Объекты конфигурации

Общая информация

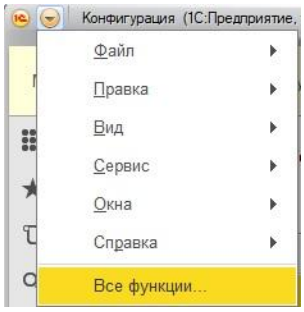
Получение информации о конфигурации

Через отчет по метаданным конфигурации (в режиме Конфигуратор меню Конфигурация – Отчет по конфигурации). Имена, показанные в отчете, соответствуют именам свойств и коллекций объектов.

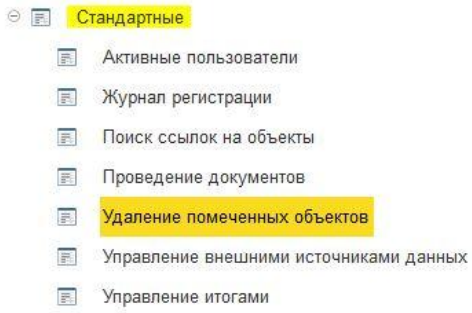
Ссылочное удаление (пометка удаления)

В конфигурации могут существовать связанные свойства у разных объектов. Предлагаемый алгоритм удаления:

- Объект помечается как удаляемый путем установки пометки на удаление (кнопка Del или контекстное меню). Все связанные объекты автоматически помечаются на удаление.
- Основное меню - Все функции



- Стандартные - Удаление помеченных объектов



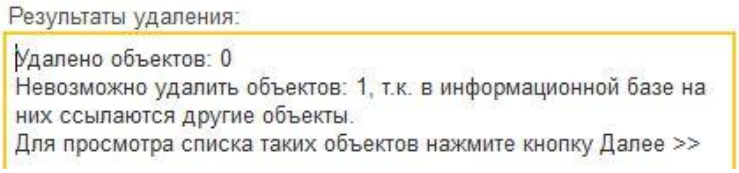
- Можно посмотреть удаляемые объекты выбрав Выборочное удаление.

- Можно выбрать удаляемые объекты и удалить.

Удаление элемента родительского справочника.

Есть элемент (Подчиненный) в подчиненном справочнике, у которого выставлен в качестве родителя удаляемый нами элемент (Основной).

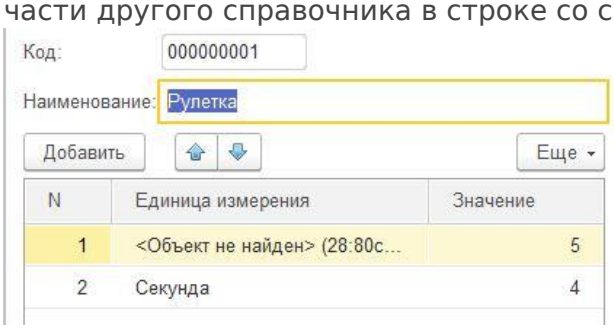
- В случае удаление через "Удалить" (Shift-Del) оба элемента тихо удалятся.
- В случае установки пометки можно увидеть все удаляемые элементы.
- При снятии пометки удаления с Основного, с Подчиненного тоже удаляется метка.
- Однако проверка ссылочной целостности происходит только в момент установки пометки. То есть если пометить на удаление Основной, затем создать второй Подчиненный, будет выдано предупреждение "Вы хотите установить в качестве владельца элемент с пометкой удаления", но пометка удаления на втором подчиненном элементе не установится. При попытке удаления через предлагаемую процедуру появится ошибка и объект не будет удален.



