Введение в складской учет

Автор: Михаил Кривошеин (mikkri@mail.ru)
Статья была впервые опубликована в журнале «Программист» (www.programme.ru), февраль 2002 г.

Целью статьи является демонстрация некоторых проблем, возникающих при решении задач автоматизации бизнеса. Одной из самых массовых задач учета является складской учет, который заключается в своевременном и точном учете товаров на складах предприятия. В этой статье мы обсудим задачу автоматизации складского учета магазина сотовой связи.

Постановка задачи

Итак, руководство сети магазинов сотовой связи решило наладить оперативный учет. При этом директор оказался достаточно дальновидным, чтобы не браться за все сразу, а начать с одного магазина. В перспективе он планирует автоматизировать все магазины, а в центральном офисе (там, где сидит бухгалтерия) организовать сбор и обработку информации.

> **Оперативный учет** – своевременный учет результатов выполнения операций основного бизнеса предприятия. Не путать с бухгалтерским учетом.

Для реализации этих «грандиозных» планов был привлечен один программист, владеющий Delphi и немного понимающий в базах данных. Перед ним была поставлена задача: разработать программу складского учета товаров для магазина сотовой связи с учетом того, что товары бывают с серийными номерами и без.

После подробного обсуждения задачи было решено, что в системе должны найти отражение три операции: приход товаров из офиса, возврат товаров в офис и ввод отчета о продажах за день. При этом на первом этапе следовало ограничиться количественным учетом телефонов, пейджеров и аксессуаров.

Теперь самое время упомянуть, что описываемая сеть магазинов отличается высоким уровнем централизации. На практике это означает, что магазины занимаются только продажами телефонов и контрактов, а все остальные операции (взаимодействие с поставщиками, гарантийное обслуживание и бухгалтерский учет) выполняют сотрудники центрального офиса. Так что простая по функциональности программа вполне подходит для решения описанных задач.

Модель программы

Начало нового проекта наш программист пошел отметить к знакомому аналитику. Детали описывать не будем, но в результате они нарисовали модель программы.

Объекты предметной области

Нумерованный товар (телефоны и пейджеры) – товары, имеющие уникальные идентификационные номера (серийные номера). Товары этой группы необходимо учитывать по номерам, то есть для каждой номенклатурной единицы нужно иметь список номеров товаров, находящихся на складе.

> **Номенклатурная единица** – конкретный тип товара, например, телефон Nokia 3210.

Товар без номеров (аксессуары) – товары, не имеющие уникальных идентификационных номеров. Наиболее часто встречающаяся задача – количес твенный учет товаров данной группы. Это значит, что для каждой номенклатурной единицы достаточно иметь одно число – остаток товаров данного типа на складе.

Накладная на приход товаров – документ, содержащий список товаров, поступивших из офиса.

Накладная на возврат товаров – документ, содержащий список товаров, возвращенных в офис.

Накладная по продажам за день – документ, содержащий список товаров, проданных за день.

Реализация учета товаров

В соответствии со спецификой задачи нам предстоит реализовать две различные схемы учета товаров: нумерованных и ненумерованных.

Учет ненумерованных товаров
Проще всего хранить информацию об остатках товаров в такой таблице:

Наименование товара	Кол-во
Sony Z5	2
Sony J5	1
…	…
Nokia 3210	6

Это решение имеет ряд недостатков:
- Если в момент изменения таблицы произойдет сбой в БД или в программе, информацию об остатке товара в магазине может потеряться.
- В алгоритмах программы может быть ошибка, редко и на маленькое количество портящая результат работы.
- Человеческий фактор: накладная может быть введена с ошибкой, что приведет в негодность информацию о складских остатках.
- И еще: при такой структуре БД нельзя получить информацию о складских остатках на некоторую прошедшую дату – весьма важная функция складской программы, о которой легко забыть в связи с ее «очевидностью».

Какие выводы можно сделать? Во-первых, следует хранить не текущую ситуацию с остатками, а историю их изменения. Во-вторых, мы должны обеспечить возможность однозначной идентификации документа по конкретной записи истории изменений остатков. И, в-третьих, самое важное, мы должны обеспечить возможность с большой скоростью получать данные по складским остаткам, как на текущий момент, так и на любую прошедшую дату. Этим условиям вполне отвечает такая таблица.

Наименование товара	Дата	Документ	Кол-во
Sony Z5	04.09.01	Прих.№23	+5
...	...	...	...
Sony Z5	09.09.01	Прод.№102	-1
...	...	...	...
Nokia 3210	11.09.01	Возвр.№6	-3

Остается добавить: для идентификации документа, зарегистрировавшего операцию, следует использовать не текстовое поле, а целых два поля: для типа документа и его номера (своеобразный foreign key на таблицу, однозначно определяемую типом документа).

Учет нумерованных товаров
Напомню, при учете нумерованных товаров нужен доступ к списку серийных номеров конкретной номенклатурной единицы. При этом конкретный товар с конкретным серийным номером существует в единственном экземпляре.

