATE\_BEG]*;

P\_DATE\_END := this.*[DATE\_END]*;

P\_CRED\_CAT := this.*[CRED\_CAT]*;

**end** **if**;

-- End initialization of parameters and variables

**end**;

**execute** **is**

**begin**

-- Проверка обязательности полей – Кредит, Категория, дата начала

**if** P\_CREDIT **is** **null** **then**

**pragma** error('Не заполнен реквизит "Кредит"');

**end** **if**;

**if** P\_CRED\_CAT **is** **null** **then**

**pragma** error('Не заполнен реквизит "Категория"');

**end** **if**;

**if** P\_DATE\_BEG **is** **null** **then**

**pragma** error('Не заполнен реквизит "Дата начала"');

**end** **if**;

/\* Проверка корректности дат – нельзя сохранить запись,

у которой дата окончания < даты начала \*/

**if** P\_DATE\_BEG > P\_DATE\_END **then**

**pragma** error('Дата начала не может быть больше даты окончания');

**end** **if**;

/\* Проверка пересечения дат – нельзя сохранить запись,

\* если есть другая запись по данному кредиту,

у которой «Дата начала» > «Даты начала» сохраняемой записи \*/

**for** (

**select** a(1 :a)

**in** ::[S226\_CRED\_CATEG]**all**

**where** a.*[DATE\_BEG]* > P\_DATE\_BEG **and** a.*[CREDIT]* = P\_CREDIT

)

**loop**

**pragma** error('По данному кредитному договору уже существует запись с датой начала позже указанной');

**end** **loop**;

-- при сохранении записи, если по данному кредиту есть другая запись,

-- у которой «Дата начала» < «Даты начала» сохраняемой записи и

-- «Дата окончания» = Пусто, у этой записи проставляется

-- «Дата окончания» = «Дате начала» сохраняемой записи минус 1 день

**declare**

record\_exists **ref**[S226\_CRED\_CATEG];

**begin**

**select** a(a**%id**)

**in** ::[S226\_CRED\_CATEG]**all**

**where** a.*[CREDIT]* = P\_CREDIT **and** a.*[DATE\_BEG]* < P\_DATE\_BEG **and** a.*[DATE\_END]* **is** **null**

**into** record\_exists;

**update** a (a.*[DATE\_END]* = P\_DATE\_BEG - 1) **in** ::[S226\_CRED\_CATEG] **where** a**%id** = record\_exists;

**exception**

**when** NO\_DATA\_FOUND **then**

**null**;

**when** TOO\_MANY\_ROWS **then**

**pragma** error('Найдено более одной записи с датой начала меньше указанной и пустой

датой окончания для данного кредитного договора');

**when** **others** **then**

**pragma** error('Необработанное исключение');

**end**;

-- при сохранении записи, если по данному кредиту есть другая запись,

-- у которой «Дата начала» = «Дате начала» сохраняемой записи, данная запись удаляется

**for** (

**select** a(a :id)

**in** ::[S226\_CRED\_CATEG]**all**

**where** a.*[DATE\_BEG]* = P\_DATE\_BEG **and** a.*[CREDIT]* = P\_CREDIT

)

**loop**

**delete** t **in** ::[S226\_CRED\_CATEG] **where** t**%id** = a.id;

**stdio**.put\_line\_buf('Запись по кредитному договору №' || P\_CREDIT || ' с такой же датой уже существует, поэтому она была удалена');

**end** **loop**;

-- Установка значения реквизита "Кредит"

*[CREDIT]* := P\_CREDIT;

-- Установка значения реквизита "Дата начала"

*[DATE\_BEG]* := P\_DATE\_BEG;

-- Установка значения реквизита "Дата окончания"

*[DATE\_END]* := P\_DATE\_END;

-- Установка значения реквизита "Категория"

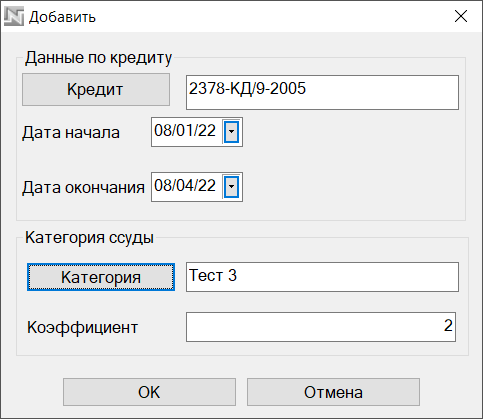
*[CRED\_CAT]* := P\_CRED\_CAT;

**end**;

**end**;

**end**;

**ЭФ операции:**

****

**Задание 3.** *Освещается в лекциях Операции, Подготовка хранилища с доработкой и Синтаксис языка. Выборки и курсоры*

Для подготовки передачи изменений на схему банка создать операцию нормализации в ТБП **«Обновления версии»** (CONV\_57), с коротким именем UYYYYMMDD\_S##\_NN (где S## – префикс студента, NN – номер вашей нормализации/конвертации по порядку), которая:

* 1. Создаст 5 произвольных записей в справочнике **«S##. Кредиты. Категория ссуды»** со ссылками на различные виды кредита (в том числе - с незаполненной ссылкой);

**Пример заполнения.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Код** | **Наименование** | **Вид кредита** | **Коэффициент** |
| 1 | Стандартная |  | 1 |
| 2 | VIP | Кредитный договор | 0.1 |
| 3 | Льготная 70% |  | 0.7 |
| 4 | Льготная 50% |  | 0.5 |

* 1. По действующим кредитным договорам («Дата закрытия» = Пусто) проставит категорию из справочника **«S##. Кредиты. Категория ссуды»** по произвольному алгоритму, исходя из даты открытия КД.

Проставлять категорию нужно только для «своих» кредитных договоров, для этого заведите вручную запись «S## Льготные кредиты» в справочнике **«Тип проекта» (KIND\_PROJECT)** и указывайте данный тип проекта при открытии договоров. При выполнении нормализации выбирайте КД только с данным типом проекта.

Операция должна допускать многократный повторный запуск.

-- Задание 3. Освещается в лекциях Операции, Подготовка хранилища с доработкой и Синтаксис языка. Выборки и курсоры

**class** CONV\_57;

*@name*('S226. Заполнение справочника')

**static** **method** U20220115S226\_01 **is**

**pragma** **include**(::[RUNTIME].[MACRO\_LIB]);

**procedure** Fill\_Category(p\_code S226\_LOAN\_CTGR.*[CODE]***%type**

,p\_name S226\_LOAN\_CTGR.*[NAME]***%type**

,p\_credit\_kind S226\_LOAN\_CTGR.*[CREDIT\_TYPE]***%type**

,p\_coef S226\_LOAN\_CTGR.*[COEFFICIENT]***%type**

) **is**

code\_exist **ref** [S226\_LOAN\_CTGR];

**begin**

[CONV].[LOG].WriteMess('Начало выполнения процедуры Fill\_Category');

**select** a(a)

**in** ::[S226\_LOAN\_CTGR]

**where** a.*[CODE]* = p\_code

**into** code\_exist;

[CONV].[LOG].WriteMess('В справочнике ' || ::[S226\_LOAN\_CTGR]**%classname** || ' уже есть запись с кодом ' || p\_code);

[CONV].[LOG].WriteMess('Окончание выполнения процедуры Fill\_Category');

**exception**

**when** NO\_DATA\_FOUND **then**

::[S226\_LOAN\_CTGR].*[NEW\_AUTO]*(*P\_CODE* == p\_code,

*P\_NAME* == name,

*P\_CREDIT\_TYPE* == p\_credit\_kind,

*P\_COEFFICIENT* == p\_coef);

[CONV].[LOG].WriteMess('В справочник ' || ::[S226\_LOAN\_CTGR]**%classname** || ' добавлена запись с кодом ' || p\_code);

[CONV].[LOG].WriteMess('Окончание выполнения процедуры Fill\_Category');

**when** TOO\_MANY\_ROWS **then**

[CONV].[LOG].WriteMess('В справочнике ' || ::[S226\_LOAN\_CTGR]**%classname** || ' найдено более одной записи

с кодом ' || p\_code);

[CONV].[LOG].WriteMess('Окончание выполнения процедуры Fill\_Category');

**when** **others** **then**

[CONV].[LOG].WriteMess('Необработанное исключение в операции ['||&METHOD$SNAME||'] '||&METHOD$NAME);

[CONV].[LOG].WriteMess('Окончание выполнения процедуры Fill\_Category');

**end**;

-- функция вернет true, если у кредита уже есть категория

-- вернет false, если нет категории

**function** Is\_Record\_Exist(credit [PR\_CRED\_REF]) **return** **boolean** **is**

category\_exist **ref** [S226\_CRED\_CATEG];

**begin**

**select** a(a)

**in** ::[S226\_CRED\_CATEG]**all**

**where** a.*[CREDIT]***%id** = credit**%id**

**into** category\_exist;

**return** **true**;

**exception**

**when** NO\_DATA\_FOUND **then**

**return** **false**;

**when** TOO\_MANY\_ROWS **then**

**return** **false**;

**when** **others** **then**

**return** **false**;

**end**;

-- присвоение категории кредитам с пустой датой окончания

**procedure** Fill\_Credit\_Cat **is**

s226\_prj\_kind [KIND\_PROJECT\_REF];

op\_date **date**;

cred\_cat [S226\_LOAN\_CT\_REF];

**begin**

[CONV].[LOG].WriteMess('Начало выполнения процедуры Fill\_Credit\_Cat');

-- находим дату ОД

op\_date := ::[system].*[op\_date]*;

-- находим нужный тип проекта

**locate** s226\_prj\_kind **in** ::[KIND\_PROJECT] **where** s226\_prj\_kind.*[CODE]* = 'S226\_SOFT\_LOANS';

**for** cred **in** [PR\_CRED]

**where** cred.*[DATE\_BEGIN]* **is** **not** **null**

**and** cred.*[DATE\_CLOSE]* **is** **null**

**and** cred.*[PRJ\_KIND\_CRED]* = s226\_prj\_kind

**loop**

-- по алгоритму высчитываем категорию и записываем в cred\_cat

**if** cred.*[DATE\_BEGIN]* = op\_date **then**

**locate** cred\_cat **in** ::[S226\_LOAN\_CTGR] **where** cred\_cat.*[CODE]* = '1';

**elsif** cred.*[DATE\_BEGIN]* < op\_date **and** cred.*[DATE\_BEGIN]* >= op\_date - 30 **then**

**locate** cred\_cat **in** ::[S226\_LOAN\_CTGR] **where** cred\_cat.*[CODE]* = '2';

**elsif** cred.*[DATE\_BEGIN]* < op\_date - 30 **and** cred.*[DATE\_BEGIN]* >= op\_date - 60 **then**

**locate** cred\_cat **in** ::[S226\_LOAN\_CTGR] **where** cred\_cat.*[CODE]* = '3';

**elsif** cred.*[DATE\_BEGIN]* < op\_date - 60 **and** cred.*[DATE\_BEGIN]* >= op\_date - 90 **then**

**locate** cred\_cat **in** ::[S226\_LOAN\_CTGR] **where** cred\_cat.*[CODE]* = '4';

**else**

**locate** cred\_cat **in** ::[S226\_LOAN\_CTGR] **where** cred\_cat.*[CODE]* = '5';

**end** **if**;

-- проверяем, есть ли у нас категория по даннному кредиту

**if** Is\_Record\_Exist(cred)= **true** **then**

-- запишем в лог, что запись уже существует

[CONV].[LOG].WriteMess('В справочнике ' || ::[S226\_CRED\_CATEG]**%classname** || ' уже есть категория по кредитному договору № ' || cred.*[NUM\_DOG]*);

