



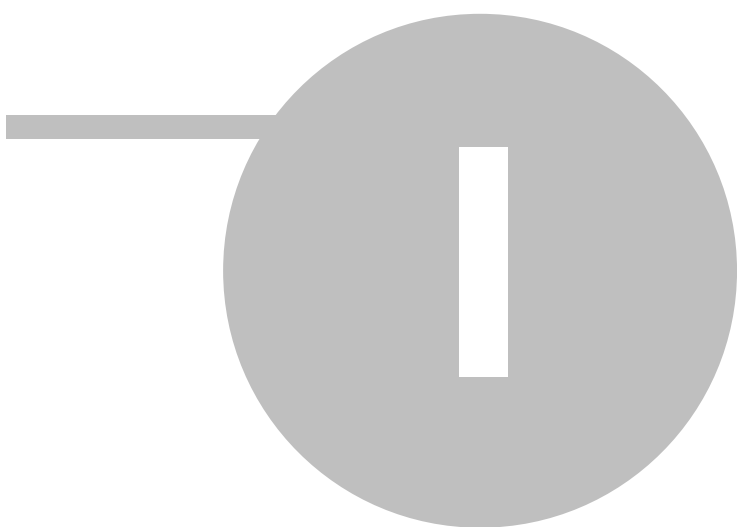
**CoolLib**

**2007**

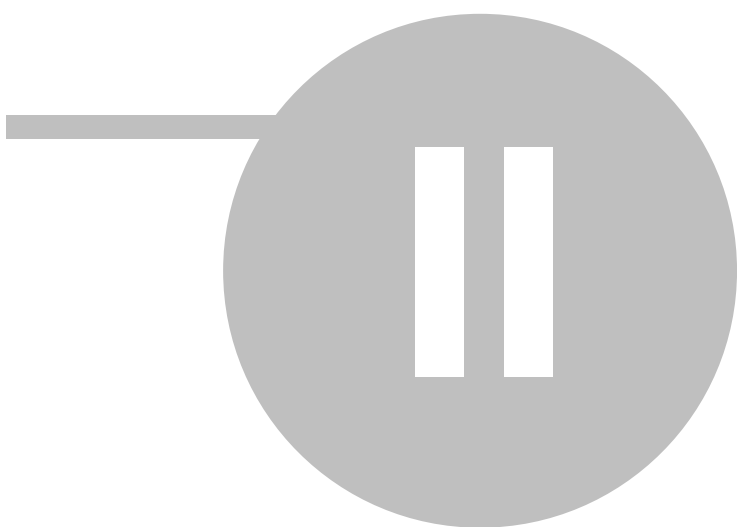
<b>I</b>		<b>2</b>
<b>II</b>		<b>4</b>
1	.....	4
2	.....	4
3	.....	4
4	.....	5
5	.....	6
6	.....	7
<b>III</b>		<b>10</b>
1	.....	11
2	.....	15
3	.....	15
<b>IV</b>		<b>19</b>
1	.....	23
2	.....	24
3	.....	25
<b>V</b>		<b>29</b>
1	TcmScriptBase .....	29
2	TcmBases .....	29
3	TcmBase .....	30
4	TcmBaseSeg .....	30
5	TcmBaseUnit .....	30
<b>VI</b>		<b>33</b>
1	TcmRecvizeitData .....	33
2	TcmParamRecvizeitData .....	34
3	TcmRecvizeitList .....	34
4	TcmRecvizitsEdit .....	34
5	TcmSprEdit .....	35
6	TcmDocEdit .....	36
7	TcmDocSpecEdit .....	39

8	TcmTableEdit .....	40
9	TcmRezervEdit .....	42
10	TcmOstatokList .....	42
11	TcmOstatokRec .....	43
<b>VII</b>		<b>45</b>
1	TcmTransaction .....	45
2	TcmBufData .....	46
3	TcmQuery .....	48
4	TcmDataSet .....	49
5	TcmRecord .....	54
<b>VIII</b>		<b>57</b>
1	TcmMenuRoot .....	57
2	TcmMenuItem .....	57
3	TcmMenuGrantData .....	58
<b>IX</b>		<b>61</b>
1	TcmGridEh .....	61
2	TcmPrintGridEh .....	62
3	TcmDataProducer .....	62
4	TcmToolBar .....	63
	TcmToolBar .....	65
5	TcmObjectEdit .....	66
	TcmObjectEdit .....	67
6	TcmEnumEdit .....	68
7	TcmImageList .....	68
<b>X</b>		<b>71</b>
1	TcmReport .....	71
2	TcmCompositeReport .....	72
3	TcmUserDataSet .....	74
4	.....	74
<b>XI</b>		<b>77</b>
1	TcmClosePeriod .....	77
<b>XII</b>		<b>82</b>
1	.....	83
	TcmInterDataParam .....	83
	TcmInterDataItem .....	85
	TcmOutProcessItem .....	87
2	TcmSendData .....	87

<b>3</b>	<b>TcmLoadData .....</b>	<b>88</b>
<b>4</b>	<b>TcmUnloadData .....</b>	<b>88</b>
<b>5</b>	<b>TcmSendToMail .....</b>	<b>89</b>



**1**



## 2

## 2.1



TcmMetadate = class(TcmUnitComponent)

Заголовок метаданных. В конфигураторе подчинен компоненту TcmUnitComponent.

Свойства и методы	Описание
function GetEnumerator: TcmEnumList	Возвращает список перечислений.
function GetEnumerator(AName: string): TcmEnum	Возвращает ссылку на перечисление по имени.
function GetSprList: TcmSprList	Возвращает список справочников.
function GetSpr(AName: string): TcmSpr	Возвращает ссылку на справочник по его имени.
function GetDocList: TcmDocList	Возвращает список документов.
function GetDoc(AName: string): TcmDoc	Возвращает ссылку на документ по его имени.
function GetTableList: TcmTableList	Возвращает список сводных таблиц.
function GetTable(AName: string): TcmTable	Возвращает ссылку на таблицу по её имени.

## 2.2

Для описания перечислений, используются следующие метаклассы:



TcmEnumList = class(TcmUnitComponent)

Список перечислений. В конфигураторе подчинен компоненту TcmMetadate.

Свойства и методы	Описание
property Enums[Index: integer]: TcmEnum	Индексированное свойство для доступа к перечислениям. Перечисления нумеруются в диапазоне от 0 до Count-1.
function Count: integer	Число определенных перечислений.
function GetNumByName(AName: string): TcmEnum	Возвращает перечисление по имени.



TcmEnum = class(TcmUnitComponent)

Метакласс перечисления. В конфигураторе подчинен компоненту TcmEnumList.

Свойства и методы	Описание
property EnumName: string	Идентификатор перечисления
property Enum: TcmEnumCollection	Список элементов перечисления
function ConvertEnumValue(Value: string): string	Проверяет корректность переменной Value. Value содержит строковое значение перечисления.

TcmEnumCollection = class(TCollection)

Список значений перечисления, каждый элемент списка содержит объект класса TcmEnumItem.

TcmEnumItem = class(TCollectionItem)

Класс для описания элемента перечисления.

Свойства и методы	Описание
property Code: integer	Уникальный ID элемента перечисления.
property Name: string	Имя элемента перечисления.

## 2.3

Реквизиты используются при описании справочников, документов, сводных таблиц.

Типы реквизита:

TcmRecvType = (cmrUnknown, cmrInteger, cmrInt64, cmrCurrency, cmrDouble, cmrDateTime, cmrDate, cmrTime, cmrString, cmrEnum,



cmrBoolean, cmrSpr, cmrTable)

- cmrUnknown - тип не определен
- cmrInteger - целое
- cmrInt64 - целое 64 битное
- cmrCurrency - деньги
- cmrDouble - вещественное
- cmrDateTime - дата и время
- cmrDate - дата
- cmrTime - время
- cmrString - строка
- cmrEnum - перечисление
- cmrBoolean - логическое
- cmrSpr - справочник
- cmrTable - таблица



TcmRecvizit = class(TcmUnitComponent)

Реквизит справочника, документа или сводной таблицы.

Свойства и методы	Описание
<b>property</b> RecvName: string	Идентификатор реквизита
<b>property</b> RecvType: TcmRecvType	Тип реквизита
<b>property</b> Precision: smallint	Число знаков после запятой для вещественных чисел
<b>property</b> Required: boolean	Если равен true - задание значения реквизита обязательно
<b>property</b> SprRef: TcmSpr	Ссылка на метакласс справочника, если RecvType=cmrSpr
<b>property</b> TableRef: TcmTable	Ссылка на метакласс сводной таблицы если RecvType=cmrTable
<b>property</b> EnumRef: TcmEnum	Ссылка на метакласс перечисления если RecvType=cmrEnum

TcmRecvizitComponent = class(TcmUnitComponent)

Список реквизитов.

Свойства и методы	Описание
<b>property</b> Recvizits[Index: integer]: TcmRecvizit	Индексированное свойство для доступа к списку реквизитов. Значение параметра Index должно находиться в диапазоне от 0 до Count-1
<b>function</b> Count: integer	Число реквизитов в списке.
<b>function</b> GetRecvizitByName (AName: string): TcmRecvizit	Возвращает ссылку на метакласс реквизита по его имени.

## 2.4

Типы справочников:

TcmSprType=(cmsList, cmsTree);


- cmsList - список
- cmsTree - дерево



TcmSprList = class(TcmUnitComponent)

Список справочников, в конфигураторе подчинен компоненту TcmMetadate.

Свойства и методы	Описание
<b>property</b> SprList[Index: integer]: TcmSpr	Индексированное свойство для доступа к списку справочников. Параметр Index должен находиться в диапазоне от 0 до Count-1.
<b>function</b> Count: integer	Число справочников в списке SprList.
<b>function</b> GetSprByName (SprName: string): TcmSpr	Возвращает ссылку на метакласс справочника по его имени (свойство TcmSpr.SprName).
<b>function</b> GetSprByCode (SprCode: integer): TcmSpr	Возвращает ссылку на метакласс справочника по его коду (свойство TcmSpr.SprCode).

 TcmSpr = class(TcmRecvzitComponent)

Метакласс справочника. В конфигураторе подчинен компоненту TcmSprList.

Свойства и методы	Описание
<b>property</b> SprCode: smallint	Уникальный код справочника.
<b>property</b> SprName: string	Идентификатор справочника. Каждый справочник должен иметь уникальный идентификатор.
<b>property</b> SprType: TcmSprType	Тип справочника.
<b>property</b> SprOwner: TcmSpr	Ссылка на метакласс справочника владельца.
<b>property</b> EditForm: string	Имя формы для редактирования записи справочника.
<b>property</b> ShowForm: string	Имя формы для выбора записи справочника.
<b>property</b> CheckForm: string	Имя формы для выбора группы записей справочника.
<b>property</b> HistoryForm: string	Имя формы для просмотра истории редактирования записи справочника.
<b>property</b> OnNewRecord: TNotifyEvent	Событие при создании новой записи справочника. Вызывается в методе TcmSprEdit.New
<b>property</b> OnBeforePost: TNotifyEvent	Событие перед сохранением записи справочника. Вызывается в методе TcmSprEdit.Post
<b>property</b> OnAfterGet: TNotifyEvent	Событие после чтения записи справочника. Вызывается в методе TcmSprEdit.Get
<b>property</b> OnBeforeDelete: TNotifyEvent	Событие перед пометкой записи на удаление. Вызывается в методе TcmSprEdit.Delete
<b>property</b> OnAfterRestore: TNotifyEvent	Событие после снятия пометки записи на удаление. Вызывается в методе TcmSprEdit.Restore

*Примечание:* Все события вызываются в соответствующих методах компонента TcmSprEdit (см. раздел Компонент TcmSprEdit). В качестве параметра Sender всем событиям передается ссылка на объект класса TcmSprEdit.

## 2.5

 TcmDocList = class(TcmUnitComponent)

Список документов. В конфигураторе подчинен компоненту TcmMetadate.

Свойства и методы	Описание
<b>property</b> DocList[Index: integer]: TcmDoc	Индексированное свойство для доступа к списку документов. Параметр Index должен находиться в диапазоне от 0 до Count-1.
<b>function</b> Count: integer	Число документов в списке DocList.
<b>function</b> GetDocByName(AName: string): TcmDoc	Возвращает ссылку на метакласс документа по его имени (свойство TcmDoc.DocName)
<b>function</b> GetDocByCode(ACode: integer): TcmDoc	Возвращает ссылку на метакласс документа по его коду (свойство TcmDoc.DocCode)


 TcmDoc = class(TcmUnitComponent)

Метакласс документа. В конфигураторе подчинен компоненту TcmDocList.

Свойства и методы	Описание
<b>property</b> DocCode: integer	Уникальный код документа.
<b>property</b> DocName: string	Уникальный идентификатор документа.
<b>property</b> DocForm: string	Имя формы просмотра документа.
<b>property</b> DocEditForm: string	Имя формы редактирования строки спецификации документа.
<b>property</b> DocHistoryForm: string	Имя формы просмотра истории редактирования документа.
<b>function</b> GetHead: TcmDocHead	Возвращает список реквизитов заголовка документа.
<b>function</b> GetSpec: TcmDocSpec	Возвращает список реквизитов спецификации документа.

<b>property</b> HeadRecvzit[Index: integer]: TcmRecvzit	Возвращает ссылку на реквизит заголовка документа по его порядковому номеру. Параметр Index должен находиться в диапазоне от 0 до HeadRecvCount-1.
<b>function</b> HeadRecvCount: integer	Возвращает число реквизитов заголовка документа.
<b>property</b> SpecRecvzit[Index: integer]: TcmRecvzit	Возвращает ссылку на реквизит спецификации документа по его порядковому номеру. Параметр Index должен находиться в диапазоне от 0 до SpecRecvCount-1.
<b>function</b> SpecRecvCount: integer	Возвращает число реквизитов спецификации документа.
<b>function</b> GetHeadRecvByName (AName: string): TcmRecvzit	Возвращает ссылку на реквизит заголовка документа по его имени (свойство TcmRecvzit.RecvName)
<b>function</b> GetSpecRecvByName (AName: string): TcmRecvzit	Возвращает ссылку на реквизит спецификации документа по его коду (свойство TcmRecvzit.RecvName)
<b>property</b> onNewRecord: TNotifyEvent	Событие при создании нового документа. Вызывается в методе TcmDocEdit.New
<b>property</b> onBeforePost: TNotifyEvent	Событие перед записью документа в базу данных. Вызывается в методе TcmDocEdit.Post
<b>property</b> onAfterGet: TNotifyEvent	Событие после чтения документа из базы данных. Вызывается в методе TcmDocEdit.Get
<b>property</b> onBeforeDelete: TNotifyEvent	Событие перед удалением документа. Вызывается в методе TcmDocEdit.Delete
<b>property</b> onRegister: TNotifyEvent	Событие при регистрации документа. Вызывается в методе TcmDocEdit.Register
<b>property</b> onUnRegister: TNotifyEvent	Событие после отмены регистрации документа. Вызывается в методе TcmDocEdit.UnRegister
<b>property</b> onSpecNewRecord: TNotifyEvent	Событие при создании новой строки спецификации документа. Вызывается в методе TcmDocSpecEdit.New
<b>property</b> onSpecLockResource: TNotifyEvent	Событие для резервирования ресурсов сводных таблиц. Вызывается в методе TcmDocSpecEdit.Post
<b>property</b> onSpecBeforePost: TNotifyEvent	Событие перед сохранением строки документа в базе. Вызывается в методе TcmDocSpecEdit.Post
<b>property</b> onSpecBeforeDelete: TNotifyEvent	Событие перед удалением строки спецификации документа. Вызывается в методе TcmDocSpec.Delete
<b>property</b> onGenerateDocNomer: TNotifyEvent	Генерирует код документа. Вызывается в методе TcmDocEdit.Post, если TcmDocEdit.DocNomer не указан (пустая строка).

**Примечание:** Все события вызываются в соответствующих методах компонента TcmDocEdit (см. раздел Компонент TcmDocEdit). В качестве параметра Sender всем событиям передается ссылка на объект класса TcmDocEdit.

 TcmDocHead = class(TcmRecvzitComponent)

Список реквизитов заголовка документа. В конфигураторе подчинен компоненту TcmDoc.

 TcmDocSpec = class(TcmRecvzitComponent)

Список реквизитов спецификации документа. В конфигураторе подчинен компоненту TcmDoc.

## 2.6

Типы реквизитов таблицы:


TcmTableRecvType = (cmScale, cmResource, cmRecvzit);

- cmScale - измерение
- cmResource - ресурс
- cmRecvzit - реквизит

 TcmTableList = class(TcmUnitComponent)


Список сводных таблиц. В конфигураторе подчинен компоненту TcmMetadate.

Свойства и методы	Описание
<b>property</b> Tables[Index: integer]: TcmTables	Индексированное свойство для доступа к списку таблиц. Параметр Index должен находиться в диапазоне от 0 до Count-1.
<b>function</b> Count: integer	Число таблиц в списке Tables.
<b>function</b> GetTableByName(AName: string): TcmTable;	Возвращает ссылку на метакласс таблицы по её имени (свойство TcmTable.TableName)


 TcmTable = **class**(TcmUnitComponent)

Сводная таблица. В конфигураторе подчинена компоненту TcmTableList.

Свойства и методы	Описание
<b>property</b> TableName: string	Уникальный идентификатор таблицы.
<b>property</b> Reserved: boolean	Если равно true - для таблицы может производиться резервирование ресурсов.
<b>function</b> GetRecvList(RecvType: TcmTableRecvType): TcmRecvzitComponent	Возвращает список реквизитов таблицы. Все реквизиты таблицы делятся на три группы измерения, ресурсы, реквизиты.

 TcmScale = **class**(TcmRecvzitComponent)

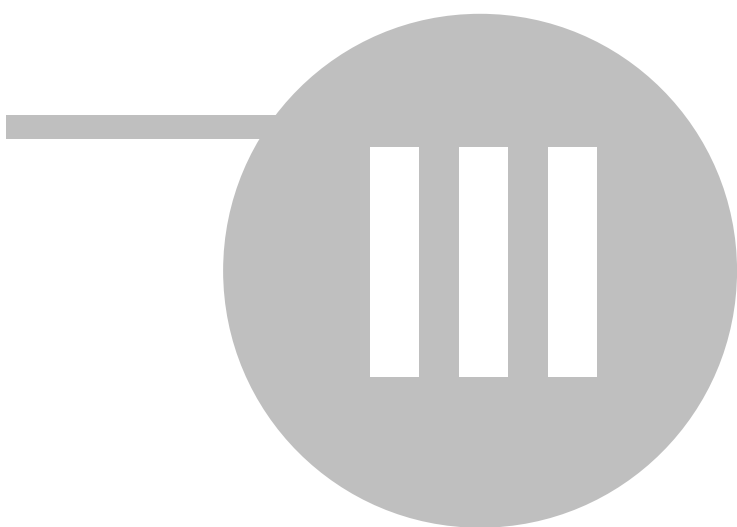
Список измерений таблицы. В конфигураторе подчинен TcmTable.

 TcmResource = **class**(TcmRecvzitComponent)

Список ресурсов таблицы. В конфигураторе подчинен TcmTable.

 TcmTableData = **class**(TcmRecvzitComponent)

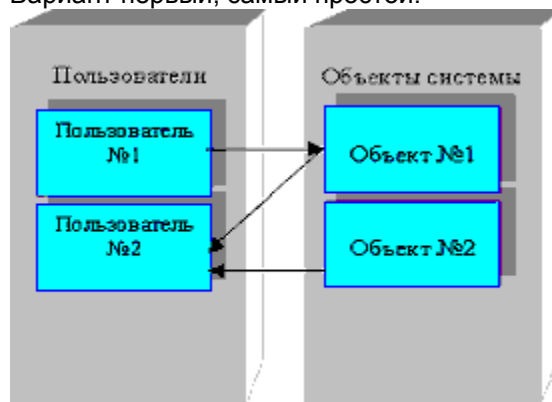
Список реквизитов таблицы. В конфигураторе подчинен TcmTable.



## 3

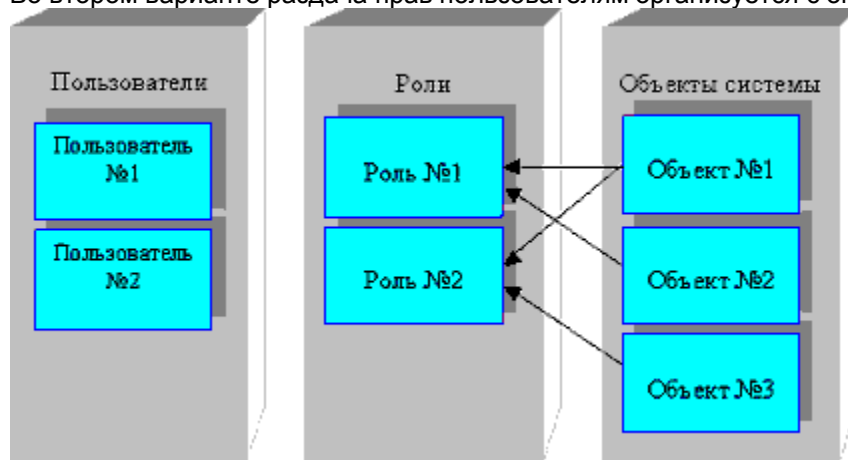
Важнейшим элементом любой системы автоматизации, является блок предоставления прав доступа к различным объектам системы. С системой автоматизации одновременно может работать не один десяток пользователей, выполняя при этом различные операции – редактирование справочников, ввод новых документов, формирование отчетов и т.д. Каждый пользователь выполняет строго определенные для его должности действия, и нет смысла давать ему полный доступ ко всем возможностям системы, скорее наоборот, следует запретить доступ к функциям не связанным с его служебными обязанностями. Для этого, каждый пользователь имеет собственное имя и пароль для регистрации в системе. При запуске системы запрашивается имя пользователя и его пароль, эта процедура называется авторизацией пользователей. Имя пользователя однозначно определяет его права доступа, а пароль позволяет закрыть доступ к системе всем посторонним лицам. Из всего многообразия систем автоматизации, можно выделить три модели используемых при создании блока авторизации пользователей.

Вариант первый, самый простой:



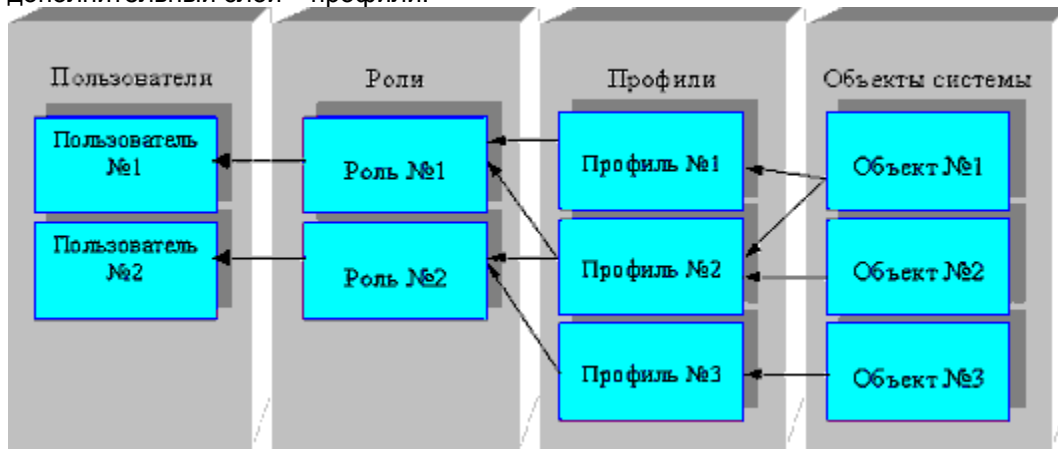
В данном случае, создается список пользователей и для каждого пользователя определяется доступ к различным объектам системы. Недостатки данного подхода очевидны, при создании каждого нового пользователя требуется заново определять для него все права доступа. При этом существуют группы пользователей выполняющих одну и ту же работу, следовательно, имеющих одинаковые права доступа.

Во втором варианте раздача прав пользователям организуется с определением типовых ролей:



Здесь определяются типовые права доступа для каждой должности (обычно такие шаблоны называются ролями пользователей), далее каждому пользователю создаваемому в системе присваивается определенная роль. Плюсы на лицо, если мы изменяем права доступа для какой-нибудь роли, автоматически изменяются права доступа у всех пользователей, которым

присвоена данная роль. При создании нового пользователя, достаточно присвоить ему определенную роль, тем самым однозначно определив права доступа к системе. Существует еще один вариант построения систем авторизации, в котором вводится дополнительный слой – профили:



Здесь права доступа к объектам присваиваются не ролям, а профилям. Права доступа для роли, есть совокупность присвоенных данной роли профилей. В принципе, может показаться, что еще один дополнительный слой профилей избыточен, вполне можно обойтись и без него. Здесь все зависит от сложности системы авторизации. В реальной жизни встречаются системы, где определяются десятки ролей и предоставляется доступ к сотням различных объектов. Поэтому, в сложной системе авторизации, правильно спроектированные и настроенные профили, могут значительно облегчить жизнь системному администратору и снизить временные затраты на настройку и сопровождение системы. В библиотеке Cool Library, для построения системы авторизации, используется последний вариант, включающий профили.

### 3.1

Права доступа определяемые в конфигурации делятся на три группы:

- Права доступа к справочникам
- Права доступа к документам
- Права доступа к конфигурации

При описании прав доступа к справочникам и документам используется тип:

```
TcmGrantType=(grInsert,grEdit,grDelete,grProv,grUnProv);
```

где:

- **grInsert** - Доступ на создание нового документа или записи справочника
- **grEdit** - Доступ для редактирования документа или записи справочника
- **grDelete** - Доступ на удаление документа или пометки на удаление записи справочника
- **grProv** - Доступ на проведение документа
- **grUnProv** - Доступ на отмену проведения документа



`TcmGrant = class(TcmUnitComponent)`

Менеджер системы прав доступа. ВМенеджер содержит описание профилей и ролей пользователей.