Первое решение: использовать такую же таблицу, как для ненумерованных товаров, добавив колонку с серийным номером. Но, например, для того, чтобы получить список серийных номеров телефонов Nokia 3210 на складе, нужно будет выполнить такой запрос:

```
SELECT serial
FROM (
  SELECT table2.serial AS serial,
    sum(table2.quantity) AS total_quantity
  FROM table2
  WHERE table2.name = 'Nokia 3210'
  GROUP BY table2.serial
  )
WHERE total_quantity > 0
```

Как видите, для достижения такой простой цели пришлось использовать вложенный запрос. Значит необходимо придумать более качественное решение. Для этого можно использовать специфику нумерованного товара. Например, воспользуемся тем, что все серийные номера уникальны, а значит, конкретная единица товара может поступить в магазин, только если ее там нет и уйти со склада, только если она до этого была оприходована. Тогда можно все складские операции по конкретной единице товара однозначно разбить на пары «документ поступления» – «документ списания», упорядочив их по времени. В результате приходим к такой таблице.

Наименование товара	Серийный номер	Дата поступления	Документ поступления	Дата списания	Документ списания
Nokia 3210	4320341123	03.09.2001	Прих.№22	15.09.2001	Прод.№108
...	...	...	...	...	...
Nokia 3210	4320341204	15.09.2001	Прих.№25	NULL	NULL
Nokia 3210	4320341206	15.09.2001	Прих.№25	NULL	NULL

Столбцы «Наименование товара» и «Серийный номер» однозначно идентифицируют конкретную единицу товара. Столбцы «Дата поступления» и «Документ поступления» содержат информацию о документе, оформившем поступление товара. «Дата списания» и «Документ списания» содержат информацию о документе, оформившем расход товара со склада магазина. Разумеется, в БД таблица будет выглядеть несколько иначе: наименование товара будет заменено foreign key на справочник товаров, «Документ поступления» и «Документ списания» будут парами полей (для типа документа и для номера документа).

Можно заметить, что этот формат хранения данных удобнее для обработки, но насколько он удовлетворяет нашим требованиям?

Нам должна быть доступна история изменения данных: легко, выполняем запрос с фильтром по конкретной единице товара и получаем всю историю.

В каждой записи таблицы содержатся ссылки на документ поступления и документ списания: требование однозначной идентификации документа по конкретной записи истории изменений складских остатков выполнено.

Третье условие: мы должны обеспечить возможность с большой скоростью получать данные по складским остаткам, как на текущий момент, так и на любую прошедшую дату. Его рассмотрим поподробнее.

В контексте нумерованных товаров доступ к данным по складским остаткам сводится к доступу к данным о серийных номерах товаров на складе. Для получения списка серийных номеров воспользуемся утверждением, что товар находится на складе, пока его оттуда не списали, и напишем такой запрос:

```
SELECT serial FROM Table3
WHERE (name = 'Nokia 3210')
  AND (out_doc is NULL)
```

Как насчет любой прошедшей даты? Для этого нужно потребовать, чтобы товар поступил на склад до этой даты и был списан после этой даты или не был списан вообще:

```
SELECT serial FROM Table3
WHERE (name = 'Nokia 3210')
  AND (in_date <= :RequestDate)
  AND ( (out_doc IS NULL) OR (out_date > :RequestDate) )
```

Запрос немного хуже, но тоже выполняется достаточно быстро. Не стану утверждать, что приведенное решение – лучшее, просто замечу, что оно эффективно и удовлетворяет поставленным критериям качества.

Структура БД

Модель структуры БД
Теперь, когда понятно, что и как мы собираемся делать, можно приступить к моделированию структуры базы данных. Напомню, что в нашей программе будет использоваться 9 таблиц для одного справочника, трех документов (каждый с табличной частью) и двух регистров:

> **Регистр** – таблица специального назначения, призванная облегчить выполнение учетных операций программой

- Справочник товаров – таблица goods.
- Журнал документов «Поступление со склада» – таблица receipt_waybills.
- Сервисная таблица для хранения состава документов «Поступление со склада» – receipt_waybills_detail.
- Журнал документов «Возврат на склад» – таблица returnable_waybills.
- Сервисная таблица для хранения состава документов «Возврат на склад» – returnable_waybills_detail.
- Журнал документов «Продажи за день» – таблица sales_waybills.
- Сервисная таблица для хранения состава документов «Накладная по продажам за день» – sales_waybills_detail.
- Таблица для учета ненумерованных товаров – reg_goods.
- Таблица для учета нумерованных товаров – reg_numbered_goods.