**else**

-- если у кредита еще нет категории, зададим её

::[S226\_CRED\_CATEG].*[NEW\_AUTO]*(*P\_CREDIT* == cred,

*P\_DATE\_BEG* == cred.*[DATE\_BEGIN]*,

*P\_DATE\_END* == **null**,

*P\_CRED\_CAT* == cred\_cat);

[CONV].[LOG].WriteMess('В справочник ' || ::[S226\_CRED\_CATEG]**%classname** || ' была добавлена категория по кредитному договору № ' || cred.*[NUM\_DOG]*);

**end** **if**;

**end** **loop**;

[CONV].[LOG].WriteMess('Окончание выполнения процедуры Fill\_Credit\_Cat');

**end**;

U20220115S226\_01

**is**

**execute** **is**

credit\_type1 [KIND\_CREDITS\_REF];

credit\_type2 [KIND\_CREDITS\_REF];

**begin**

[CONV].[LOG].InitLog(&METHOD$SNAME||'\_mess.log',**true**);

[CONV].[LOG].WriteMess('Начало выполнения операции ['||&METHOD$SNAME||'] '||&METHOD$NAME);

-- находим тип кредита №1

**locate** credit\_type1 **in** ::[KIND\_CREDITS] **where** credit\_type1.*[CODE]* = '10\_1' **and** **rownum** = 1;

-- заполнение справочника категорий с типом кредита №1

Fill\_Category(01, 'Стандартная', credit\_type1, 1);

Fill\_Category(02, 'Льготная 20%', credit\_type1, 0.2);

-- находим тип кредита №2

**locate** credit\_type2 **in** ::[KIND\_CREDITS] **where** credit\_type2.*[CODE]* = '01' **and** **rownum** = 1;

-- заполнение справочника категорий с типом кредита №1

Fill\_Category(03, 'Льготная 40%', credit\_type2, 0.4);

-- заполнение справочника категорий с пустым типом кредита

Fill\_Category(04, 'VIP', **null**, 0.1);

Fill\_Category(05, 'Штрафная', **null**, 1.5);

-- присвоение категории кредитам

Fill\_Credit\_Cat();

[CONV].[LOG].WriteMess('Окончание выполнения операции ['||&METHOD$SNAME||'] '||&METHOD$NAME);

[CONV].[LOG].InitLog;

**end**;

**end**;

**end**; /\* конeц static method U20220115S226\_01 \*/

**Задание 4.** *Освещается в лекциях Операции, Методы расширения дистрибутивной логики и Синтаксис языка. Выборки и курсоры*

Для возможности задания Категории для кредитного договора (далее КД), необходимо выполнить следующие доработки:

* 1. Скопировать операцию конструктора (PR\_CRED.NEW\_DOG\_N) и редактора (PR\_CRED.EDIT\_LAST) ТБП **«Кредиты».**

Создать для копии каждой операции расширение, в котором предусмотреть возможность пользовательского выбора записи из справочника **«S##. Кредиты. Категория ссуды»**. Для этого на ЭФ (экранной форме операции) на закладке «Расширение» добавить кнопку **«S##. Категория»**, а также вывести «Наименование» и «Коэффициент» для категории.

При выборе категории из справочника **«S##. Кредиты. Категория ссуды»** должны предлагаться только записи, у которых «Вид кредита» пустой, либо соответствует виду кредита КД.

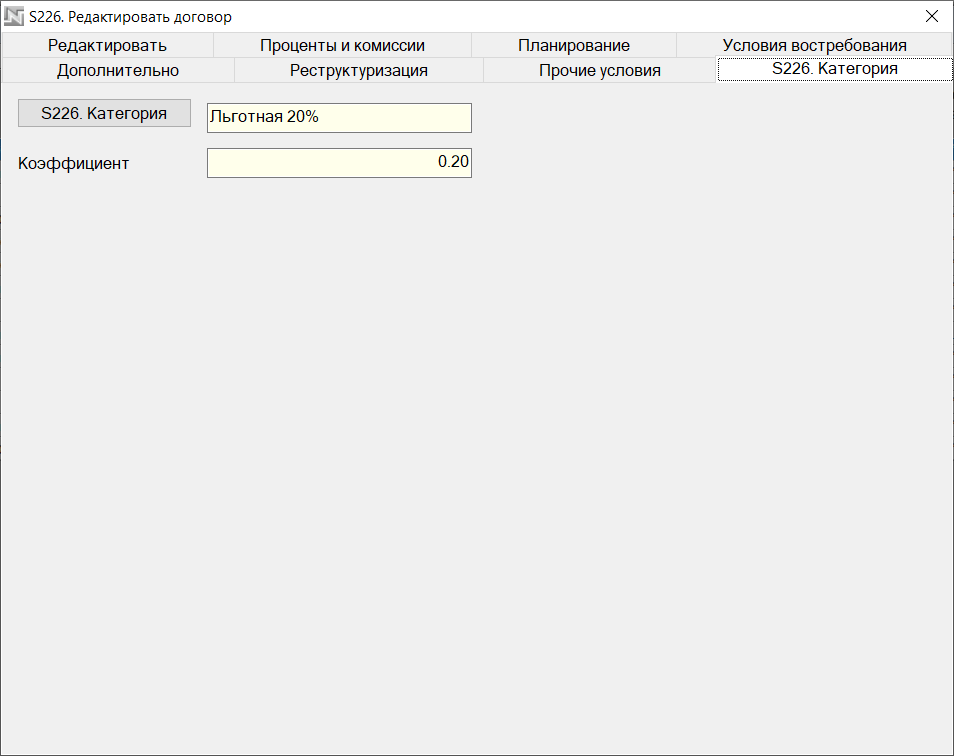
При поднятии ЭФ **«S##. Категория»** автоматически заполняется текущим значением категории КД на дату операционного дня филиала (далее - дата ОД), т.е. выбирается из справочника **«S##. Кредиты. Принадлежность к категории».**

**«S##. Категория»** - не обязательно для заполнения.

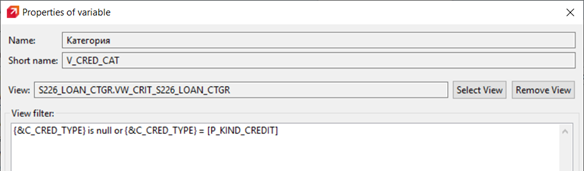
При сохранении ЭФ, значение поля **«S##. Категория»** необходимо сохранять в справочник **«S##. Кредиты. Принадлежность к категории»**. Реквизит «Дата начала» добавляемой записи заполнять значением текущей даты ОД, «Дата окончания» не заполняется.

При редактировании договора существующие до сохранения записи **«S##. Кредиты. Принадлежность к категории»** с датой начала более или равной дате текущего операционного дня филиала необходимо удалять. У существующей до изменения записи с пустым реквизитом «Дата окончания» данный реквизит заполнить значением (Дата ОД-1). В случае очистки значения **«S##. Категория»** на ЭФ новая запись не добавляется, изменяется «Дата окончания» для существующей по тому же алгоритму. Для добавления записей необходимо использовать конструктор справочника **«S##. Кредиты. Принадлежность к категории»**.

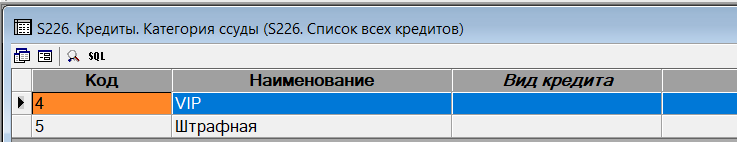
**ЭФ операции расширения:**



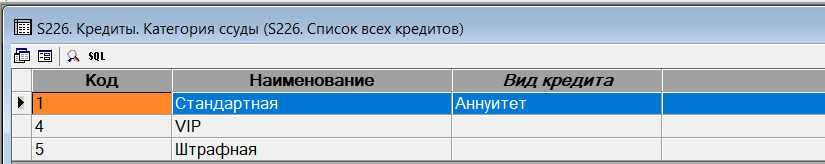
Установка фильтра для отображения категорий с пустым значением или значением, равным соответствующему виду КД



Работа фильтра с пустым видом КД



Работа фильтра с непустым видом КД



**Расширение операции конструктора КД:**

**class** PR\_CRED;

*@import\_plsql*(**true**)

*@this*(create\_on\_start:=**true**)

*@name*('S226. Открыть договор') /\* Свойство операции "Наименование" \*/

*@tag*('EXTENSION')

**constructor** S226\_NEW\_DOG\_EXT **instead** **of** S226\_NEW\_DOG **is**

-- категория кредита

*@name*('Категория') **public** *V\_CRED\_CAT* [S226\_LOAN\_CT\_REF];

S226\_NEW\_DOG\_EXT

**return** **ref** [PR\_CRED] **is**

**validate** **is**

/\* Секция "Проверка" \*/

**begin**

&BASE$SETVARS;

&BASE$VALIDATE;

&BASE$GETVARS;

**end**;

**execute** **is**

/\* Секция "Тело" \*/

**begin**

&BASE$SETVARS;

&BASE$EXECUTE;

**if** *V\_CRED\_CAT* **is** **not** **null** **then**

::[S226\_CRED\_CATEG].*[NEW\_AUTO]*(*P\_CREDIT* == this,

*P\_DATE\_BEG* == [system].*[OP\_DATE]*,

*P\_DATE\_END* == **null**,

*P\_CRED\_CAT* == *V\_CRED\_CAT*

);

**end** **if**;

**end**;

**end**;

**end**; /\* конeц constructor S226\_NEW\_DOG\_EXT \*/

**Расширение операции редактирования КД:**

**class** PR\_CRED;

*@import\_plsql*(**true**)

*@name*('S226. Редактировать договор') /\* Свойство операции "Наименование" \*/

*@tag*('EXTENSION')

**method** S226\_EDIT\_LAST\_E **instead** **of** S226\_EDIT\_LAST **is**

-- категория кредита

*@name*('Категория') **public** *V\_CRED\_CAT* [S226\_LOAN\_CT\_REF];

old\_cat [S226\_LOAN\_CT\_REF];

S226\_EDIT\_LAST\_E

**is**

**validate** **is** /\* Секция "Проверка" \*/

category\_exist **ref**[S226\_CRED\_CATEG];

**begin**

&BASE$SETVARS;

&BASE$VALIDATE;

&BASE$GETVARS;

**if** p\_message = 'DEFAULT' **then**

-- При поднятии ЭФ «S##. Категория» автоматически заполняется

-- текущим значением категории КД на дату операционного дня филиала

-- (далее - дата ОД), т.е. выбирается из справочника

-- «S##. Кредиты. Принадлежность к категории».