Свойства и методы	Описание
<code>function GetRights: TcmRights</code>	Возвращает ссылку на объект прав доступа к конфигурации
<code>function GetProfiles: TcmProfiles</code>	Возвращает ссылку на объект списка профилей
<code>function GetRoles: TcmRoles</code>	Возвращает ссылку на объект списка ролей пользователей



`TcmRights = class(TcmUnitComponent)`


Список прав доступа к конфигурации. В конфигураторе подчинен классу TcmMetadate. Список

имеет древовидную структуру.

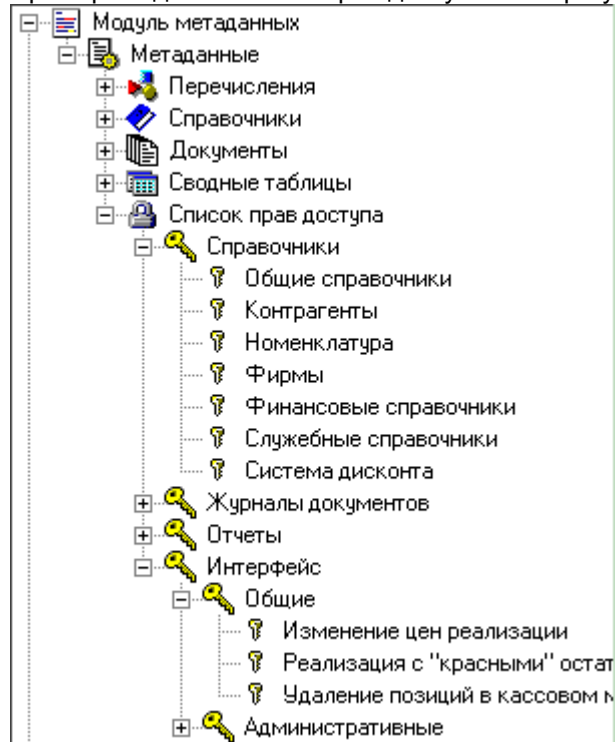
Для создания группы прав используется класс TcmRightGroup:

 TcmRightGroup = **class**(TcmUnitComponent)

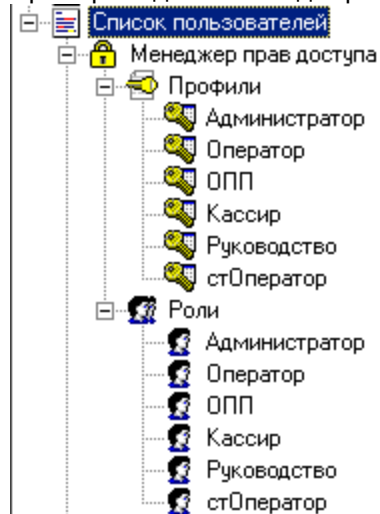
Для создания права доступа используется класс TcmRight:

 TcmRight = **class**(TcmUnitComponent)

Пример создания списка прав доступа в конфигураторе:




Пример создания менеджера прав доступа:



 TcmProfiles = **class**(TcmUnitComponent)

Класс для создания списка профилей. В конфигураторе подчинен классу TcmMetadate.

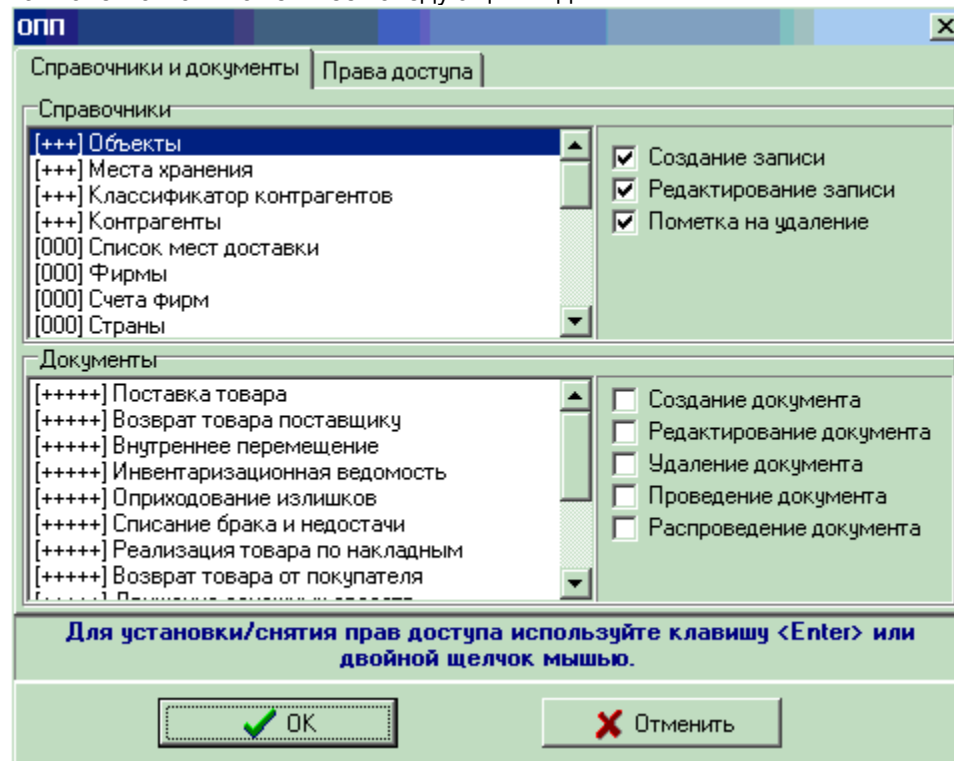
 TcmProfile = **class**(TcmUnitComponent)



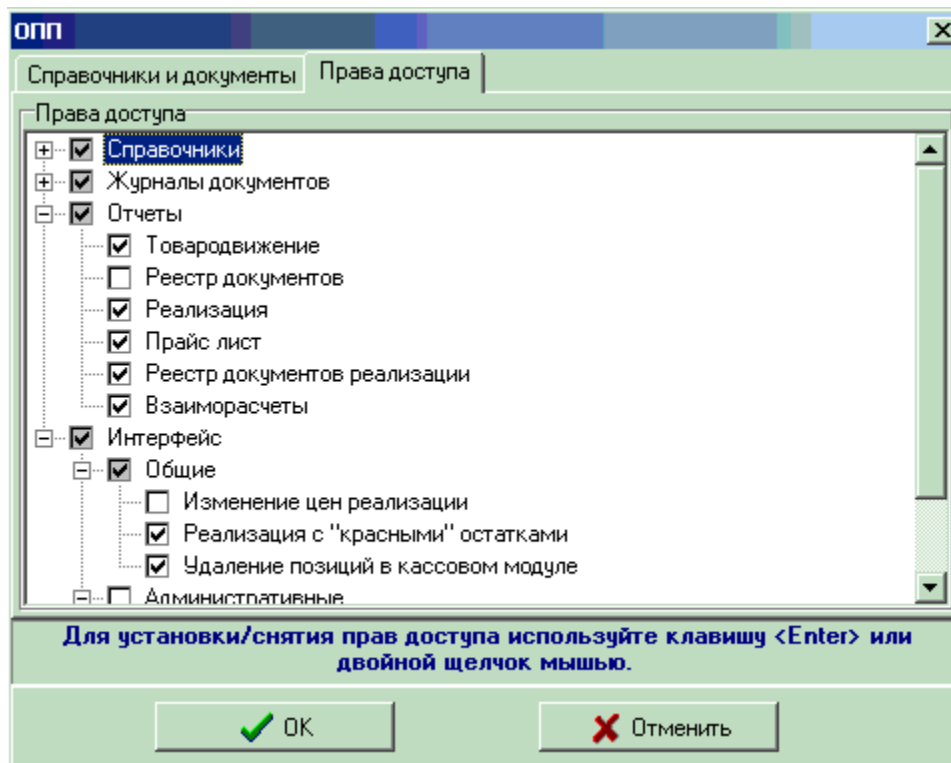
Класс для создания профиля прав доступа. В конфигураторе подчинен классу TcmProfiles.

Свойства и методы	Описание
property Spr	Список прав доступа к справочникам.
property Doc	Список прав доступа к документам.
property Rights	Список прав доступа к конфигурации.


С помощью класса TcmProfile задаются права доступа к объектам конфигурации. Редактор компонента TcmProfile имеет следующий вид:




Установка прав доступа к справочникам и документам.



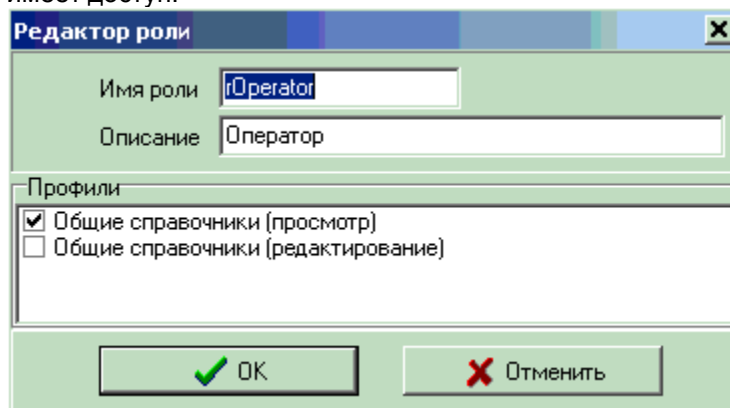
Настройка специальных прав доступа.

 `TcmRoles = class(TcmUnitComponent)`

Список ролей. В конфигураторе подчинен классу `TcmGrant`.

 `TcmRole = class(TcmUnitComponent)`

Роль пользователя. В конфигураторе подчинена классу `TcmRoles`. Каждому пользователю системы присваивается роль, права роли определяются списком профилей, к которым роль имеет доступ:



На рисунке изображен редактор роли. Для каждой роли задается:

- Имя роли - свойство `Name` класса `TcmUnitComponent`
- Описание - свойство `Description` класс `TcmUnitComponent`
- Профили - Указываются профили которые составляют набор прав доступа для текущей роли.

## 3.2


 TcmUsers= **class**(TcmUnitComponent)

Список пользователей. В конфигураторе подчинен классу TcmUnitComponent.

Свойства и методы	Описание
<b>property</b> Grant: TcmGrant	Ссылка на систему прав доступа. Система прав доступа содержит список ролей присваиваемых пользователям.

 TcmUserGroup= **class**(TcmUnitComponent)

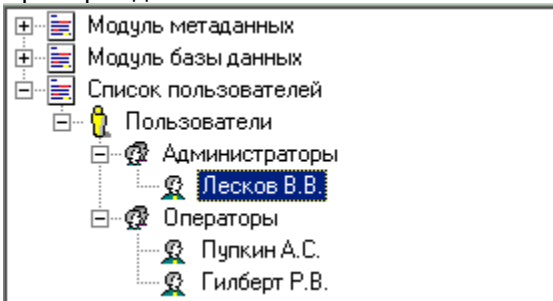
Группа пользователей. В конфигураторе подчинен классу TcmUsers.

 TcmUser= **class**(TcmUnitComponent)

Пользователь системы. В конфигураторе подчинен классу TcmUserGroup.

Свойства и методы	Описание
<b>property</b> Role: TcmRole	Роль пользователя.
<b>property</b> Password: <b>string</b>	Пароль пользователя для входа в систему.

Пример задания списка пользователей:



Редактор класса TcmUser имеет следующий вид:

- Имя пользователя - свойство TcmUnitComponent.Description
- Роль - роль пользователя
- Пароль - пароль пользователя
- Повторить ввод пароля - повторный ввод пароля для проверки корректности введенного значения.

## 3.3

Для авторизации пользователя при запуске конфигурации следует воспользоваться методом класса TcmBases.SelectUser.

Пример главного скрипта конфигурации:

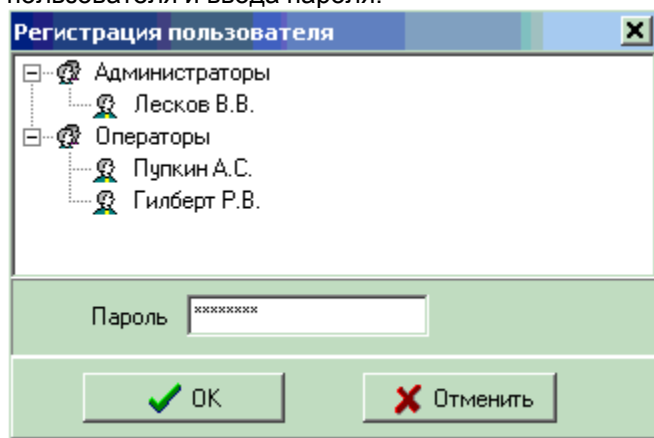
```
-----
var DataBase: TcmDataBase;
begin
//создаем главную форму конфигурации
```

```

Application.CreateCoolForm('MainForm',true);
//получаем ссылку на метаобъект базы данных
DataBase:=TcmBases(CoolConfig.GetComponentByName('DataBase'));
//если метаобъект базы не найден - прерываем выполнение конфигурации
if DataBase=nil then
begin
ShowMessage('Не найден метаобъект базы данных.');//сообщение
CoolConfig.MainForm.Close; //закрываем главную форму
Exit; //прерываем выполнение скрипта
end;
//авторизация пользователя
if DataBase.SelectUser then
begin
CoolConfig.MainForm.Show; //показываем главную форму
end else CoolConfig.MainForm.Close; //закрываем главную форму
end.

```

При вызове метода DataBase.SelectUser на экран будет выведена форма для выбора пользователя и ввода пароля:



После успешной авторизации, свойство объекта DataBase.User содержит ссылку на активного пользователя. Для получения информации о правах пользователя можно воспользоваться следующими методами класса TcmBases:

<b>function</b> SprGrant(ASprName: <b>string</b> ; AGrantType: TcmGrantType): boolean	Возвращает true если для справочника ASprName установлен режим доступа AGrantType. Возможны следующие значения AGrantType: <ul style="list-style-type: none"> <li>• grInsert - создание новой записи справочника</li> <li>• grEdit - редактирование записи справочника</li> <li>• grDelete - пометка на удаление записи справочника.</li> </ul>
<b>function</b> DocGrant(ADocName: <b>string</b> ; AGrantType: TcmGrantType): boolean	Возвращает true если для документа ADocName установлен режим доступа AGrantType. Возможны следующие значения AGrantType: <ul style="list-style-type: none"> <li>• grInsert - создание нового документа</li> <li>• grEdit - редактирование документа</li> <li>• grDelete - удаление документа</li> <li>• grProv - проведение документа</li> <li>• grUnProv - Отмена проведения документа</li> </ul>
<b>function</b> RightGrant(ARightName: <b>string</b> ): boolean	Возвращает true если для активного пользователя установлен доступ к правилу с именем ARightName.

Если авторизация пользователя не производилась (не вызывался метод SelectUser), все вышеперечисленные методы будут возвращать true.

Пример настройки доступа к пунктам главного меню приложения с использованием метода RightGrant:

```
procedure SetMenuGrant;  
begin  
PlaceListItem.Enabled:=DataBase.RightGrant('rgPlaceList');  
KontragentListItem.Enabled:=DataBase.RightGrant('rgKontragentList');  
end;
```

-----  
где:

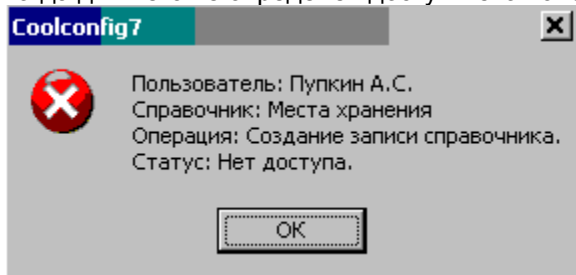
- PlaceListItem и KontragentListItem - пункты главного меню (объекты класса TMenuItem).
- DataBase - Объект класса TcmBases
- 'rgPlaceList' и 'rgKontragentList' - права доступа описанные в конфигурации.

При работе с классами

- TcmSprEdit - редактирование записи справочника
- TcmDocEdit - редактирование документа

Права доступа активного пользователя проверяются автоматически и если текущий пользователь пытается выполнить запрещенную для его роли операцию, генерируется исключение.

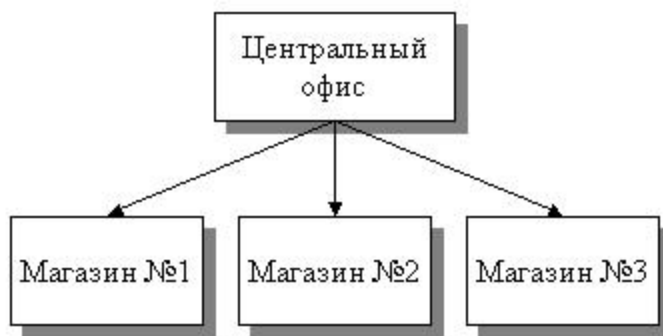
Пример сообщения при попытке пользователя вставить запись в справочник мест хранения, когда для него не определен доступ к этой операции:





## 4

В качестве сервера баз данных, библиотека CoolLib использует InterBase версии 6.0. или выше, могут также использоваться клоны данной БД FireBird или Yaffil. Концепция "базы данных" в системе CoolManager существенно отличается от классического толкования этого термина, используемого при создании реляционных БД. В случае InterBase, под базой данных подразумевается файл с расширением (.gdb), расположенный на одном из дисков компьютера с установленным сервером InterBase. В библиотеке CoolLib под базой данных подразумевается набор различных баз данных, содержащих заранее оговоренные структуры метаданных (таблиц, триггеров, сохраненных процедур и т.д.). Файлы баз данных (gdb), являются составными компонентами СУБД CoolManager и отображают логическую структуру объектов и подразделений предприятия, использующего CoolManager в качестве инструмента автоматизации учета. Объясним вышесказанное на примере. На рисунке №1 изображена типичная структура торговой сети розничных магазинов:



**Рис. №1**

Сеть состоит из Центрального офиса и ряда розничных магазинов. В магазинах происходит хранение и реализация товара, в Центральном офисе находится бухгалтерия, отделы закупа, маркетинга и прочие службы торговой сети. С физической точки зрения, сеть состоит из набора территориально удаленных объектов, каждый из которых имеет собственную локальную сеть и базу данных. Но, для детального анализа деятельности торговой сети, базы данных магазинов должны быть доступны сотрудникам Центрального офиса, кроме того, информация содержащаяся в базе Центрального офиса (либо часть этой информации) должна быть доступна в магазинах сети. Например, для оперативного доступа к данным о взаиморасчетах с поставщиками товара, о наличии товаров на Центральном складе сети и т.д. Система автоматизации должна содержать встроенные средства для репликации данных между территориально удаленными объектами торговой сети. Т.е. если существует база данных магазина №1, то в Центральном офисе должно быть расположено "зеркало" этой базы, под зеркалом подразумевается копия базы магазина №1, доступная для чтения и постоянно поддерживаемая в актуальном состоянии посредством встроенных механизмов межбазового обмена данными. Конкретная реализация межбазового обмена может быть организована любым удобным для предприятия способом (передача информации на электронных носителях, автоматическая отправка писем по электронной почте, передача пакетов данных по выделенному каналу и т.д.).

Как уже говорилось, СУБД CoolManager состоит из набора логически взаимосвязанных баз под управлением сервера InterBase. Структура базы данных одного из объектов сети представлена на рис. №2:



**Рис. №2**

База данных объекта состоит из двух типов файлов базы данных:

- База справочников
- База документов

Все эти базы имеют стандартную InterBase архитектуру (файлы с расширением .gdb), но содержат различные наборы объектов метаданных. Рассмотрим все три типа баз данных.

#### **База справочников.**

Любая система автоматизации содержит блок справочной информации (справочники мест хранения, номенклатуру товаров, ценовые схемы, списки поставщиков и покупателей и многое другое). Не вдаваясь в технические подробности реализации таблиц справочников, следует отметить, что все базы данных должны использовать единую систему справочной информации. Поэтому, вся справочная информация, используемая в системе, вынесена в отдельную базу. Каждый объект предприятия содержит только ОДНУ базу справочников, изменения вносимые в базу любого объекта, автоматически передаются в базы справочников других объектов. За счет этого достигается создание единого информационного справочного пространства всех объектов работающих в рамках системы учета CoolManager. Помимо справочников, в базе хранятся оперативные остатки сводных таблиц и сведения о зарезервированных ими ресурсах.

#### **База документов.**

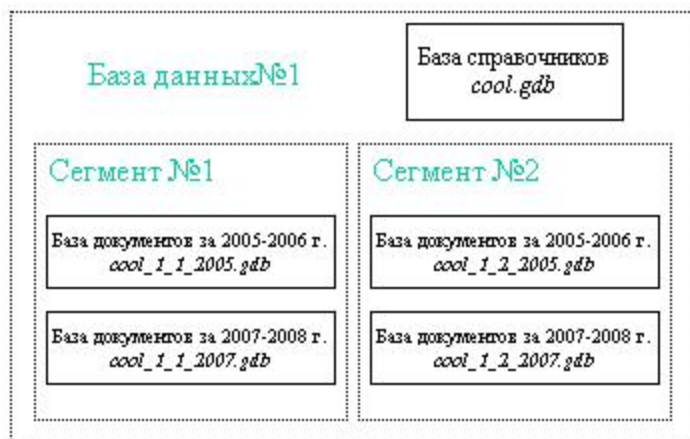
Одной из основных проблем любой системы автоматизации является проблема физического роста размеров используемой базы данных. Поэтому, со временем, приходится выполнять различные технологические процедуры по "обрезке" данных находящихся в базе с целью оптимизации её размера и быстродействия. Библиотека CoolLib предлагает другой путь решения проблемы роста физического размера базы. Для каждого отчетного периода создается своя база данных, под отчетным периодом понимается календарный год (или несколько лет). Т.е. все документы созданные в системе, хранятся в базе соответствующего периода. Базы документов могут находиться на различных физических дисках компьютера и даже на различных компьютерах в рамках одной локальной сети.

Кроме того, база данных может быть разбита на сегменты, каждый сегмент реализуется в виде отдельного gdb файла базы. Т.е. если в базе определено три сегмента, то для каждого отчетного периода будет создано три различных базы (сегмента). Каждый сегмент хранит определенные типы документов. Например, сегмент №1 может хранить документы реализации товара через кассовые терминалы, сегмент №2 бухгалтерские документы о взаиморасчетах с поставщиками и покупателями, сегмент №3 складские документы о движении товаров. Каждому виду сопоставлен сегмент, в котором будут храниться документы указанного вида.

Приведем пример. Предположим существует база данных объекта состоящая из двух сегментов. В базе открыто два отчетных периода за 2005-2006 и 2007-2008 года. База данных и



её сегменты имеют уникальные целочисленные кода. В нашем случае база будет иметь код - 1, сегменты соответственно кода 1 и 2. В этом случае будут созданы следующие файлы баз данных:



**Рис. №3**

База в нашем примере состоит из пяти файлов:

- База справочников (cool.gdb). Наименование базы и её местоположение задается в конфигураторе ИСР.
- Базы документов (cool\_1\_1\_2005.gdb, cool\_1\_1\_2007.gdb, cool\_1\_2\_2005.gdb, cool\_1\_2\_2007.gdb). В конфигураторе задается только местоположение каждой из баз, имена баз генерируются автоматически и имеют формат: <Name>\_X1\_X2\_X3.gdb, где <Name> - имя главной базы данных (в нашем случае: cool), X1 - код базы данных, X2 - код сегмента, X3 - календарный год начала периода.

Итак, база данных объекта состоит из ОДНОЙ базы справочников и набора баз документов. Каждая база документов хранит документы за определенный период.

Кроме главной СУБД объекта, база может включать любое количество зеркальных копий СУБД других объектов:

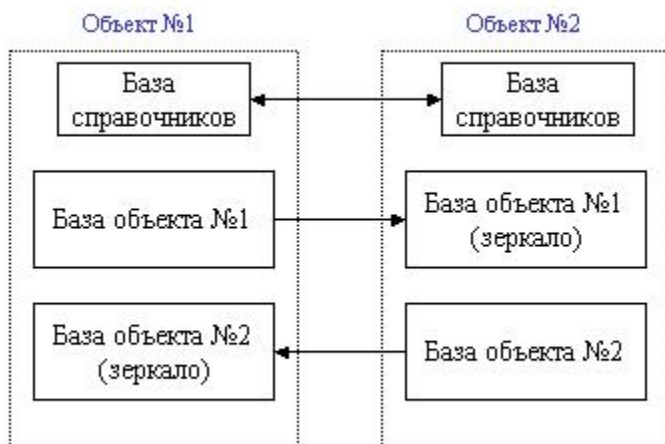


**Рис. №4**

Зеркальные копии содержат СУБД других объектов и периодически обновляются посредством встроенных в библиотеку механизмов межбазового обмена данными. Зеркала используются в качестве источника данных для различных отчетов формируемых в системе и аналитических

выборок.

Схематически процесс обмена данными между двумя удаленными объектами сети можно представить следующим рисунком:



**Рис. №5**

Предполагается, что Объект №1 и Объект №2 содержат полный набор данных, т.е. информация доступная в локальной сети объекта №1, так же доступна и в локальной сети объекта №2. Каждый объект содержит базу справочников, данные которых синхронизируются посредством системы межбазового обмена. Кроме того, объект содержит собственную базу данных, в которую вводятся все документы создаваемые на данном объекте, а так же зеркало базы другого объекта, в которую поступают все изменения документов произведенные на этом объекте.

Следовательно, можно выделить два типа межбазового обмена данными:

- Синхронизация баз справочников
- Репликация баз документов, т.е. передача сделанных изменений в удаленно расположенное зеркало базы. Зеркальные копии используются в режиме "только для чтения", т.е. запрещено редактирование документов содержащихся в этих базах.

В заключении раздела следует сказать несколько слов о разбиении базы документов на несколько сегментов. При первом знакомстве со структурой СУБД библиотеки Cool Manager может показаться, что подобное дробление данных является излишним и достаточно создавать только одну базу для каждого периода. Приведем несколько примеров, когда разбиение базы документов на сегменты позволяет оптимизировать процесс работы в целом.

#### **Пример №1.**

База данных розничного магазина. Документы создаваемые в системе можно разбить на две группы:

- реализация товара через кассовые терминалы
- прочие документы для ведения складского учета (приход товара, возврат, внутренние перемещения и т.д.).