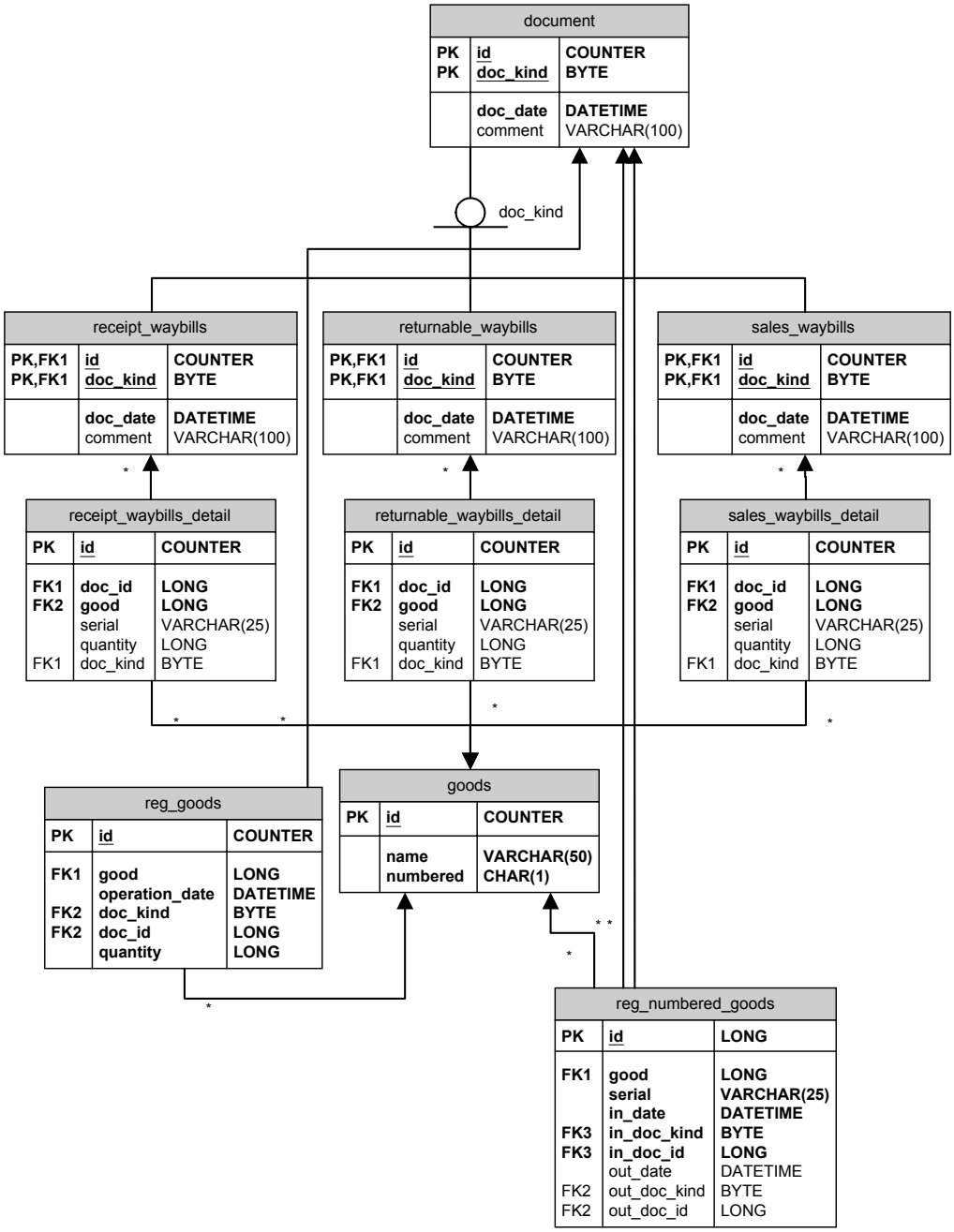
Справочник товаров
Для начала опишем справочник товаров. В принципе, можно было бы завести две таблицы: одну для нумерованных товаров, вторую для ненумерованных. Но для наших целей это решение не подходит. Дело в том, что во всех документах нашей программы нумерованные и ненумерованные товары выступают как равноправные сущности, и если их хранить в одной таблице, это упростит реализацию пользовательского интерфейса документов. Тем не менее, для разделения товаров на нумерованные и ненумерованные добавим в таблицу поле numbered типа char(1). В него будем записывать “Y”, если товар нумерованный, и “N” – в противном случае.

> Фактически, поле numbered содержит булевскую величину, и можно было бы использовать тип данных boolean. Но так как большинство checkbox’ов для Delphi ориентированы на работу с полем строкового типа, такая подмена оправдана. При использовании других инструментальных средств, такая «хитрость» может не понадобиться.

Документы: общие замечания
Рассмотрим абстрактную сущность document. Любой документ обязательно должен обладать двумя атрибутами: номером (id) и датой создания (doc_date). Вероятно, пользователям может понадобиться добавить к документу какое-то неформализованное замечание. Для этого введем текстовое поле comment.

Если бы программа поддерживала авторизацию пользователей, то имело бы смысл добавить поле author для имени пользователя, добавившего документ.