**select** a(a)

**in** ::[S226\_CRED\_CATEG]**all**

**where** a.*[CREDIT]***%id** = this**%id**

**and** a.*[DATE\_BEG]* <= ::[SYSTEM].*[OP\_DATE]*

**and** (a.*[DATE\_END]* >= ::[SYSTEM].*[OP\_DATE]* **OR** a.*[DATE\_END]* **is** **null**)

**into** category\_exist;

*V\_CRED\_CAT* := category\_exist.*[CRED\_CAT]*;

old\_cat := *V\_CRED\_CAT*;

**end** **if**;

**exception**

**when** NO\_DATA\_FOUND **then**

old\_cat := **null**;

**when** TOO\_MANY\_ROWS **then**

**stdio**.put\_line\_buf('По кредитному договору №' || this.*[NUM\_DOG]* || ' найдено более одной категории на текущую дату операционного дня');

**end**;

**execute** **is** /\* Секция "Тело" \*/

**begin**

&BASE$SETVARS;

&BASE$EXECUTE;

--У существующей до изменения записи с пустым реквизитом «Дата окончания»

--данный реквизит заполнить значением (Дата ОД-1).

**for** credit **in** [S226\_CRED\_CATEG]

**where** credit.*[DATE\_END]* **is** **null** **and** credit.*[CREDIT]* = this

**loop**

credit.*[DATE\_END]* := [system].*[OP\_DATE]* - 1;

**end** **loop**;

/\* При редактировании договора существующие до сохранения записи

«S##. Кредиты. Принадлежность к категории»

с датой начала более или равной дате текущего операционного дня филиала

необходимо удалять.\*/

**delete** t **in** ::[S226\_CRED\_CATEG] **where** t.*[DATE\_BEG]* >= ::[SYSTEM].*[OP\_DATE]* **and** t.*[CREDIT]***%id** = this**%id**;

-- изменение категории

**if** old\_cat != *V\_CRED\_CAT* **and** *V\_CRED\_CAT* **is** **not** **null** **then**

-- установим дату окончания для старой записи

**update** t (t.*[DATE\_END]* = ::[SYSTEM].*[OP\_DATE]* - 1) **in** ::[S226\_CRED\_CATEG]

**where** t.*[CREDIT]* = this **and** t.*[DATE\_END]* **is** **null**;

-- добавим новую запись

::[S226\_CRED\_CATEG].*[NEW\_AUTO]*(*P\_CREDIT* == this,

*P\_DATE\_BEG* == ::[SYSTEM].*[OP\_DATE]*,

*P\_DATE\_END* == **null**,

*P\_CRED\_CAT* == *V\_CRED\_CAT*);

**end** **if**;

**if** old\_cat **is** **null** **and** *V\_CRED\_CAT* **is** **not** **null** **then**

-- добавим новую запись

::[S226\_CRED\_CATEG].*[NEW\_AUTO]*(*P\_CREDIT* == this,

*P\_DATE\_BEG* == ::[SYSTEM].*[OP\_DATE]*,

*P\_DATE\_END* == **null**,

*P\_CRED\_CAT* == *V\_CRED\_CAT*);

**end** **if**;

--В случае очистки значения «S##. Категория» на ЭФ

--новая запись не добавляется,

--изменяется «Дата окончания» для существующей по тому же алгоритму

**if** old\_cat **is** **not** **null** **and** *V\_CRED\_CAT* **is** **null** **then**

**update** t (t.*[DATE\_END]* = ::[SYSTEM].*[OP\_DATE]* - 1) **in** ::[S226\_CRED\_CATEG]

**where** t.*[CREDIT]* = this;

**end** **if**;

**end**;

**end**;

**end**; /\* конeц method S226\_EDIT\_LAST\_E \*/

* 1. Скопировать операцию деструктора (PR\_CRED.DELETE#AUTO) ТБП **«Кредиты»**. Создать для копии операцию расширения, удаляющую связанные записи **«S##. Кредиты. Принадлежность к категории»** при удалении кредита.

**class** PR\_CRED;

*@import\_plsql*(**true**)

*@this*(create\_on\_start:=**true**)

*@name*('S226. Удаление кредитного договора')

*@tag*('EXTENSION')

**destructor** S226\_DELETE\_EXT **instead** **of** S226\_DELETE#AUTO **is**

S226\_DELETE\_EXT

**is**

**validate** **is**

**begin**

&BASE$SETVARS;

&BASE$VALIDATE;

&BASE$GETVARS;

**end**;

**execute** **is**

**begin**

&BASE$SETVARS;

&BASE$EXECUTE;

-- удаление связанной категории

**delete** rec **in** ::[S226\_CRED\_CATEG] **all** **where** rec.*[CREDIT]* = this;

**end**;

**end**;

**end**;

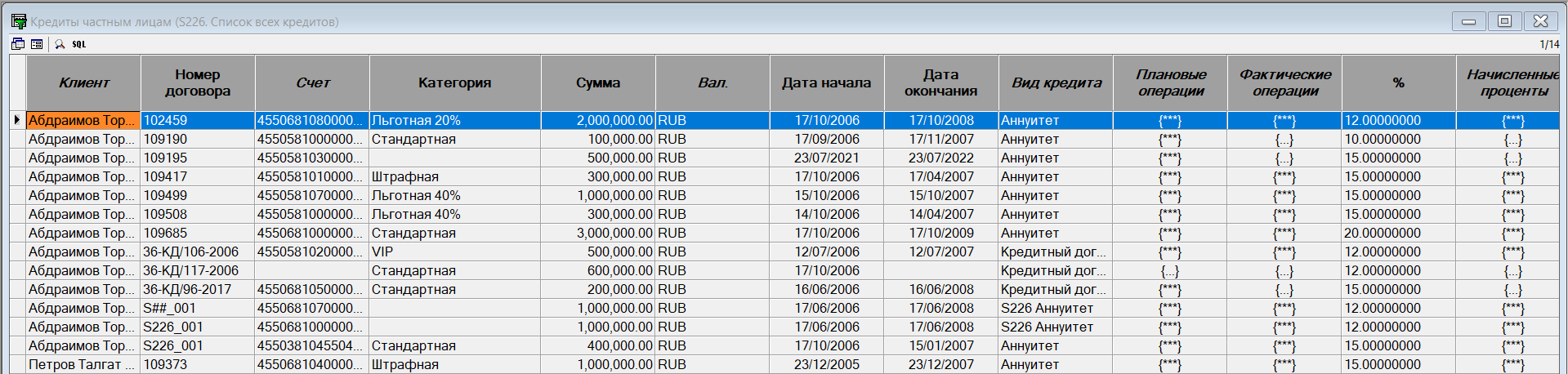
**Задание 5.** *Освещается в лекциях Представления и Синтаксис языка. Выборки и курсоры*

Для вывода пользователю введённого значения при просмотре кредитов скопировать представление **«Список всех кредитов» (VW\_CRIT\_PR\_CRED)** ТБП **«Кредиты»**.

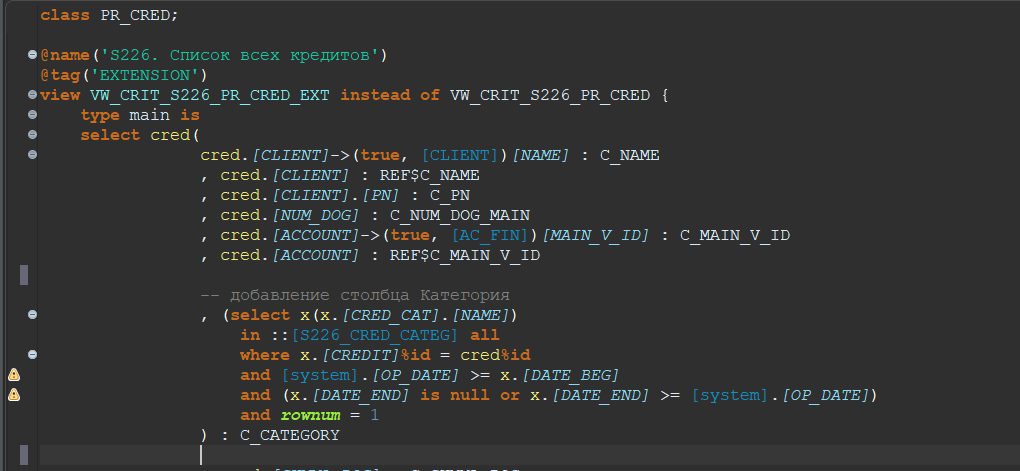
Создать для копии расширение, добавив колонку с значением реквизита **«Наименование»** справочника **«S##. Кредиты. Категория ссуды»** с заголовком «Категория».

Запись из **«S##. Кредиты. Принадлежность к категории»** выбирается по условию: дата ОД >= «Дате начала», «Дата окончания» пуста, либо >= дате ОД.

**Представление с фильтром по типу проекта:**



**Фрагмент кода в операции расширения, отвечающий за добавление колонки:**



**Задание 6.** *Освещается в лекциях Операции, Методы расширения дистрибутивной логики, Кредиты*

Необходимо реализовать возможность вычисления ставки тарифа в зависимости от значения коэффициента, указанного для категории ссуды.

Для этого:

* 1. Создать групповую операцию в ТБП **«Кредиты» (PR\_CRED) «S##. Определение льготного коэффициента»** с параметрами:
  + Код (P\_CODE) - Строка 50
  + Документ (P\_DOC) - Ссылка на "Документы"
  + Продукта (P\_PROD) - Ссылка на "Продукты"
  + Дополнительные параметры (P\_ADD\_PAR) – Строка 32000

Операция должна выполнять следующее:

* Если значение параметра P\_CODE = ‘S##\_ЛЬГОТ\_КОЭФ’, то необходимо:
  + Определить дату расчета. Для этого необходимо определить значение доппараметра DATE\_OP из строки дополнительных параметров P\_ADD\_PAR. Если значение – пусто, то в качестве даты расчета берется дата ОД
  + Найти записи в справочнике **«S##. Кредиты. Принадлежность к категории»** по данному КД (который передается через параметр P\_PROD) на дату расчета (определенную выше).