Для каждой группы документов можно создать отдельный сегмент, базы каждого сегмента могут быть расположены на различных компьютерах локальной сети, что позволит достигнуть максимальной производительности при работе кассовых модулей (обслуживаются отдельным сервером) и при этом сохранить целостность логической структуры базы в целом.

#### **Пример №2.**

База Центрального офиса розничной торговой сети. База содержит зеркала всех баз удаленных магазинов. На основании данных о поставках товара по каждому магазину формируются документы оплаты товара поставщикам, кроме того, в базе Центрального офиса ведется

финансовый учет. Необходимо, что бы в магазинах товароведы имели возможность просматривать текущее состояние взаиморасчетов с поставщиками товаров. В этом случае передавать ВСЮ базу Центрального офиса в каждый магазин не представляется целесообразным. Проще разбить базу Центрального офиса на два сегмента, в одном хранить документы о взаиморасчетах с поставщиками, в другом всю прочую документацию. В этом случае достаточно передавать в магазины сегмент с документами о взаиморасчетах.

#### **Пример №3.**

Оптовая торговая фирма. Содержит в своем составе Центральный офис, удаленные склады, на которых хранится отгружаемый товар. Менеджеры отдела прямых продаж в течении дня обходят предприятия розничной торговли, составляют заявки на поставку товара. Вечером, в Центральном офисе, менеджеры вводят в систему заявки, которые затем поступают на удаленный склад, где на основании данных заявок формируются накладные отпуска товара. В этом случае целесообразно разбить базу офиса на два сегмента - в одном хранить заявки, в другом всю прочую документацию. На удаленные склады передавать только сегмент с заявками.

## 4.1

### **Генераторы**

Генератор	Описание
Spr_gen	Генератор кодов записей справочника.

### **Таблица cmSpr**

Поле	Тип	Описание
Code	varchar(32) primary key	Код записи справочника
State	char(1)	Текущий статус записи. Возможны следующие значения: '+' - нормальное состояние. '-' - запись удалена.
SprCode	smallint	Код справочника
RecordOwner	varchar(32)	Код записи в справочнике владельце
BaseCode	smallint	Код базы данных, в которой было сделано последнее изменение записи.
LastTime	TimeStamp	Дата и время последнего изменения записи
UserName	varchar(32)	Имя пользователя, сделавшего последнее изменение.
RecordName	varchar(256)	Наименование записи справочника
RecordParent	varchar(32)	Код записи верхнего уровня, для справочников имеющих древовидную структуру.
OldCode	varchar(32)	Код записи в другой системе. Используется при переносе данных.
Recvizits	varchar(16384)	Список реквизитов справочника.

### **Таблица cmSprLog**

Поле	Тип	Описание
Code	varchar(32) primary key	Код записи справочника
State	smallint	Текущий статус записи. Возможны следующие значения: 0 - нормальное состояние. 1 - запись удалена.
LastTime	varchar(32) primary key	Дата и время изменения записи
SprCode	smallint	Код справочника
RecordOwner	varchar(32)	Код записи в справочнике владельце

BaseCode	smallint	Код базы данных, в которой было сделано последнее изменение записи.
UserName	varchar(32)	Имя пользователя, сделавшего последнее изменение.
RecordName	varchar(256)	Наименование записи справочника
RecordParent	varchar(32)	Код записи верхнего уровня, для справочников имеющих древовидную структуру.
OldCode	varchar(32)	Код записи в другой системе. Используется при переносе данных из другой системы автоматизации.
Recvizits	varchar(16384)	Список реквизитов справочника.

#### Таблица cmVar

Поле	Тип	Описание
Name	varchar(32) primary key	Наименование переменной
Data	varchar(1024)	Значение переменной
LastTime	TimeStamp	Дата и время последнего изменения.

## 4.2

#### Таблица cmVar

Поле	Тип	Описание
Name	varchar(32) primary key	Наименование переменной
Data	varchar(1024)	Значение переменной
LastTime	TimeStamp	Дата и время последнего изменения.

#### Таблица cmDoc

Поле	Тип	Описание
Code	varchar(32) primary key	Код документа.
State	char	Текущий статус документа. Возможны следующие значения: ' ' - не проведен. '+' - проведен. '-' - удален.
DocType	smallint	Тип документа
Owner	varchar(32)	Документ, на основании которого был создан текущий.
LastTime	TimeStamp	Дата и время последнего изменения документа.
UserName	varchar(32)	Имя пользователя, сделавшего последнее изменение.
DocNomer	varchar(32)	Номер документа.
DocDate	date	Дата выписки документа.
Recvizits	varchar(16384)	Список реквизитов шапки документа.

#### Таблица cmDocLog

Поле	Тип	Описание
Code	varchar(32) primary key	Код документа.
LastTime	TimeStamp primary key	Дата и время последнего изменения записи.
State	char	Текущий статус документа. Возможны следующие значения: ' ' - не проведен. '+' - проведен. '-' - удален.

DocType	smallint	Тип документа
Owner	varchar(32)	Документ, на основании которого был создан текущий.
UserName	varchar(32)	Имя пользователя, сделавшего последнее изменение.
DocNomer	varchar(32)	Номер документа.
DocDate	date	Дата выписки документа.
Recvizits	varchar(16384)	Список реквизитов шапки документа.

### Таблица smDocSpec

Поле	Тип	Описание
Code	Int64 primary key	Код строки документа.
DocumCode	varchar(32)	Код документа.
LastTime	TimeStamp	Дата и время последнего изменения записи.
Recvizits	varchar(16384)	Список реквизитов шапки документа.

Примечания:

- Имя базы документов генерируется в формате: cool\_<BaseCode>\_<SegCode>\_<Year>. Где: BaseCode - код базы данных, SegCode - код сегмента базы, Year - календарный год.
- Формирование PrimaryKey документа: DocCode = Year-BaseCode-SegCode-DocumCode
- Помимо вышеописанных таблиц, база документов содержит сводные таблицы, формируемые на основании описания метаданных конфигурации.

## 4.3

Сводные таблицы используются для накопления информации. Структура сводных таблиц описывается в конфигураторе. При создании новой базы данных, на основании описаний сделанных в конфигураторе, создаются физические таблицы в базе. Информация о результатах хозяйственных операций, которая появляется при оформлении документов, записывается в сводные таблицы при проведении документов, затем эта информация используется при формировании отчетов. При создании сводной таблицы определяется, как будет обрабатываться, группироваться и сохраняться сводная информация. Сводные таблицы могут быть двух видов: оборотные таблицы и таблицы остатков.

**Таблица остатков** - это объект, предназначенный для ведения остатков "ресурсов" на момент времени с привязкой к некоторому набору "измерений".

**Таблица оборотов** - это объект, предназначенный для регистрации некоторого набора данных, без расчета остатков и оборотов "ресурсов".

Иначе говоря, если при описании сводной таблицы в конфигураторе указан некоторый набор "ресурсов" и "измерений" - это таблица остатков. Если при создании таблицы не указываются "измерения" или "ресурсы" - это таблица оборотов.

Ресурсами таблицы могут являться любые категории учета, которые могут быть представлены в числовом виде. Измерения таблицы - это оси координат, на пересечении которых таблица хранит конкретные значения ресурсов. Кроме того для таблицы можно задавать "реквизиты". Значения реквизитов просто сопровождают запись в сводной таблице и хранят дополнительную информацию.

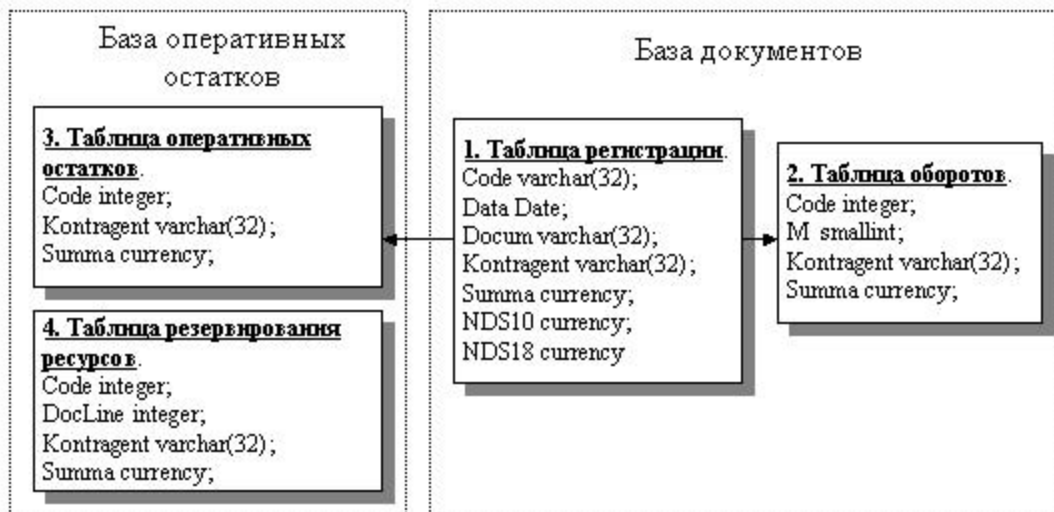
Поясним принцип работы сводной таблицы на примере. Предположим, требуется вести учет взаиморасчетов с покупателями товара. В этом случае можно создать следующую сводную таблицу:

Поле таблицы	Тип поля	Описание
Kontragent	измерение	Код контрагента (покупателя товара). Обычно, поле имеет тип справочника и указывает на справочник контрагентов.

Summa	ресурс	Сумма взаиморасчетов. Информация об отгрузке товара и об оплатах за отгруженный товар. Суммы отгруженного товара указываются с минусом, суммы оплат с плюсом. Следовательно, при итоговом сальдо меньше нуля, покупатель имеет задолженность перед предприятием, при положительном сальдо, предприятие имеет долг перед покупателем.
NDS10	реквизит	Реквизиты содержат суммы НДС в т.ч. по ставке 10 и 18%.
NDS18		

Предположим, сводная таблица будет называться Raschet.

В базе данных, для работы со сводной таблицей, будут созданы следующие объекты:



#### **Таблица регистрации (Raschet).**

Включает информацию о движениях денежных средств, таблица содержит следующие служебные поля:

- Code - уникальный код записи таблицы, генерируется автоматически.
- Data - дата совершения операции.
- Docum - код документа, при проведении которого была сформирована операция.

#### **Таблица оборотов (Raschet\_m).**

Обороты по месяцам расчетного периода, используется для оптимизации расчета итогов на указанную дату. Таблица содержит следующие служебные поля:

- Code - уникальный код записи таблицы, генерируется автоматически.
- M - номер календарного месяца (1..12).

#### **Таблица резервирования ресурсов (Raschet\_res).**

Содержит данные о зарезервированных ресурсах сводной таблицы. Создается, если свойство Reserved сводной таблицы равно true.

- Code - уникальный код записи таблицы, генерируется автоматически.
- DocLine - код строки спецификации документа, по которой был установлен резерв.

### **Пример записи данных в сводную таблицу.**

#### **Отгрузка товара.**

Предположим, было выписано две накладных отпуска товара покупателям:

Код документа	Дата выписки	Код контрагента	Сумма	НДС в т.ч.
2006-1-1-1-1	05.06.2006	01-001	23 000.00	2 090.00
2006-1-1-1-2	05.06.2006	01-002	15 600.00	2 378.00

В этом случае, в таблицах базы данных будут сделаны следующие записи:

**Raschet**

Code	Data	Docum	Kontragent	Summa	NDS10	NDS18
1	05.06.2006	2006-1-1-1-1	01-001	-23 000.00	2 090.00	0
2	05.06.2006	2006-1-1-1-2	01-002	-15 600.00	0	2 378.00

**Raschet\_m**

Code	M	Kontragent	Summa
1	6	01-001	-23 000.00
2	6	01-002	-15 600.00

**Raschet\_ost**

Code	Kontragent	Summa
1	01-001	-23 000.00
2	01-002	-15 600.00

**Оплата товара.**

01.07.2006 была сформирована банковская выписка со следующими данными об оплате:

Код документа	Код строки	Дата выписки	Код контрагента	Сумма	НДС в т.ч.
2006-1-1-2-3	1	01.07.2006	01-001	20 000.00	1 818.00
2006-1-1-2-3	2	01.07.2006	01-002	16 000.00	2 440.68

При проведении документа, в таблицы базы, будут внесены следующие изменения:

**Raschet**

Code	Data	Docum	Kontragent	Summa	NDS10	NDS18
3	01.07.2006	2006-1-1-2-3	01-001	20 000.00	1 818.00	0
4	01.07.2006	2006-1-1-2-3	01-002	-16 000.00	0	2 440.68

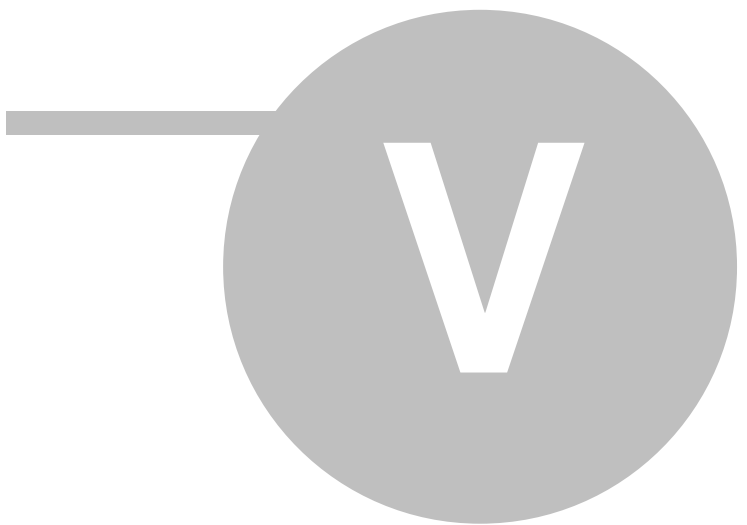
**Raschet\_m**

Code	Month	Kontragent	Summa
3	7	01-001	20 000.00
4	7	01-002	16 000.00

**Raschet\_ost**

Code	Kontragent	Summa
1	01-001	-3 000.00
2	01-002	400.00

Таким образом, использование сводных таблиц позволяет получить детальную информацию о движении объектов учета, оперативных остатках ресурсов, а так же рассчитать текущие значения ресурсов на любую дату отчетного периода.





## 5

## 5.1 TcmScriptBase

TcmScriptBase = **class**(TcmUnitComponent)

Базовый класс от которого наследуются все компоненты владельцы баз данных.

Свойства и методы	Описание
<b>property</b> Host: <b>string</b>	Возвращает имя компьютера на котором расположена база данных. Примеры: server01 //с использованием сервиса DNS 192.168.0.1 //указание IP адреса компьютера в сети
<b>property</b> Path: <b>string</b>	Путь к каталогу в котором находится база данных. Пример: c:\base\database\
<b>property</b> Password: <b>string</b>	Пароль пользователя SYSDBA установленный на сервере InterBase.

## 5.2 TcmBases

 TcmBases = **class**(TcmScriptBase)

Главный компонент базы данных. В конфигураторе подчинен компоненту TcmUnitComponent.

Свойства и методы	Описание
<b>property</b> Users: TcmUsers	Ссылка на список пользователей конфигурации.
<b>property</b> User: TcmUser	Активный пользователь. Устанавливается при помощи функции SelectUser.
<b>property</b> DirectBaseCode: smallint	Код главной базы данных. СУБД содержит одну главную или рабочую базу данных и любое количество зеркальных копий баз других объектов.
<b>property</b> BaseName: <b>string</b>	Имя файла базы данных без расширения. По умолчанию: <b>Cool</b>
<b>property</b> FullConnect: boolean	Определяет режим доступа к базе данных для записи. Если равен false возможно внесение изменений в файлы ТОЛЬКО главной БД, иначе возможна запись данных в копии зеркальных баз. Файл базы справочников доступен для записи в любом режиме.
<b>property</b> Metadate: TcmMetadate	Ссылка на метаданные системы учета.
<b>procedure</b> SetVariable(AName, AValue: <b>string</b> )	Устанавливает значение глобальной переменной AName равной AValue.
<b>function</b> GetVariable(AName: <b>string</b> ): <b>string</b>	Возвращает значение глобальной переменной AName.
<b>function</b> Gen_id(AName: <b>string</b> ): integer	Генерирует следующее значение глобальной переменной с именем AName. Если Value - значение глобальной переменной, то функция сперва возвращает Value+1, затем записывает полученное значение в глобальную переменную.
<b>function</b> SelectUser: boolean	При вызове функции на экран выводится диалоговое окно для выбора пользователя и ввода пароля. Если авторизация прошла успешно - возвращает true.
<b>Трассировка запросов к базе данных.</b>	
<b>property</b> TraceMode: boolean	Если равно true, режим трассировки активен.
<b>property</b> LogFileName: <b>string</b>	Имя лог-файла в котором сохраняются сообщения сгенерированные в режиме трассировки.
<b>procedure</b> AddLogFile(Msg: <b>string</b> )	Запись строки в лог-файл.

Описанные ниже функции используются для определения прав доступа активного пользователя. Активный пользователь устанавливается посредством вызова функции <b>SelectUser</b> . Если активный пользователь не был выбран, все функции возвращают true.	
<b>function</b> SprGrant(ASprName: <b>string</b> ; AGrantType: TcmGrantType): boolean	Возвращает true если для справочника ASprName установлен режим доступа AGrantType. Возможны следующие значения AGrantType: <ul style="list-style-type: none"> <li>• grInsert - создание новой записи справочника</li> <li>• grEdit - редактирование записи справочника</li> <li>• grDelete - пометка на удаление записи справочника.</li> </ul>
<b>function</b> DocGrant(ADocName: <b>string</b> ; AGrantType: TcmGrantType): boolean	Возвращает true если для документа ADocName установлен режим доступа AGrantType. Возможны следующие значения AGrantType: <ul style="list-style-type: none"> <li>• grInsert - создание нового документа</li> <li>• grEdit - редактирование документа</li> <li>• grDelete - удаление документа</li> <li>• grProv - проведение документа</li> <li>• grUnProv - Отмена проведения документа</li> </ul>
<b>function</b> RightGrant(ARightName: <b>string</b> ): boolean	Возвращает true если для активного пользователя установлен доступ к правилу с именем ARightName.

### 5.3 TcmBase



TcmBase = **class**(TcmUnitComponent)

Компонент базы данных объекта. В конфигураторе подчинен компоненту TcmBases.

Свойства и методы	Описание
<b>property</b> BaseCode: smallint	Уникальный код базы данных.
<b>property</b> BaseDerec: boolean	Признак главной базы данных.
<b>property</b> StoredBase: integer	Код базы из которой следует фактически получать данные. Если не равен 0, то все запросы переадресуются к базе с указанным номером.
<b>procedure</b> SetVariable(AName, AValue: <b>string</b> )	Устанавливает значение глобальной переменной AName равной AValue.
<b>function</b> GetVariable(AName: <b>string</b> ): <b>string</b>	Возвращает значение глобальной переменной AName.
<b>function</b> Gen_id(AName: <b>string</b> ): integer	Генерирует следующее значение глобальной переменной с именем AName. Если Value - значение глобальной переменной, то функция сперва возвращает Value+1, затем записывает полученное значение в глобальную переменную.

### 5.4 TcmBaseSeg



TcmBaseSeg = **class**(TcmScriptBase)

Компонент сегмента базы данных. В конфигураторе подчинен компоненту TcmBase.

Свойства и методы	Описание
<b>property</b> SegCode: smallint	Уникальный код сегмента базы данных.

### 5.5 TcmBaseUnit



TcmBaseUnit = **class**(TcmScriptBase)

Компонент базы документов. В конфигураторе подчинен компоненту TcmBaseSeg.

Свойства и методы	Описание
<b>property</b> Year1: integer	Year1 и Year2 определяют период. Документы, созданные в течении указанного периода, хранятся в текущей базе.
<b>property</b> Year2: integer	

<b>procedure</b> SetVariable(AName,AValue: <b>string</b> )	Устанавливает значение глобальной переменной AName равной AValue.
<b>function</b> GetVariable(AName: <b>string</b> ): <b>string</b>	Возвращает значение глобальной переменной AName.
<b>function</b> Gen_id(AName: <b>string</b> ): integer	Генерирует следующее значение глобальной переменной с именем AName. Если Value - значение глобальной переменной, то функция сперва возвращает Value+1, затем записывает полученное значение в глобальную переменную.



## 6

### 6.1 TcmRecvizitData

Класс **TcmRecvizitData** по своему назначению похож на стандартный компонент Delphi TField, только в отличие от него работает не с полями таблиц, а с реквизитами объектов библиотеки CoolLib. Реквизиты имеют справочники, документы, сводные таблицы, отчеты и т.д., в любом случае, при обращении к значению реквизита используется класс **TcmRecvizitData**.

TcmRecvizitData = class(TObject)

Свойства и методы	Описание
<b>property</b> Name: string {read only}	Возвращает имя реквизита
<b>property</b> Required: boolean {read only}	Если равен true - задание значения реквизита обязательно
<b>property</b> DataSetMode: boolean; {read only}	Если true - устанавливает режим работы со списком значений. При присваивании значения реквизиту, присвоенное значение сохраняется в буфере данных. Свойство AsString возвращает список присвоенных значений. См.пример работы с реквизитом-множеством в конце раздела.
<b>property</b> AsString: string	Преобразует значение поля типа string
<b>property</b> AsInteger: integer	Преобразует значение поля типа integer
<b>property</b> AsInt64: Int64	Преобразует значение поля типа Int64
<b>property</b> AsDouble: double	Преобразует значение поля типа double
<b>property</b> AsCurrency: currency	Преобразует значение поля типа currency
<b>property</b> AsBoolean: boolean	Преобразует значение поля типа boolean
<b>property</b> AsDateTime: TDateTime	Преобразует значение поля типа TDateTime
<b>property</b> AsDate: TDateTime	Преобразует значение поля типа TDateTime учитывает только дату
<b>property</b> AsTime: TDateTime	Преобразует значение поля типа TDateTime учитывает только время

#### Запись и чтение реквизитов перечислений (strEnum).

Перечислению можно присваивать значение как по коду так и по имени.

Предположим существует перечисление TFirmType:

1 - ООО

2 - ЧП

а так же реквизит FirmType типа TFirmType.

Значение реквизита можно устанавливать одним из двух способов:

```
//присваиваем значение реквизита
Recvizits['FirmType'].AsInteger:=2;
ShowMessage(Recvizit['FirmType'].AsString); //возвращает строку 'ЧП'
Recvizits['FirmType'].AsString:='ООО';
ShowMessage(Recvizits['FirmType'].AsInteger); //возвращает 1
```

#### Работа с реквизитом-множеством.

```
procedure AddDataSetList(R: TcmRecvizitData);
var S: string;
begin
//проверяем режим работы с множеством
if not R.DataSetMode then Exit;
//заполняем список значений
R.AsInteger:=1;
R.AsDouble:=2;
```

```
R.AsString:=3;
ShowMessage(R.AsInteger); //вернет последнее присвоенное значение: 3
ShowMessage(R.AsDouble); //вернет последнее присвоенное значение: 3
S:=R.AsString; //вернет список присвоенных значений: '1;2;3'
R.AsString:=""; //очищаем список присвоенных значений
R.AsString:=S; //альтернативный способ заполнения списка значений
end;
```

## 6.2 TcmParamRecvzitData

Класс **TcmParamRecvzitData** наследован от класса **TcmRecvzitData** и предназначен для работы со списком параметров класса **TcmQuery.Params**.

```
TcmParamRecvzitData = class(TcmRecvzitData)
```

Свойства и методы	Описание
<b>property</b> DataSetMode: boolean; {read only}	Если true - устанавливает режим работы со списком значений. При присваивании значения реквизиту, присвоенное значение сохраняется в буфере данных. Свойство <b>AsString</b> возвращает список присвоенных значений.

## 6.3 TcmRecvzitList

Класс **TcmRecvzitList** является списком реквизитов, каждый элемент списка имеет тип **TcmRecvzitData**.

```
TcmRecvzitList = class(TObject)
```

Свойства и методы	Описание
<b>property</b> Recvzit[Index: integer]: TcmRecvzitData	Получить реквизит по его порядковому номеру. Значение параметра <b>Index</b> находится в диапазоне от 0 до <b>Count-1</b>
<b>default property</b> ByName[AName: string]: TcmRecvzitData	Получить реквизит по его имени. Если реквизит с указанным именем не найден, генерируется исключение.
<b>function</b> FindRecvzit(AName: string): TcmRecvzitData	Поиск реквизита по его имени, если реквизит не найден возвращает <b>nil</b> .
<b>function</b> Count: integer	Возвращает число реквизитов в списке.

## 6.4 TcmRecvzitsEdit

Тип определяющий режим редактирования объекта:

```
TcmEditType = (cmsNone, cmsNew, cmsGet)
```

- **cmsNone** - документ не загружен
- **cmsNew** - новый документ
- **cmsGet** - редактирование сохраненного документа

Базовый класс от которого наследуются компоненты редактирования объектов системы учета:

- **TcmSprEdit** - редактирование записей справочников
- **TcmDocEdit** - редактирование документов
- **TcmDocSprEdit** - редактирование спецификации документов

```
TcmRecvzitsEdit = class(TComponent)
```

Свойства и методы	Описание
<b>property</b> RecState: TcmEditType {read only}	Текущий режим редактирования объекта.
<b>property</b> Base: TcmBases	Ссылка на метаобъект базы данных. Метаобъект создается в конфигураторе ИСП <b>CoolManager</b> .
<b>property</b> Modified: boolean {read only}	Если равно true - свойства объекта были изменены

<b>default property</b> Recvizits[Name: string]: TcmRecvizitData;	Индексированное свойство для доступа к значениям реквизитов. Name - имя реквизита. При установке значения реквизита производится проверка наличия у объекта реквизита с указанным именем, а так же проверка корректности заданного значения.
<b>property</b> BufData: TcmBufData	Свойство для оперативного доступа к данным справочников и сводных таблиц.
<b>property</b> OnModified: TNotifyEvent	Событие вызывается при любых изменениях редактируемого объекта. Таких как изменение значения реквизита или выполнения над объектом операций записи, чтения, удаления и т.д.
<b>function</b> IsRecvizit(AName: string): boolean	Если для объекта определен реквизит с именем AName возвращает true.