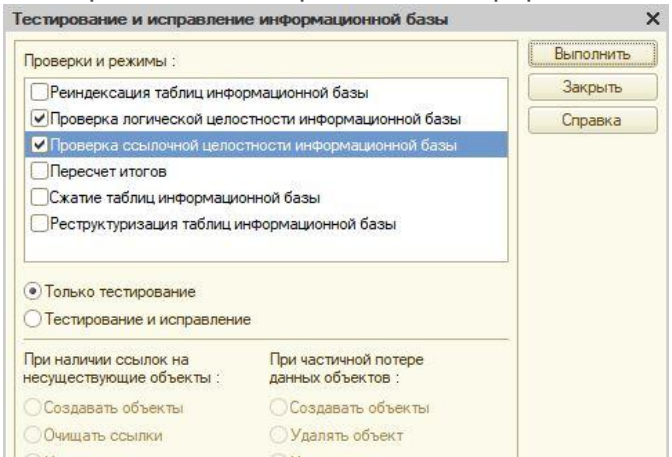
  Но при удалении через "Удалить" (Shift-Del) все элементы (и помеченный, и непомеченный) удалятся.
- Если при просмотре удаляемых объектов убрать пометку с подчиненного, оставив основной, то удаление не произойдет и будет ошибка из предыдущего пункта.

Удаление элемента справочника в случае наличия ссылки на него в табличной части.

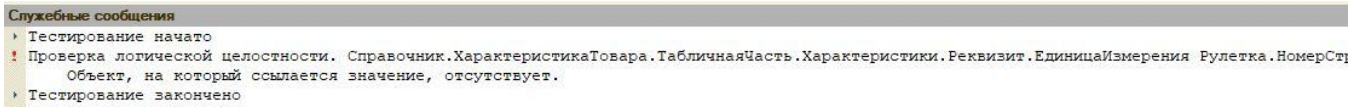
- При установке пометки на удаление, в табличной части другого справочника элемент не помечается на удаление. В списке элементов будет отражен один элемент, однако при попытке удаления будет выдано предупреждение о наличии ссылок и элемент не будет удален и можно будет посмотреть список элементов, в которых есть ссылки.
- При удалении через "Удалить" (Shift-Del) элемент будет удален, но в табличной части другого справочника в строке со ссылкой будет надпись "Объект не найден".



- Для исправления предыдущей ситуации можно использовать Администрирование - Тестирование и исправление информационной базы



В служебных сообщениях будет выведен список элементов с битыми ссылками.



Это считается медленной процедурой и требует монопольного доступа, можно использовать внешнюю обработку

Типы данных встроенных классов

Раздел относится к классам Справочник, Документ, ПланВидовХарактеристик, ПланСчетов и ПланВидовРасчета. Идея общая для всех классов, рассмотрен пример класса Справочник.
При создании объекта появляются следующие типы (так их называют в документации) - СправочникМенеджер, СправочникСсылка, СправочникОбъект, СправочникВыборка. Эти слова используются часто. Но есть тьма вопросов, основной из которых - нахрена было сделано данное разделение терминологий. Ответ: понять и простить.

Названия типов не связаны с их использованием в коде.

СправочникМенеджер

Название типа	Отображение в коде	Назначение
СправочникиМенеджер	Справочники	Содержит все СправочникМенеджер и метод ТипВсеСсылки(), позволяющий получить ссылки на типы всех справочников для последующего сравнения с типом переменной.
СправочникМенеджер	Справочники.ИмяСправочника	Это конкретный элемент СправочникиМенеджера. Понять и простить. текномenкл = Справочники.Номенклатура; //текномenкл это справочникменеджер. Общие действия, относящиеся к конкретному справочнику, а не к его конкретным объектам. Например, методы позволяют создать новый объект или найти объект по коду.

СправочникСсылка

Хранит ссылку, идентифицирующую запись в таблице справочника. Внутренний идентификатор, хранящийся в реквизите Ссылка таблицы справочника. Через

СправочникСсылка можно получить данные всего объекта через методы. При программной работе объект СправочникСсылка получают при поиске элемента справочника. Используется только для чтения данных. При получении ссылки на одну и ту же запись справочника разными способами, ссылки будут одинаковые (в отличие от объекта). Получение СправочникОбъект из ссылки:

```
ЗаписьНоменклатураСсылка = Справочники.Номенклатура.НайтиПоНаименованию("Ручка");
ЗаписьНоменклатураОбъект = ЗаписьНоменклатураСсылка.ПолучитьОбъект();
// Здесь идут какие-то изменения
ЗаписьНоменклатураОбъект.Записать();
```

При первом обращении к свойству происходит запрос к БД и данные кэшируются. При обращении или раз в 20 секунд происходит проверка неизменности данных. При внешнем изменении данных, свойство, к которому происходит обращение, обновляется. Объекты в кэше очищаются через 20 минут.