Если такая запись есть, то из операции возвращается значение коэффициента для категории ссуды из этой записи.

* Во всех остальных случаях возвращается строка ‘EMPTY’

**class** PR\_CRED;

*@name*('S226. Определение льготного коэффициента') /\* Свойство операции "Наименование" \*/

**static** **method** S226\_LOAN\_COEFF **is**

S226\_LOAN\_COEFF (

*@name*('Код') P\_CODE **in** [STRING\_50],

*@name*('Документ') P\_DOC **in** [DOCUMENT\_REF],

*@name*('Продукт') P\_PROD **in** [A\_PROD\_REF],

*@name*('Дополнительные параметры') P\_ADD\_PAR **in** [STRING\_32000]

) **return** [MEMO]

**is**

**execute** **is**

date\_calc [DATE];

category\_coeff **number**;

**begin**

**if** P\_CODE = 'S226\_ЛЬГОТ\_КОЭФ' **then**

-- если не нашли дату ОД в доп параметрах, берем дату ОД филиала

date\_calc := **nvl**(STR.GET\_DATE\_PAR(P\_ADD\_PAR,'DATE\_OP'),

P\_PROD.*[FILIAL]*.*[OP\_DATE]*);

-- найдем коэффициент

**select** a(a.*[CRED\_CAT]*.*[COEFFICIENT]*)

**in** ::[S226\_CRED\_CATEG]**all**

**where** a.*[CREDIT]***%id** = P\_PROD**%id** **and** a.*[DATE\_BEG]* <= date\_calc **and** (date\_calc <= a.*[DATE\_END]* **or** a.*[DATE\_END]* **is** **null**)

**into** category\_coeff;

**return** category\_coeff;

**else**

**return** 'EMPTY';

**end** **if**;

**exception**

-- вернем коэффициент по умолчанию, если возникло исключение

**when** NO\_DATA\_FOUND **then**

**return** '1';

**when** TOO\_MANY\_ROWS **then**

**return** '1';

**when** **others** **then**

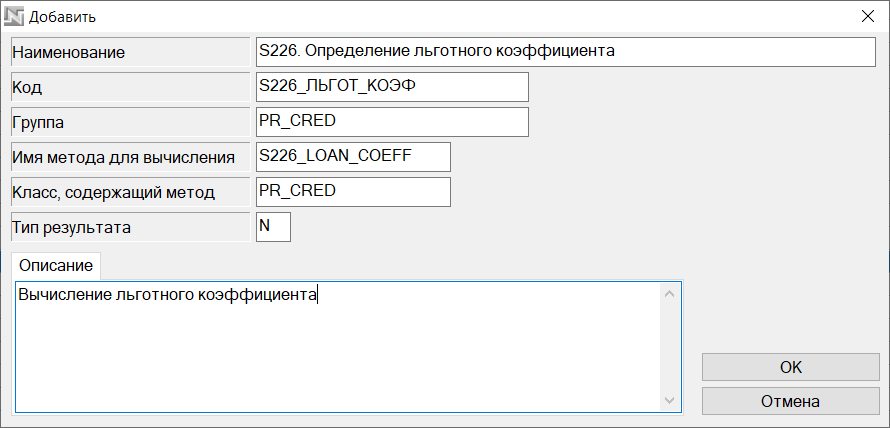
**return** '1';

**end**;

**end**;

**end**; /\* конeц static method S226\_LOAN\_COEFF \*/

* 1. Создать (вручную) запись в справочнике **«Вычисляемые параметры» (CALC\_PARAMS)**
     + Код = S##\_ЛЬГОТ\_КОЭФ
     + Наименование = S##. Определение льготного коэффициента
     + Тип результата = N
     + Класс содержащий метод = PR\_CRED
     + Имя метода для вычисления = короткое имя операции, созданной выше
     + Группа = PR\_CRED



* 1. Проверить работоспособность реализованного.

Для этого:

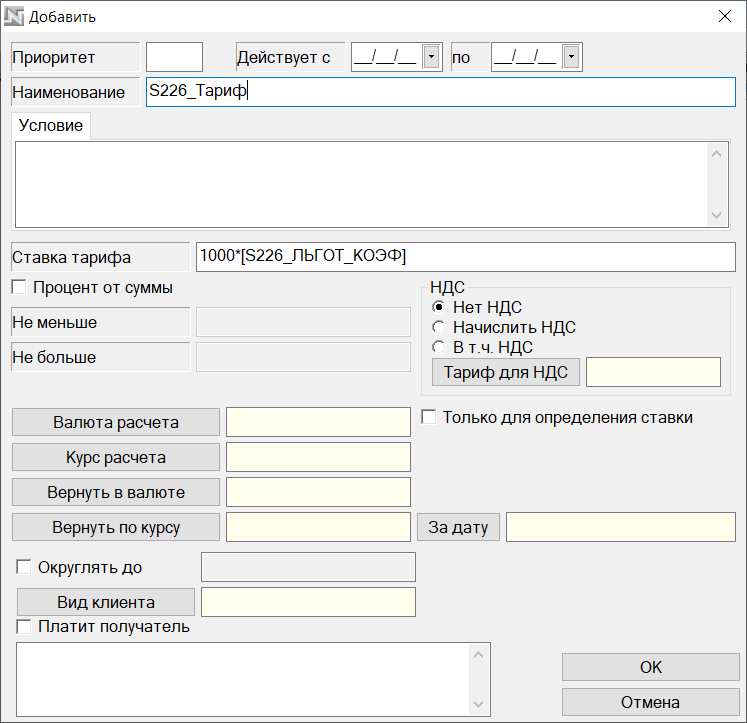
1. Необходимо настроить (вручную) тариф в справочнике **«Тарифы, комиссии, налоги» (TARIFFS\_COMMISS).**

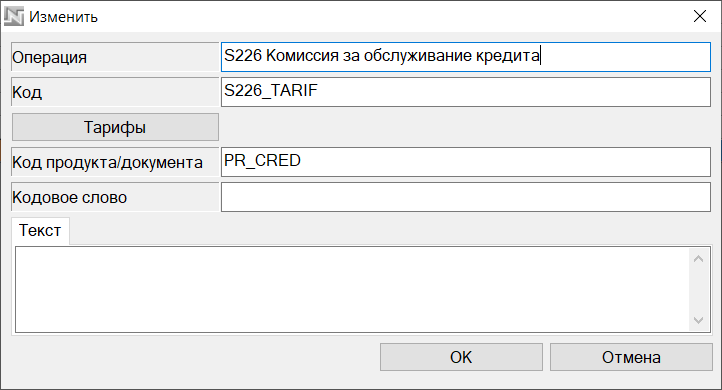
Значение ставки тарифа указать равным:

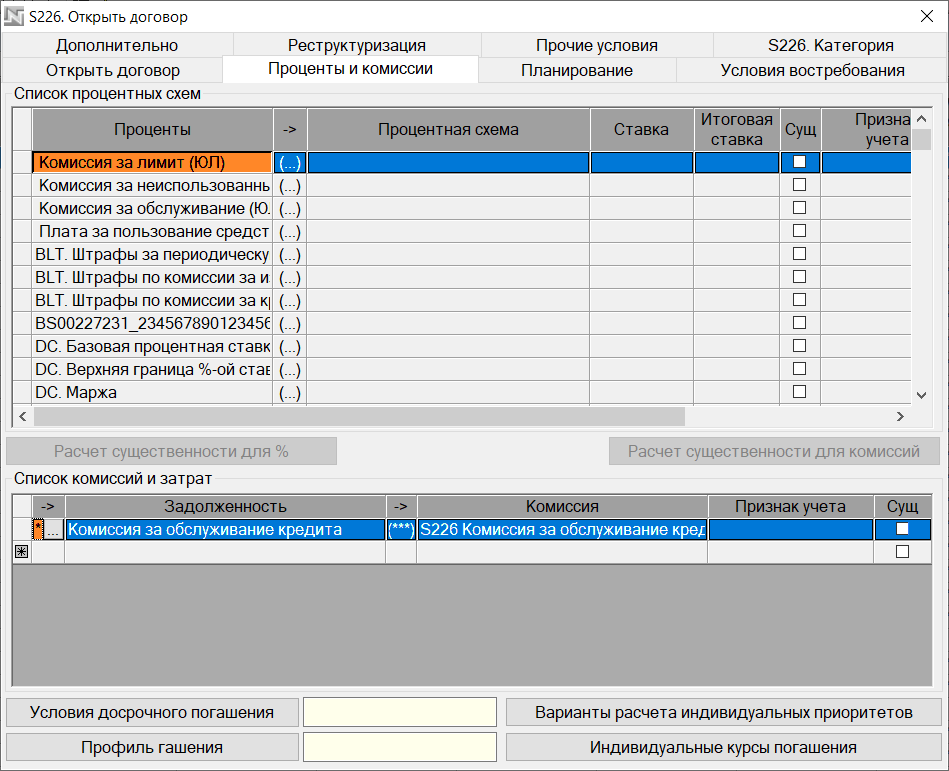
1000 \* [S##\_ЛЬГОТ\_КОЭФ]

1. Завести новый КД, указать для него принадлежность к категории
2. Для вида задолженности **«Комиссия за обслуживание кредита»** на закладке **«Проценты и комиссии»** карточки КД задать тариф, созданный выше
3. Выполнить планирование по КД – убедиться, что в плановом графике появились операции гашения комиссии, рассчитанные в сумме, зависящей от коэффициента категории ссуды