## 6.5 TcmSprEdit

Статус записи справочника:

TcmSprState = (cmsNormal, cmsDelete);

- cmsNormal - нормальное состояние
- cmsDelete - запись помечена на удаление

Для редактирования записей справочников используется класс **TcmSprEdit**, в палитре компонентов он расположен на закладке CoolMap.

cool



TcmSprEdit = class(TcmRecvizitEdit)

Свойства и методы	Описание
<b>property</b> Spr: TcmSpr	Ссылка на объект метаданных справочника.
<b>property</b> Code: string(32) {read only}	Уникальный код записи справочника. Код справочника может быть задан вручную, либо сгенерирован автоматически. При автогенерации код имеет следующий формат: <BaseCode>-<Code>, где: <BaseCode> - код рабочей базы данных. <Code> - уникальный код записи справочника.
<b>property</b> State: TcmSprState {read only}	Статус записи справочника.
<b>property</b> RecordOwner: string	Владелец записи справочника.
<b>property</b> RecordParent: string	Код записи верхнего уровня, если справочник имеет иерархическую структуру.
<b>property</b> BaseCode: smallint {read only}	Код базы данных в которой произошло последнее изменение записи справочника.
<b>property</b> LastTime: TDateTime {read only}	Дата и время последнего изменения записи справочника.
<b>property</b> UserName: string	ФИО пользователя выполнившего последнее изменение записи справочника.
<b>property</b> RecordName: string	Наименование записи справочника
<b>property</b> OldCode: string	Код записи в другой системе учета. Используется при начальной загрузке данных из других систем автоматизации.
<b>property</b> SprName: string	Идентификатор метакласса справочника с которым работает компонент TcmSprEdit.
<b>property</b> UserGrant: boolean;	Использование системы авторизации пользователей. По умолчанию равно true.
<b>constructor</b> Create(AOwner: TComponent)	Конструктор компонента.
<b>procedure</b> New(AObjName, ACode: string)	Создание новой записи справочника. <ul style="list-style-type: none"> <li>• AObjName - идентификатор метакласса справочника.</li> <li>• ACode - код новой записи справочника.</li> </ul>

<b>function</b> Get(ACode: <b>string</b> ): boolean	Загружает запись с кодом ACode справочника из базы в компонент, возвращает true если запись успешно загружена.
<b>procedure</b> Post	Сохраняет редактируемую запись в базе данных.
<b>procedure</b> Delete	Метит редактируемую запись на удаление.
<b>procedure</b> Restore	Снимает пометку на удаление с редактируемой записи.
<b>function</b> GetGlobalValue(AName: <b>string</b> ): <b>string</b>	Возвращает значение глобальной переменной с именем AName.
<b>procedure</b> SetGlobalValue(AName, AValue: <b>string</b> )	Присваивает глобальной переменной AName значение AValue.
<b>function</b> Gen_id(AName: <b>string</b> ): integer	Генерирует следующее значение глобальной переменной с именем AName. Если Value - значение глобальной переменной с именем AName, то функция сперва возвращает Value+1, затем записывает полученное значение в глобальную переменную.

Пример работы с компонентом **TcmSprEdit**:

```

procedure TestWork;
var Spr: TcmSprEdit;
    SprCode, GlobalVar: string;
begin
  //создаем компонент для редактирования справочника
  Spr:=TcmSprEdit.Create(nil);
  try
    //получаем ссылку на метаобъект базы данных
    Spr.Base:=TcmBases(CoolConfig.GetComponentByName('DataBaseObj'));
    if Spr.Base=nil then RaiseException('Не найдена база данных.');
```

*Spr.New('ObjectList,'); //новая запись*

*Spr.RecordName:='Объект №1';*

*Spr.UserName:='None';*

*Spr.Post; //сохраняем запись в базе данных*

*SprCode:=Spr.Code; //получаем уникальный код записи*

*Spr.SprName:=''; //очищаем объект Spr*

*Spr.Get(SprCode); //загружаем запись из базы*

*Spr.Delete; //метим запись на удаление*

*Spr.Restore; //снимаем пометку на удаление*

*//сохраняем код записи в глобальной переменной DerecoObject*

*Spr.SetGlobalValue('DerecoObject',SprCode);*

**finally**

*Spr.Free; //удаляем компонент*

**end;**

## 6.6 TcmDocEdit

Тип определяющий текущий статус документа:

TcmDocState = (cmdNone, cmdProv, cmdDelete)

- cmdNone - документ не проведен
- cmdProv - документ проведен
- cmdDelete - документ удален

Тип описывающий положение глобальной переменной. Используется в качестве одного из параметров методов GetGlobalValue, SetGlobalValue, Gen\_id:

TcmDocVarType = (cmvGlobal, cmvBase, cmvCurrent)

- cmvGlobal - переменная находится в базе справочников
- cmvBase - переменная находится в базе оперативных остатков
- cmvCurrent - переменная находится в той же базе, что и редактируемый документ.

Для редактирования документов используется компонент **TcmDocEdit**, в палитре компонентов он расположен на закладке CoolMan.





TcmDocEdit = class(TcmRecvzitEdit)

Свойства и методы	Описание
<b>property</b> Docum: TcmDoc	Ссылка на объект метаданных документа
<b>property</b> Code: <b>string(32)</b> { <i>read only</i> }	Уникальный код документа. Код документа генерируется автоматически и имеет следующий формат: <Year>-<BaseCode>-<SegCode>-<DocumCode> где: <ul style="list-style-type: none"> <li>• &lt;Year&gt; - календарный год выписки документа</li> <li>• &lt;BaseCode&gt; - код базы данных</li> <li>• &lt;SegCode&gt; - код сегмента в базе данных</li> <li>• &lt;DocumCode&gt; - уникальный код документа в базе</li> </ul>
<b>property</b> State: TcmDocState { <i>read only</i> }	Текущий статус документа
<b>property</b> DocumOwner: <b>string</b> (32)	Код документа владельца. Указывается в случае, если документ был создан на основании другого документа.
<b>property</b> LastTime: TDateTime { <i>read only</i> }	Дата и время последнего изменения документа
<b>property</b> UserName: <b>string(32)</b>	ФИО пользователя внесшего последние изменения в документ
<b>property</b> DocNomer: <b>string(32)</b>	Номер документа. Содержит сквозной номер документа для печатных форм.
<b>property</b> DocDate: TDateTime	Дата выписки документа.
<b>property</b> DocName: <b>string</b>	Идентификатор метакласса для документа (свойство TcmDoc.DocName).
<b>property</b> UserGrant: boolean;	Использование системы авторизации пользователей. По умолчанию равно true.
<b>property</b> Reserved: boolean	Активация режима резервирования ресурсов сводных таблиц. По умолчанию равно true.
<b>property</b> ShowProgress: boolean	Если равно true - при вызове методов RegisterDoc и UnRegisterDoc на экран выводится форма заставки.
<b>property</b> SetFormTitle: boolean	Если равно true - автоматически обновляется заголовок формы на которой расположен компонент. При создании нового документа в заголовке выводится "[New]", при сохранении строка в формате: " <b>[Код документа]</b> <b>Номер документа от Дата выписки</b> "
<b>property</b> SpecEdit: TcmDocSpecEdit	Ссылка на компонент для редактирования спецификации документа (см. раздел Компомент TcmDocSpecEdit)
<b>constructor</b> Create(AOwner: TComponent)	Конструктор компонента.
<b>function</b> DocCaption: boolean	Возвращает краткое описание документа. Строка содержит код, номер и дату выписки документа, а так же метку, если документ проведен.
<b>procedure</b> ClearRezerv(DocLine: integer)	Снимает резерв ресурсов установленный для документа. Если DocLine=0, снимает резерв по всем строкам документа, иначе только для строки с кодом DocLine.
<b>procedure</b> New(AObjName: <b>string</b> )	Создание нового документа. AObjName - идентификатор метакласса для документа (свойство TcmDoc.DocName).
<b>function</b> Get(ACode: <b>string</b> ): boolean	Загрузка документа из базы данных. ACode - код документа. Если документ успешно загружен возвращает true.
<b>procedure</b> Post	Запись текущего документа в базу данных.
<b>procedure</b> Delete	Удаление текущего документа. Удалить можно только непроведенный документ. Из базы физически удаляется только спецификация документа. Заголовок документа остается в базе, свойству State присваивается значение cmdDelete. Над документом помеченным на удаление запрещены все операции редактирования.
<b>procedure</b> UnDelete	Снимает с документа пометку на удаление, после это документ вновь доступен для редактирования.

<b>function</b> DeleteQuery: boolean	Удаление текущего документа. Перед удалением на экран выводится диалог для ввода подтверждения операции. Если операция успешно выполнена возвращает true.
<b>procedure</b> RegisterDoc	Метод выполняет процедуру регистрации документа. При регистрации происходит запись информации в сводные таблицы базы данных. После регистрации документ становится недоступным для редактирования.
<b>procedure</b> UnRegisterDoc	Метод отменяет регистрацию документа. При отмене регистрации удаляется вся информация из сводных таблиц записанная при регистрации документа и документ становится доступным для редактирования.
<b>function</b> RegisterQuery: boolean	Проведение/распределение документа (в зависимости от текущего состояния). Метод выводит диалоговое окно для ввода подтверждения. Если операция выполнена успешно - возвращает true.
<b>function</b> GetGlobalValue(AVarType: TcmDocVarType; AName: <b>string</b> ): <b>string</b>	Возвращает значение глобальной переменной с именем AName. Параметр AVarType указывает из какой базы следует получить переменную.
<b>procedure</b> SetGlobalValue(AVarType: TcmDocVarType; AName, AValue: <b>string</b> )	Записывает значение AValue в глобальную переменную AName. Параметр AVarType указывает в какую базу следует записать переменную.
<b>function</b> Gen_id(AVarType: TcmDocVarType; AName: <b>string</b> ): integer	Генерирует следующее значение глобальной переменной с именем AName. Если Value - значение глобальной переменной, то функция сперва возвращает Value+1, затем записывает полученное значение в глобальную переменную. Параметр AVarType указывает в какую базу следует записать переменную.
<b>function</b> GetTableEditObject(TableName: <b>string</b> ): TcmTableEdit	Создает объект для вставки записей в сводную таблицу. TableName - имя сводной таблицы. Вызов метода возможен только в процессе проведения документа, т.е. в обработчике события OnRegister, Попытка вызова метода в любом другом месте программного кода приведет к генерации исключительной ситуации.
<b>function</b> GetRezervObject(TableName: <b>string</b> ): TcmRezervEdit	Создает объект для резервирования ресурсов сводных таблиц. TableName - имя сводной таблицы. Вызов метода возможен только в обработчике события TcmDoc.onSpecLockResource. Попытка вызова метода в любом другом месте программного кода приведет к генерации исключительной ситуации. <u>Примечание:</u> Режим резервирования для сводной таблицы определяется значением свойства TcmTable.Reserved: =true.
<b>function</b> GetOstatokObject(TableName: <b>string</b> ): TcmOstatokList	Создает объект для просмотра списка оперативных остатков сводных таблиц. Для таблиц с резервированием ресурсов, возвращает список остатков с учетом установленного резерва.

Пример работы с компонентом **TcmDocEdit**:

```

procedure EditDoc;
var DocEdit: TcmDocEdit;
begin
  DocEdit:=TcmDocEdit.Create(nil);
  try
    //получаем ссылку на метаобъект базы данных
    DocEdit.Base:=TcmBases(CoolConfig.GetComponentByName('DataBaseObj'));
    if DocEdit.Base=nil then RaiseException('Не найдена база данных.');
```

```

    DocEdit.New('PrihodDoc'); //создаем новый документ
```

```

DocEdit.UserName:='Tester'; //имяпользователя
DocEdit.DocNomer:='1'; //номер документа
DocEdit.DocDate:=Date; //датавыписки документа
//устанавливаемреквизитышапки документа
DocEdit['Firm'].AsString:='1-1';
DocEdit['Kontragent'].AsString:='1-3';
DocEdit['Place'].AsString:='1-5';
DocEdit.Post; //сохраняем документ
DocEdit.SpecEdit.New; //создаем строку в спецификации документа
//устанавливаемреквизитыстроки спецификации
DocEdit.SpecEdit['Articul'].AsString:='1-7';
DocEdit.SpecEdit['Quantity'].AsDouble:=20;
DocEdit.SpecEdit['Summa'].AsCurrency:=100;
DocEdit.SpecEdit['NDS'].AsCurrency:=152.42;
DocEdit.SpecEdit.Post; //сохраняем строку спецификации
DocEdit.RegisterDoc; //регистраруем документ
DocEdit.UnRegisterDoc; //отменяем регистрацию документа
//записываем в базу документа глобальную переменную с именем LastDoc
DocEdit.SetGlobalValue(cmvCurrent, 'LastDoc', Code);
DocEdit.Delete; //удаляем документ
finally
DocEdit.Free;
end;
end;

```

## 6.7 TcmDocSpecEdit

Компонент **TcmDocSpecEdit** предназначен для редактирования записей спецификации документа. При создании объекта класса TcmDocEdit автоматически создается объект для редактирования спецификации документа. Ссылка на него находится в свойстве TcmDocEdit.SpecEdit.

TcmDocSpecEdit = class(TcmRecvizitEdit)

Свойства и методы	Описание
<b>property</b> Code: integer { <i>read only</i> }	Уникальный код строки спецификации документа.
<b>property</b> LastTime: TDateTime { <i>read only</i> }	Дата и время последнего редактирования строки спецификации
function Get(ACode: integer): boolean	Загружает строку спецификации с кодом ACode.
<b>procedure</b> New(EditMode: boolean = true; ACode: integer = 0)	Создает новую строку спецификации документа. <b>EditMode</b> - см. примечание в конце таблицы. <b>ACode</b> - уникальный код записи. Код записи генерируется автоматически, задавать его вручную может понадобиться только при переносе документа из другой базы.
<b>procedure</b> Post(EditMode: boolean = true)	Сохраняет редактируемую строку спецификации.
<b>procedure</b> Delete	Удаляет редактируемую строку спецификации. <b>EditMode</b> - см. примечание в конце таблицы.
<b>function</b> DeleteQuery(Code: integer): boolean	Удаляет строку спецификации с кодом Code. Перед выполнением операции выводит диалог для ввода подтверждения. Если операция успешно завершена, возвращает true.
<b>procedure</b> DeleteAll	Удаляет все строки спецификации.
<b>function</b> DeleteAllQuery: boolean	Удаляет все строки спецификации. Перед выполнением операции выводит диалог для ввода подтверждения. Если операция успешно завершена, возвращает true.
<b>function</b> FirstRecord: boolean	Загружает первую строку спецификации, если строка найдена возвращает true.
<b>function</b> NextRecord: boolean	После вызова метода FirstRecord, используется для загрузки следующей записи. Если запись найдена возвращает true.

```
function GetCodeByRecvizit(ARecvName, AValue): integer
```

Возвращает код строки спецификации для которой значение реквизита с именем ARecvName равно AValue. Если запись не найдена, возвращает -1.

**Примечание:**

Параметр **EditMode** передается методам **New** и **Post**, по умолчанию данный параметр равен true. При создании и сохранении записи спецификации, генерируется ряд дополнительных запросов к базе данных - проверяется проведен ли документ, входит ли он в закрытый период и т.д. При "стандартной" работе, связанной с редактированием документа пользователем, все эти проверки необходимы. Но представим ситуацию когда спецификация документа генерируется автоматически в скрипте (например, при создании инвентаризационной ведомости). Стандартный алгоритм вставки записи в спецификацию документа выглядит так:

```
//DocSpec-объект класса TcmDocSpecEdit
DocSpec.New;
//инициализируем реквизиты записи
DocSpec.Post;
```

В спецификацию могут быть вставлены сотни и тысячи записей, и при каждой новой вставке будет выполняться множество лишних, дублирующихся запросов проверки состояния документа. Это может значительно замедлить работу в целом. Для того, что бы отключить дополнительные проверки, следует передавать методам **EditMode=false**, т.е. код должен выглядеть следующим образом:

```
//DocSpec-объект класса TcmDocSpecEdit
DocSpec.New(false);
//инициализируем реквизиты записи
DocSpec.Post(false);
```

## 6.8 TcmTableEdit

При проведении документа производится запись информации в сводные таблицы. Для описания процесса проведения следует перекрыть событие on\_Register метаобъекта TcmDoc. Событию on\_Register передается ссылка на регистрируемый документ (объект класса TcmDocEdit). Для доступа к сводной таблице следует создать объект класса **TcmTableEdit**. Создать объект можно при помощи метода TcmDocEdit.GetTableEditObject:

```
function GetTableEditObject(TableName: string): TcmTableEdit;
```

Методу передается имя таблицы, к которой следует получить доступ. Имя таблицы находится в свойстве TcmTable.TableName. Вызывать данный метод можно только при проведении документа (в событии TcmDoc.OnRegister), при вызове этого метода в любом другом месте программного кода будет сгенерировано исключение.

Класс **TcmTableEdit** позволяет получить доступ к измерениям, ресурсам и реквизитам сводной таблицы:

```
TcmTableEdit = class(TObject)
```

Свойства и методы	Описание
<b>property</b> LineCode: integer	Код записи в таблице.
<b>property</b> LineSubCode: integer	Дополнительный код записи в таблице. Используется если по одной строке документа формируется несколько записей в таблице.
<b>property</b> PostFix: <b>string</b>	Постфикс кода записи таблицы. Используется для обеспечения генерации уникальных кодов записей.
<b>property</b> Scale[AName: string]: TcmRecvizitData	Список измерений сводной таблицы.
<b>property</b> Resource[AName: <b>string</b> ]: TcmRecvizitData	Список ресурсов сводной таблицы.
<b>property</b> Recvizit[AName: <b>string</b> ]: TcmRecvizitData	Список реквизитов сводной таблицы.

```
function Post: string
```

Записывает данные в сводную таблицу и возвращает уникальный код записи в таблице.

При распорядении документа, все записи из сводных таблиц удаляются автоматически. Запрещено вносить изменения в проведенный документ.

#### Генерирование уникального кода записи документа.

Записи, вносимые в сводные таблицы, должны иметь уникальный код в пределах ВСЕЙ СУБД.

Код записи генерируется в следующем формате:

```
<DocumCode>-<LineCode>-<LineSubCode>-<PostFix>
```

Примеры:

Код документа: 06-1-1-23, остальные параметры не указаны.

Сгенерированный код: "06-1-1-23"

Код документа: 06-1-1-23, LineCode=1

Сгенерированный код: "06-1-1-23-1"

Код документа: 06-1-1-23, LineCode=1, LineSubCode=2

Сгенерированный код: "06-1-1-23-1-2"

Код документа: 06-1-1-23, LineCode=1, LineSubCode=2, PostFix="K"

Сгенерированный код: "06-1-1-23-1-2-K"

Код документа: 06-1-1-23, PostFix="K"

Сгенерированный код: "06-1-1-23-K"

#### Пример процедуры проведения документа:

*{Проведение документа поставки товара}*

```
procedure PrihodDoc_onRegister(Sender: TObject);
```

```
var DocumTable, PartyTable, MoveTable: TcmTableEdit;
```

```
Doc: TcmDocEdit;
```

```
PartyCode: string;
```

```
begin
```

```
Doc:=TcmDocEdit(Sender); //получаем ссылку на документ
```

*//вставка записи в регистр документов*

```
DocumTable:=Doc.GetTableEditObject('DocumTable');
```

```
with DocumTable do
```

```
try
```

```
Recvzit['Firm'].AsString:='1-1';
```

```
Recvzit['Kontragent'].AsString:='1-3';
```

```
Recvzit['Place'].AsString:='1-5';
```

```
Post;
```

```
finally
```

```
DocumTable.Free;
```

```
end;
```

*//таблицы партий и движения товара*

```
PartyTable:=Doc.GetTableEditObject('PartyTable');
```

```
MoveTable:=Doc.GetTableEditObject('MoveTable');
```

```
try
```

*//в цикле просматриваем спецификацию документа*

```
with Doc.SpecEdit do
```

```
if FirstRecord then
```

```
repeat
```

*//.....добавляем информацию по партии товара.....*

```
PartyTable.LineCode:=Code;
```

```
PartyTable.Recvzit['PrihodCost'].AsDouble:=Recvzits['Summa'].AsDouble/
```

```
Recvzits['Quantity'].AsDouble;
```

```
PartyTable.Recvzit['Kontragent'].AsString:=Doc.Recvzits['Kontragent'].AsString;
```

```
PartyCode:=PartyTable.Post;
```

```
//.....добавляем информацию о движении товара.....
MoveTable.LineCode:=Code;
MoveTable.Scale['Firm'].AsString:=Doc.Recvizits['Firm'].AsString;
MoveTable.Scale['Place'].AsString:=Doc.Recvizits['Place'].AsString;
MoveTable.Scale['Articul'].AsString:=Recvizits['Articul'].AsString;
MoveTable.Scale['Party'].AsString:=PartyCode;
MoveTable.Resource['Quantity'].AsDouble:=Recvizits['Quantity'].AsDouble;
MoveTable.Post;
  until not NextRecord;
finally
PartyTable.Free;
MoveTable.Free;
end;
end;
```

## 6.9 TcmRezervEdit

Класс **TcmRezervEdit** предназначен для резервирования ресурсов сводных таблиц. Получить экземпляр объекта можно с помощью метода класса TcmDocEdit.GetRezervEdit. Резервирование ресурсов производится в обработчике события TcmDoc.onSetLockResource, попытка вызвать метод GetRezervEdit в любом другом месте программного кода приведет к генерации исключения.

TcmRezervEdit = class(TObject)

Свойства и методы	Описание
property Scale[AName: string]: TcmRecvizitData	Список измерений сводной таблицы.
property Resource[AName: string]: TcmRecvizitData	Список ресурсов сводной таблицы.
property DocLine: integer	Код строки документа для которой производится резервирование ресурсов.
function Post: string	Резервирование ресурсов.

Пример реализации обработчика события TcmDoc.onSetLockResource:

```
procedure SailDoc_onSpecLockResource(Sender: TObject);
var Doc: TcmDocEdit;
    RezObj: TcmRezervEdit;
begin
Doc:=TcmDocEdit(Sender);
RezObj:=Doc.GetRezervObject('MoveTable');
try
RezObj.DocLine:=Doc.SpecEdit.Code;
RezObj.Scale['Firm'].AsString:=Doc.Recvizits['Firm'].AsString;
RezObj.Scale['Place'].AsString:=Doc.Recvizits['Place'].AsString;
RezObj.Scale['Articul'].AsString:=Doc.SpecEdit.Recvizits['Articul'].AsString;
RezObj.Scale['Party'].AsString:='0';
RezObj.Resource['Quantity'].AsDouble:=Doc.SpecEdit.Recvizits['Quantity'].AsDouble;
RezObj.Post;
finally
RezObj.Free;
end;
end;
```

## 6.10 TcmOstatokList

Класс предназначен для просмотра списка оперативных остатков сводных таблиц. Если для таблицы установлен режим резервирования ресурсов, остатки возвращаются с учетом установленного резерва.

TcmOstatokList = class(TObject)

Свойства и методы	Описание
-------------------	----------



<b>property</b> FindScale[AName: string]: TcmRecvizitData	Список измерений по которым производится выборка данных из таблицы оперативных остатков.
<b>property</b> Current[Index: integer]: TcmOstatokRec	Список строк загруженных из таблицы оперативных остатков. Загрузка производится при помощи метода GetOstatok. Список индексируется в диапазоне от 0 до Count.
<b>function</b> Count: integer	Возвращает число строк загруженных из таблицы оперативных остатков.
<b>procedure</b> GetOstatok	Загружает список строк из таблицы оперативных остатков.

Создать объект класса TcmOstatokList можно с помощью метода TcmDocEdit.GetOstatokObject. Использование класса TcmOstatokList бывает крайне полезно при проведении документа. Когда сумме списываемых ресурсов, указанных в документе, может быть сопоставлено несколько записей из сводной таблицы. Например, в накладной указывается количество отгружаемого товара, а в таблице товарных остатков, все товарные остатки хранятся в разрезе партий товара, следовательно, по одной строке документа в сводную таблицу может понадобиться вставить несколько записей по разным партиям для того, что бы набрать требуемое количество.

Последовательность работы с объектом класса следующая:

- Создаем объект при помощи метода TcmDocEdit.GetOstatokObject
- В свойстве FidnScale устанавливаем значения реквизитов по которым следует производить отбор записей остатков
- С помощью метода GetOstatok извлекаем записи из таблицы
- Используем свойство Current для доступа к списку полученных записей. Параметр Index находится в диапазоне от 0 до Count-1.

## 6.11 TcmOstatokRec

Класс предназначен для доступа к реквизитам записи оперативного остатка сводной таблицы. Ссылку на объект данного класса возвращает метод TcmOstatokList.Current.

TcmOstatokRec= class(TObject)

Свойства и методы	Описание
<b>property</b> Scale[AName: string]: TcmRecvizitData	Список измерений сводной таблицы.
<b>property</b> Resource[AName: string]: TcmRecvizitData	Список ресурсов сводной таблицы.

Примечание. Класс предназначен только для просмотра информации об оперативных остатках сводной таблицы. Следовательно, присваивание типа Rec.Resource[Quantity].AsDouble:=5; не изменит значение ресурса в сводной таблице (предполагается что Rec объект класса TcmOstatokRec).





## 7

## 7.1 TcmTransaction

Библиотека CoolLib изолирует разработчика от особенностей реализации сервера InterBase. Нет необходимости знать структуру таблиц базы, строить SQL запросы, явно запускать и завершать транзакции и т.д. Вместо этого библиотека предоставляет ряд компонентов, позволяющих разработчику абстрагироваться от уровня управления базами данных и сосредоточиться на разработке прикладной задачи. Одной из особенностей библиотеки является механизм автоматического управления транзакциями.