Теперь настало время вспомнить, что наша программа содержит несколько документов. А значит нужно поле для уточнения типа документа. Обозначим его как doc_kind типа byte.



Документ «Поступление со склада»

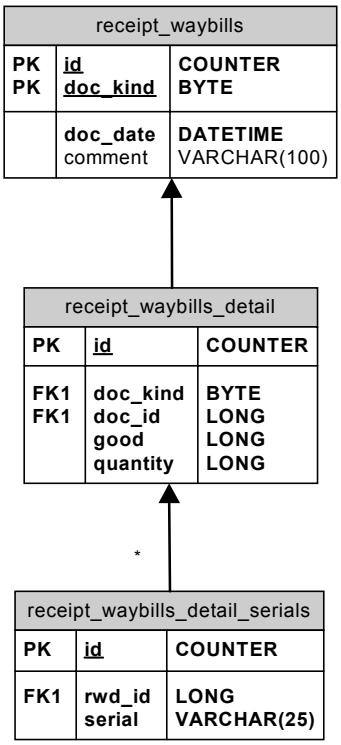
Этот документ является категорией описанного абстрактного документа, а значит наследует все его поля. Введенное поле doc_kind для данного типа документов всегда будет принимать значение «1».

Для хранения этого типа документов заведем таблицу receipt_waybills. Заметим, что документ содержит список товаров, поступивших со склада. Для хранения этих данных заведем таблицу receipt_waybills_detail и придем к соглашению, что поле doc_id – foreign key (внешний ключ) на таблицу receipt_waybills. В итоге мы получили классическое отношение master-detail между двумя таблицами.

Осталось сделать несколько замечаний по таблице receipt_waybills_detail:

- Поле good имеет тип long, так как мы используем его в качестве foreign key на таблицу goods, поле id.
- Поле serial имеет тип varchar(25) – строка переменной длины для серийного номера товара.

Рассмотрим специфику использования поля serial. Во-первых, оно часто содержит пустое значение. Во-вторых, если со склада пришло, скажем, десять телефонов одной модели, то придется ввести десять строк, отличающихся значением только одного поля. Эти недостатки должны натолкнуть нас на мысль использовать еще одну таблицу уже для уточнения строки накладной. В итоге документы «Поступление со склада» будут храниться в структуре данных, показанной на рисунке.



Если товар ненумерованный, то в таблицу receipt_waybills_detail_serials записей не добавляется и память БД впустую не расходуется. А при поступлении нескольких телефонов одной модели мы добавляем только одну строку в таблицу receipt_waybills_detail и несколько в receipt_waybills_detail_serials. Так как последняя содержит только список серийных номеров, можно говорить об эффективном использовании БД, да еще с потенциалом разработки удобного пользовательского интерфейса.

Данное решение имеет один недостаток, который нужно будет учесть при написании пользовательского интерфейса (или написать специальную хранимую процедуру на сервере БД): пользователь вводит строку в receipt_waybills_detail с информацией о нумерованном товаре, далее он вводит ровно N строк в таблицу receipt_waybills_detail_serial, а программа должна отследить, чтобы значение поля quantity таблицы receipt_waybills_detail совпало с N.

Один из возможных вариантов: введем понятие «проведенный документ» – документ, который полностью заполнен и должен учитываться программой. Соответственно, изначально документ не проведен, то есть новый. Для поддержания такого механизма надо добавить в таблицу receipt_waybills поле state, которое означает состояние документа. В нашем случае можно считать, что если документ новый, то значение поля quantity может отличаться от количества введенных серийных номеров. Такое положение позволяет пользователю начать заполнять накладную, потом выключить компьютер, а в следующий раз продолжить заполнение документа. В результате операция ввода накладной перестает быть атомарной! Эта возможность, несомненно, улучшает потребительские свойства продукта, так как ввод одной длинной накладной с большим количеством нумерованного товара может занять значительное время. Когда накладная будет заполнена, пользователь нажатием кнопки приведет ее в состояние «Проведена», и только в этот момент программа осуществит операции по модификации регистров. Программа, перед тем как осуществлять проведение, может

> В общем случае документ может иметь большое количество состояний, например: «новый», «на проверке», «утвержден», «отменен», «удален».

проверить введенные данные на согласованность, таким образом, устранив потенциальную ошибку, когда значение поля quantity не совпадает с количеством фактически введенных серийных номеров.

Документ «Возврат на склад»
Этот документ почти полностью аналогичен «Поступлению со склада». Единственное отличие –он совершает операции списания по регистрам, а не прихода.

Документ «Продажи за день»
Учет продаж можно организовать двумя способами: информация о продаже может попадать в систему в момент ее совершения (так работают POS-терминалы в крупных супермаркетах) или вводиться в конце некоторого отчетного периода (способ, не требующий компьютеризации вообще).