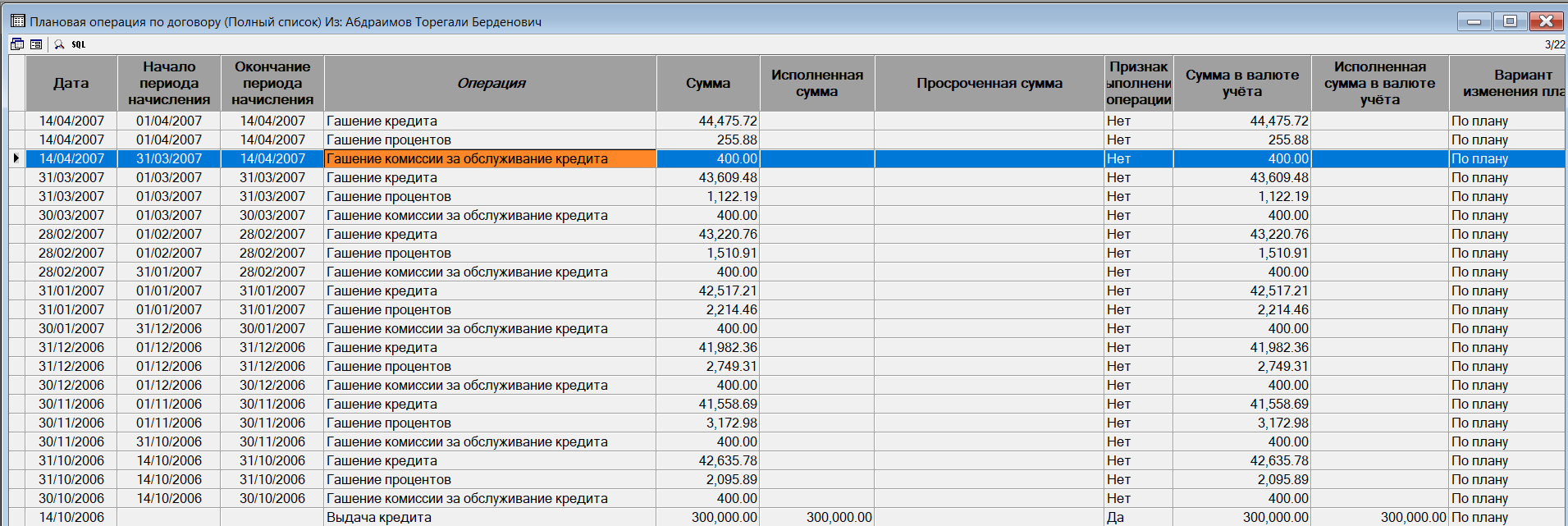
**Создание тарифа**







**Пример расчета категории для льготной комиссии (40%) в массиве плановых операций:**



**Задание 7.** *Освещается в лекциях Методы расширения дистрибутивной логики, УЯ*

Для маркирования документов, порождённых КД с льготной категорией, и сохранения значения категории ссуды на момент проводки необходимо выполнить следующие доработки:

В справочнике **«Расширения и перекрытия. Точки вызова»** для точки вызова **«Платежные документы. Анализ счета деб/кр при проводке пл. документа» (MD\_FOR\_PROV)** создать метод, анализирующий проводки кредитной подсистемы.

В случае, если проводится платёжный документ (MAIN\_DOCUM) по кредиту, на который есть актуальная ссылка из справочника **«S##. Кредиты. Принадлежность к категории»** (на дату проводки документа), в массив **«История состояний» (HISTORY\_STATE)** документа необходимо добавить запись с состоянием (STATE) = S##\_NOTE\_CAT. В реквизит **«Дополнительная информация»** записи указывается текущее значение реквизита «Коэффициент» справочника **«S##. Кредиты. Категория ссуды»**.

Для того, чтобы возможные ошибки в методе не мешали проводке документов остальных студентов, необходимо использовать для метода обработку всех исключений с выводом текста ошибки в монитор сессии.

**class** HOOK;

*@name*('S226. Анализ счета деб/кр при проводке пл. документа') /\* Свойство операции "Наименование" \*/

**static** **method** S226\_MD\_FOR\_PROV **is**

**pragma** **include**(::[RUNTIME].[DEBUG\_MACRO]);

S226\_MD\_FOR\_PROV (

*@name*('Документ') DOCUMENT **in** [MAIN\_DOCUM\_REF],

*@name*('Счет дебета') ACCOUNT\_DT **in** [AC\_FIN\_REF],

*@name*('Счет кредита') ACCOUNT\_CT **in** [AC\_FIN\_REF],

*@name*('Признак счета дебета') IT\_DT **in** [BOOLEAN],

*@name*('Признак успешности анализа') ANALYSE\_OK **in** **out** [BOOLEAN],

*@name*('Дополнительные параметры') P\_ADDS **in** **out** [STRING\_32000],

*@name*('Тип вызова') P\_MESS **in** [STRING\_100]

) **is**

/\*В случае, если проводится платёжный документ (MAIN\_DOCUM) по кредиту,

\*на который есть актуальная ссылка из справочника «S##. Кредиты. Принадлежность к категории»

\*(на дату проводки документа), в массив «История состояний» (HISTORY\_STATE) документа

\*необходимо добавить запись с состоянием (STATE) = S##\_NOTE\_CAT.

\*В реквизит «Дополнительная информация» записи указывается текущее значение

\*реквизита «Коэффициент» справочника «S##. Кредиты. Категория ссуды».

\*/

**execute** **is**

coef **number**;

h\_state [HISTORY\_STATES];

**begin**

-- ищем категорию

**select** a(a.*[CRED\_CAT]*.*[COEFFICIENT]*) **in** ::[S226\_CRED\_CATEG]

-- дата проводки должна лежать в пределах даты действия категории

**where** a.*[CREDIT]***%id** = DOCUMENT.*[PRODUCT\_CT]*.*[ACC\_PROD]***%id**

**or** a.*[CREDIT]***%id** = DOCUMENT.*[PRODUCT\_DT]*.*[ACC\_PROD]***%id**

**and** DOCUMENT.*[DATE\_PROV]* >= a.*[DATE\_BEG]*

**and** (DOCUMENT.*[DATE\_PROV]* <= a.*[DATE\_END]* **or** a.*[DATE\_END]* **is** **null**)

**into** coef;

-- если категория найдена, записываем данные в массив истории состояний

h\_state**%init**;

&debug('coef ' || coef, 0)

h\_state.*[ADD\_INFO]* := **to\_char**(coef, '09D00');

h\_state.*[DATE]* := **sysdate**;

h\_state.*[STATE]* := 'S226\_NOTE\_CAT';

**insert** h\_state **into** DOCUMENT.*[HISTORY\_STATE]*;

**exception**

**when** NO\_DATA\_FOUND **then**

&debug('S226. Категории для данного кредитного договора не найдено', 0)

**when** TOO\_MANY\_ROWS **then**

&debug('S226. Найдено больше одной категории для данного кредитного договора', 0)

**when** **others** **then**

&debug('S226. Ошибка при проводке', 0)

**end**;

**end**;

**end**; /\* конeц static method S226\_MD\_FOR\_PROV \*/

**Задание 8.** *Освещается в лекции* *Реализация выгрузки в файл*

В ТБП **«Платёжные документы» (MAIN\_DOCUM)** создать групповую операцию **«S##. Выгрузка документов льготных кредитов»**.

На экранной форме (ЭФ) операции выгрузки должны присутствовать параметры «Дата начала», «Дата окончания».

Операция должна выгрузить в формате xls проведенные документы с реквизитом «Дата проводки», соответствующим указанному на ЭФ периоду, у которых в массиве «История состояний» есть запись с состоянием S##\_NOTE\_CAT.

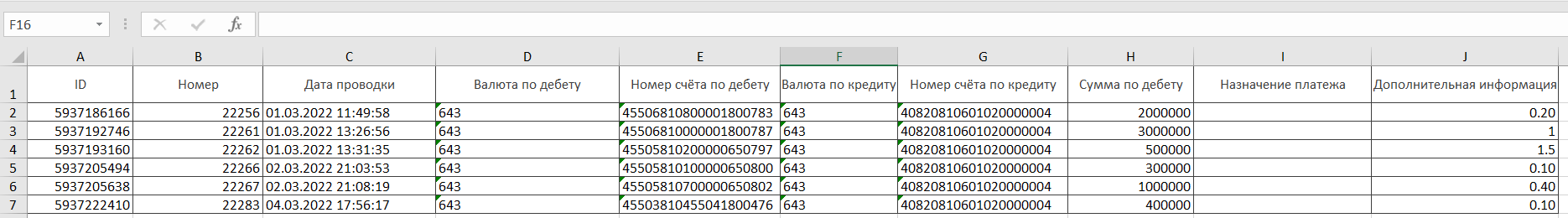
В файл должны выгружаться реквизиты:

* id
* Номер
* Астрономическая дата проводки (в формате DD.MM.YYYY HH24:MI:SS)
* Валюта по дебету (код ISO)
* Номер счёта по дебету
* Валюта по кредиту (код ISO)
* Номер счёта по кредиту
* Сумма по дебету
* Назначение платежа
* Дополнительная информация (выбирается из массива «История состояний» из записи с состоянием S##\_NOTE\_CAT). Если в истории более одной записи со состоянием S##\_NOTE\_CAT, выбирается запись с максимальным значением реквизита «Дата».

В первой строке файла указываются заголовки колонок.

Для выгрузки необходимо использовать шаблон из «Справочника шаблонов документов».

**Сформированный отчет:**

****

**Реализацию см. после задания 9**

**Задание 9.** *Освещается в лекции* *Реализация выгрузки в файл*

В операции из задания 8 добавить возможность выбора экспорта в формат xml, разместив на ЭФ чекбоксы «В формат xls», «В формат xml». Выгружаются записи ТБП **«Платёжные документы»**, выбранные по алгоритму задания 8 и все записи порождённых ими массивов **«История состояний»**.

Реквизиты платёжного документа для выгрузки:

* Номер
* ФИО исполнителя
* Номер счёта по дебету
* Номер счёта по кредиту
* Сумма по дебету
* Наименование клиента из структуры «Реквизиты плательщика».

Также выгружаются все несистемные реквизиты всех записей массива **«История состояний»** документа.

Параметры xml: кодировка WINDOWS-1251, версия 1, заголовок xml файла:

exportDesc created="дата\_создания" author="ФИО пользователя" comment="Выгрузка документов льготных кредитов"

Запись определяется тегом record с заголовком idx="id платёжного документа".

Наименования выгружаемого поля определяется тегом field name. Значение определяется тегом value.

