§1 БАЗЫ ДАННЫХ

Грибанова-Подкина М.Ю.

ПРОГРАММНАЯ РЕАЛИЗАЦИЯ УЧЕТА ТОВАРА ПО ТЕХНОЛОГИИ FIFO

Аннотация: *В статье рассматривается модель партионного учета по технологии FIFO. Такая технология подразумевает последовательное списание (или расход) сырья на производство, начиная с самых ранних поставок. При этом ставится задача учета цены каждой расходуемой партии.Описанная технология является востребованной при ведении учета товаров, так как позволяет определить, какой товар, в каком количестве и по какой цене присутствует на складе. Это позволяет, например, в любой момент времени рассчитать реальную себестоимость продукции. Реализация представлена моделью базы данных и клиентскими компонентами, которые осуществляют регистрацию поставок и расхода товара. Особенностью и новизной предлагаемого подхода является избыточность в логической модели данных, за счет чего появляется более прозрачный механизм регистрации поставок.Приведенная реализация может рассматриваться как общая схема технологии списания сырья FIFO в автоматизированной информационной системе, и может быть использована на любой программной платформе.*

Ключевые слова: *FIFO, информационная система, программное обеспечение, автоматизация, списание сырья, программная реализация, расход товара, логическая модель, база данных, клиент-серверное приложение*

При разработке автоматизированной информационной системы по учету в кафе была поставлена задача списания сырья по технологии FIFO (first-in-first-out). Эта технология подразумевает последовательное списание (или расход) сырья на производство, начиная с самых ранних поставок [1, С.163]. То есть, имеются различные по времени поставки одного и того же сырья, оформленные, как правило, по разным ценам. В первую очередь расходуется сырье, ранее всех поступившее на склад, при этом учитывается цена расходуемой партии. На рисунке 1 списание сырья изображено нумерованными стрелками. Например, сырье из первой поставки было списано расходом 1. Далее, для расхода 2 оказалось недостаточно сырья из поставки 1, поэтому дополнительно списывается сырье из поставки 2. Для расходов 2 и 3 используется тот же принцип: списывается

DOI: 10.7256/2305-6061.2014.4.13738

оставшееся сырье из самой ранней поставки и, при необходимости, добирается сырье из следующей поставки.

Рисунок 1. Партионный учет FIFO

Реализация схемы FIFO в автоматизированной информационной системе представляет собой актуальную задачу для многих разработчиков, так как необходимо вписать эту схему в логическую модель данных системы. Предлагаемый далее подход вносит избыточность в логическую модель данных, за счет чего, во-первых, появляется более прозрачный механизм регистрации поставок и расхода сырья, а во-вторых, возникает необходимость отражения операций по поставкам и расходу в нескольких таблицах базы данных. Второй момент можно считать негативным, но за счет механизма транзакций его нельзя считать препятствием для реализации предложенного подхода.

Для программной реализации рассматриваемой схемы моделируется база данных, хранящая информацию обо всех необходимых сущностях (см. рисунок 2). Для поступления товара потребуются две таблицы:

- invoice — накладная, будет содержать сведения о номере накладной, дате поставки и поставщике;
- invoice_item – табличная часть накладной, содержит номер позиции, собственно приходуемый ресурс (товар), его цену и количество.

Аналогично расход оформляется двумя таблицами:

- outgo – документ на расход, содержит номер, дату и код клиента;

DOI: 10.7256/2305-6061.2014.4.13738

- outgo_item – табличная часть расходного документа, содержит номер позиции, расходуемый ресурс (товар) и количество товара в расходе.

Еще одна дополнительная сущность будет являться регистрационным журналом прихода и расхода товара. Это таблица register, содержащая сведения о дате операции, номере накладной и ее позиции, регистрируемом ресурсе, приходуемом и расходуемом количестве ресурса, а также о расходном документе и его позиции.