В контексте маленького магазина сотовой связи лучшее, на что можно надеятся (и что экономически оправдано) – это один офисный компьютер, в который вводится информация о продажах за прошедший день. В принципе, такой уровень оперативности данных вполне удовлетворит потребности данного вида бизнеса и при этом не потребует существенных материальных затрат. Итак, нашей программе нужен документ, позволяющий вводить информацию о продажах за день.

Рассмотрим его реализацию. По сравнению с документом «Возврат на склад», можно указать только одно существенное отличие: в контексте этого документа замечание по поводу целесообразности использования вместо поля serial в таблице sales_waybills_detail целой таблицы sales_waybills_detail_serials становится спорным, так как операция продажи, как правило, связана с одним телефоном.

Поле doc_kind
В реальной базе данных нет необходимости заводить поле doc_kind, так как тип документа однозначно определяется названием таблицы. На рисунке 1 это поле было введено для большей наглядности при описании ссылок на документы из таблиц регистров (reg_goods и reg_numbered_goods). Так что положим, что для документа «Поступление со склада» doc_kind равно 1, для «Возврата на склад» – 2, а для «Продаж за день» – 3. Этим мы воспользуемся при проведении документов.

Регистры
Таблицы регистров уже были весьма подробно описаны, так что здесь приведем только несколько комментариев по их структуре.

reg_goods		
PK	**id**	**COUNTER**
FK1	**good**	**LONG**
	operation_date	**DATETIME**
FK2	**doc_kind**	**BYTE**
FK2	**doc_id**	**LONG**
	quantity	**LONG**

reg_numbered_goods		
PK	**id**	**LONG**
FK1	**good**	**LONG**
	serial	**VARCHAR(25)**
	in_date	**DATETIME**
FK3	**in_doc_kind**	**BYTE**
FK3	**in_doc_id**	**LONG**
	out_date	DATETIME
FK2	out_doc_kind	BYTE
FK2	out_doc_id	LONG

В обоих регистрах для указания учитываемого товара используется foreign key на таблицу goods.

Обе таблицы ссылаются на документы, в которых зарегистрированы соответствующие операции. Для регистра reg_goods это поля doc_kind и doc_id. Для регистра reg_numbered_goods это поля in_doc_kind, in_doc_id и out_doc_kind, out_doc_id, ссылающиеся, соответственно, на документ поступления нумерованного товара и на документ расхода нумерованного товара.

Также заслуживает пояснения использование полей operation_date в reg_goods и in_date, out_date в reg_numbered_goods. Хотя значения данных полей и являются функцией от ссылок на документы, тем самым нарушая требования третьей нормальной формы, нам имеет смысл пойти на

компромисс, так как в ряде случаев это существенно ускорит выполнение запросов. Но при реализации проведения документов необходимо уделить особое внимание поддержанию целостности данных регистров.

Индексы
Для реализации такой простой программы нам потребовалось завести достаточно много таблиц, связанных между собой. Соответственно, нужно проиндексировать все primary key и foreign key нашей базы данных.

Вопрос, как индексировать – выходит за рамки этой статьи, посвященной подходу к решению задачи автоматизации учета, тем более, что это зависит от СУБД.

Реализация
Перейдем к реализации нашего проекта на Delphi 5 с использованием библиотеки ADOExpress (компоненты Delphi для работы с Microsoft ADO).

> Мой опыт использования ADOExpress говорит, что следует обязательно установить первый и второй service pack'и на Delphi 5, а так же MDAC 2.6. Если вы используете Microsoft Jet OLE DB Provider (для баз данных Access), нелишне установить SP3 и SP5 к нему.

В первую очередь определимся с составом проекта. Для интерфейса программы в стиле SDI нам понадобятся:
- Главная экранная форма с меню.
- Экранная форма справочника товаров.
- Журнал документов «Поступление со склада».
- Экранная форма документа «Поступление со склада».
- Журнал документов «Возврат на склад».
- Экранная форма документа «Возврат на склад».
- Журнал документов «Продажи за день».
- Экранная форма документа «Продажи за день».
- Экранная форма для отчета по движениям товаров.
- Экранная форма для отчета по складским остаткам.

Главная форма
Экранную форму можно сделать предельно простой: строка меню и несколько «быстрых» кнопок. В принципе, пользователь может открыть одновременно много экранных форм, поэтому полезно будет предоставить возможность просмотреть их список, например, через меню «Окна».

Справочник товаров
Достаточно простая экранная форма, содержащая всего один DBGrid. Один совет – не забыть указать сортировку в SQL-запросе – неупорядоченные таблицы очень раздражают пользователей.

Журналы документов
Журнал документов – список документов некоторого вида, введенных в БД. В журнале должны быть предусмотрены следующие возможности:
- Отбор документов за некоторый период времени.
- Фильтрация по конкретному автору.
- Выбор состояния отображаемых документов (новый, проведенный, отмененный).
- Фильтрация по значению одного из полей документа. Например, по поставщику или покупателю приходных и расходных накладных соответственно.
- Поиск документа по специфическим критериям.