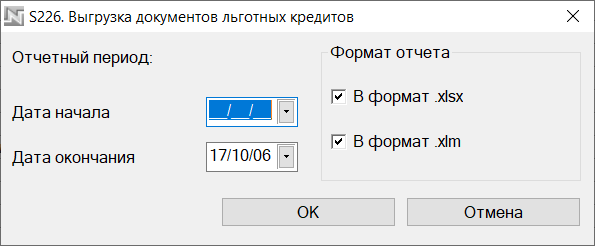
Таблица соответствий наименований реквизитов и кодов выгрузки для ТБП **«Платёжные документы»**:

|  |  |
| --- | --- |
| **Наименование реквизита** | **field name** |
| Номер | document\_num |
| ФИО исполнителя | empl\_name |
| Номер счёта по дебету | num\_dt |
| Номер счёта по кредиту | num\_ct |
| Сумма по дебету | totsum |
| Наименование клиента | clname |
| История статуса | state\_hist |

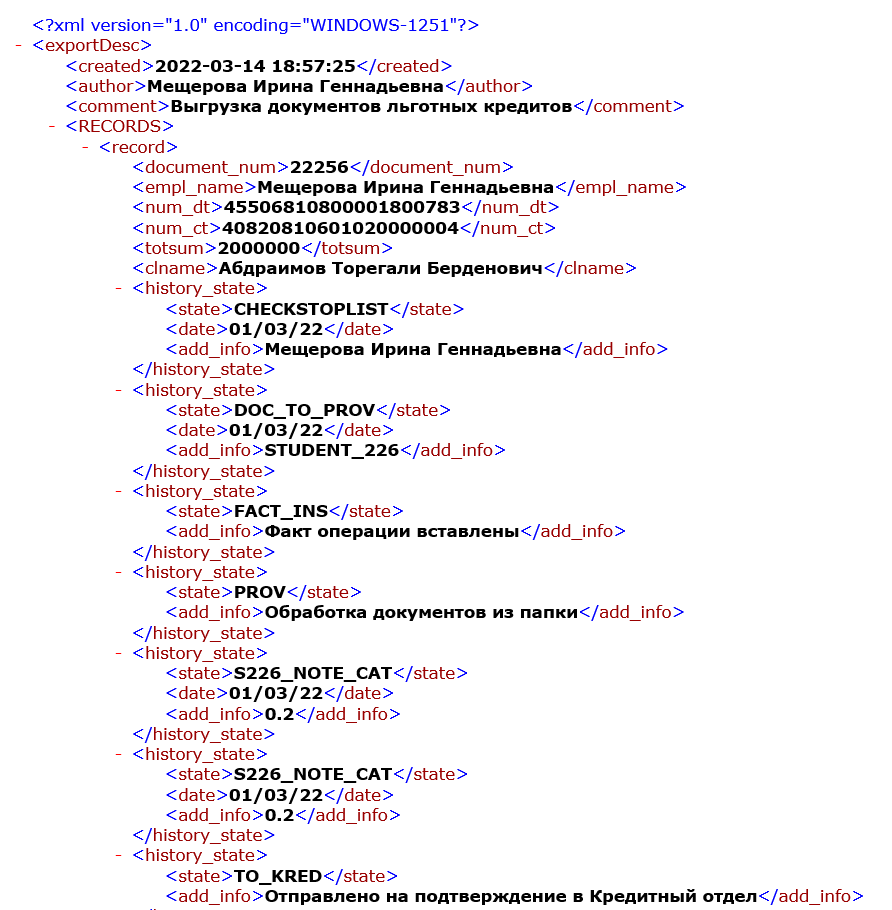
Таблица соответствий наименований реквизитов и кодов выгрузки для ТБП **«История состояний»:**

|  |  |
| --- | --- |
| **Наименование реквизита** | **field name** |
| Состояние | state |
| Дата | date |
| Дополнительная информация | add\_info |

**Задание 8 и 9 находятся в одной операции. ЭФ операции:**



**Фрагмент сформированного отчета в .xml:**

****

**Код операции:**

**class** MAIN\_DOCUM;

*@import\_plsql*(**true**)

*@name*('S226. Выгрузка документов льготных кредитов') /\* Свойство операции "Наименование" \*/

**static** **method** S226\_DOC\_PRINT **is**

**pragma** **macro**(xl,'::[OOXML].[API\_XL]');

**pragma** **macro**(lib\_ooxml,'::[SHABLON\_DOC].[LIB\_OOXML]');

**pragma** **macro**(xml,'::[RUNTIME].[XML\_DOM]');

**pragma** **include**(::[RUNTIME].[DEBUG\_MACRO]);

**pragma** **include**(::[RUNTIME].[CSMD]);

**pragma** **include**(::[RUNTIME].[DEBUG\_MACRO]);

**pragma** **include**(::[OOXML].[API\_XL]);

-- в процедуре содержится выгрузка данных и форматирование отчета

**procedure** Draw\_Report\_XLSX(date\_from **date**, date\_to **date**) **is**

n\_i **integer** **default** 1;

iBorder **integer**; -- Идентификатор рамки

**begin**

&xl.Open\_Sheet(1);

iBorder := &xl.get\_border

( *p\_top* == &xl.xl\_b\_thin

, *p\_bottom* == &xl.xl\_b\_thin

, *p\_left* == &xl.xl\_b\_thin

, *p\_right* == &xl.xl\_b\_thin

);

-- выгрузка реквизитов

**for** (**select** a(a**%id** : C\_PROD\_ID

,a.*[DOCUMENT\_NUM]* : C\_NUMBER

,a.*[ASTR\_DATE\_PROV]* : C\_ASTR\_DATE\_PROV

,a.*[VALUTA]*.*[CODE\_ISO]* : C\_VALUTA\_ISO\_DT

,a.*[NUM\_DT]* : C\_NUM\_DT

,a.*[VALUTA\_PO]*.*[CODE\_ISO]* : C\_VALUTA\_ISO\_KT

,a.*[NUM\_KT]* : C\_NUM\_KT

,a.*[SUM]* : C\_SUM\_DT

,a.*[NAZN]* : C\_NAZN

,a.*[HISTORY\_STATE]*.*[ADD\_INFO]* : C\_ADDS

--Если в истории более одной записи со состоянием S##\_NOTE\_CAT, выбирается запись с максимальным значением реквизита «Дата».

,analytic(row\_number(), 'over(partition by [1] order by [2] desc)',a**%id**, a.*[ASTR\_DATE\_PROV]*) : C\_ROW\_NUM

)

**in** ::[MAIN\_DOCUM]

**where** a.*[HISTORY\_STATE]*.*[STATE]* = 'S226\_NOTE\_CAT'

**and** **trunc**(a.*[DATE\_PROV]*) >= date\_from

**and** **trunc**(a.*[DATE\_PROV]*) <= date\_to

**and** a**%state** **in** ('PROV','PROV\_IN\_BANK'))

-- заполнение ячеек

**loop**

-- выбираем запись с максимальным значением реквизита дата

**if** a.C\_ROW\_NUM = 1 **then**

n\_i := n\_i + 1;

&xl.cell(*p\_row* == n\_i, *p\_col* == 1, *p\_value* == a.C\_PROD\_ID, *p\_borderId* == iBorder);

&xl.cell(*p\_row* == n\_i, *p\_col* == 2, *p\_value* == a.C\_NUMBER, *p\_borderId* == iBorder);

&xl.cell(*p\_row* == n\_i, *p\_col* == 3, *p\_value* == **to\_char**(a.C\_ASTR\_DATE\_PROV, 'DD.MM.YYYY HH24:MI:SS'), *p\_borderId* == iBorder);

&xl.cell(*p\_row* == n\_i, *p\_col* == 4, *p\_value* == a.C\_VALUTA\_ISO\_DT, *p\_borderId* == iBorder);

&xl.cell(*p\_row* == n\_i, *p\_col* == 5, *p\_value* == a.C\_NUM\_DT, *p\_borderId* == iBorder);

&xl.cell(*p\_row* == n\_i, *p\_col* == 6, *p\_value* == a.C\_VALUTA\_ISO\_KT, *p\_borderId* == iBorder);

&xl.cell(*p\_row* == n\_i, *p\_col* == 7, *p\_value* == a.C\_NUM\_KT, *p\_borderId* == iBorder);

&xl.cell(*p\_row* == n\_i, *p\_col* == 8, *p\_value* == a.C\_SUM\_DT, *p\_borderId* == iBorder);

&xl.cell(*p\_row* == n\_i, *p\_col* == 9, *p\_value* == a.C\_NAZN, *p\_borderId* == iBorder);

&xl.cell(*p\_row* == n\_i, *p\_col* == 10, *p\_value* == a.C\_ADDS, *p\_borderId* == iBorder);

**end** **if**;

**end** **loop**;

**end**;

-- процедура для сохранения XLSX отчета

**procedure** Save\_Report\_XLSX(date\_from **date**, date\_to **date**, file\_path **varchar2**(1000)) **is**

blob\_iblob\_new **blob**;

s\_file\_name **varchar2**(500);

**begin**

--загрузка шаблона

&lib\_ooxml.*load\_blob*('S226\_PRINT\_LOAN\_DOC', blob\_iblob\_new, **null**, 'S226\_MAIN\_DOCUM', **null**, **null**);

&xl.init\_xls;

-- если шаблон не найден

**if** **not** &xl.open\_file(blob\_iblob\_new) **then**

**pragma** error('В справочнике "' || ::[SHABLON\_DOC]**%classname** ||'" не найден шаблон с кодом "S226\_PRINT\_LOAN\_DOC"');

**else**

&debug('Шаблон найден', 0)

**end** **if**;

--загрузка данных в шаблон

Draw\_Report\_XLSX(date\_from, date\_to);

&xl.finish(blob\_iblob\_new);

-- сохранение в файл

&debug('s\_file\_path: ' || file\_path,0)

s\_file\_name := 'S226\_LOAN\_DOC';

s\_file\_name := s\_file\_name || **to\_char**(**sysdate**, '\_yy\_dd\_mm\_hh24\_mi\_ss')||'.xlsx';

&debug('s\_file\_name: ' || s\_file\_name, 0)

&lib\_ooxml.*write\_to\_file*(blob\_iblob\_new, file\_path, s\_file\_name);

[OOXML]::[API\_ORSA].putReport2Run(blob\_iblob\_new,'xlsx');

**stdio**.put\_line\_buf('Выгрузка отчета ' || s\_file\_name || ' успешно завершена');

**end**;

-- процедура добавления полей в узел

**procedure** Add\_Text\_Node(p\_node **in** **out** **nocopy** &xml.DOMNode,

p\_xml\_doc **in** **out** **nocopy** &xml.DOMDocument,

p\_node\_name **in** **varchar2**,

p\_value **in** [STRING\_1000],

p\_write\_empty **in** **boolean** **default** **false**) **is**

**begin**

**if** p\_value **is** **not** **null** **then**

&xml.*appendChild*(&xml.*appendChild*(p\_node, &xml.*makeNode*(&xml.*createElement*(p\_xml\_doc, p\_node\_name))),

&xml.*makeNode*(&xml.*createTextNode*(p\_xml\_doc, p\_value)));

**elsif** p\_value **is** **null** **and** p\_write\_empty **then**

&xml.*appendChild*(p\_node,&xml.*makeNode*(&xml.*createElement*(p\_xml\_doc, p\_node\_name)));