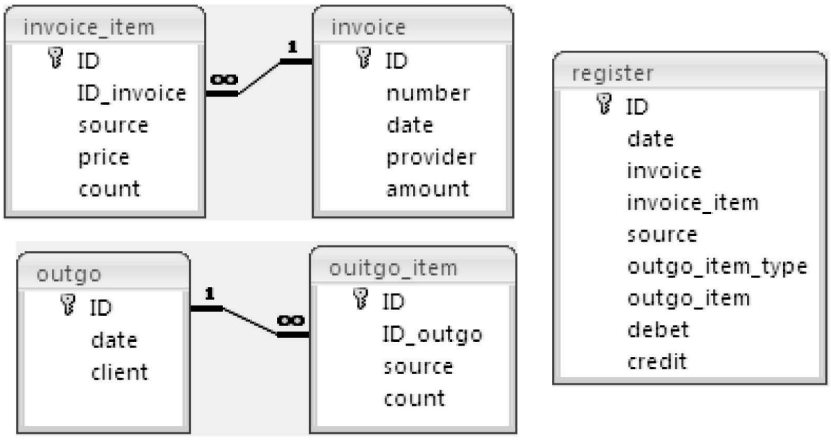

Рисунок 2. Логическая модель базы данных

Допустим, имеются две поставки товара с кодом 5, каждая поставка в количестве 10 единиц. В таблице invoice_item они будут зафиксированы, соответственно, двумя записями (см. таблица 1).

Таблица 1. Записи таблицы invoice_item

ID	ID_invoice	source	price	count
1	1	5	300	10
2	2	5	320	10

Теперь пусть товар с кодом 5 расходуется дважды, и каждый раз в количестве 7 единиц. В таблице outgo_item фиксируем также две записи (см. таблица 2).

Таблица 2. Записи таблицы outgo_item

ID	ID_outgo	source	count
1	1	5	7
2	2	5	7

Теперь необходимо зарегистрировать соответствие между поставками товара и его расходом. Для этого, согласно технологии FIFO, в таблице register будет пять записей: две на приход товара, одна на первый расход (7 единиц) и две на второй расход (3 единицы и 4 единицы). Каждый расход привязывается к соответствующему приходу (см. таблица 3). Эта привязка позволяет, при необходимости, выяснить и себестоимость каждого расхода.

DOI: 10.7256/2305-6061.2014.4.13738

Таблица 3. Записи таблицы register

ID	date	invoice	Invoice_item	source	debet	credit
1		1	1	5	10	
2		2	2	5	10	
3		1	1	5		7
4		1	1	5		3
5		2	2	5		4

Пользуясь таблицей register, всегда можно вычислить остатки по каждой партии товара. Для этого на серверной стороне обеспечивается наличие представления view_saldo_source, SQL-код которого приведен ниже:

```
select invoice_item.id_invoice as invoice, register.invoice_item,
register.source, sum(debet)-sum(credit) as saldo
from register, invoice_item
where register.invoice_item=invoice_item.id
group by invoice_item.id_invoice, register.invoice_item, register.source
order by register.invoice_item;
```

Разберем технологию формирования записей о списании в регистре с использованием этого представления. Перед списанием первых 7 единиц товара (до формирования строки 3 в таблице 3) представление view_saldo_source даст следующие строки:

Таблица 4. Записи представления view_saldo_source перед списанием первых 7 единиц товара

ID	invoice	Invoice_item	source	saldo
1	1	1	5	10
2	2	2	5	10

Списание первых 7 единиц товара должно происходить с партии, указанной в первой записи представления view_saldo_source (invoice_item=1). После этого списания формируется строка 3 в таблице 3, и представление view_saldo_source будет выглядеть так:

Таблица 5. Записи представления view_saldo_source перед первой итерацией процесса списания следующих 7 единиц товара

ID	invoice	Invoice_item	source	saldo
1	1	1	5	3
2	2	2	5	10

То есть, от первой партии осталось 3 единицы товара, от второй – 10 единиц. Далее, нужно списать еще 7 единиц товара. Используя первую строку представления view_saldo_source, списываем 3 единицы с первой партии (строка 4 таблицы 3) и вновь формируем представление view_saldo_source:

Таблица 6. Записи представления view_saldo_source перед второй итерацией процесса списания следующих 7 единиц товара

ID	invoice	Invoice_item	source	saldo
2	2	2	5	10

DOI: 10.7256/2305-6061.2014.4.13738

Снова используя первую строку представления, списываем оставшиеся 4 единицы товара уже из второй партии (invoice_item=1), формируя строку 5 таблицы 3.

В общем, нужное количество товара может списываться из нескольких партий несколькими записями в таблице реестра. Однако, может возникнуть такая ситуация, что расход будет превышать приход, поэтому необходимо проводить проверку наличия товара. Это можно делать в транзакции на каждой итерации процесса списания. Если на очередном этапе товара окажется недостаточно, то транзакция откатывается, и предыдущие записи реестра не фиксируются.

Таким образом, в клиентской части информационной системы (реализована в Delphi 7) для списания товара необходима реализация двух задач:

- сформировать остатки по данному товару;
- в цикле, пока не наберем нужное количество единиц товара для списания, формируем записи на расход товара из первой по порядку партии, полученной в обновленном представлении view_saldo_source.