СправочникОбъект.

Используется для создания, изменения и удаления объекта, для отображения и редактирования всех данных элемента справочника в форме элемента. Так как содержит весь объект, тяжелее ссылки. Очень похоже на передачу данных по ссылке и по значению. При каждом запросе на получение объекта создается новый временный объект.

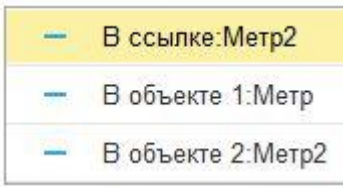
Объект может быть создан с помощью менеджера справочника. В этом случае создается новый объект, которого еще нет в базе данных. Если его записать, то появится новый объект в базе данных. Объект может быть получен из ссылки. В этом случае из базы данных считывается существующий объект, на который указывает ссылка. Объект также может быть получен из выборки (СправочникВыборка). В этом случае данные объекта заполняются данными, полученными в процессе считывания выборки.

СправочникОбъект оптимизирует запись изменений в базу данных, сохраняя только изменения. Обеспечивается оптимистическая блокировка - запись заблокируется в случае изменении в базе данных после считывания. Это обеспечивает логическую целостность изменения объектов.
Существует пессимистическая блокировка, запрещающая изменения до снятия блокировки. Включается методом Заблокировать(). В основном он предназначен для блокировки объектов, редактируемых в форме. Расширение формы элемента справочника автоматически включает блокировку, чтобы пользователь был уверен что, начав редактировать объект, он сможет его записать.

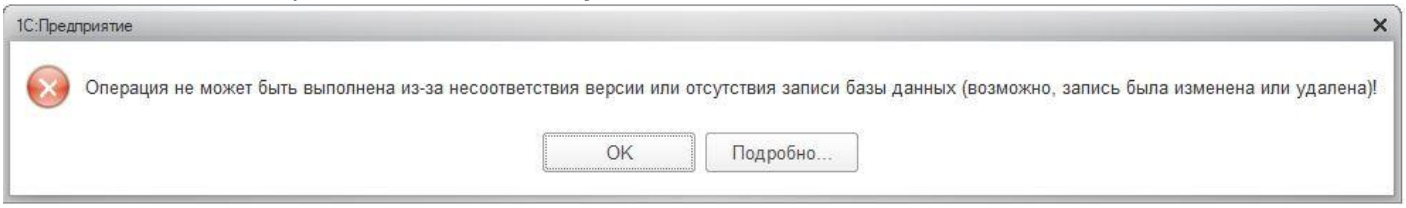
Предоставляется доступ через свойства к данным объекта. Значения СправочникОбъект заполняются при считывании существующего объекта значениями из базы данных. Проверка факта изменения в СправочникСсылка и отсутствия изменений в одном из объектов:

```
&НаСервереБезКонтекста
Процедура ПроверитьАктуальностьНаСервере()
	ЕдиницаСсылка = Справочники.ЕдиницыИзмерения.НайтиПоНаименованию("Метр");
	ЕдиницаОбъект1 = ЕдиницаСсылка.ПолучитьОбъект();
	ЕдиницаОбъект2 = ЕдиницаСсылка.ПолучитьОбъект();
	ЕдиницаОбъект2.Наименование = "Метр2";
	ЕдиницаОбъект2.Записать();
	Сообщить("В ссылке:" + ЕдиницаСсылка.Наименование);
	Сообщить("В объекте 1:" + ЕдиницаОбъект1.Наименование);
	Сообщить("В объекте 2:" +ЕдиницаОбъект2.Наименование);
	  //ЕдиницаОбъект1.Наименование = "Метр3";
	//ЕдиницаОбъект1.Записать();
КонецПроцедуры

&НаКлиенте
Процедура ПроверитьАктуальность(Команда)
	ПроверитьАктуальностьНаСервере();
КонецПроцедуры
```



Наименование в ЕдиницаОбъект1 действительно не изменилось. Если раскомментировать команды записи первого объекта, то будет ошибка:



Похоже, что обновление СправочникОбъект возможно только через повторное получение, поэтому в продуктовом решении минимум необходимо добавить обработку ошибок и, возможно, запись в цикле до успешного сохранения + понимание что же изменилось, почему, и нужно ли в принципе обновлять объект.