**end** **if**;

**end**;

-- процедура формирования и сохранения XML отчета

**procedure** Save\_Report\_XML(date\_from **date**, date\_to **date**) **is**

s\_file\_name **varchar2**(500);

parser &xml.Parser;

xml\_doc &xml.DOMDocument;

main\_node &xml.DOMNode;

header\_node &xml.DOMNode;

record\_node &xml.DOMNode;

root\_dir **varchar2**;

slash **varchar2**(1) := SysInfo.GetValue('DEF\_SLASH');

**begin**

-- инициализация xml

&xml.*initialize*; -- инициализация библиотеки

parser := &xml.*newParser*; -- создание парсера

&xml.*parseBuffer*(parser, '<?xml version="1.0" encoding="Windows-1251"?><elm/>'); -- задание параметров

xml\_doc := &xml.*getDocument*(parser);

-- заполнение данных

main\_node := &xml.*makeNode*(xml\_doc); -- создание верхнего узла

header\_node := &xml.*makeNode*(&xml.*createElement*(xml\_doc, 'exportDesc'));

-- добавление полей в узел

Add\_Text\_Node(header\_node, xml\_doc, 'created', **sysdate**);

Add\_Text\_Node(header\_node, xml\_doc, 'author', stdlib.userid.*[NAME]*);

Add\_Text\_Node(header\_node, xml\_doc, 'comment', 'Выгрузка документов льготных кредитов');

-- пустой узел второго уровня, к которому будем привязывать записи

record\_node := &xml.*makeNode*(&xml.*createElement*(xml\_doc, 'RECORDS'));

**for** (**select** doc(doc**%id** : C\_PROD\_ID

,doc.*[DOCUMENT\_NUM]* : C\_NUMBER

,doc.*[PROV\_USER]*.*[NAME]* : C\_PROV\_USER

,doc.*[NUM\_DT]* : C\_NUM\_DT

,doc.*[NUM\_KT]* : C\_NUM\_KT

,doc.*[SUM]* : C\_SUM\_DT

,**nvl**(doc.*[KL\_DT]*.*[1]*.*[1]*.*[NAME]*,

doc.*[KL\_DT]*.*[2]*.*[2]*) : C\_PAYER\_NAME

,doc.*[HISTORY\_STATE]* : C\_HIST\_STATE

--Если в истории более одной записи со состоянием S##\_NOTE\_CAT, выбирается запись с максимальным значением реквизита «Дата».

,analytic(row\_number(), 'over(partition by [1] order by [2] desc)',doc**%id**, doc.*[ASTR\_DATE\_PROV]*) : C\_ROW\_NUM

)

**in** ::[MAIN\_DOCUM]

**where** doc.*[HISTORY\_STATE]*.*[STATE]* = 'S226\_NOTE\_CAT'

**and** **trunc**(doc.*[DATE\_PROV]*) >= date\_from

**and** **trunc**(doc.*[DATE\_PROV]*) <= date\_to

**and** doc**%state** **in** ('PROV','PROV\_IN\_BANK')

)

**loop**

**declare**

value\_node &xml.DOMNode;

**begin**

**if** doc.C\_ROW\_NUM = 1 **then**

-- создание узла для записи

value\_node := &xml.*makeNode*(&xml.*createElement*(xml\_doc, 'record'));

&xml.*appendChild*(record\_node, value\_node);

--&xml.setAttribute(level03\_elem, 'idx=', to\_char(doc.C\_PROD\_ID));

-- добавление полей в узел

add\_text\_node(value\_node, xml\_doc, 'document\_num', **to\_char**(doc.C\_NUMBER));

add\_text\_node(value\_node, xml\_doc, 'empl\_name', doc.C\_PROV\_USER);

add\_text\_node(value\_node, xml\_doc, 'num\_dt', **to\_char**(doc.C\_NUM\_DT));

add\_text\_node(value\_node, xml\_doc, 'num\_ct', **to\_char**(doc.C\_NUM\_KT));

add\_text\_node(value\_node, xml\_doc, 'totsum', **to\_char**(doc.C\_SUM\_DT));

add\_text\_node(value\_node, xml\_doc, 'clname', doc.C\_PAYER\_NAME);

-- вывод записей из массива истории состояний

**for** hs **in** doc.C\_HIST\_STATE

**loop**

**declare**

history\_node &xml.DOMNode;

**begin**

history\_node := &xml.*makeNode*(&xml.*createElement*(xml\_doc, 'history\_state'));

&xml.*appendChild*(value\_node, history\_node);

&debug('xml 4',0)

add\_text\_node(history\_node, xml\_doc, 'state', hs.*[STATE]*);

add\_text\_node(history\_node, xml\_doc, 'date', **to\_char**(hs.*[DATE]*, 'DD.MM.YYYY HH24:MI:SS'));

add\_text\_node(history\_node, xml\_doc, 'add\_info', hs.*[ADD\_INFO]*);

**end**;

**end** **loop**;

**end** **if**;

**end**;

**end** **loop**;

root\_dir := **stdio**.GET\_ENV('FIO\_ROOT\_DIR');

&debug('root\_dir = '||root\_dir,0)

&xml.*appendChild*(header\_node, record\_node);

&xml.*replaceChild*(main\_node, header\_node, &xml.*makeNode*(&xml.*getDocumentElement*(xml\_doc))); -- Подменяем верхний узел на созданный

s\_file\_name := 'S226\_LOAN\_DOC';

s\_file\_name := s\_file\_name || **to\_char**(**sysdate**, '\_yy\_dd\_mm\_hh24\_mi\_ss')||'.xml';

&debug('s\_file\_name = '||s\_file\_name,0)

&xml.*writeToFile*(xml\_doc, root\_dir||slash||s\_file\_name);

**stdio**.put\_line\_buf('Выгрузка отчета ' || s\_file\_name || ' успешно завершена');

**end**;

S226\_DOC\_PRINT (

/\*На экранной форме (ЭФ) операции выгрузки должны присутствовать параметры «Дата начала», «Дата окончания».

\* Операция должна выгрузить в формате xls проведенные документы с реквизитом «Дата проводки»,

\* соответствующим указанному на ЭФ периоду, у которых в массиве «История состояний» есть запись с состоянием S##\_NOTE\_CAT.

\*/

*@name*('Дата начала') P\_DATE\_FROM **in** [DATE],

*@name*('Дата окончания') P\_DATE\_TO **in** [DATE],

-- для выбора формата выгрузки отчета

*@name*('Checkbox .xlsx') P\_XLSX\_BOOLEAN **in** [BOOLEAN],

*@name*('Checkbox .xlm') P\_XML\_BOOLEAN **in** [BOOLEAN]

) **is**

**validate** **is**

**begin**

**if** P\_MESSAGE = 'DEFAULT' **then**

-- предварительное заполнение полей

P\_DATE\_TO := ::[system].*[OP\_DATE]*;

P\_XLSX\_BOOLEAN := **true**;

P\_XML\_BOOLEAN := **true**;

**end** **if**;

**end**;

**execute** **is**

s\_file\_path **varchar2**(1000);

**begin**

-- проверка заполнения дат

**if** P\_DATE\_FROM **is** **null** **or** P\_DATE\_TO **is** **null** **then**

**pragma** error('Должны быть заполнены обе даты!');

**end** **if**;

-- проверка корректности дат

**if** P\_DATE\_FROM > P\_DATE\_TO **then**

**pragma** error('Дата начала не может быть больше даты окончания!');

**end** **if**;

-- проверка формата

**if** P\_XLSX\_BOOLEAN = **false** **and** P\_XML\_BOOLEAN = **false** **then**

**pragma** error('Выберите хотя бы один формат выгрузки!');

**end** **if**;

s\_file\_path := ::[SYSTEM\_PARAMS].*[GET]*('PATH', **true**);

**if** P\_XLSX\_BOOLEAN = **true** **then**

Save\_Report\_XLSX(P\_DATE\_FROM, P\_DATE\_TO, s\_file\_path);

**end** **if**;

**if** P\_XML\_BOOLEAN = **true** **then**

Save\_Report\_XML(P\_DATE\_FROM, P\_DATE\_TO);

**end** **if**;

**end**;

**end**;

**end**; /\* конeц static method S226\_DOC\_PRINT \*/

**Задание 10.** *Освещается в лекции Подготовка хранилища с доработкой, создание операций конвертации*

Для подготовки передачи изменений на схему банка создать операцию конвертации в ТБП **«Обновления версии»** (CONV\_57), с коротким именем UYYYYMMDD\_S##\_NN (где S## – префикс студента, NN – номер по порядку), которая:

* 1. Добавит в справочник **«Расширения и перекрытия. Точки вызова»** для записи с кодом MD\_FOR\_PROV в массив **«Методы»** операцию из задания 7 и запустит операцию **«Перекомпиляция точки вызова»** данного справочника над этой строкой;
  2. Добавит в справочник **«Вычисляемые параметры»** параметр S##\_ЛЬГОТ\_КОЭФ из задания 6;

Операция должна допускать многократный повторный запуск.

**class** CONV\_57;

*@name*('S226. Добавление системных записей')

**static** **method** U20220305S226\_02 **is**

--2.Добавит в справочник «Вычисляемые параметры» параметр S##\_ЛЬГОТ\_КОЭФ из задания 6;

**procedure** New\_Calc\_Params **is**

rec\_exist **ref** [CALC\_PARAMS];

**begin**

**select** a(a)

**in** ::[CALC\_PARAMS]

**where** a.*[CODE]* = 'S226\_ЛЬГОТ\_КОЭФ'

**into** rec\_exist;

[CONV].[LOG].WriteMess('В справочнике ' || ::[CALC\_PARAMS]**%classname** || ' уже существует запись с кодом "S226\_ЛЬГОТ\_КОЭФ"');

**exception**

**when** NO\_DATA\_FOUND **then**

[CALC\_PARAMS].*[NEW#AUTO]*(*P#NAME* == 'S226. Определение льготного коэффициента',

*P#CODE* == 'S226\_ЛЬГОТ\_КОЭФ',

*P#GROUP* == 'PR\_CRED',

*P#METHOD* == 'S226\_LOAN\_COEFF',

*P#CLASS* == 'PR\_CRED',

*P#TYPE\_RESULT* == 'N',

*P\_NOTE* == **null**

);

[CONV].[LOG].WriteMess('В справочник ' || ::[CALC\_PARAMS]**%classname** || ' добавлена запись с кодом "S226\_ЛЬГОТ\_КОЭФ"');

::[CONV].[LOG].WRITEMESS(TB$ || TB$ || 'Начата компиляция вычисляемых параметров');

::[CALC\_PARAMS].*[MAKE\_CP\_LIBS]*;

::[CONV].[LOG].WRITEMESS(TB$ || TB$ || 'Компиляция вычисляемых параметров завершена');