Рассмотрим как реализуется работа с транзакциями в стандартных системах управления базами данных:

```
-----
StartTransaction; //стартуемтранзакцию

//выполняемкакиелибодействиянадобъектамибазы

if Ok then CommitTransaction //есливсёхорошо,сохраняемтранзакцию
  else RollbackTransaction; //иначеоткатываемвседеланныеизменения
-----
```

Ключевым моментом в приведенном фрагменте является то, что программист явно указывает начало транзакции, контролирует результаты работы и программирует действия при сохранении и откате внесенных изменений. Разработчику системы CoolManager никогда не придется писать подобный код. Он вообще может не иметь ни малейшего представления о системе управления транзакциями в InterBase, их типах, уровнях изоляции и о многом другом. Всю рутинную работу за него делают компоненты библиотеки CoolLib. Теперь посмотрим как разработчик может управлять транзакциями в системе CoolManager. Вообще все транзакции можно разделить на три вида:

- читающие - в рамках таких транзакций выполняются стандартные SELECT запросы SQL.
- пишущие - выполняют запросы обновления данных базы (INSERT, UPDATE, DELETE)
- композитные - в рамках одной транзакции выполняется как извлечение, так и модификация данных базы.

При работе в системе CoolManager все транзакции стартуют и завершаются автоматически, без участия разработчика. При этом разработчик всё же должен иметь возможность прервать транзакцию и откатить внесенные изменения. Справочники и документы описанные в конфигураторе, имеют ряд обработчиков событий. Эти события возникают при редактировании записей справочников, проведении и распроведении документов, резервировании ресурсов сводных таблиц и т.д. Каждый обработчик стартует в рамках отдельной транзакции, которая автоматически завершается при выходе из процедуры обработчика. Если разработчику требуется прервать выполнение обработчика и откатить все сделанные изменения, достаточно сгенерировать исключение в теле процедуры.

Пример:

```
-----
//обработчиксобытияпроведениядокументаPrihodDoc
procedure doc_PrihodDoc_onRegister(Sender: TObject);
begin
  //выполняемобновлениесводныхтаблицбазы
  //.....

  //чтотопошлонетак,следуетпрерватьпроведениедокументаиоткатитьвсевнесенныеизменения
  RaiseException('Проведение документа отменено. ');

  //другиекомандыобновленияданных
end;
-----
```

Процедура RaiseException генерирует исключение. При этом выполнение процедуры будет

прервано, все изменения, сделанные в рамках текущей транзакции будут отменены. Т.е. фактически выполняется команда RollbackTransaction.

Но иногда требуется явно запустить транзакцию, в рамках которой выполняется некая последовательность команд обновления данных (взаимосвязанное обновление записей справочников, создание и проведение ряда документов и т.д.). Для выполнения таких транзакций служит компонент **TcmTransaction**.



TcmTransaction = class(TComponent)

Свойства и методы	Описание
property Base: TcmBases	Ссылка на метаобъект базы данных.
procedure Execute;	Выполняет процедуру заданную в обработчике onExecute
property onExecute: TNotifyEvent	Обработчик события вызываемый в методе Execute.

Работа с компонентом достаточно проста. Следует написать обработчик события OnExecute, в котором задается последовательность команд, выполняемых в рамках одной транзакции. Для старта транзакции вызвать метод Execute. Если во время выполнения требуется прервать транзакцию и откатить все сделанные изменения, сгенерировать исключение в теле обработчика события OnExecute.

## 7.2 TcmBufData

Класс **TcmBufData** предназначен для оперативного доступа к данным справочников и сводных таблиц. При извлечении данных из базы, они сохраняются во внутреннем кэше компонента. При последующем обращении к той же записи, данные извлекаются не из базы, а из кэша, что позволяет значительно повысить скорость доступа к данным.



TcmBufData = class(TComponent)

Свойства и методы	Описание
property Base: TcmBases	Ссылка на метаобъект базы данных.
function GetSprRecvizit(ACode: string): TcmRecvizitList;	Возвращает список реквизитов записи справочника. ACode - код записи справочника. Список реквизитов содержит все реквизиты описанные в конфигураторе, а так же набор стандартных реквизитов. Описание стандартных реквизитов справочника приведено в конце раздела.
function GetSprRecvizitByDate(ACode: string; Data: TDateTime): TcmRecvizitList	Возвращает список реквизитов записи справочника, но на указанную дату. Поиск производится в таблице истории редактирования записей справочников. ACode - код записи справочника, Data - дата на которую следует вернуть реквизиты. Если подходящей записи в истории редактирования не найдено - возвращает nil.
function GetTableRecvizit(ATableName: string; ACode: string): TcmRecvizitList	Возвращает список реквизитов записи сводной таблицы. ATableName - имя сводной таблицы, ACode - код записи таблицы. Список реквизитов содержит все реквизиты описанные в конфигураторе, а так же набор стандартных реквизитов. Описание стандартных реквизитов сводной таблицы приведено в конце раздела.

<b>function</b> GetTableOperRes(ATableName, AResName: <b>string</b> ; Params: TStringList; UseReserv: boolean): double	Возвращает текущее значение оперативных остатков ресурса таблицы. <ul style="list-style-type: none"> <li>• ATableName - имя сводной таблицы</li> <li>• AResName - имя ресурса таблицы</li> <li>• Params - список измерений таблицы и их текущих значений. Каждая строка списка имеет формат: &lt;ResName&gt;=&lt;ResData&gt;</li> <li>• UseReserv - если равно true, возвращает остатки ресурса с учетом установленного резерва.</li> </ul>
<b>procedure</b> Clear	Очищает кэш буфер компонента.
<b>procedure</b> ClearRecord(ACode: <b>string</b> )	Удаляет из буфера запись с кодом ACode.
<b>procedure</b> AddSprCode(ACode: <b>string</b> )	Добавляет код записи справочника во временный буфер. Реквизиты всех сохраненных в буфере записей извлекаются одновременно, при вызове метода GetRecords.
<b>procedure</b> AddTableCode(ATable, ACode: <b>string</b> )	Добавляет код записи сводной таблицы во временный буфер. Реквизиты всех сохраненных в буфере записей извлекаются одновременно, при вызове метода GetRecords.
<b>procedure</b> GetRecords	Извлекает из базы записи, чьи кода предварительно были сохранены во временном буфере посредством вызова методов AddSprCode и AddTableCode.

**Список стандартных реквизитов справочника:**

Имя и тип реквизита	Описание
Code: <b>string</b> [32]	Уникальный код записи справочника.
SprCode: integer;	Уникальный код справочника. Задается при описании справочника в конфигураторе.
State: <b>string</b> [1]	Статус записи. Может принимать одно из двух значений: '+' - активная запись '-' - запись помечена на удаление.
RecordOwner: <b>string</b> [32]	Запись владелец. Если в конфигураторе справочник описан как подчиненный другому справочнику, данное поле хранит код записи владельца.
LastTime: TDateTime	Дата и время последнего изменения записи.
UserName: <b>string</b> [32]	Имя пользователя выполнившего последнее изменение записи.
RecordName: <b>string</b> [256]	Имя записи.
BaseCode: smallint	Код базы данных, в которой было сделано последнее изменение записи.
OldCode: <b>string</b> [32]	Код записи в другой системе автоматизации. Обычно используется при начальном переносе данных.
RecordParent: <b>string</b> [32]	Если тип справочника stTree (дерево), данное поле хранит ссылку на запись верхнего уровня.

**Список стандартных реквизитов сводной таблицы:**

Имя и тип реквизита	Описание
Code: <b>string</b> [32]	Уникальный код записи таблицы.
Docum: <b>string</b> [32]	Код документа, при проведении которого была создана данная запись сводной таблицы.
Data: TDate	Дата совершения операции.

## 7.3 TcmQuery

Класс **TcmQuery** предназначен для выполнения запросов к базе данных. С помощью данного компонента можно формировать следующие виды запросов:

- Список записей справочника
- История редактирования записи справочника
- Список документов
- История редактирования документа
- Спецификация документа
- Список записей сводной таблицы
- Оперативные значения ресурсов сводной таблицы
- Значения ресурсов сводной таблицы на заданную дату
- Обороты ресурсов сводной таблицы за указанный период
- Список зарезервированных ресурсов сводной таблицы

Настройка запроса производится с помощью специализированного визуального редактора, который позволяет максимально упростить и ускорить создание запросов любой степени сложности.



TcmQuery = class(TComponent)

Свойства и методы	Описание
<b>property</b> BufData: TcmBufData	Ссылка на компонент, используемый для извлечения записей справочников и сводных таблиц из базы данных.
<b>property</b> BufSize: integer	Количество записей извлекаемых из базы данных за один раз. Если <b>BufSize=0</b> извлекаются все записи удовлетворяющие заданному запросу.
<b>property</b> UseDocumSegment: boolean	Используется для оптимизации запросов выбирающих список документов. Если данное свойство равно true, просматривается только сегмент базы, с которым связан метакласс документа, иначе просматриваются все сегменты базы.
<b>property</b> RunProcess: boolean	Если производится выборка данных посредством метода OpenQuery - возвращает true. Можно прервать выборку данных установив значение свойства равным false.
<b>property</b> OrderBy: <b>string</b>	Порядок сортировки записей набора данных. Задается список полей таблицы разделенных запятыми. Если значение BufSize<>0, сортировка всегда производится по ключевому полю таблицы.
<b>property</b> FilterStr: <b>string</b>	Строка условия для оператора WHERE формируемого запроса.
<b>property</b> Params[Index: integer]: TcmParamRecvzitData	Возвращает параметр запроса по его порядковому номеру. Значение <b>Index</b> находится в диапазоне от 0 до <b>ParamCount-1</b> .
<b>property</b> Fields[Index: integer]: TcmRecvzitData	Возвращает поле запроса по его порядковому номеру. Значение <b>Index</b> находится в диапазоне от 0 до <b>FieldCount-1</b> .
<b>default property</b> ByName[AName: <b>string</b> ]: TcmRecvzitData	Возвращает поле запроса по его имени.
<b>function</b> ParamCount: integer	Возвращает число параметров запроса.
<b>function</b> FieldCount: integer	Возвращает число полей запроса.
<b>function</b> ParamByName(AName: <b>string</b> ): TcmParamRecvzitData	Возвращает ссылку на параметр по его имени.
<b>function</b> FieldByName(AName: <b>string</b> ): TcmRecvzitData	Возвращает ссылку на поле по его имени.
<b>procedure</b> Prepare	Инициализирует запрос и подготавливает его к работе.
<b>procedure</b> OpenQuery	Открываем запрос, извлекаем данные.

<b>procedure</b> Next	Возвращает следующий набор записей. Число записей возвращаемых за один раз определяется параметром <b>BufSize</b> .
<b>procedure</b> Prior	Возвращает предыдущий набор записей. Число записей возвращаемых за один раз определяется параметром <b>BufSize</b> .
<b>function</b> Eof: boolean	Возвращает true, если был достигнут конец набора данных.
<b>function</b> Bof: boolean	Возвращает true, если было достигнуто начало набора данных.
<b>property</b> onGetRecord: TNotifyEvent	Событие вызываемое при выборке новой записи из базы данных. Данное событие следует перекрыть, что бы определить алгоритм обработки записей, извлекаемых из базы данных.

## 7.4 TcmDataSet

Класс **TcmDataSet** предназначен для создания табличных наборов данных. Отображается сформированный набор с помощью компонента **TcmGridEh**. Используя **TcmDataSet** можно создавать следующие виды наборов данных:

- **стандартные наборы данных**
- **многоуровневые наборы данных поддерживающие отношения Master-Detail**
- **древовидные структуры данных**
- **cross-tab наборы. Используются, когда фактическое число столбцов набора заранее не известно.**

Все выше перечисленные виды могут группироваться в одном наборе. Т.е. набор может быть многоуровневым, верхний уровень может быть представлен в виде древовидного списка, при этом набор может содержать любое количество различных cross-tab групп данных.

Компонент спроектирован таким образом, что бы позволить разработчику создавать наборы данных любой степени сложности с минимумом трудозатрат. Компонент содержит визуальный редактор, позволяющий создать структуру будущего набора без написания программного кода. Так же компонент содержит ряд событий, обработчики которых позволяют гибко настроить внешний вид и структуру набора. При создании набора требуется написание минимального кода, т.к. большую часть работы выполняет компонент **TcmDataSet**.



TcmDataSet = class(TComponent)

Свойства и методы	Описание
<b>property</b> AutoFitColWidths: boolean	Если равно true выравнивает ширину столбцов по ширине сетки.
<b>property</b> AutoSort: boolean	Если равно true, щелчок левой кнопкой мыши по заголовку столбца сетки (св-во Grid) сортирует набор данных в порядке возрастания значений данного столбца, повторный щелчок сортирует данные в порядке убывания.
<b>property</b> BufData: TcmBufData	Ссылка на компонент, используемый для извлечения записей справочников и сводных таблиц из базы данных.
<b>property</b> Grid: TDBGridEh	Ссылка на компонент, используемый для отображения набора данных.
<b>property</b> SortIndex: integer	Номер столбца сетки по которому производится сортировка данных набора, после его создания (сортировка по умолчанию).

<b>property</b> SortField: <b>string</b>	Имя поля по которому производится сортировка данных набора, после его создания. Имеет более высокий приоритет, чем SortIndex, т.е. сортировка по SortIndex производится только в том случае, если не указано значение свойства SortField.
<b>property</b> Data: TcmRecvizitList	Буфер используемый при формировании новой записи. Сперва список реквизитов <b>Data</b> заполняется данными для новой записи, затем посредством метода <b>AddRecord</b> производится вставка новой записи в набор данных.
<b>default property</b> ByName[AName: <b>string</b> ]: TcmRecvizitData	Возвращает ссылку на реквизит буфера Data, поиск производится по имени реквизита.
<b>property</b> Root: TcmRecord (read only)	Возвращает ссылку на корневую запись отчета.
<b>property</b> ActiveRecord: TcmRecord (read only)	Возвращает ссылку на выбранную в Grid-е запись, если Grid не содержит записей возвращает nil.
<b>property</b> ActiveRecordKey: <b>string</b>	Возвращает ключ выбранной в Grid-е записи, если Grid не содержит записей возвращает пустую строку.
<b>property</b> ImageList: TImageList	Коллекция картинок для сетки Grid.
<b>property</b> Title: TStringList	Заголовок отчета. Используется при экспорте данных в Excel
<b>property</b> UseGroupBy: boolean	Если равно true (по умолчанию) использовать выражение GROUP BY при формировании запросов к сводным таблицам.
<b>procedure</b> Prepare	Инициализирует компонент, вызывается перед вставкой данных в набор посредством метода <b>AddRecord</b> .
<b>function</b> AddRecord: TcmRecord	Вставка новой записи в набор данных. Предварительно значения реквизитов записи должны быть записаны в свойстве <b>Data</b> . Возвращает ссылку на запись, если набор данных состоит из нескольких уровней, возвращается ссылка на запись самого нижнего уровня.
<b>procedure</b> AddFromQuery(Query: TcmQuery)	Присваивает реквизитам буфера Data текущие значения полей запроса (Query.Fields). Присваивание производится если имя реквизита и поля совпадают.
<b>function</b> FindRecord(ARecord: TcmRecord): boolean	Ищет в наборе данных запись ARecord. Если запись найдена, делает её активной в Gridе и возвращает true.
<b>function</b> Locate(AFields, AData: <b>string</b> ): boolean	Ищет наборе данных запись. Если запись найдена, делает её активной в Gridе и возвращает true. AFields - список полей, имена в списке разделяются - ',' AData - значения полей, данные в списке разделяются - ','
<b>procedure</b> CreateDataSet	Вызывается после того, как в набор добавлены все записи. Настраивает сетку для отображения полученных данных.
<b>procedure</b> Refresh	Пересоздает набор данных и пересчитывает итоги.
<b>procedure</b> Update	Обновляет информацию выводимую в Grid-е, при этом набор данных не пересоздается.
<b>procedure</b> First	Перейти к первой записи в Grid-е
<b>procedure</b> Last	Перейти к последней записи в Grid-е.



<b>function</b> Active: boolean;	Возвращает true, если набор данных активен.
<b>function</b> SetGroupKey(AGroupName, AKey: <b>string</b> ): TcmCGItem	Устанавливает текущее значение ключа для группы с именем <b>AGroupName</b> . Текущее значение ключа содержится в свойстве <b>AKey</b> . Возвращает ссылку на структуру с описанием параметров активного ключа.
<b>procedure</b> ClearLocaFilter	Очищает локальный фильтр набора данных. Для отбора записей удовлетворяющих условиям фильтрации следует определить обработчик события onFilterRec.
<b>procedure</b> SetLocalFilter	Устанавливает локальный фильтр для набора данных.
<b>procedure</b> SortByName(AFieldName: <b>string</b> ; Ascending: boolean)	Сортирует набор данных по значению поля с именем AFieldName. Параметр Ascending задает порядок сортировки, если равно true данные сортируются в порядке увеличения значений.
<b>procedure</b> SaveAsExcel(FileName: <b>string</b> )	Записывает набор данных в Excel файл. FileName - имя файла.
<b>procedure</b> SaveExcelDialog	Записывает набор данных в Excel файл. Предварительно выводит стандартный диалог для указания имени файла.
<b>function</b> NextQuery(Query: TcmQuery): boolean	При чтении данных блоками (когда параметр TcmQuery.BufSize <> 0) получает следующую группу записей и заносит её в набор данных. Если из запроса выбраны все записи, возвращает true.
<b>function</b> PriorQuery(Query: TcmQuery): boolean	При чтении данных блоками (когда параметр TcmQuery.BufSize <> 0) получает предыдущую группу записей и заносит её в набор данных. Если из запроса выбраны все данные, возвращает true.
<b>procedure</b> SetColumnFooterText(AName, AData: <b>string</b> )	Устанавливает текст AData в строку итогов столбца с меткой AName. Метка столбца - это произвольный набор символов, устанавливается в диалоге редактора столбца при настройке компонента TcmDataSet.
<b>procedure</b> SetColumnsVisible(ATag: integer; AVisible: boolean)	Устанавливает свойство Visible столбцов Grid-a равным AVisible, отбираются столбцы у которых свойство Tag равно ATag.
<b>procedure</b> FullCollapse	Если набор данных имеет древовидную структуру, свертывает все подразделы.
<b>procedure</b> FullExpand	Если отчет имеет древовидную структуру, разворачивает все подразделы.
<b>property</b> onNewRec: TcmNewRec	Событие при вставке новой записи в набор данных.
<b>property</b> onEditRec: TcmEditRec	Событие при редактировании записи набора данных.
<b>property</b> onNewTreeNode: TcmNewTreeNode	Событие при вставке нового узла в дерево.
<b>property</b> onCalcData: TcmCalcData	Событие при расчете итогов по строке набора данных.
<b>property</b> onDrawHighLight: TcmDrawHighLight	Событие для задания цвета фона и шрифта текста в ячейке по условию.
<b>property</b> onNewRecGroup: TcmNewReporRecGroup	Событие при вставке нового элемента группы.
<b>property</b> onEditRecGroup: TcmEditRecGroup	Событие при редактировании элемента группы.
<b>property</b> onCalcDataGroup: TcmNewRecGroup	Событие при расчете итогов по элементу группы.

<b>property</b> onEndBuild: TNotifyEvent	Событие после завершения построения набора данных. Используется для окончательной настройки данных, расчета итогов, удаления пустых строк и т.д. Событию передается ссылка на набор данных.
<b>property</b> onFilterRec: TcmFilterRec	Событие для установки локального фильтра набора данных.
<b>property</b> onChoiceData: TcmChoiceData	Событие при выборе реквизита набора данных.
<b>property</b> onReportGroupData: TcmChoiceGroupData	Событие при выборе реквизита элемента группы.
<b>property</b> onDataChange: TNotifyEvent	Событие при смене в Grid-е активной записи.
<b>property</b> onMessage: TcmMessage	Обработка сообщений генерируемых в процессе работы метода CreateDataSet.

### Список событий используемых компонентом TcmDataSet.

Событие	Описание
TcmNewRec= <b>procedure</b> (Sender: TObject; ALevel: integer; ARecvList: TcmRecvzitList) <b>of object</b> ;	Событие при вставке новой записи в набор данных. <ul style="list-style-type: none"> <li>• Sender - ссылка на компонент TcmDataSet</li> <li>• ALevel - номер текущего уровня набора данных</li> <li>• ARecvList - буфер реквизитов (редактирование)</li> </ul>
TcmEditRec= <b>procedure</b> (Sender: TObject; ALevel: integer; ANewData, ARecvList: TcmRecvzitList) <b>of object</b> ;	Событие при редактировании записи набора данных. <ul style="list-style-type: none"> <li>• Sender - ссылка на компонент TcmDataSet</li> <li>• ALevel - номер текущего уровня набора данных</li> <li>• ANewData - буфер реквизитов (просмотр)</li> <li>• ARecvList - ревизиты записи текущего уровня (редактирование)</li> </ul>
TcmNewRecGroup= <b>procedure</b> (Sender: TObject; ALevel: integer; AGroupName: <b>string</b> ; ARecvList: TcmRecvzitList) <b>of object</b> ;	Событие при вставке нового элемента группы. <ul style="list-style-type: none"> <li>• Sender - ссылка на компонент TcmDataSet</li> <li>• ALevel - номер текущего уровня набора данных</li> <li>• AGroupName - имя группы</li> <li>• ARecvList - буфер реквизитов (редактирование)</li> </ul>
TcmEditRecGroup= <b>procedure</b> (Sender: TObject; ALevel: integer; AGroupName: <b>string</b> ; ANewData, ARecvList, AGroupRecv: TcmRecvzitList) <b>of object</b> ;	Событие при редактировании элемента группы. <ul style="list-style-type: none"> <li>• Sender - ссылка на компонент TcmDataSet</li> <li>• ALevel - номер текущего уровня набора данных</li> <li>• AGroupName - имя группы</li> <li>• ANewData - буфер реквизитов (просмотр)</li> <li>• ARecvList - реквизиты записи текущего уровня (просмотр)</li> <li>• AGroupRecv - текущие значения реквизитов элемента группы (редактирование)</li> </ul>



TcmCalcData= <b>procedure</b> (Sender, ARec: TObject; ALevel: integer)	Событие при расчете итогов по строке отчета <ul style="list-style-type: none"> <li>• Sender - ссылка на набор данных (TcmDataSet)</li> <li>• ARec - ссылка на текущую запись набора (TcmRecord)</li> <li>• ALevel - номер текущего уровня набора данных</li> </ul>
TcmCalcGroupData= <b>procedure</b> (Sender, ARec: TObject; ALevel: integer; AGroupName, AKey: <b>string</b> ; ARecvList: TcmRecvzitList)	Событие при расчете итогов по элементу группы <ul style="list-style-type: none"> <li>• Sender - ссылка на набор данных (TcmDataSet)</li> <li>• ARec - ссылка на текущую запись набора (TcmRecord)</li> <li>• ALevel - номер текущего уровня набора данных</li> <li>• AGroupName - имя группы</li> <li>• AKey - код элемента группы</li> <li>• ARecvList - реквизиты элемента группы (редактирование)</li> </ul>
TcmNewTreeNode= <b>procedure</b> (Sender: TObject; ARecvList: TcmRecvzitList) <b>of object</b> ;	Вставка нового узла в дерево. <ul style="list-style-type: none"> <li>• Sender - ссылка на компонент TcmDataSet</li> <li>• ARecvList - список реквизитов текущей записи.</li> </ul>
TcmDrawHighLight = <b>function</b> (Sender: TObject; ALevel: integer; RecvName, Value: <b>string</b> ; ARecvList: TcmRecvzitList): boolean <b>of object</b> ;	Событие для проверки условий параметров условного выделения ячейки. Если возвращает true - ячейку следует выделить. <ul style="list-style-type: none"> <li>• Sender - ссылка на компонент TcmDataSet</li> <li>• ALevel - номер текущего уровня набора данных</li> <li>• RecvName - имя реквизита</li> <li>• Value - текущее значение реквизита</li> <li>• ARecvList - список реквизитов текущей записи.</li> </ul>
TcmFilterRec= <b>function</b> (Sender: TObject; ALevel: integer; ARecvList: TcmRecvzitList): boolean <b>of object</b> ;	Событие при установке локального фильтра набора данных. Если возвращает true текущая запись удовлетворяет текущим условиям фильтрации. <ul style="list-style-type: none"> <li>• Sender - ссылка на компонент TcmDataSet</li> <li>• ALevel - номер текущего уровня набора данных</li> <li>• ARecvList - список реквизитов текущей записи.</li> </ul>
TcmChoiceData= <b>procedure</b> (Sender: TObject; ALevel: integer; ARecvName: <b>string</b> ; ARecvList: TcmRecvzitList) <b>of object</b> ;	Событие при выборе реквизита набора данных. Выбор осуществляется двойным щелчком мыши по ячейке сетки, либо нажатием клавиши <Enter>. <ul style="list-style-type: none"> <li>• Sender - ссылка на компонент TcmDataSet</li> <li>• ALevel - номер текущего уровня набора данных</li> <li>• ARecvName - имя реквизита</li> <li>• ARecvList - список реквизитов текущей записи.</li> </ul>

<p>TcmChoiceGroupData = <b>procedure</b>(Sender: TObject; ALevel: integer; ARecvName, AGroupName, AKeyData: <b>string</b>; ARecvList, AGroupList: TcmRecvzitList) <b>of object</b>;</p>	<p>Событие при выборе реквизита элемента группы. Выбор осуществляется двойным щелчком мыши по ячейке сетки, либо нажатием клавиши <b>&lt;Enter&gt;</b>.</p> <ul style="list-style-type: none"> <li>• Sender - ссылка на компонент TcmDataSet</li> <li>• ALevel - номер текущего уровня набора данных</li> <li>• ARecvName - имя реквизита</li> <li>• AGroupName - имя группы</li> <li>• AKeyData - ключ элемента группы</li> <li>• ARecvList - список реквизитов текущей записи.</li> <li>• AGroupList - список реквизитов элемента группы.</li> </ul>
<p>TcmMessage = <b>procedure</b>(Sender: TObject; Msg: <b>String</b>; Index: Integer; Data: TObject);</p>	<p>Событие для передачи сообщений в скрипт.</p> <ul style="list-style-type: none"> <li>• Ссылка на объект-отправитель сообщения.</li> <li>• Msg - текст сообщения</li> <li>• Index - генератор, используемый для отображения длительного процесса.</li> <li>• Data - ссылка на связанный с сообщением объект.</li> </ul>

## 7.5 TcmRecord

Класс **TcmRecord** используется для описания записей наборов данных TcmDataSet.