Получение ссылки из объекта:

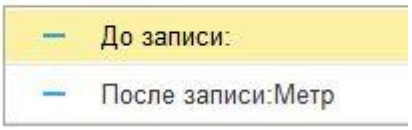
```
СсылкаНаОбъект = МойОбъект.Ссылка;
```

Поскольку реквизит ссылка - это параметр в базе данных, то до записи нового объекта в ней ничего нет. Проверка:

```
&НаКлиенте
Процедура ДобавитьЕдиницуИзмерения(Команда)
СоздатьЕдиницуИзмерения(Объект.НоваяЕдиницаИзмерения);
КонецПроцедуры

&НаСервереБезКонтекста
Процедура СоздатьЕдиницуИзмерения(НазваниеЕдиницы)
НовыйОбъект = Справочники.ЕдиницыИзмерения.СоздатьЭлемент();
НовыйОбъект.Наименование = НазваниеЕдиницы;
Сообщить("До записи:" + НовыйОбъект.Ссылка);
НовыйОбъект.Записать();
Сообщить("После записи:" + НовыйОбъект.Ссылка);
КонецПроцедуры
```

Сообщения:



СправочникВыборка

Предназначен для обхода всех элементов справочника. Считанные элементы могут быть получены в качестве значений типа СправочникОбъект, при этом они не будут повторно считываться из базы данных. Выборка считывает данные объектов целиком (все поля и все табличные части).

```
&НаСервереБезКонтекста
Процедура ВывестиВсеЕдиницыНаСервере()
Выборка = Справочники.ЕдиницыИзмерения.Выбрать();
Пока Выборка.Следующий() Цикл
Сообщить(Выборка.Наименование);
КонецЦикла;
КонецПроцедуры

&НаКлиенте
Процедура ВывестиВсеЕдиницы(Команда)
ВывестиВсеЕдиницыНаСервере();
КонецПроцедуры
```

СправочникСписок

Доп. информация

Предназначен для динамического просмотра данных справочника в элементе управления ТабличноеПоле (свойство Данные). Возможность считывать данные порциями. В том случае, если ни один из индексов таблицы не соответствует заданному пользователем упорядочиванию, механизм динамического просмотра не будет задействован. Может быть настроен состав считываемых полей (колонок), отбор и сортировка. Список осуществляет считывание данных порциями в процессе навигации пользователем в табличном поле. Обращение к реквизитам на чтение аналогично СправочникСсылка.

Регламентные задания

Разобраться, интересная статья

Программное создание конфигурации.

Хер. Внешний обработчик может только уведомить об отсутствии чего-либо в конфигурации, создать кодом не получится.

Тестирование

Через жопу. Статья о тестировании Используется уже другой (!) язык. Им видимо двух не хватало...

Git

Еще нужно разобраться, но вроде есть.

Класс Константы и Перечисления

Константы

Т к константы хранятся в базе данных, доступ к ним нужно получать через процедуру на сервере. В случае обращения к несуществующей константе будет сгенерировано исключение.

```
&НаКлиенте
Процедура ВыводКонстант(Команда)
Отобразить();
КонецПроцедуры

&НаСервере
Процедура Отобразить()
Сообщить(Константы.НазваниеОрганизации.Получить());
    Константы.НазваниеОрганизации.Установить("ООО");
КонецПроцедуры
```

Получение одним запросом нескольких констант, их перебор и сохранение.

```
&НаСервере
Процедура Отобразить()
Набор = Константы.СоздатьНабор("НазваниеОрганизации, ЕщеКонстанта");
Набор.Прочитать();
Набор.ЕщеКонстанта = "Сохраненная из кода";
Набор.Записать();
КонецПроцедуры
```

Можно сделать единую форму ввода констант в контекстном меню "Создать форму констант". Если после создания формы создана новая константа - зайти в реквизиты созданной формы и перетащить мышью новую константу на форму.