**when** TOO\_MANY\_ROWS **then**

[CONV].[LOG].WriteMess('В справочнике ' || ::[CALC\_PARAMS]**%classname** || ' существует более одной с кодом "S226\_ЛЬГОТ\_КОЭФ"');

**end**;

U20220305S226\_02 **is**

**execute** **is**

max\_priority ::[AUXAPP\_METHOD].*[PRIORITY]***%type**;

record\_exist ::[APPAUX\_CALLPOINT];

**begin**

[CONV].[LOG].InitLog(&METHOD$SNAME||'\_mess.log',**true**);

[CONV].[LOG].WriteMess('Начало выполнения операции ['||&METHOD$SNAME||'] '||&METHOD$NAME);

--2.Добавит в справочник «Вычисляемые параметры» параметр S##\_ЛЬГОТ\_КОЭФ из задания 6;

New\_Calc\_Params();

/\*1.Добавит в справочник «Расширения и перекрытия. Точки вызова» для записи с кодом MD\_FOR\_PROV

в массив «Методы» операцию из задания 7 и запустит операцию «Перекомпиляция точки вызова»

данного справочника над этой строкой;\*/

-- если нашли такой метод в точке вызова

**select** md\_for\_pr(md\_for\_pr)

**in** ::[APPAUX\_CALLPOINT]

**where** md\_for\_pr.*[CODE]* = 'MD\_FOR\_PROV'

**and** md\_for\_pr.*[METHODS]*.*[SNAME]* = 'S226\_MD\_FOR\_PROV'

**into** record\_exist;

[CONV].[LOG].WriteMess('Метод S226\_MD\_FOR\_PROV уже существует в точке вызова MD\_FOR\_PROV');

[CONV].[LOG].WriteMess('Окончание выполнения операции ['||&METHOD$SNAME||'] '||&METHOD$NAME);

[CONV].[LOG].InitLog;

**exception** **when** NO\_DATA\_FOUND **then**

**for** (**select** md\_for\_pr(md\_for\_pr.*[METHODS]* : hook\_methods)

**in** ::[APPAUX\_CALLPOINT]

**where** md\_for\_pr.*[CODE]* = 'MD\_FOR\_PROV'

)

**loop**

**select** a(**max**(**nvl**(a.*[PRIORITY]*, 0))) **in** md\_for\_pr.hook\_methods **into** max\_priority;

max\_priority := max\_priority + 1;

md\_for\_pr.hook\_methods.*[NEW#AUTO]*(*P\_GRP* == **null**,

*P\_PRIORITY* == max\_priority,

*P\_CLS* == 'HOOK',

*P\_SNAME* == 'S226\_MD\_FOR\_PROV',

*P\_PLP\_CODE* == **null**,

*P\_COMP* == **true**

);

[CONV].[LOG].WriteMess('В справочник ' ||::[APPAUX\_CALLPOINT]**%classname**|| ' для точки вызова MD\_FOR\_PROV был добавлен метод S226\_MD\_FOR\_PROV с приоритетом ' || max\_priority);

[CONV].[LOG].WriteMess('Окончание выполнения операции ['||&METHOD$SNAME||'] '||&METHOD$NAME);

[CONV].[LOG].InitLog;

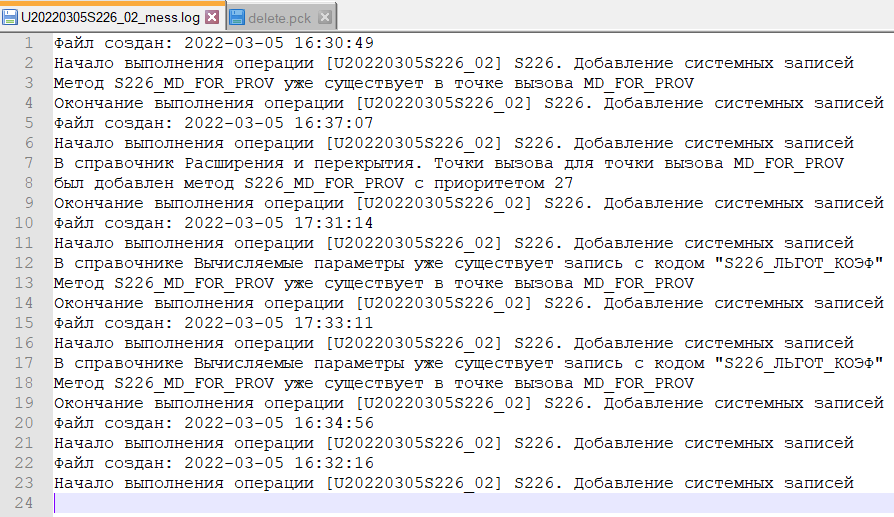
**end** **loop**;

**end**;

**end**;

**end**;

**Пример лога операции:**



**Задание 11**. *Освещается в лекции Подготовка хранилища с доработкой, создание операций конвертации*

* 1. Создать операцию конвертации в ТБП **«Обновления версии» (CONV\_57)**, с коротким именем UYYYYMMDD\_S##\_NN (где S## – префикс студента, NN – номер по порядку), которая удалит доработку со схемы:
* удалит данные из созданных справочников
* удалит добавленный метод из справочника **«Расширения и перекрытия. Точки вызова»** с компиляцией точки вызова
* удалит вычисляемый параметр.

**class** CONV\_57;

*@name*('S226. Удаление доработки') /\* Свойство операции "Наименование" \*/

**static** **method** U20220307S226\_03 **is**

-- удаление данных из справочников

**procedure** Delete\_Data **is**

s226\_prj\_kind [KIND\_PROJECT\_REF];

**begin**

-- удаление всех записей из справочника принадлежности кредита категории

**delete** t **in** ::[S226\_CRED\_CATEG]**all**;

[CONV].[LOG].WriteMess('В справочнике ' || ::[S226\_CRED\_CATEG]**%classname** || ' были удалены все записи"');

-- удаление всех записей из справочника категорий

**delete** t **in** ::[S226\_LOAN\_CTGR]**all**;

[CONV].[LOG].WriteMess('В справочнике ' || ::[S226\_LOAN\_CTGR]**%classname** || ' были удалены все записи"');

**exception**

**when** **others** **then**

[CONV].[LOG].WriteMess('Произошла ошибка при удалении записей из справочников'|| ::[S226\_CRED\_CATEG]**%classname** || ' и ' || ::[S226\_LOAN\_CTGR]**%classname**);

**end**;

-- удаление метода из справочника «Расширения и перекрытия. Точки вызова» с компиляцией точки вызова

**procedure** Delete\_Method **is**

mdForProvCallPoint **ref** [APPAUX\_CALLPOINT];

s226method **ref** [AUXAPP\_METHOD];

**begin**

**locate** mdForProvCallPoint **in** ::[APPAUX\_CALLPOINT]

**where** mdForProvCallPoint.*[CODE]* = 'MD\_FOR\_PROV';

**locate** s226method **in** mdForProvCallPoint.*[METHODS]*

**where** s226method.*[SNAME]* = 'S226\_MD\_FOR\_PROV';

s226method.[DELETE#AUTO](**true**);

-- вызвать перекомпиляцию точки вызова

mdForProvCallPoint.*[RECOMPILE]*;

[CONV].[LOG].WriteMess('В справочнике ' || ::[APPAUX\_CALLPOINT]**%classname** || ' в точке вызова "MD\_FOR\_PROV" был удален метод "S226\_MD\_FOR\_PROV"');

**exception**

**when** **others** **then**

[CONV].[LOG].WriteMess('Произошла ошибка при удалении метода "S226\_MD\_FOR\_PROV" в справочнике' || ::[APPAUX\_CALLPOINT]**%classname** || ' в точке вызова "MD\_FOR\_PROV"');

**end**;

-- удаление вычисляемого параметра

**procedure** Delete\_Calc\_Param **is**

**begin**

**delete** t **in** ::[CALC\_PARAMS] **where** t.*[CODE]* = 'S226\_ЛЬГОТ\_КОЭФ';

[CONV].[LOG].WriteMess('В справочнике ' || ::[CALC\_PARAMS]**%classname** || ' была удалена запись с кодом "S226\_ЛЬГОТ\_КОЭФ"');

**exception**

**when** **others** **then**

[CONV].[LOG].WriteMess('Произошла ошибка при удалении записи с кодом "S226\_ЛЬГОТ\_КОЭФ" в справочнике' || ::[CALC\_PARAMS]**%classname**);

**end**;

U20220307S226\_03 **is**

**execute** **is**

**begin**

[CONV].[LOG].InitLog(&METHOD$SNAME||'\_mess.log',**true**);

[CONV].[LOG].WriteMess('Начало выполнения операции ['||&METHOD$SNAME||'] '||&METHOD$NAME);

-- запускаем операции удаления

Delete\_Data;

Delete\_Method;

Delete\_Calc\_Param;

[CONV].[LOG].WriteMess('Окончание выполнения операции ['||&METHOD$SNAME||'] '||&METHOD$NAME);

[CONV].[LOG].InitLog;

**end**;

**end**;

**end**; /\* конeц static method U20220307S226\_03 \*/

* 1. Создать файл delete.pck с полным перечнем удаляемых объектов (представлений, операций и ТБП, созданных при выполнении задания).

**Содержимое файла delete.pck:**

METH S226\_LOAN\_CTGR DELETE\_AUTO

METH S226\_LOAN\_CTGR EDIT\_AUTO

METH S226\_LOAN\_CTGR NEW\_AUTO

CRIT VW\_CRIT\_S226\_LOAN\_CTGR

TYPE S226\_LOAN\_CTGR

METH S226\_CRED\_CATEG DELETE\_AUTO

METH S226\_CRED\_CATEG EDIT\_AUTO

METH S226\_CRED\_CATEG NEW\_AUTO

CRIT VW\_CRIT\_S226\_CRED\_CATEG

TYPE S226\_CRED\_CATEG

TYPE S226\_LOAN\_CT\_REF

CRIT PR\_CRED VW\_CRIT\_S226\_PR\_CRED

CRIT PR\_CRED VW\_CRIT\_S226\_PR\_CRED\_EXT

METH PR\_CRED S226\_DELETE\_EXT

METH PR\_CRED S226\_DELETE#AUTO

METH PR\_CRED S226\_EDIT\_LAST\_E

METH PR\_CRED S226\_EDIT\_LAST

METH PR\_CRED S226\_LOAN\_COEFF

METH PR\_CRED S226NEW\_DOG\_EXT

METH PR\_CRED S226NEW\_DOG

METH MAIN\_DOCUM S226\_DOC\_PRINT

METH HOOK S226\_MD\_FOR\_PROV

METH CONV\_57 U20220115S226\_01

METH CONV\_57 U20220305S226\_02

METH CONV\_57 U20220307S226\_03