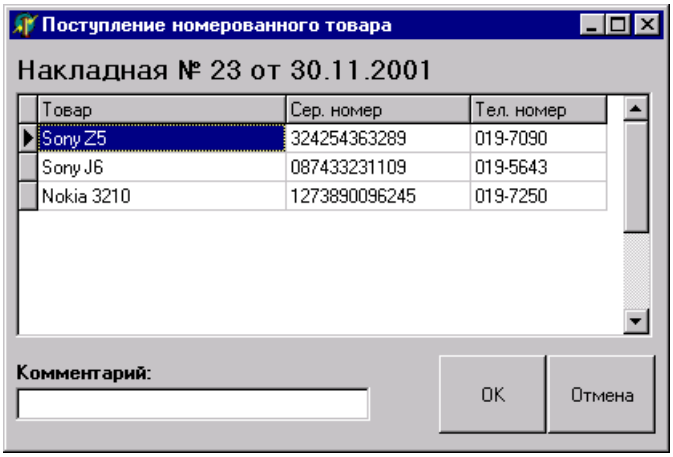
В нашей простой программе реализуем только отбор документов за определенный период времени. Неплохой идеей будет автоматический отбор документов за последние десять дней.

Второй ключевой особенностью журналов документов по отношению к справочникам является то, что они не должны допускать непосредственного изменения данных в экранной форме. Для внесения изменений необходимо предусмотреть три кнопки: ввести новый, открыть

существующий и удалить документ. Причем после того, как пользователь введет новый документ или изменит существующий, программа должна обновить данные экранной формы журнала.

Экранные формы документов
Самая сложная часть пользовательского интерфейса – экранные формы документов. Общепринята следующая структура пользовательского интерфейса документа:

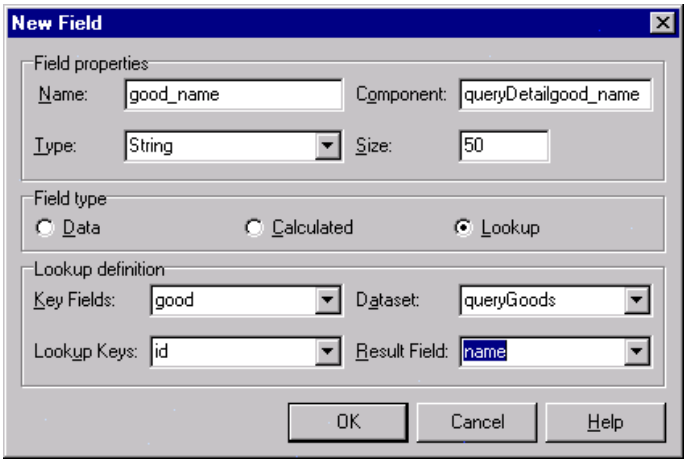
- В самом верху – название документа, номер и дата.
- Чуть пониже – поля документа. В нашем случае их нет.
- Еще ниже – табличная часть документа.
- В самом низу – строка комментария.
- В правом нижнем углу – кнопки «ОК» и «Отмена».



Поле good является ссылкой, и у нас появляются две проблемы:

- Для выбора товара хотелось бы использовать экранную форму справочника товаров.
- Вместо отображения значения поля следует отображать значение поля name соответствующей записи таблицы goods.

Начнем со второй задачи. Для доступа к данным можно использовать компоненты TADOQuery и queryDetail – источник данных для табличной части. Дополним экранную форму документа компонентом queryGoods типа TADOQuery с запросом к таблице товаров (SELECT * FROM goods). Теперь настроим компонент queryDetail: по двойному щелчку откроется редактор состава полей. В нем следует добавить все поля (right click → Add all fields); после этого необходимо создать еще одно поле (right click → New field…). В появившемся диалоге укажите имя поля good_name, тип string, размер 50, установите Lookup; далее заполните поля в соответствии с рисунком.



После нажатия «ОК» появится еще одна колонка, которая будет содержать текстовое представление значения поля good. Значение этого поля можно будет выбрать из списка товаров, введенных в таблицу goods. Осталось только установить в свойствах поля good visible=false.

Теперь разберемся с первой задачей. В свойстве Columns у DBGrid для колонки good_name в свойстве ButtonStyle укажем cbsEllipsis. В результате поле good_name не будет редактироваться в grid'е, а будет появляться кнопка с тремя точками для редактирования. По нажатию этой кнопки будет вызываться обработчик события OnEditButtonClick нашей таблицы.

Процедура-обработчик должна выполнить следующие действия:
1. Считать текущее значение поля good.
2. Модально открывать форму справочника товаров с курсором, установленным на текущий товар.
3. При нажатии кнопки «ОК» или по двойному щелчку форма справочника будет закрываться.
4. Учитывая значение ModalResult формы справочника, изменять значение поля good на значение поля id текущей записи экранной формы справочника товаров.

Экранная форма справочника товаров должна работать в двух режимах:
- Для ввода данных.
- Для выбора значения.