TcmRecord = **class**(TObject)

Свойства и методы	Описание
<p><b>default property</b> Items[Name: <b>string</b>]: TcmRecvzitData</p>	<p>Доступ к реквизитам по имени.</p>
<p><b>property</b> Level: integer;</p>	<p>Номер уровня отчета к которому относится текущая запись.</p>
<p><b>property</b> Owner: TcmRecord</p>	<p>Владелец записи.</p>
<p><b>property</b> RecordKey: <b>string</b>;</p>	<p>Возвращает в виде строки значение уникального ключа записи.</p>
<p><b>property</b> Recvizits: TcmRecvzitList;</p>	<p>Список реквизитов текущей записи.</p>
<p><b>function</b> Count: integer;</p>	<p>Возвращает количество подчиненных записей.</p>
<p><b>property</b> Child[Index: integer]: TcmRecord;</p>	<p>Доступ к списку подчиненных записей. Параметр Index должен находиться в диапазоне от 0 до Count-1.</p>
<p><b>function</b> GetGroup(AGroup, AKey: <b>string</b>; AddMode: boolean = false): TcmRecvzitList</p>	<p>Возвращает элемент группы AGroup по ключу AKey. Если текущая запись не содержит запрошенного элемента, возвращает nil (при AddMode=false). Если AddMode=true и элемент не найден, создается новый элемент.</p>
<p><b>procedure</b> PostGroup(AGroup, AKey: <b>String</b>; Recv: TcmRecvzitList)</p>	<p>Сохраняет реквизиты элемента группы. Список реквизитов передается в свойстве Recv, имя группы AGroup, ключ группы AKey.</p>
<p><b>function</b> GetGroupCount(AGroup: <b>string</b>): integer</p>	<p>Возвращает число элементов группы AGroup связанных с текущей записью.</p>
<p><b>function</b> GetGroupByIndex(AGroup: <b>string</b>; AIndex: integer; var AKey: <b>string</b>): TcmRecvzitList</p>	<p>Возвращает элемент группы AGroup с индексом AIndex. Параметр AIndex должен находиться в диапазоне от 0 до GetGroupCount(AGroup). Параметр AKey возвращает ключ полученного элемента.</p>

<b>procedure</b> Sort(RecvName: <b>string</b> ; Ascending: boolean)	Сортировка записей по значению поля RecvName. Ascending определяет порядок сортировки, если равно true данные сортируются в порядке увеличения значений.
<b>procedure</b> Post	Сохраняет внесенные изменения. Вызывается после изменения значений реквизитов записи.
<b>procedure</b> Delete(i: integer)	Удаляет подчиненную запись по её порядковому номеру. Доступ к списку подчиненных записей производится посредством свойства Child.
<b>procedure</b> Remove(Rec: TcmRecord)	Удаляет подчиненную запись Rec.
<b>procedure</b> GetRecvzitSum(RecvName: <b>string</b> ; ALevel: integer; <b>var</b> Summa: double);	Возвращает сумму реквизита с именем RecvName, расположенному на уровне ALevel. Сумма возвращается в параметре Summa. Реквизит должен иметь целый или вещественный тип.



## 8

Библиотека **CoolLib** определяет два метакласса для описания главного меню приложения на уровне конфигурации. Пункты меню, описанные в конфигураторе создаются динамически в момент запуска приложения.

Для создания меню, следует вызвать процедуру **MakeMenu**:

```
procedure MakeMenu(MainMenu: TMainMenu; ATag: integer);
```

процедуре передается два параметра:

**MainMenu** - ссылка на объект главного меню приложения

**ATag** - целочисленное свойство используемое для дополнительной фильтрации добавляемых пунктов меню. В **MainMenu** добавляются только пункты (объекты класса **TcmMenuItem** в конфигурации), свойство **Tag** которых равно нулю, либо значению переданному в параметре **ATag**.

### 8.1 TcmMenuRoot

Метакласс **TcmMenuRoot** используется для описания "точки входа" подчиненных пунктов меню (**TcmMenuItem**) в главное меню приложения. **TcmMenuRoot** в иерархии метаклассов конфигурации подчинен непосредственно **TcmUnitComponent**. Следовательно, конфигурация может содержать описание нескольких групп пунктов меню, которые, к тому же, могут быть расположены в различных модулях конфигурации. Например стандартная конфигурация "Торговля" включает два модуля:

\units\MainMenu.cm4 - описание стандартного меню конфигурации.

\local\UserMenu.cm4 - используется для описания "пользовательских" пунктов меню, определенных при настройке конфигурации под конкретного заказчика.

```
TcmMenuRoot = class(TcmUnitComponent)
```

Свойства и методы	Описание
property Metadate: TcmMetadate	Ссылка на объект метаданных, с которыми связан объект
property RootItem: string	Имя пункта главного меню которому подчинена текущая группа пунктов. Если не указан - добавляется новый пункт в главное меню приложения.

### 8.2 TcmMenuItem

Объекты класса **TcmMenuItem** используются для описания пунктов главного меню приложения. Главное меню формируется автоматически, при вызове процедуры **MakeMenu**. В конфигураторе, объекты класса **TcmMenuItem** подчинены метаклассу **TcmMenuRoot**. Кроме того, объекты класса **TcmMenuItem** могут образовывать древовидные структуры.

```
TcmMenuItem = class(TcmUnitComponent)
```

Свойства и методы	Описание
property Picture: TBitmap	Изображение связанное с пунктом меню.
property GrantData: TcmMenuGrantData	Описание прав доступа к пункту меню.
property FormName: string	Имя формы вызываемой при выборе пункта меню.
property ShowModal: boolean	Если равно true, связанная с пунктом меню форма выводится в модальном режиме.
property Unique: boolean	Второй параметр, передаваемый методу Application.CreateCoolForm при создании не модальной формы.
property OnClick: TNotifyEvent	Событие при выборе пункта меню.

Если задан обработчик события **onClick**, он вызывается при выборе пункта меню. Иначе вызывается форма имя которой указано в свойстве **FormName**. Если имя формы не указано,

ничего не происходит, например пункт меню является разделителем или владельцем набора подчиненных пунктов меню.

## 8.3 TcmMenuGrantData

Класс **TcmMenuGrantData** используется для задания прав доступа к пункту меню, описанному в конфигураторе (объекты класса TcmMenuItem).

При описании пункта меню в конфигураторе можно определить права доступа к данному пункту. Если конфигурация использует стандартную процедуру авторизации пользователей, то при динамическом создании меню с помощью процедуры MakeMenu, проверяются права доступа текущего пользователя. Если права для пользователя определены, пункт добавляется к главному меню приложения, иначе игнорируется. Возможно задание трех типов прав доступа:

- по правам доступа к справочнику
- по правам доступа к документу
- по "дополнительным" правам доступа, описанным в метаданных прикладной задачи (объекты класса TcmRight).

type

TcmMenuGrantType=(mgNone,mgSpr,mgDoc,mgRight);

TcmMenuGrantData = class (TPersistent)

Свойства и методы	Описание
property GrantType: TcmMenuGrantType	Тип прав доступа к пункту меню. Возможны следующие варианты: <ul style="list-style-type: none"> <li>• mgNone - права доступа не заданы.</li> <li>• mgSpr - по правам доступа к справочнику</li> <li>• mgDoc - по правам доступа к документу</li> <li>• mgRight - по праву доступа описанному в конфигурации</li> </ul>
property ObjectName: string	Имя объекта определяющего права доступа к пункту меню. при GrantType=mgSpr - имя справочника при GrantType=mgDoc - имя документа при GrantType=mgRight - имя права доступа (тип TcmRight)
property GrantMask: byte	Битовая маска определяющая, какие права доступа к справочнику или документу должен иметь текущий пользователь, чтобы пункт меню был для него доступен. Маска формируется из множества следующих прав: <ul style="list-style-type: none"> <li>• Доступ на создание</li> <li>• Доступ на изменение</li> <li>• Доступ на удаление</li> <li>• Доступ на проведение (только для документов)</li> <li>• Доступ на распределение (только для документов)</li> </ul>

Права доступа настраиваются визуально, с помощью специального редактора. Внешний вид редактора зависит от типа определяемых прав доступа.

Настройка по правам доступа к справочнику:

**Редактор прав доступа**

Тип объекта: Справочник

Список объектов:

- Объекты
- Места хранения
- Классификатор контрагентов
- Контрагенты**
- Список мест доставки
- Фирмы
- Счета фирм
- Страны
- Единицы измерения
- Торговые марки

Права:

- Создание
- Редактирование
- Удаление
- Проведение
- Отмена проведения

OK Отменить

Настройка по правам доступа к документу:

**Редактор прав доступа**

Тип объекта: Документ

Список объектов:

- Поставка товара**
- Возврат товара поставщику
- Внутреннее перемещение
- Инвентаризационная ведомость
- Оприходование излишков
- Списание брака и недостачи
- Реализация товара по накладным
- Возврат товара от покупателя
- Движение денежных средств
- Кассовая смена

Права:

- Создание
- Редактирование
- Удаление
- Проведение
- Отмена проведения

OK Отменить

Настройка по "дополнительным" правам доступа:

**Редактор прав доступа**

Тип объекта: Право доступа

Список объектов:

- Журналы документов
- Отчеты
- Интерфейс
  - Общие
  - Административные
    - Настройка параметров**
    - Групповое перепроведение документов
    - Администрирование

OK Отменить





## 9

## 9.1 TcmGridEh



TcmGridEh = class(TDBGridEh)

Свойства и методы	Описание
procedure SaveColumns	Сохраняет в ini файле текущие настройки таблицы (ширину столбцов и порядок их следования). Файл инициализации называется grids.ini и располагается в каталоге с исполняемыми файлами ИСР CoolManager.

Компонент **TcmGridEh** - это сетка для отображения табличного набора данных. Компонент наследован от **TDBGridEh** - одного из самых продвинутых и многофункциональных Grid компонентов созданных для среды Delphi. **TDBGridEh** является одним из основных компонентов библиотеки EhLib разрабатываемой Дмитрием Большаковым ([www.ehlib.com](http://www.ehlib.com)). В стандартную поставку CoolManager включен скомпилированный bpl пакет библиотеки, полную версию с исходными текстами можно скачать на сайте разработчика. **TcmGridEh** не добавляет никакой новой функциональности к стандартному компоненту и создан с одной лишь целью - для подключения Grid-а к палитре компонентов. **TcmGridEh** используется компонентом TcmDataSet для визуального отображения сформированного набора данных. Не смотря на огромное количество опубликованных свойств **TDBGridEh**, разработчику использующему ИСР CoolManager нет необходимости вникать во все тонкости настройки данного компонента, всю основную работу за него делает TcmDataSet. Достаточно просто положить на форму два компонента: TcmDataSet и **TcmGridEh**, после чего указать в свойстве **TcmDataSet.Grid** ссылку на **TcmGridEh**. Настройка внешнего вида Gridа и его функциональности производится при вызове метода **TcmDataSet.CreateDataSet**.

Следует отметить, что в отличие от среды Delphi, CoolManager использует совершенно другой подход к отображению табличных наборов данных. Стандартный Delphi подход состоит в использовании следующей трехзвенной архитектуры:

**TDataSet -> TDataSource -> TDBGrid**

где:

- **TDataSet** - набор данных
- **TDataSource** - источник данных для TDBGrid, извлекает данные из связанного с ним **TDataSet**
- **TDBGrid** - сетка для визуального отображения набора данных, источником данных служит **TDataSource**.

Компонент TcmDataSet не является наследником стандартного **TDataSet** и не "привязан" ни к какому конкретному набору данных. Структура набора описывается визуально, на этапе проектирования. Набор данных формируется программно, при выполнении скрипта. Сформированный набор может быть многоуровневым, иметь древовидную структуру, включать любое количество cross-tab групп (группа столбцов, точное количество элементов которой заранее не известно). TcmDataSet предоставляет набор методов для навигации по сформированному набору, но для визуального отображения данных требуется компонент **TDBGridEh**. Конечно стандартная трехзвенная Delphi архитектура реализуется "за кадром", но с точки зрения разработчика, для визуального представления данных достаточно подключить к **TcmDataSet -> TDBGridEh** и вызвать метод **TcmDataSet.CreateDataSet**. Подобный подход изолирует разработчика от большинства тонкостей и нюансов реализации компонентной модели Delphi и позволяет в кратчайшие сроки создавать достаточно сложные визуальные структуры данных без написания объемного программного кода.

## 9.2 TcmPrintGridEh



TcmPrintGridEh = class(TPrintDBGridEh)

Невизуальный компонент **TcmPrintGridEh** используется для вывода на печать с возможностью предварительного просмотра сетки TcmGridEh. Компонент наследован от **TPrintDBGridEh**, данный компонент включен в библиотеку EhLib. **TcmPrintGridEh** не добавляет никакой новой функциональности реализованной в классе **TPrintDBGridEh** и создан с одной лишь целью - для подключения к палитре компонентов.

## 9.3 TcmDataProducer

**TcmDataProducer** невизуальный компонент, используемый при создании форм редактирования записей справочников, шапок документов и записей спецификации документа. Компонент получает ссылку на класс доступа к данным (TcmSprEdit, TcmDocEdit или TcmDocSpecEdit) и "связывает" реквизиты редактируемого объекта с визуальными компонентами расположенными на форме. Для отображения значений реквизитов в визуальных компонентах используется метод **SetRecvzits**, для чтения данных из визуальных компонент метод **GetRecvzits**. Для того, что бы **TcmDataProducer** правильно настроил связи, компонент для редактирования реквизита должен называться так же как реквизит (св-во Name компонента). Кроме того, визуальные компоненты должны иметь определенный тип. Ниже приведена таблица, где указаны типы реквизитов системы CoolManager и соответствующие им визуальные компоненты:

Тип реквизита	Визуальный компонент	Страница палитры компонент
cmrInteger, cmrInt64, cmrCurrency, cmrDouble	TcmNumEdit	CoolMan Edit
cmrString, cmrDateTime	TcmEdit	CoolMan Edit
cmrDate, cmrTime	TcmDateEdit	CoolMan Edit
cmrEnum	TcmEnumEdit	CoolMan Edit
cmrBoolean	TCheckBox	Standard
cmrSpr, cmrTable	TcmObjectEdit	CoolMan Edit



TcmDataProducer = class(TComponent)

Свойства и методы	Описание
<b>property</b> SectionName: string	Имя секции ini файла, в которой сохраняются текущие значения выбранных полей.
<b>property</b> RecvSaved: string	Список визуальных компонент, чьи текущие значения следует сохранять в ini файле.
<b>property</b> BufData: TcmBufData	Компонент для работы с записями справочников и сводных таблиц. Используется при настройке компонентов класса TcmObjectBox.
<b>property</b> DataObject: TcmUnitComponent	Объект доступа к данным (TcmSprEdit, TcmDocEdit или TcmDocSpecEdit).
<b>property</b> IsSetRecvzits: boolean (read only)	Устанавливается равным true при выполнении метода SetRecvzits. Данное свойство может быть использовано в обработчике события OnChange визуальных компонент. Позволяет определить кто производит изменение значения, пользователь или метод SetRecvizit.
<b>procedure</b> SetRecvzits	Заносит значения реквизитов текущего объекта доступа к данным в визуальные компоненты формы.

<b>procedure</b> GetRecvizits	Считывает значения реквизитов текущего объекта доступа к данным из визуальных компонент формы.
<b>procedure</b> SaveControls	Сохраняет текущее значение визуальных полей ввода данных в ini файл. Список полей, текущие значения которых следует запоминать в Ini файле, задаются с помощью редактора компоненты.
<b>procedure</b> ReadControls	Читает текущее значение визуальных полей ввода данных формы из ini файла.

## 9.4 TcmToolBar

Компонент **TcmToolBar** предназначен для создания панелей управления. Компонент наследован от **TToolBar**, к базовой функциональности добавлены следующие возможности:

- Новый редактор компонента. Позволяет создавать кнопки на панели управления, выбирая их из стандартного списка. При создании служебной кнопки автоматически настраиваются её свойства (имя, подсказка, индекс картинки, режим работы и т.д). В список изображений **ImageList** автоматически добавляется картинка, связанная с создаваемой кнопкой. Каждая служебная кнопка имеет уникальный целочисленный код, этот код сохраняется в свойстве **Tag** кнопки.
- Метод **ButtonActions**. По коду введенной клавиши определяет связанную с ней стандартную кнопку и выполняет обработчик события кнопки **onClick**.
- Автоматическая обработка событий стандартных кнопок, если для кнопки не задан обработчик события **onClick**.

























TcmToolBar = class(TToolBar)

Свойства и методы	Описание
<b>property</b> DataSet: TcmDataSet	Связанный с панелью набор данных.
<b>property</b> PrintGrid: TcmPrintGridEh	Компонент для печати набора данных.
<b>property</b> Query: TcmQuery	Компонент для выполнения запросов к базе данных.
<b>property</b> Doc: TcmDocEdit	Ссылка на объект редактирования документа.
<b>procedure</b> ButtonActions( <b>var</b> Key: Word; Shift: TShiftState);	Получает код клавиши. Определяет связанную с этим кодом кнопку панели управления и выполняет обработчик события <b>onClick</b> кнопки. Метод вызывается в обработчике <b>OnKeyDown</b> компонента <b>TDBGridEh</b> (сетка для отображения набора данных).
<b>procedure</b> SaveDocBtn	Сохраняет документ. При сохранении выполняется ряд дополнительных проверок, поэтому сохранение документа настоятельно рекомендуется выполнять вызовом данного метода, а не TcmEditDoc.Post.
<b>procedure</b> SaveNewDoc	Работает так-же как и SaveDocBtn, но если статус состояния документа (TcmDocEdit.RecState) равен smsNew, предварительно выводит запрос: "Сохранить документ ?"
<b>procedure</b> ProvDocBtn	Вызывает обработчик события определенный для кнопки btnProvDoc (проведение документа).
<b>property</b> OnOpenQuery: TNotifyEvent	Обработчик события вызываемого при открытии набора данных.
<b>property</b> OnClearFilter: TNotifyEvent	Обработчик события вызываемого при очистке условий фильтрации данных.
<b>property</b> OnStopProcess: TNotifyEvent	Обработчик события вызываемого по команде пользователя прервать формирование текущего набора данных.

<b>property</b> OnTitleStr: TcmTitleStr	Обработчик события возвращающий строку с параметрами формирования набора данных. Используется при формировании отчетов и экспорте данных в Excel.
<b>property</b> OnBeforeEditForm: TNotifyEvent	Событие вызывается перед выводом на экран формы редактирования записи. Событию передается ссылка на форму редактирования записи.
<b>property</b> OnChange: TNotifyEvent	Вызывается при внесении изменений в спецификацию документа.
<b>property</b> OnSaveDoc: TNotifyEvent	Событие при сохранении документа.
<b>property</b> OnEditValid: TNotifyEvent	Вызывается перед любым изменением документа. Если по каким-либо причинам редактирование документа запрещено, в процедуре обработчика должно быть сгенерировано исключение.

#### Список стандартных кнопок панели:



Картинка	Наименование	Горячая клавиша	Checked	Описание
	btnClose	<b>Alt-F4</b>		Закрыть окно
	btnNewRec	<b>Insert</b>		Новая запись
	btnEditRec	<b>Enter</b>		Редактировать запись
	tnDelRec	<b>Delete</b>		Пометить запись на удаление/Снять метку
	btnProvDoc	<b>F2</b>		Провести документ/Отменить проведение
	btnPrintDoc	<b>F5</b>		Печать документов
	btnHistory	<b>F7</b>		История редактирования
	btnQuery	<b>F3</b>		Запрос к базе данных
	btnPred	<b>Ctrl-L</b>		К предыдущей странице данных
	btnNext	<b>Ctrl-N</b>		К следующей странице данных
	btnFilter	<b>F4</b>	Да	Локальный фильтр
	btnNotFilter	<b>Alt-F3</b>		Очистить условия фильтрации данных
	btnShowProv	<b>Ctrl-R</b>	Да	Показывать проведенные
	btnShowDel	<b>Ctrl-D</b>	Да	Показывать помеченные на удаление
	btnShowOwner	<b>Ctrl-W</b>	Да	Показывать без учета владельца
	btnPreview	<b>F6</b>		Печать набора данных
	btnExportExcel	<b>Ctrl-E</b>		Экспорт данных в Excel
	btnGridTool	<b>Ctrl-C</b>		Настройка таблицы
	btnSaveDoc	<b>Ctrl-S</b>		Запись документа
	btnStopProcess			Прервать формирование отчета

	btnExpand		Раскрыть все узлы дерева
	btnCollapse		Свернуть дерево

### 9.4.1

### TcmToolBar

Стандартные кнопки панели TcmToolBar предназначены для управления связанным с панелью набором данных. Можно просто задать обработчики событий для стандартных кнопок, либо настроить свойства и события панели для работы с набором данных. При настройке панели следует указать следующие свойства и события:

- Набор данных указывается в свойстве DataSet.
- Компонент для просмотра и печати набора данных указывается в свойстве PrintGrid.
- Запрос для формирования набора данных указывается в свойстве Query. Указывается если набор используется для редактирования записей справочника или спецификации документа.
- OnOpenQuery - формирование набора данных
- OnClearFilter - очистка условий отбора данных. Вызывается при нажатии на стандартную кнопку .
- OnStopProcess - прервать формирование набора данных. Вызывается при нажатии на стандартную кнопку .
- OnTitleStr - возвращает список параметров используемых при формировании набора данных. Используется при печати набора и экспорте данных в Excel.
- У формы, на которой расположена панель управления, свойство KeyPreview должно быть равно true и в обработчике onKeyDown следует вызвать метод панели ButtonActions.

#### Просмотр истории редактирования.

Для автоматического вызова формы истории редактирования должны быть выполнены следующие условия:

- При описании справочников и документов в конфигураторе следует указать имя формы для просмотра истории в свойстве HistoryForm. Свойство определено для справочников (TcmSpr) и документов (TcmDoc).
- Указать набор данных (свойство DataSet)
- Указать запрос для формирования набора данных (свойство Query)
- Уникальный ключ набора данных (DataSet) должен содержать код записи справочника (при редактировании записей справочника), либо код документа (при редактировании спецификации документа), поле ключа набора данных должно называться 'Code'.
- Форме просмотра истории передается внешняя переменная Code: string, содержащая код записи справочника или код документа.
- Если форма просмотра истории должна выводиться в модальном режиме, то она должна возвращать внешнюю переменную IsModal: boolean равную true.

Пример обработчика события onButtonClick (реализация стандартного алгоритма):

```
procedure btnHistory1_OnClick(Sender: TObject);
var Form: TCoolForm;
begin
if cmDataSet1.ActiveRecord=nil then Exit; //если нет активной записи в наборе данных
Form:=Application.CreateCoolForm('CountryHistoryForm', true); //создаем форму просмотра истории
Form.Variables['Code']:=cmDataSet1.ActiveRecordKey; //передаем форме код записи
Form.Show; //показываем форму
end;
```


#### Редактирование записи справочника.

Для режима автоматического редактирования текущей записи справочника должны быть выполнены следующие условия:

- Для панели должны быть установлены значения свойств DataSet и Query.
- Поле ключевой записи набора данных должно называться 'Code' и содержать код записи

справочника.

- В конфигураторе следует указать имя формы используемой для редактирования записи справочника. Имя формы редактирования записи задается свойством TcmSpr.EditForm
- Форма редактирования записи должна содержать обработчики событий onCoolSetValue и onCoolGetValue. В обработчиках описывается переменная 'Code', для доступа к коду выбранной записи справочника.

Пример обработчика события onClick для кнопки редактирования записи  (реализация стандартного алгоритма):

```
procedure btnEditRec1_OnClick(Sender: TObject);
var Form: TCoolForm;
begin
  //если нет записей - выход
  if cmDataSet1.ActiveRecord=nil then Exit;
  //создаем форму для редактирования записи справочника
  Form:=Application.CreateCoolForm('CountryEditForm',true);
  try
    Form.Variables['Code']:=cmDataSet1.ActiveRecordKey; //код текущей записи
    if Form.ShowModal=mrOk then MakeDataSet; //редактируем запись
  finally
    Form.Free;
  end;
end;
```

### Редактирование спецификации документа.

Для режима автоматического редактирования текущей записи справочника должны быть выполнены следующие условия:

- Для панели должны быть установлены значения свойств DataSet, Query и Doc.
- Поле ключевой записи набора данных должно называться 'Code' и содержать код строки документа.
- В конфигураторе следует указать имя формы используемой для редактирования записи справочника. Имя формы редактирования записи задается свойством TcmDoc.EditForm
- Форма редактирования записи должна содержать обработчики событий onCoolSetValue и onCoolGetValue. Форме передается три параметра: Code - код строки документа, Docum - ссылка на объект редактирования документа (св-во Doc), DocForm - ссылка на форму документа (св-во Owner).

## 9.5 TcmObjectEdit

Визуальный компонент **TcmObjectEdit** используется для редактирования значений реквизитов типа "справочник" или "таблица".



TcmObjectEdit = class(TTryBtnEdit)

Свойства и методы	Описание
<b>property</b> AutoNext: boolean	Если равно true, то после выбора записи(ей) фокус ввода автоматически перемещается к следующему визуальному компоненту формы.
<b>property</b> CheckMode: boolean	Определяет режим работы компонента. Если равно false, компонент используется для выбора одной записи (по умолчанию), иначе можно выбрать несколько записей.
<b>property</b> EditMode: boolean	Если равно true (значение по умолчанию), разрешает вывод диалога редактирования записи по клавише <F3>.
<b>property</b> Code: string	Код записи справочника или сводной таблицы.



<b>property</b> BufData: TcmBufData	Компонент для доступа к записям справочников и сводных таблиц.
<b>property</b> BoxType: TcmObjectBoxType	Вид редактируемого реквизита: <ul style="list-style-type: none"> <li>• TcmSprBox - справочник</li> <li>• TcmTableBox - сводная таблица</li> </ul>
<b>property</b> MetaObject: TcmUnitComponent	Объект метаданных реквизита (TcmSpr, TcmTable).
<b>property</b> onEditRecord: TNotifyEvent	Обработчик события вызываемого при редактировании реквизитов текущей записи.
<b>property</b> onBeforeShowForm: TNotifyEvent	Вызывается перед выводом на экран формы списка. Событию передается ссылка на форму списка. С помощью данного события можно передать форме дополнительные параметры используя свойство Variables.