Можно настроить обработки "ОбработкаПроверкиЗаполнения", "ПередЗаписью", "ПриЗаписи". На каждой константе нужно щелкнуть правой кнопкой -> Открыть модуль менеджера значений. Через модуль менеджера. Пример ограничения значения константы

(должна быть больше 4), процедура определена в модуле менеджера значений константы Константа4.

```
Процедура ПередЗаписью(Отказ)
    Если Значение < 5 Тогда
        Сообщить("Значение константы не может быть меньше 4.");
        Отказ = Истина;
    КонецЕсли;
КонецПроцедуры
```

Перечисления

Для неизменяемых в процессе использования конфигурации значений. Вот им больше подходит термин "Константа". Задается в конфигураторе в разделе Данные. Обращения через объектный вариант напрямую по имени.

```
Перечисления.ВидыНоменклатуры.Материал
```

Класс Справочники

У всех справочников есть стандартные реквизиты: Код и Наименование. Справочник обновляется в момент обращения или изменения со стороны пользователя, при изменении другим пользователем или программным изменением автоматического обновления нет, данные кэшируются, поэтому нужно вручную обновить справочник.

Существует тип данных Составной реквизит, создается при помощи троеточия в поле настройки типа данных в свойстве реквизита.

Настройка внешнего вида

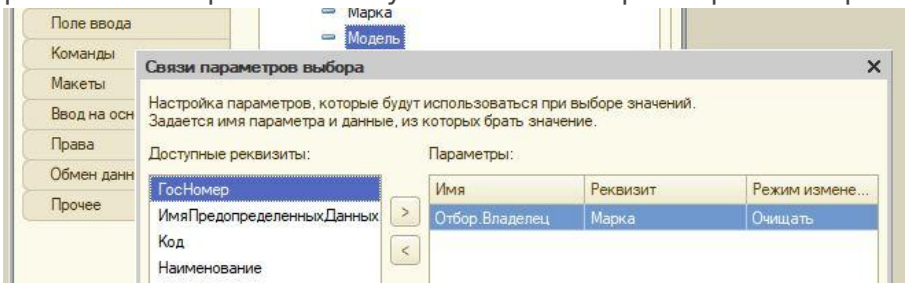
У реквизита есть свойство Использование. При создании иерархических справочников можно указать, для чего использовать этот реквизит. При использовании "Для элемента" этого реквизита не будет у группы.

Предопределенные элементы. ПКМ на справочнике -> Открыть предопределенные данные. Используется в частности для настройки значений по умолчанию в Документах, ... Для запрета удаления предопределённых элементов нужно выключить во всех ролях следующие права (по умолчанию выключены):

- «ИнтерактивноеУдалениеПредопределённыхДанных»
- «ИнтерактивнаяПометкаУдаленияПредопределённыхДанных»
- «ИнтерактивноеСнятиеПометкиУдаленияПредопределённыхДанных»
- «ИнтерактивноеУдалениеПомеченныхПредопределённыхДанных».

Подчиненные справочники. В свойствах справочника -> Владелец добавить справочник-владелец. Может быть несколько справочников-владельцев, но владелец конкретного элемента подчиненного справочника может быть один элемент одного из справочников-владельцев. Множественное владение не поддерживается. Детали удаления элемента родительского справочника

Связь параметров выбора. Предположим, необходимо хранить список наших автомобилей. Среди параметров в частности есть модель, марка, и т.д. (гос. номер, ...). У одной модели может быть несколько марок. Логично создать справочник Модель и подчиненный справочник Марка. Однако если просто в справочнике Автомобили в реквизитах указать ссылки на Модель и Марка, то при поиске марки будут отображаться все марки, а не только для выбранной модели. Для исправления этого поведения в свойствах реквизита Марка используется "Связи параметров выбора".



Табличные части. Используется для отражения информации, связанной с данным элементом, но не имеющей связанной объектной сущности. Например, для справочника "Товары" может быть создана табличная часть "ЕдиницыИзмерения".