Первая из этих задач обеспечивается выборкой из представления view_saldo_source по заданному параметру source. В среде Delphi это компонент ADOQSaldoSource класса TADOTable со следующим текстом:

```
select invoice,invoice_item, source, saldo
from view_saldo_source
where source=:source
```

Вторая задача подразумевает наличие компонента ADOCreditRegister класса TADOCommand с командой

```
insert into register
(invoice,invoice_item,source,outgo_item_type,outgo_item,debet,credit)
values
(:invoice,:invoice_item,:source,:outgo_item_type,:outgo_item,:debet,:credit)
```

Формирование записей на расход товара по партиям описано функцией клиентской части:

```
function CreditRegister(outgo_item_type_:String;outgo_item_,source_:integer;
count_:double): boolean;
      var credit,saldo: double;
begin
try
 DMForm.ADOConnection1.BeginTrans;
 credit:=count_;
 while credit>0 do begin
     //обновление представления с остатками
     DMForm.ADOQSaldoSource.Active:=false;
     DMForm.ADOQSaldoSource.Parameters.ParamByName('source').Value:=source_;
     DMForm.ADOQSaldoSource.Active:=true;
```


DOI: 10.7256/2305-6061.2014.4.13738

```
        //общие параметры для списания
        DMForm.ADOCreditRegister.Parameters.ParamByName('source').Value:=source_;
        DMForm.ADOCreditRegister.Parameters.ParamByName('outgo_item_type').
Value:=outgo_item_type_;
        DMForm.ADOCreditRegister.Parameters.ParamByName('outgo_item').Value:=
outgo_item_;
        DMForm.ADOCreditRegister.Parameters.ParamByName('debet').Value:=0;
        //если больше нет записей, то сырья нет в нужном количестве
        if DMForm.ADOQSaldoSource.RecordCount<1 then begin
          DMForm.ADOConnection1.RollbackTrans; //прерываем транзакцию
          MessageDlg(' Недостаточно товара на складе для списания!',mtInformation,
[mbOk],0);
          CreditRegister:=false;
          exit;
        //если сырье есть, то списываем
        DMForm.ADOQSaldoSource.First; //первая запись = первая партия товара
        //считываем из ADOQSaldoSource общие данные по накладной
        saldo:=DMForm.ADOQSaldoSource.FieldByName('saldo').AsFloat;
        DMForm.ADOCreditRegister.Parameters.ParamByName('invoice').Value:=
DMForm.ADOQSaldoSource.FieldByName('invoice').AsInteger;
        DMForm.ADOCreditRegister.Parameters.ParamByName('invoice_item').
Value:=DMForm.ADOQSaldoSource.FieldByName('invoice_item').AsInteger;
        if saldo<=credit then begin
          //если количество в партии недостаточное, то запись на расход
          //с количеством в партии
          ADOCreditRegister.Parameters.ParamByName('credit').Value:=saldo;
          credit:=credit-saldo; //уменьшаем следующий расход
        end else begin
          //иначе запись на расход с оставшимся несписанным количеством
          ADOCreditRegister.Parameters.ParamByName('credit').Value:=credit;
          credit:=0; //следующего расхода не будет, цикл закончится
        end;
        ADOCreditRegister.Execute; // формируем запись в таблице register
        end; //конец цикла
        DMForm.ADOConnection1.CommitTrans; //фиксируем транзакцию
        CreditRegister:=true; //процедура возвращает true
    except on E:Exception do begin
        DMForm.ADOConnection1.RollbackTrans;
        CreditRegister:=false;
        ShowMessage(E.Message);
```


DOI: 10.7256/2305-6061.2014.4.13738

```
        end
    end;
    end; //конец процедуры
```

Приведенная реализация может рассматриваться как общая схема технологии списания сырья FIFO и быть руководством для программной разработки. В частности, при использовании многослойной архитектуры приложения списание может быть реализовано как хранимая процедура в составе базы данных или процедура сервера приложений.

Библиография :

1. Кондраков Н.П. Бухгалтерский учет [Текст]: Учебник/Н.П. Кондраков.– М.:ИНФРА-М, 2007.– 592 с.

References:

1. Kondrakov N.P. Bukhgalterskii uchet [Tekst]: Uchebnik/N.P. Kondrakov.– M.:INFRA-M, 2007.– 592 s.

DOI: 10.7256/2305-6061.2014.4.13738