Когда свойство Code не содержит данных, возможен ввод с клавиатуры в поле редактирования компонента. Если значение Code указано, поле ввода доступно только для чтения и в нем отражается наименование записи справочника или код записи сводной таблицы. Возможны следующие варианты выбора нужной записи:

- Ввести код записи и нажать клавишу **<F2>**, будет произведен поиск записи с указанным кодом в базе данных. Если записи с указанным кодом нет, выводится соответствующее сообщение. Для работы в этом режиме должно быть указано значение свойства **BufData**.
- Ввести шаблон поиска по наименованию записи и нажать клавишу **<Enter>**. Будет выведено диалоговое окно для выбора записи. В диалоге показываются только записи удовлетворяющие введенному шаблону.
- Для сброса выбранного значения следует нажать клавишу **BackSpace**.
- **<F3>** - выводит диалог для редактирования реквизитов текущей записи, если установлено значение свойства Code, иначе диалог для создания новой записи.

## 9.5.1

### TcmObjectEdit

#### Выбор записи справочника.

Для выбора записи справочника из связанного списка, следует перекрыть обработчик события onButtonClick.

Если обработчик для onButtonClick не задан, компонент попытается получить значение используя "стандартный" алгоритм выбора записи. Для корректной работы "стандартного" алгоритма должны быть выполнены следующие условия:

- указано значение свойства MetaObject - ссылка на метаобъект справочника в конфигураторе.
- в конфигураторе, для справочника, следует указать имя формы используемой для выбора записи. Для справочника (класс TcmSpr) определены две формы выбора: ShowForm - используется при выборе одной записи справочника, CheckForm - используется при выборе нескольких записей справочника.
- Свойство CheckMode определяет режим выбора записей справочника. Если равно false - выбирается одна запись, иначе несколько. Соответственно, при CheckMode=false используется TcmSpr.ShowForm, иначе TcmSpr.CheckForm.
- Форма для выбора записи должна содержать обработчик события onCoolSetValue. Обработчик получает переменную класса TcmObjectEdit, переменная должна называться 'Edit'.

Пример обработчика события onButtonClick (реализация стандартного алгоритма):

```

procedure Place_onButtonClick(Sender: TObject);
var Form: TCoolForm;
begin
  Form:=Application.CreateCoolForm('PlaceShowList',true);
  try
    Form.SetRectOwner(Place);
    if Place.Code<>" then Form.Variable['Code']:=Place.Code;
    if Form.ShowModal=mrOk then Place.Code:=Form.Variable['Code'];
  finally

```

```
Form.Free;
end;
end;
```

### Редактирование записи справочника.

Для редактирования текущей записи справочника или создания новой следует перекрыть обработчик события onEditRecord. Если обработчик для onEditRecord не задан, используется "стандартный" алгоритм редактирования записи. Для корректной работы "стандартного" алгоритма должны быть выполнены следующие условия:

- указано значение свойства MetaObject - ссылка на метаобъект справочника в конфигураторе.
- в конфигураторе следует указать имя формы используемой для редактирования записи справочника. Имя формы редактирования записи задается свойством TcmSpr.EditForm
- форма редактирования записи должна содержать обработчики событий onCoolSetValue и onCoolGetValue. В обработчиках описывается переменная 'Code', для доступа к коду выбранной записи справочника.

Пример обработчика события onEditButton (реализация стандартного алгоритма):

```
procedure PlaceList_OnButtonClick(Sender: TObject);
var Form: TCoolForm;
begin
Form:=Application.CreateCoolForm('PlaceShowForm',false);
try
Form.SetRectOwner(TcmObjectBox(Sender));
Form.Variables['Edit']:=TcmObjectEdit(Sender);
Form.ShowModal;
finally
Form.Free;
end;
end;
```

## 9.6 TcmEnumEdit

Визуальный компонент **TcmEnumEdit** предназначен для редактирования перечислений. Список перечислений задается в конфигураторе и описывается объектами класса TcmEnum.



TcmEnumEdit = class(TComboBox)

Свойства и методы	Описание
property Enum: TcmEnum	Указатель на объект перечисления.
property Value: integer	Текущее значение перечисления.
property UseEmptyValue: boolean	Если равно true, то в список со значениями перечисления добавляется пустая строка, при выборе её свойство Value возвращает -1.

## 9.7 TcmImageList

Компонент **TcmImageList** наследуется от **TImageList** и определяет несколько дополнительных свойств и методов позволяющих получать ссылки на объекты TBitMap не только по порядковому номеру, но и по связанному с картинкой строковому идентификатору.

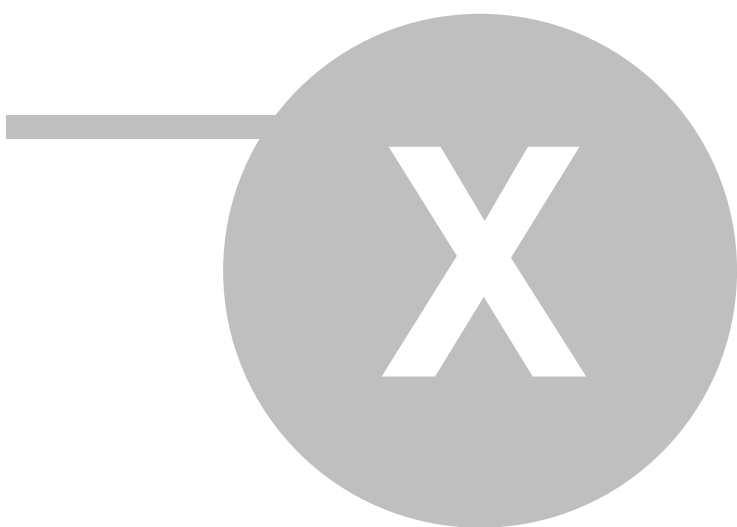


TcmImageList = class(TImageList)

Свойства и методы	Описание
-------------------	----------






<b>property</b> ItemList: TStrings	Список идентификаторов и связанных с ними индексов изображений. Список содержит информацию в формате <Identificator> = <Index>, где: Identificator - строковый идентификатор, Index - порядковый номер картинки связанной с идентификатором.
<b>procedure</b> AddImageByName(BitMap: TBitMap; BitMapName: <b>string</b> )	Добавляет изображение BitMap в список. В свойстве ItemList делается соответствующая запись связывающая идентификатор BitMapName с добавленной картинкой.
<b>procedure</b> GetImageByName(BitMap: TBitMap; BitMapName: <b>string</b> )	Возвращает картинку связанную с идентификатором BitMapName. Если картинка найдена, она возвращается в параметре BitMap. Загрузка картинки в объект BitMap производится вызовом метода Assign.



## 10

Для создания форм первичных документов (счетов-фактур, накладных и т.д.) используется библиотека FreeReport от компании FastReport ([www.fast-report.com](http://www.fast-report.com)). В стандартную поставку системы CoolManager библиотека FreeReport включена в исходных текстах. Модули библиотеки расположены в каталоге \source\FreeReport. В том же каталоге находится файл FR\_RUS.doc - документация с описанием компонентов библиотеки и дизайнера отчетов. На основании компонент библиотеки FreeReport создано три невидимых компонента, которые расположены в палитре компонентов на странице CoolMan:

	Компонент	наследован от	Описание
	TcmReport	TfrReport	Основной компонент - генератор отчета. В design-time двойной щелчок на компоненте открывает дизайнер.
	TcmCompositeReport	TfrCompositeReport	Композитный (составной) отчет. Предназначен для «склеивания» нескольких отчетов в один. Для этого программно заполняется список Reports ссылками на нужные объекты TfrReport.
	TcmUserDataSet	TfrUserDataSet	Источник данных, ориентированный на использование совместно с компонентом TcmDataSet. У стандартного компонента TfrUserDataSet, для навигации по набору данных, следует перекрыть события OnFirst, OnNext, OnCheckEOF. TcmUserDataSet перекрывает эти обработчики, поэтому, для работы с набором данных достаточно определить свойство OwnerRecord.

Как и все прочие классы и компоненты включенные в библиотеку CoolLib, вышеперечисленные компоненты предназначены для того, что бы максимально облегчить и автоматизировать процесс создания конфигураций в ИСП CoolManager. Ниже приводится описание свойств и методов добавленных к базовым компонентам библиотеки FreeReport. Для информации о стандартных свойствах и методах данных компонент обратитесь к файлу документации библиотеки - FR\_RUS.doc.

### 10.1 TcmReport



TcmReport = class(TfrReport)

Основной компонент - генератор отчета. В design-time двойной щелчок на компоненте открывает дизайнер.

Свойства и методы	Описание
<b>property</b> ReportName: string	Имя отчета. Задается без указания пути к файлу отчета, так же можно отпустить расширение. Например, если отчет расположен в файле: 'c:\config\report\test.frp', то свойству ReportName присвоить значение 'test.frp' или просто - 'test'. Если в параметрах конфигурации указан правильный путь к каталогу в котором располагаются шаблоны отчетов, файл будет найден. В design-time при вызове редактора компонента, открывается дизайнер и сразу же загружается файл указанный, имя которого указано в свойстве ReportName.
<b>property</b> Vars[AName: string]: string	Доступ к переменным отчета по имени. См. описание свойства Variables компонента TfrReport.
<b>procedure</b> LoadReport(AReportName: string = "")	Загружает шаблон отчета из файла с именем AReportName, если имя файла не указывается, производится поиск шаблона указанного в свойстве ReportName.

<b>procedure</b> ShowReport	Формирует отчет и выводит на экран.
<b>procedure</b> DesignReport	Выводит дизайнер отчета. Используется в целях отладки.
<b>procedure</b> PrintReport(PageNumbers: <b>string</b> ; Copies: integer)	Отправляет отчет на печать без предварительного просмотра. Для вывода используется принтер установленный по умолчанию. <ul style="list-style-type: none"> <li>• PageNumbers - список страниц для вывода (см. FR_RUS.doc)</li> <li>• Copies - число копий отчета.</li> </ul>
<b>procedure</b> AddDataSet(BandName: <b>string</b> ; DataSet: TcmUserDataSet)	Добавляет к отчету пользовательский набор данных <ul style="list-style-type: none"> <li>• BandName - имя бэнда отчета, с которым связан набор данных.</li> <li>• DataSet - набор данных.</li> </ul>

## 10.2 TcmCompositeReport



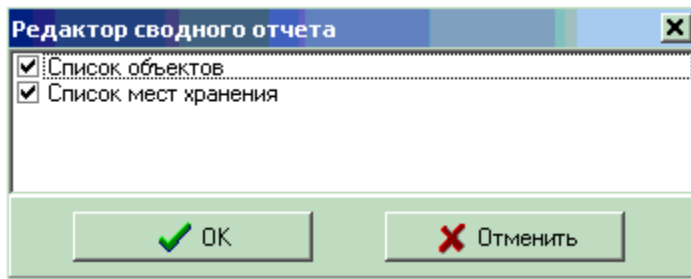
TcmCompositeReport = **class**(TfrCompositeReport)

Композитный (составной) отчет. Предназначен для «склеивания» нескольких отчетов в один. При использовании компонента TfrCompositeReport для этого следует программно заполнить свойство Reports ссылками на нужные объекты. Для TcmCompositeReport разработан редактор компонента, который позволяет создавать список отчетов в Design time. Для выбора и формирования отчетов в run time предназначен метод Execute.

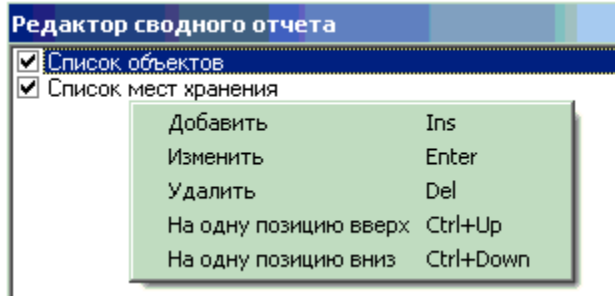
Свойства и методы	Описание
<b>property</b> Items[Index: integer]: TcmReport	Индексированное свойство для доступа к списку отчетов присоединенных к компоненту. Параметр Index должен находиться в диапазоне от 0 до ReportCount-1.
<b>property</b> Checked[Index: integer]: boolean	Каждый отчет связанный с компонентом имеет "флаг активности" (тип boolean). Если флаг активен, отчет включается в общий список при выводе на печать, иначе будет пропущен. Параметр Index должен находиться в диапазоне от 0 до ReportCount-1.
<b>function</b> ReportCount: integer	Возвращает число отчетов подключенных к компоненту.
<b>procedure</b> CheckAll(AChecked: boolean)	Устанавливает "флаги активности" всех отчетов равными значению переданному в параметре AChecked.
<b>procedure</b> Execute	Выводит на экран диалоговое окно, которое позволяет выбрать отчеты включаемые в композитный отчет.
<b>property</b> Copies: integer	Число копий отчета формируемых при выводе на печать.
<b>property</b> ReportList	Список отчетов подключенных к компоненту. Настраивается при помощи редактора компонента. Программного доступа к данному свойству нет.
<b>property</b> onBeforeExecute: TNotifyEvent	Событие вызываемое в методе Execute перед выводом диалогового окна на экран.

### **Редактор компонента.**

Редактор позволяет визуально создать и настроить список отчетов подключенных к компоненту:

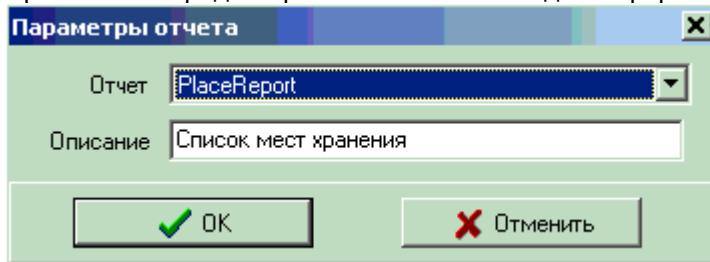


Со списком отчетов связано Popup menu:



Горячая клавиша	Команда
<b>Insert</b>	Добавить отчет в список.
<b>Enter</b>	Редактировать текущую запись в списке.
<b>Delete</b>	Удалить текущую запись.
<b>Ctrl-Up</b>	Переместить текущий отчет в списке на одну позицию вверх.
<b>Ctrl-Down</b>	Переместить текущий отчет в списке на одну позицию вниз.

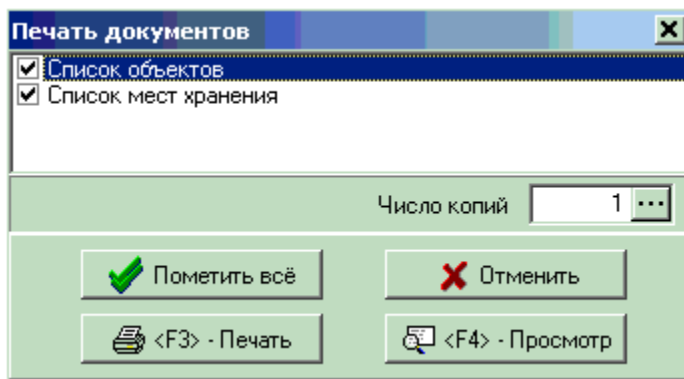
При вставке и редактировании отчета выводится форма для выбора отчета:



- **Отчет** - список компонентов TcmReport расположенных на той же форме что и компонент композитного отчета.
- **Описание** - описание отчета (св-во TfrReport.Title).

### **Формирование отчетов в Run-time.**

При выполнении конфигурации, для настройки и печати композитного отчета можно воспользоваться методом **Execute**. При этом на экран будет выведена диалоговое окно для выбора списка отчетов:



В списке галочками следует пометить отчеты, которые нужно включить в комPOSITE отчет. Кнопка "Пометить всё" позволяет снять/установить пометки на все отчеты в списке.

- <F3> - формирует комPOSITE отчет и выводит его на печать без предварительного просмотра.
- <F4> - формирует комPOSITE отчет и выводит его на экран.

### 10.3

## TcmUserDataSet



TcmUserDataSet = class(TfrUserDataSet)

Источник данных, ориентированный на использование совместно с компонентом TcmDataSet. У стандартного компонента TfrUserDataSet, для навигации по набору данных, следует перекрыть события OnFirst, OnNext, OnCheckEOF. **TcmUserDataSet** перекрывает эти обработчики, поэтому, для работы с набором данных достаточно определить свойство OwnerRecord.

Свойства и методы	Описание
<b>property</b> UseLevel: boolean	Если равно false просматриваются только записи текущего уровня (все записи подчиненные OwnerRecord). Иначе просматриваются записи всех уровней (последовательный обход дерева).
<b>property</b> OwnerRecord: TcmRecord	Владелец списка записей. При работе с TcmDataSet можно воспользоваться свойством TcmDataSet.Root
<b>property</b> ActiveRecord: TcmRecord	Ссылка на текущую запись набора данных.
<b>property</b> onGetRecord: TNotifyEvent	Событие вызывается при перемещении курсора на новую запись набора данных. Используется вместо OnFirst и OnNext компонента TfrReport. Ссылка на текущую запись набора данных находится в свойстве ActiveRecord.

### 10.4

Порядок создания отчета рассмотрим на "типовом" примере. Предположим, мы формируем накладную отпуска товара. Такой отчет состоит из двух частей - заголовка и табличной части со спецификацией документа.

Для простоты предположим что заголовок и табличная части документа содержат следующие поля:

#### Заголовок

- DocNomer - номер документа.
- DocDate - дата выписки документа.
- Kontragent - покупатель товара.

#### Табличная часть

- Code - код товара
- Name - Наименование товара
- Quantity - Количество товара

Для создания отчета следует выполнить следующие действия:

- Расположить на форме компонент TcmReport (свойство Name:='Report').
- Дважды щелкнув по компоненту вызвать дизайнер отчета
- Создать шаблон отчета
- Создать список переменных отчета (перечислены выше)
- Сохранить шаблон отчета в файле на диске.
- В свойстве отчета ReportName указать имя файла шаблона без пути к нему.
- Расположить на форме компонент TcmDataSet (свойство Name:='DataSet')
- Сформировать список записей набора данных (TcmDataSet)
- Расположить на форме компонент TcmUserDataSet (свойство Name:='UserDataSet')
- Написать обработчик события Report.onBeginDoc:

```
procedure Report_OnBeginDoc;
begin
  Report.LoadReport; //загрузить шаблон отчета
  UserDataSet.OwnerRecord:=DataSet.Root; //ссылка на главную запись набора данных
  Report.AddDataSet('DataSet', UserDataSet); //подключаем набор данных к отчету
  //установка значений переменных заголовка отчета
  Report.Vars['DocNomer']:= 'pacx0034';
  Report.Vars['DocDate']:= '12.06.2006';
  Report.Vars['Kontragent']:= 'ООО Перспектива';
end;
```

- Сформировать отчет и вывести для просмотра:

```
Report.ShowReport
```

Если переменные табличной части отчета имеют те-же имена, что и соответствующие поля набора данных (TcmDataSet) больше делать ничего не нужно, иначе следует перекрыть событие UserDataSet.onGetRecord:

```
procedure DataSet_onGetRecord(Sender: TObject);
var Rec: TcmRecord;
begin
  Rec:=TcmUserDataSet(Sender).ActiveRecord; //получаем ссылку на текущую запись отчета
  if Rec<>nil then
    Report.Vars['Name']:=Rec.Recvizits.ByIndex[1].AsString;
end;
```

В приведенном примере мы сперва получаем ссылку на активную запись набора данных, затем переменной отчета с именем Name присваиваем значение реквизита с именем RecordName.





## 11

## 11.1 TcmClosePeriod

Компонент **TcmClosePeriod** предназначен для расчета итогов сводных таблиц за отчетный период (календарный год или несколько лет). Одновременно с расчетом итогов производится закрытие периода, все документы входящие в закрытый период не подлежат изменению. Для каждой сводной таблицы (если, конечно, это не таблица оборотов), в базе документов существует таблица оборотов по месяцам. Таблица оборотов включает список измерений сводной таблицы и список ресурсов. При записи данных в сводную таблицу, автоматически обновляется и таблица оборотов, ресурсы суммируются в разрезе месяцев (каждая таблица оборотов содержит поле M: smallint, в котором хранится номер календарного месяца, от 1..12). Таблицы оборотов используются для оптимизации расчета значений ресурсов сводной таблицы на определенную дату или оборотов за указанный период. Предположим, существует сводная таблица MoveTovar, содержащая информацию о движении товара в разрезе номенклатурных кодов и мест хранения. Подобная таблица будет иметь два измерения:

- **Articul** - Номенклатурный код товара
  - **Place** - Место хранения товара
- и одно поле ресурса:
- **Quantity** - количество товара

Предположим так-же, что наша база данных содержит два открытых периода за 2005 и 2006 год, информация храняемая в сводной таблице MoveTovar имеет следующий вид:

**MoveTable (2005 год)**

Data	Articul	Place	Quantity	Примечание
01.05.2005	1	1	100	Оприходован товар в количестве 100 единиц
05.06.2005	1	1	-20	Перемещено 20 единиц товара из места хранения № 1 в место хранения № 2.
05.06.2005	1	2	20	
10.06.2005	1	2	-10	Реализован товар в количестве 10 единиц

При записи данных в таблицу MoveTable автоматически происходит обновление данных в таблице MoveTable\_M (обороты товара по месяцам), данные внесенные в эту таблицу будут иметь следующий вид:

**MoveTable\_M (2005 год)**

M	Articul	Place	Quantity
5	1	1	100
6	1	1	-20
6	1	2	10

Обратите внимание на то, что две операции по 2-м месту хранения за 6-й месяц были просуммированы и записаны в таблицу оборотов одной строкой с общим итогом равным 10 единицам товара (20 ед - 10 ед. = 10 ед.). Теперь, если просуммировать обороты товара в разрезе номенклатурных кодов и мест хранения получим:

- код товара 1, место хранения 1 - остаток 80 единиц.
- код товара 1, место хранения 2 - остаток 10 единиц

Опишем движение товара в базе 2006 года:

Data	Articul	Place	Quantity	Примечание
01.07.2006	1	1	50	Оприходован товар в количестве 50 единиц

05.07.2006	1	1	-100	Перемещено 100 единиц товара из места хранения № 1 в место хранения № 2.
05.07.2006	1	2	100	
10.08.2006	1	2	-20	Реализован товар в количестве 20 единиц
12.08.2006	1	2	-5	Реализован товар в количестве 5 единиц

**MoveTable\_M (2006 год)**

M	Articul	Place	Quantity
7	1	1	-50
7	1	2	100
8	1	2	-25

Если теперь, с помощью компонента TcmQuery рассчитать остатки на 20.08.2006, данные из базы будут извлекаться в следующем порядке:

База данных	Таблица	Условие отбора данных
2006 год	MoveTovar	Data between '1.8.06' and '19.08.06'
2006 год	MoveTovar_M	M between 0 and 7
2005 год	MoveTovar_M	M between 0 and 12

Для нашего примера TcmQuery вернет следующий набор данных:

База	Таблица	Articul	Place	Quantity
2006	MoveTovar	1	2	-25
2006	MoveTovar_M	1	1	-50
2006	MoveTovar_M	1	2	100
2005	MoveTovar_M	1	1	80
2005	MoveTovar_M	1	2	10

Просуммировав полученные данные определяем остатки товара на 20.08.2006 (остатки рассчитываются на начало дня, т.е. обороты за 20.08.06 в расчет не включаются). Полученные итоги:

Articul	Place	Quantity
1	1	30
1	2	85

На первый взгляд может показаться, что алгоритм расчета итогов на определенную дату чрезмерно усложнен. Было бы достаточно просуммировать данные хранимые в таблице MoveTovar и получить тот же самый результат. Такой подход упрощает как алгоритм расчета, так и структуру таблиц хранимых в базе (таблица MoveTovar\_M в этом случае вообще не нужна). Так оно и есть, но зато расчет остатков с использованием таблиц оборотов более производителен. При интенсивном товародвижении, таблица MoveTovar может содержать сотни тысяч или даже миллионы записей. Таблицы оборотов, суммируя ресурсы в разрезе месяцев, содержат записей в несколько раз меньше, что безусловно позволяет добиться очень высокой производительности.

**Пример:** Возьмем типичный розничный магазин. В начале месяца была произведена поставка товара в количестве 200 единиц, в течении месяца товар был полностью реализован. При продаже товара было сформировано 75 кассовых чеков. Следовательно, что бы получить остаток на начало следующего месяца, в таблице MoveTovar следует просуммировать 76 записей, а в таблице MoveTovar\_M только две. Умножьте это на среднестатистический ассортимент розничного магазина (от 5 до 15 тыс. позиций) и на частоту поставок (2-3 поставки в месяц), теперь почувствуйте разницу.

Дополнительным механизмом, призванным увеличить производительность работы при расчете остатков, является операция закрытия отчетного периода. Поясним смысл этой операции на примере. Используем данные описанных выше таблиц. Предположим, мы хотим закрыть 2005 год. В этом случае, в таблицу MoveTovar\_M за 2006 год будут добавлены следующие записи:  
MoveTable\_M (2006 год)

M	Articul	Place	Quantity
0	1	1	80
0	1	2	10

Фактически, были просуммированы данные таблицы MoveTovar за 2005, затем полученные итоги были добавлены в таблицу MoveTovar за 2006 (у записей сформированных при закрытии периода поле M=0).

Теперь, если с помощью компонента TcmQuery рассчитывать остатки на любую дату 2006 года, достаточно просмотреть две таблицы:

- MoveTovar (2006 год)
- MoveTovar\_M (2006 год)

Обращаться к базам за предыдущие периоды (2005 год) не нужно, т.к. остатки на начало отчетного периода теперь хранятся в таблице MoveTovar\_M (2006), следовательно, будут учтены при расчетах.