В первом режиме в правом нижнем углу должна располагаться единственная кнопка «Закрыть». Во втором – две кнопки: «ОК» и «Отмена».

Теперь следует вспомнить о самом документе, точнее о том, как его ввод или изменение влияет на базу данных. Дело в том, что по ряду причин мы используем несколько избыточную структуру базы данных. Соответственно, необходимо следить за ее согласованностью, поэтому все изменения документов следует выполнять в транзакции. Дополнительной причиной является проведение документов, то есть по нажатию кнопки «ОК» программа должна вносить изменения в таблицы регистров. Очевидно, что по кнопке «Отмена» она должна отменять все изменения, что достигается отменой транзакции.

Если вы используете базы данных MS Access, то необходимо учесть, что при изменении таблицы в рамках транзакции происходит ее монопольная блокировка. Если программа должна поддерживать многопользовательский режим, лучше либо отказаться от транзакций, либо использовать MSDE или другую клиент-серверную базу данных.

Проведение

Опишем теперь проведение документов на примере «Поступления со склада». Для начала следует добавить на форму компонент TADOCommand, с помощью которого мы будем выполнять SQL-запросы. При этом для повышения производительности следует установить свойство prepared=true.

На этапе проведения программа должна перебрать в цикле все строки табличной части документа и сделать по ним движения регистров:
- Для ненумерованных товаров:
```
INSERT INTO reg_goods
(good, operation_date, doc_kind, doc_id, quantity)
VALUES (:good, :doc_date, 1, :doc_id, :quantity)
```
- Для нумерованных товаров:
```
INSERT INTO reg_numbered_goods
(good, serial, in_date, in_doc_kind, in_doc_id)
VALUES (:good, :serial, :doc_date, 1, :doc_id)
```

Перед выполнением SQL-запроса необходимо инициализировать значения параметров. Аналогично работают другие документы.

В заключение сделаю одно замечание. Возможна ситуация, когда пользователь повторно откроет документ, изменит его данные, и нажмет «ОК». Для корректной обработки таких ситуаций перед проведением документа с новым содержанием следует отменить результаты предыдущего

проведения документа. Сделать это достаточно просто. Например, для «Поступление со склада» – это две SQL-команды:

```
DELETE *
FROM reg_goods
WHERE (doc_kind = 1) AND (doc_id = :doc_id)
```

и

```
DELETE *
FROM reg_numbered_goods
WHERE (in_doc_kind = 1) AND (in_doc_id = :doc_id)
```

Перед выполнением второго запроса нужно проверить возможность отмены предыдущего проведения. Ведь если товар, оприходованный в прошлый раз по этому документу, был уже списан со склада, корректная отмена затруднена. Фактически для этого понадобится перепровести также все расходные документы по нумерованным товарам, оприходованным этим документом. В нашем случае сделаем простую проверку на возможность отмены проведения и, если были расходы товаров, уведомим пользователя о невозможности повторного проведения.

Для реализации последнего замечания нужно узнать, есть ли в БД записи, удовлетворяющие запросу:

```
SELECT *
FROM reg_numbered_goods
WHERE (id_doc_kind = 1)
  AND (in_doc_id = :doc_id)
  AND NOT (out_doc_id IS NULL)
```

Как вы могли заметить, перепроведение документов может нарушить структуру базы данных и разработчик должен относиться к нему с особым вниманием.

Дополнения

Стоимостной учет
Стоимостной учет заключается в расчете стоимости складских остатков товаров. Причем делать это нужно в соответствии с нормами бухгалтерского учета.

При учете нумерованного товара его списание со склада должно происходить по той же цене, что и поступление. Учет ненумерованных товаров требует использования одной из методик расчета стоимости складских остатков.

Аналогично приход товара на склад осуществляется по цене из накладной. Но стоимость списываемого со склада товара следует вычислять по одной из трех методик: по среднему, FIFO или LIFO. Далее мы рассмотрим их требования и особенности программной реализации.

По среднему
В методике складского учета по среднему стоимость списываемого товара рассчитывается по формуле:

```
IF К_Расх = К_Ост
THEN С_Расх = С_Ост
ELSE С_Расх = К_Расх * (С_Ост / К_Ост)
```

Где К_Расх, С_Расх, К_Ост и С_Ост – количество списываемого товара, стоимость списываемого товара, остаток товара на складе и стоимость остатков товара на складе соответственно.

FIFO
В буквальном переводе название обозначает «Первым пришел – первым ушел». Суть методики заключается в том, что мы учитываем не стоимость складских остатков в целом, а стоимость остатков товаров из конкретных партий. Соответственно, когда мы списываем товар со склада, он

сперва списывается из самой «старой» партии, а только потом из более новых, в соответствии с понятием очереди.