Следует отметить, что при закрытии периода, все документы содержащиеся в базе 2005 года становятся недоступны для изменений (их нельзя редактировать, создавать, удалять, проводить и распродать). Т.к. в противном случае, любые изменения вносимые в базу 2005 года делают некорректными начальные остатки в базе 2006 года. Если все же требуется внести изменения в документы уже закрытого периода, период сперва следует открыть, внести изменения и затем вновь закрыть.

Для выполнения операций закрытия и открытия периода используется класс **TcmClosePeriod**, в палитре компонентов он расположен на закладке CoolMan Admin.



TcmClosePeriod= class(TComponent)

Свойства и методы	Описание
<b>property</b> Base: TcmBase	Ссылка на базу данных.
<b>procedure</b> ClosePeriod(ABaseCode, ABaseSeg, AYear: integer);	Закрытие периода. <ul style="list-style-type: none"> <li>• ABaseCode - код базы данных</li> <li>• ABaseSeg - код сегмента</li> <li>• AYear - календарный год закрываемого периода.</li> </ul>
<b>procedure</b> OpenPeriod(ABaseCode, ABaseSeg, AYear: integer);	Отмена закрытия периода. Параметры те же, что и у метода ClosePeriod.
<b>property</b> onBaseClosed: TNotifyEvent	Событие при обращении к базе данных. В качестве параметра передается ссылка на объект класса TcmBaseUnit - текущую базу данных.
<b>property</b> onTableClosed: TNotifyEvent	Событие при старте операции расчета остатков для сводной таблицы. В качестве параметра Sender передается ссылка на объект класса TcmTable

#### Примеры вызова метода ClosePeriod:

```
ClosePeriod(1,1,2005); //закрывает в первой базе, в первом сегменте период за 2005 год.
ClosePeriod(1,0,2005); //закрывает в первой базе период за 2005 год по всем сегментам.
ClosePeriod(0,0,2005); //закрывает период за 2005 год по всем базам и всем сегментам.
ClosePeriod(0,1,2005); //закрывает по всем базам период за 2005 год в первом сегменте.
```

**Примечание:** При закрытии периода за 2005 год в конфигураторе должна быть описана и создана база для 2006 года.

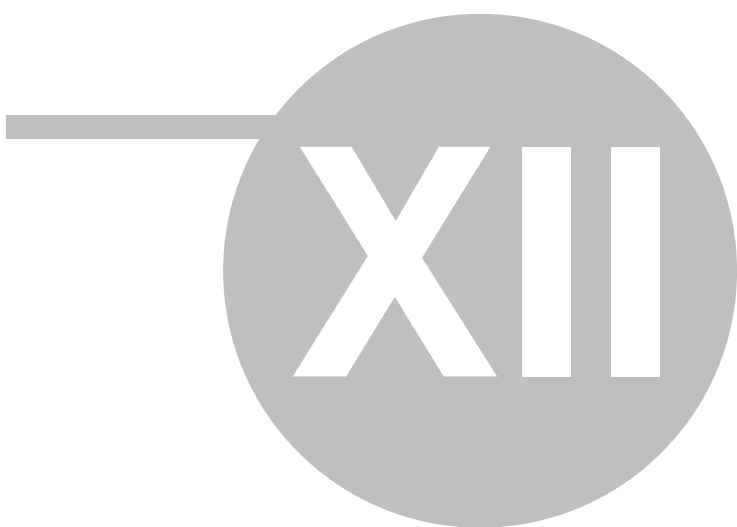
#### Пример обработчика события onBaseClosed:

```
procedure onBaseClosed(Sender: TObject);
var UBase: TcmBaseUnit;
    SBase: TcmBaseSeg;
    ABase: TcmBase;
begin
    UBase:=TcmBaseUnit(Sender);
    SBase:=TcmBaseSeg(UBase.Owner);
    ABase:=TcmBase(SBase.Owner);
    CoolMonitor.LogMsg('База:'+IntToStr(ABase.BaseCode)+
```

```
        'Сегмент:' + IntToStr(SBase.SegCode) +  
        'Год: ' + IntToStr(UBase.Year));  
end;
```

*Пример обработчика события onTableClosed:*

```
procedure cmClosePeriod_onTableClosed(Sender: TObject);  
var ATable: TcmTable;  
begin  
    ATable := TcmTable(Sender);  
    CoolMonitor.LogMsg(ATable.TableName + ' ' + ATable.Description);  
end;
```



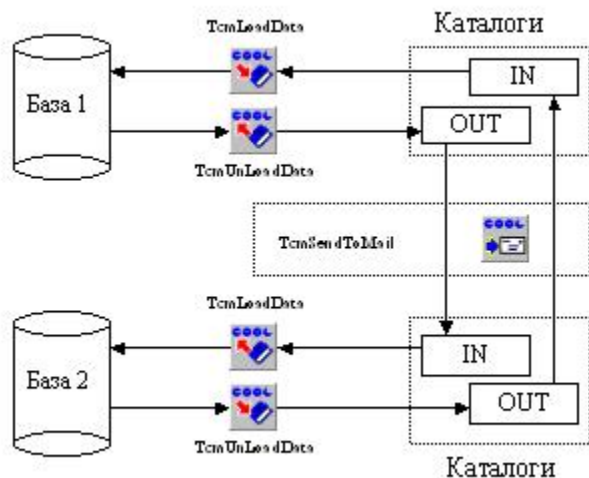
## 12

База данных системы Cool Manager состоит из набора реляционных баз, работающих под управлением СУБД InterBase (см. раздел Структура баз данных). В связи с этим крайне важным является вопрос межбазового обмена данными (репликация баз данных). Можно выделить два типа обмена данными:

- обмен записями справочников
- обмен документами

Каждый территориально удаленных объект предприятия имеет собственную копию базы справочников. На любом из объектов можно вносить изменения в справочники, поэтому при обмене данными справочников следует синхронизировать изменения сделанные на различных объектах. Т.е. каждая база со справочной информацией может как передавать внесенные в неё изменения, так и принимать изменения сделанные в других базах. При обмене документами ситуация выглядит несколько иначе, на каждом объекте существует "главная" база, которая содержит все документы созданные сотрудниками данного подразделения. Если требуется получить доступ к документам созданным в базах других объектов, создается зеркальная копия базы удаленного объекта. Информация хранимая в зеркальной копии доступна только для просмотра, т.е. в этом случае обмен информацией носит строго односторонний характер, от главной базы к её копиям расположенным на удаленных объектах предприятия.

Схема межбазового обмена данными в ИСП CoolManager:



На рисунке изображен процесс обмена данными между базами 1 и 2. Предполагается, что база 1 содержит зеркальную копию второй базы и наоборот. Обмен информацией носит дискретный характер, через равные промежутки времени формируются файлы, которые содержат информацию об изменениях внесенных в базу с момента последней выгрузки. Файлы имеют XML формат и называются пакетами. Периодичность формирования пакетов и их содержание определяется в конфигураторе ИСП. Компоненты конфигурации, отвечающие за обмен данными, будут подробно рассмотрены в следующем разделе.

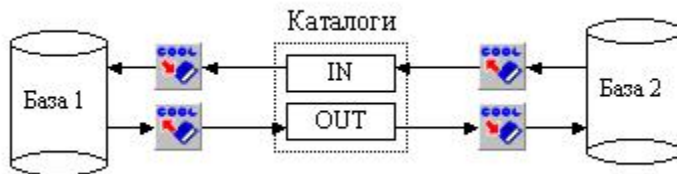
Палитра компонентов ИСП содержит закладку "CoolMan Admin" на которой расположены невидимые компоненты предназначенные для организации межбазового обмена данными:

- **TcmLoadData** - загрузка в базу полученных пакетов
- **TcmUnloadData** - формирование исходящих пакетов
- **TcmSendToMail** - отправка и прием пакетов средствами электронной почты.

Итак, компонент **TcmUnloadData** формирует исходящие пакеты, пакеты сохраняются в специально предназначенном для этого каталоге, на нашем рисунке каталог называется OUT. Пакеты содержат информацию об изменениях внесенных в справочники и документы текущей БД. Далее, с помощью компонента **TcmSendToMail** производится прием и отправка

электронных писем содержащих сформированные пакеты. При отправке писем используется SMTP сервер, при приеме POP3 сервер. Подробно конфигурирование серверов и почтовых ящиков будет описано в следующих разделах. При приеме писем компонент **TcmSendToMail** извлекает вложенные в них пакеты и помещает их в каталог IN. Компонент **TcmLoadData** производит загрузку полученных пакетов в БД. Следует отметить, что формируемые пакеты имеют сквозную нумерацию, которая позволяет соблюдать правильную последовательность загрузки пакетов.

В ряде случаев не требуется отправка пакетов по электронной почте, например когда сервера баз данных, между которыми производится обмен пакетами, находятся в одной локальной или VPN сети, либо если передача пакетов производится с помощью съемных flash носителей. В этом случае схема обмена пакетами будет иметь следующий вид:




В таком варианте обмена не используется компонент **TcmSendToMail**, а передача и прием пакетов осуществляется через одни и те же "общие" каталоги.

## 12.1


Порядок формирования, отправки, приема и загрузки пакетов задается в конфигураторе ИСР. Для этих целей существует ряд специализированных метаклассов. Корневым классом при описании процесса данными является **TcmInterData**:

 **TcmInterData** = class(TcmUnitComponent)


Метаклассу **TcmInterData** подчинены три метакласса:

 **TcmInterDataParams** = class(TcmUnitComponent)

Список шаблонов используемых при формировании пакетов.

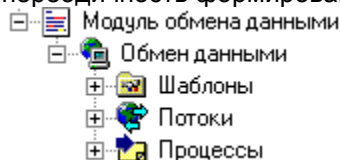
 **TcmInterDataList** = class(TcmUnitComponent)

Список потоков. Потоком называется последовательность пакетов, пересылаемых от одной базы к другой.

 **TcmOutProcess** = class(TcmUnitComponent)


Список процессов. Процессом называется процедура формирования исходящих пакетов.

Таким образом, в конфигураторе определяется содержание формируемых пакетов, порядок и периодичность формирования пакетов, а так-же способы межбазового обмена информацией:



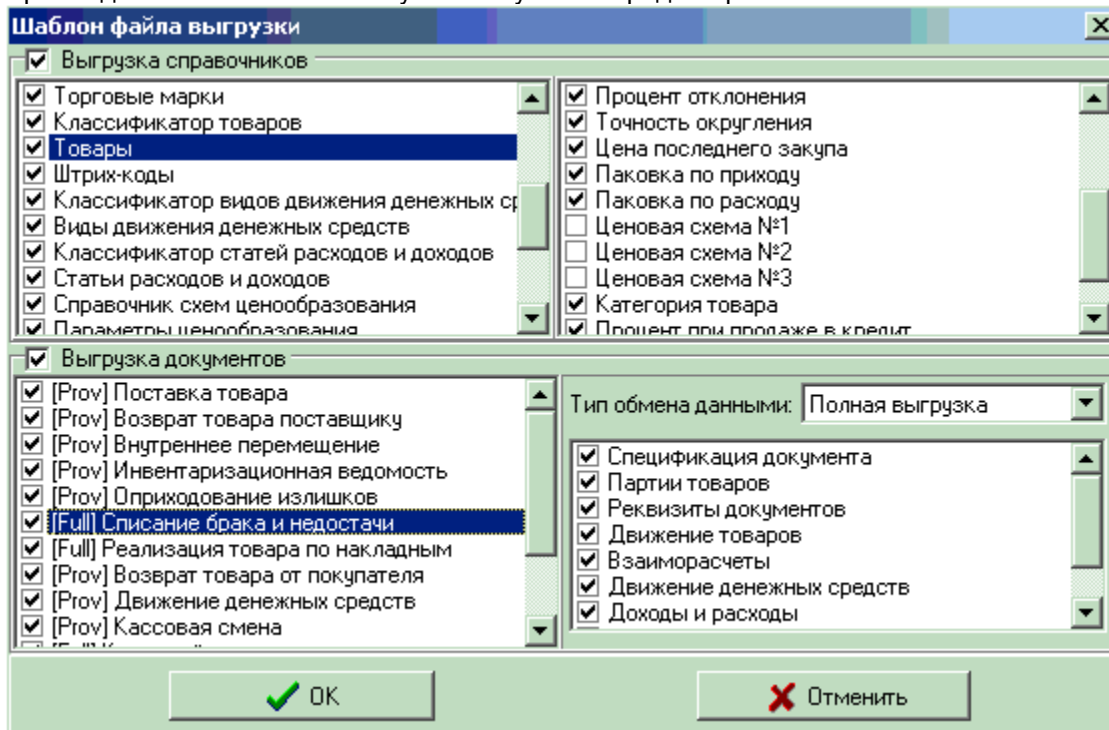
### 12.1.1 TcmInterDataParam

Шаблон используемый при формировании пакета. В конфигураторе подчинен метаклассу **TcmInterDataParams**.

 TcmInterDataParam = class(TcmUnitComponent)

Свойства и методы	Описание
property OutSpr: boolean	Признак выгрузки справочников.
property OutDoc: boolean	Признак выгрузки документов
property SprList: string	Список выгружаемых справочников
property DocList: string	Список выгружаемых документов.
property NotSpr: string	Список справочников которые не следует включать в пакет.
property Tables: string	Список выгружаемых сводных таблиц.

При создании шаблона используется визуальный редактор компонента:



В верхней части формы редактора расположен список справочников описанных в конфигурации и их реквизитов. Галочкой метятся справочники и реквизиты, которые следует включать в пакет. Под списком справочников расположен список документов конфигурации. Существует три варианта выгрузки документов:

Вариант выгрузки	Описание
Без движения регистров	В пакет включается заголовок и спецификация документа.
Полная выгрузка	В пакет включается не только документ, но и все записи, внесенные им в сводные таблицы при проведении.
Шаблон документа	В пакет включается заголовок и спецификация документа.

Различные варианты выгрузки влияют на то, каким образом будет загружаться документ в копию базы удаленного объекта:

- В режиме "Без движения регистров" создается новый документ, затем вызывается стандартная процедура проведения TcmDocEdit.RegisterDoc.
- В режиме "Полная выгрузка" из пакета загружается не только шапка и спецификация документа, но и все движения сводных таблиц.

Если процедура проведения документа носит детерминированный характер, т.е. документ всегда заносит одни и те-же записи в сводные таблицы, предпочтительнее использовать режим "Без движения регистров". Это позволяет значительно уменьшить количество данных



включаемых в пакет, а следовательно и его размер. Но при проведении ряда документов записи, вносимые в сводные таблицы, зависят от текущего состояния оперативных остатков сводных таблиц. Например, при проведении кассового чека записи вносимые в таблицу движения товаров зависят от того, какие товары и по каким партиям имеются на остатке в момент проведения документа. Следовательно, если изменятся оперативные остатки товаров изменится и информация внесенная в таблицу движения товаров. В стандартной конфигурации "Торговля" к таким документам относятся:

- Накладная реализации товара
- Кассовый чек
- Акт списания недостачи


Для того, что бы добиться идентичности записей сводных таблиц во всех копиях базы, используется режим "Полная выгрузка".

Режим "Шаблон документа" используется когда следует создать документ не в зеркальной копии базы, а в главной базе удаленного объекта. Такой режим работы используется, когда удаленные объекты работают в "режиме выписки", а все товародвижение ведется в базе Центрального офиса. Удаленные объекты посылают копии выписанных накладных или кассовых чеков. В Центральном офисе документы заносятся в главную базу. Для связи с базой удаленного объекта, код документа в удаленной базе, заносится в поле RecordOwner заголовка документа, созданного в главной базе.

Под полем редактирования режима выгрузки располагается список всех сводных таблиц конфигурации. Следует пометить таблицы, движения которых включаются в пакет. Используется только в режиме "Полная выгрузка", в этом случае можно задать список сводных таблиц движения которых следует включать в пакет.

## 12.1.2 TcmInterDataItem

Класс **TcmInterDataItem** предназначен для описания потоков данных. Поток называется последовательность пакетов, пересылаемых от одной базы к другой. В конфигураторе **TcmInterDataItem** подчинен классу TcmInterDataList.

 TcmInterDataItem = class(TcmUnitComponent)

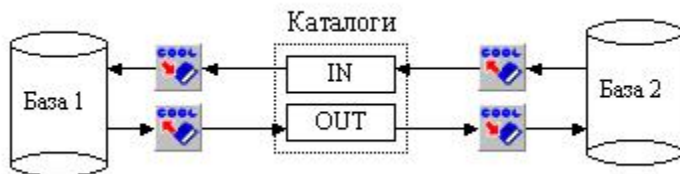
Свойства и методы	Описание
<b>property</b> DataDir: string	Каталог используемый для хранения исходящих и входящих пакетов потока.
<b>property</b> POP3: TcmPOP3Data	Параметры настройки POP3 сервера используемого для приема электронных писем с входящими пакетами.
<b>property</b> SMTP: TcmSMTPData	Параметры настройки SMTP сервера используемого для отправки электронных с исходящими пакетами.
<b>property</b> Mirror: boolean	Если равно true - включен режим зеркалирования каталогов (см. ниже)

В свойстве DataDir хранится имя каталога используемого для хранения пакетов. В каталоге автоматически создаются два подкаталога:

- \Out - для хранения исходящих пакетов.
- \In - для хранения входящих пакетов.

Обмен пакетами с удаленной базой производится посредством электронной почты. При отправке писем используется SMTP сервер, при приеме POP3 сервер.

При обмене пакетами между базами расположенными в одной локальной или VPN сети, обмен может производиться "напрямую" без пересылки пакетов по электронной почте:



т.е. обе базы используют одни и те же каталоги для доступа к пакетам. Обратите внимание, в этом случае для Базы №2 каталоги "зеркалируются", т.е. каталог IN используется для хранения исходящих пакетов, а в каталоге OUT находятся входящие пакеты. Для того, чтобы поток Базы №2 "понимал" то, что используемые им каталоги поменялись местами, свойство Mirror этого потока должно быть установлено равным true. По умолчанию Mirror всегда равно false.

TcmPOP3Data = class (TPersistent)

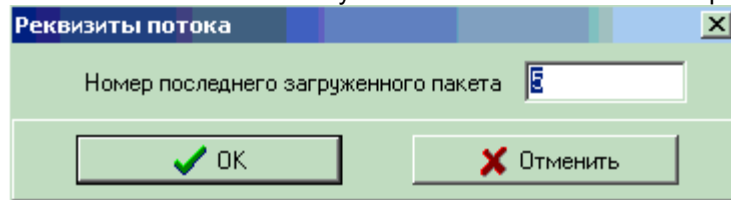
Свойства и методы	Описание
property Active: boolean	Если равно true, процесс приема электронных писем активирован.
property EMail: string	Почтовый ящик.
property Host: string	Имя POP3 сервера.
property Ident: string	Идентификатор писем с входящими пакетами. Заголовок полученного письма должен начинаться с указанной в этом свойстве последовательности символов. Если это не так, письмо классифицируется как спам и автоматически удаляется из почтового ящика.
property Password: string	Пароль для доступа к почтовому ящику.
property Port: integer	Номер порта POP3 сервера (стандартно 110).

TcmSMTPData = class (TPersistent)

Свойства и методы	Описание
property Active: boolean	Если равно true, процесс отправки электронных писем активирован.
property EMail: string	Электронный адрес получателя письма.
property EMailForm: string	Электронный адрес отправителя письма.
property Host: string	Имя SMTP сервера.
property Ident: string	Идентификатор писем с исходящими пакетами, указывается в заголовке письма. Должно совпадать со свойством POP3.Ident потока принимающего почту.
property Password: string	Пароль для доступа к SMTP серверу.
property Port: integer	Порт для доступа к SMTP серверу (стандартно 25).
property UserName: string	Имя пользователя для доступа к SMTP серверу.

*Примечание.* Если SMTP сервер требует авторизации, должны быть установлены значения свойств UserName и Password.


Создаваемые пакеты имеют сквозную нумерацию. Загрузка пакетов в базе-получателе производится в той же последовательности, в какой они были созданы. Двойной щелчок левой клавишей мыши по объекту класса TcmInterDataItem открывает редактор компонента:



В редакторе отображается номер последнего загруженного пакета, при необходимости можно изменить текущее значение.

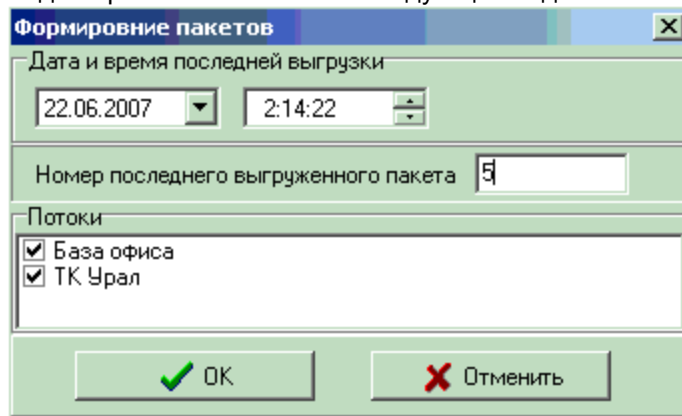
### 12.1.3 TcmOutProcessItem

Класс **TcmOutProcessItem** предназначен для описания процессов. Процессом называется процедура формирования исходящих пакетов. В конфигураторе **TcmOutProcessItem** подчинен классу TcmOutProcess.

 TcmOutProcessItem = class(TcmUnitComponent)

Свойства и методы	Описание
<b>property</b> ObjectList: string	Список потоков которые получают пакеты сформированные текущим процессом.
<b>property</b> Param: TcmInterDataParam	Шаблон на основании которого формируются исходящие пакеты.
<b>property</b> Period: integer	Периодичность формирования исходящих пакетов в минутах.

Редактор компонента имеет следующий вид:



- **Дата и время последней выгрузки** - дата и время формирования последнего пакета, в следующий пакет будут включаться изменения сделанные позднее указанного времени. Если требуется повторить выгрузку данных за какой либо период, достаточно указать здесь дату и время после которых следует выгружать сделанные изменения.
- **Номер последнего выгруженного пакета** - формируемые пакеты имеют сквозную нумерацию. В этом поле указывается номер последнего выгруженного пакета. При необходимости номер может быть изменен.
- **Потоки** - указываются потоки для которых сформирован исходящий пакет. Пакет сохраняется в каталоге \OUT всех помеченных потоков.

### 12.2 TcmSendData

Базовый класс для создания классов загрузки и выгрузки пакетов.

Свойства и методы	Описание
<b>property</b> InterData: TcmInterData	Ссылка на объект метаданных с описанием процесса обмена пакетами.

<b>function</b> Execute(ForceExecute: boolean=false): boolean	Запуск процесса формирования или загрузки пакетов. Обработываются все потоки описанные в конфигураторе. Если хоть один пакет был сформирован (загружен), метод возвращает true. Параметр ForceWrite влияет на порядок формирования исходящих пакетов. Если равен false - пакет формируется по истечении установленного периода (св-во TcmOutProcessItem.Period), если со времени формирования последнего пакета не прошло указанное время, формирование пакета будет отменено. При значении true пакет формируется в любом случае.
---	--

## 12.3 TcmLoadData

Компонент используется для загрузки файлов обмена. Для загрузки данных следует вызвать метод Execute.

**type**

TcmPostDocumEvent = **procedure**(ADoc: TObject; AShbName: **string**);

**ADoc** - ссылка на загруженный документ (TcmDocEdit)

**AShbName** - имя шаблона, в рамках которого производится загрузка пакета (TcmInterDataParam).



TcmLoadData = **class**(TcmSendData)

Свойства и методы	Описание
<b>property</b> AutoSkip: boolean	Загружаемые пакеты имеют сквозную нумерацию и загружаются в порядке возрастания номеров. Свойство AutoSkip определяет реакцию процесса загрузки на "потерянные" пакеты. Например, каталог загрузки содержит пакеты с номерами 12,13,15, пакет №14 был потерян в процессе обмена. Если AutoSkip=false будут загружены пакеты 12,13 затем процесс загрузки остановится ожидая пакет с 14 номером. Если AutoSkip=true, 14 пакет будет пропущен и сразу же начнется загрузка пакета с номером 15.
<b>property</b> onMessage: TcmMessage	Событие для обработки сообщений возвращаемых компонентом в момент выполнения метода Execute.
<b>property</b> onPostDocum: TcmPostDocumEvent	Событие вызываемое после успешной загрузки документа. Параметр Sender содержит ссылку на созданный документ.

## 12.4 TcmUnloadData

Компонент используется для формирования файлов обмена данными. Для формирования исходящих пакетов следует воспользоваться методом Execute.

**type**

TcmSendDocEvent = **function**(ADoc: TcmDocEdit; AShbName: **string**): boolean;

**ADoc** - ссылка на загруженный документ (TcmDocEdit)

**AShbName** - имя шаблона, в рамках которого производится загрузка пакета (TcmInterDataParam).



TcmUnloadData = **class**(TcmSendData)

Свойства и методы	Описание
-------------------	----------

<b>property</b> onMessage: TcmMessage	Событие для обработки сообщений возвращаемых компонентом в момент выполнения метода Execute.
<b>property</b> onIsSendDocum: TcmSendDocEvent	Вызывается перед записью документа в файл обмена данными. Если документ следует сохранять в файле выгрузки возвращает true.

## 12.5 TcmSendToMail

Компонент используется для приема и отправки пакетов средствами электронной почты. Для запуска процесса обмена данными следует вызвать метод Execute.



TcmSendToMail=(TcmSDProcessor)

Свойства и методы	Описание
<b>property</b> InterData: TcmInterData	Ссылка на объект метаданных с описанием процесса обмена пакетами.
<b>property</b> DialUpActive: boolean	Если равно true, доступ к InterNet устанавливается с помощью DialUp соединения.
<b>property</b> DialUpEntry: <b>string</b>	Имя используемого DialUp соединения.
<b>procedure</b> Execute(AMode: smallint= 0)	Запуск процесса обмена пакетами. Параметр AMode может принимать одно из следующих значений. 0 - загрузка входящих и отправка исходящих пакетов. 1 - загрузка входящих пакетов 2 - отправка исходящих пакетов
<b>property</b> onMessage: TcmMessage	Событие для обработки сообщений возвращаемых компонентом в момент выполнения метода Execute.