Рассмотрим пример. Пусть у нас на складе три партии товара (одной и той же товарной номенклатуры):

```
П1: в количестве 10 шт. по цене 99 у.е.
П2: в количестве 5 шт. по цене 105 у.е.
П3: в количестве 15 шт. по цене 89 у.е.
```

Тогда при списании 5 телефонов их «складская» стоимость будет 5 * 99 = 495 у.е., так как первая партия состоит из более чем 5 телефонов.

Рассмотрим другой случай. Нам необходимы 17 телефонов. Произведем расчет их стоимости:

```
C_Расх = (10*99 + 5*105 + 2*89) у.е. = 1693 у.е.
```

Как мы видим, такой «большой» расход товара привел к полному списанию первой и второй партий и к списанию двух телефонов из третей партии.

LIFO
LIFO соответствует концепции стека: последним пришел – первым ушел. В целом его использование аналогично FIFO, проиллюстрируем его тем же примером:

При списании 5 телефонов их «складская» стоимость будет 5 * 89 = 445 у.е., так как последняя партия состоит из более чем 5 телефонов.

Теперь рассчитаем стоимость 17 телефонов:

```
C_Расх = (15*89 + 2*105) у.е. = 1545 у.е.
```

При списании 17 единиц товара будет полностью исчерпана третья партия, а так же будут списаны два телефона из второй партии.

Особенности реализации
Для начала сделаю одно замечание. Если программа должна позволять вести учет на нескольких складах, разумно завести еще один регистр для учета стоимости складских остатков (независимо от методики), так как все методики расчета складских остатков игнорируют факт возможности использования нескольких складов.

Например, реализация FIFO или LIFO может потребовать создания такой таблицы.

Наименование товара	Дата	Партия	Стоимость
Sony Z5	04.09.01	231	525
...	...	...	...
Sony Z5	09.09.01	231	125
...	...	...	...
Nokia 3210	11.09.01	131	-560

В качестве уникального кода партии можно использовать выражение DOC_ID*10+DOC_KIND. Стоит обратить внимание на то, что в приведенной таблице при списании товара со склада идет списание с партии, порожденной приходной накладной. То есть расходные документы не должны создавать новых партий.

Даже если вы не планируете реализацию учета на нескольких складах, следует серьезно рассмотреть вопрос об использовании одного регистра для количественного учета и еще одного для стоимостного.

При удачном выборе структуры регистров реализация методик складского учета не должна вызывать алгоритмических затруднений.

Ввод «задним числом»
К сожалению, на практике часто встречаются ситуации, когда данные вводятся «задним числом». Проблема заключается в том, что работа многих программ существенно зависит от порядка ввода информации. Наглядным примером может служить стоимостный учет складских остатков. Если мы «забыли» ввести приходную накладную, весьма вероятно, что при списании товаров стоимость списываемого товара будет рассчитана неверно.

При вводе приходной накладной «задним числом» программа должна заново провести все расходные накладные, введенные после нее. На мой взгляд, в корректной обработке ввода документов «задним числом» заключается основная сложность складских программ.

Импорт-экспорт данных
Вспомним о том, что мы пишем программу для сети магазинов сотовой связи, а значит, придется изучить вопрос обмена данными «офис-магазин». С учетом современных тенденций оформления данных целесообразно использовать для передачи XML-файлы.
Задача обмена информацией разбивается на две составляющие:

- Загрузка приходных накладных: из офиса вместе с бумажными накладными на товар доставляются их электронные версии в формате XML, по которым мы генерируем документы в нашей базе данных.
- Выгрузка расходных накладных: результаты продаж и накладные на возврат в офис мы оформляем в нашей программе и выгружаем во внешний файл. При этом надо выгружать только новые накладные (с точки зрения офисной программы).

В дополнение к перечисленным функциям обмена можно предусмотреть механизмы отказоустойчивости. Например, мы отправили в офис накладные с 23 по 27. Офисная программа их подгрузила и вернула отчет о получении накладных с 23 по 27. Загрузив файл отчета, программа в магазине устанавливает счетчик последней выгруженной накладной на 27. Если же по каким-то причинам программа в магазине не получила подтверждение, при следующей выгрузке данных она повторно пошлет накладные с 23 по 27 и так далее.

Заключение

Пора подвести итог. Надеюсь, вы получили общее представление о разработке складских программ, и, может быть, пополните ряды программистов автоматизирующих магазины и торговые фирмы. По крайней мере, если теперь вы не считаете, что складские программы – это нечто большое и ужасное, моя задача выполнена.

Кто-то может задать вопрос: сколько стоит разработка такой складской программы? На него нельзя дать точного ответа, так как каждая программа имеет свои особенности. Очень важно учесть то, что на рынке присутствует огромное количество готовых программ для складского учета, и если они не отвечают специфике организации, может оказаться проще и дешевле переделать уже готовые продукты.

Если у вас есть вопросы или замечания, пишите мне на mikkri@mail.ru, особо интересные вопросы и ответы на них будут выложены по адресу http://mikkri.narod.ru/library/article1.htm.