Выбор между подчиненным справочником и табличной частью.

Это близкие элементы, поэтому нужно хорошо подумать перед выбором.

- При использовании табличной части нельзя ссылаться на строки табличной части, т.е. нельзя будет создать реквизит, соответствующий понятию «строка табличной части».
- Если в перспективе может возникнуть потребность ссылаться на подчиненные объекты, например, создавать ссылающиеся на них реквизиты документов или других справочников, то лучше сразу завести подчиненный справочник. Если же ссылка на такие сведения не имеет смысла и никогда не может быть типом какого-либо реквизита, тогда можно завести табличную часть.

Пример ситуаций и выбор между подчиненным справочником и табличной частью.

Ситуация 1 В базе данных необходимо хранить список расчетных счетов каждого контрагента. Почти наверняка в платежных документах будет необходимо, кроме контрагента, указывать его расчетный счет. Такая информация имеет объектную связь и может быть идентифицирована как "расчетный счет". Возможное решение: Справочник "Контрагенты" и подчиненный ему справочник "РасчетныеСчета" с полями "Банк", "Номер", "КоррСчет".

Ситуация 2 Для справочника "Номенклатура" нужно хранить список единиц измерения для каждого товара с указанием коэффициента пересчета в основную единицу измерения. Эти сведения подбираются из справочника "ЕдиницыИзмерения", который хранит все существующие в природе единицы измерений. Каждая строка такого подчиненного списка не имеет собственной объектной сущности, а нужна только для пересчета из одной единицы измерения в основную. Возможное решение: Справочник "Номенклатура" и табличная часть "ЕдиницыИзмерения" с полями "ЕдиницаИзмерения" и

"КоэффициентПересчета".
Ситуация 3 Допустим, встроенной системы задания прав пользователей не хватает для некоторых специальных приложений. Например, часто требуется разработать механизм утверждения документов разными пользователями. Тогда для сотрудника нужно хранить список документов, которые он может утверждать, и с этим отлично справится табличная часть "УтверждаемыеДокументы" справочника "Сотрудники". Такая информация не имеет объектной связи, а просто связывает документ и сотрудника, поэтому вряд ли понадобится когда-либо в будущем создавать ссылки на нее. Возможное решение: Справочник "Сотрудники" и табличная часть "УтверждаемыеДокументы" с полем "Документ" и флажком "Утверждается". Заметим, что эту задачу также можно решить с использование регистров сведений.
Ситуация 4 Для справочника "Сотрудники" требуется хранить сведения о составе семьи сотрудника, т.е. вносить информацию о членах семьи и их родственных отношениях к сотруднику (муж, жена, сын, дочь и т.д.). Обычно эта списковая информация полностью подчинена элементу справочника "Сотрудники", и возникает мысль о том, чтобы завести табличную часть. Но если подумать, то такая информация имеет четкую объектную природу и может быть идентифицирована как "член семьи". Для некоторых приложений может потребоваться создавать ссылки на членов семьи сотрудника. Аналогичная ситуация наблюдается со сведениям об образовании и о предыдущих местах работы сотрудника. Возможное решение: Выбор между подчиненным справочником и табличной частью зависит от назначения конфигурации. Если в будущем может возникнуть потребность создавать ссылки на такие сведения, то лучше завести подчиненные справочники "СоставСемьи", "Образование" и "ТрудоваяДеятельность".

Важное замечание. Необходимо помнить, что при обращении к элементу справочника он весь, вместе со всеми табличными частями, считывается из базы данных в память. Если табличная часть содержит достаточно большое количество строк это может заметно сказаться на производительности системы.

Таким образом, табличную часть стоит использовать, если не надо хранить ссылки на элементы, и количество элементов ограничено.

Программная работа с элементами справочника

Подборка примеров кода ссылка

Поиск по реквизиту. Код обработки, есть справочник Цвета.

```
&НаКлиенте
Процедура ПолучитьОбъектПоНаименованию(Команда)
	ТекстНаименования = НайтиОбъект(Объект.Наименование);
	Сообщить(ТекстНаименования);
КонецПроцедуры

&НаСервереБезКонтекста
```

```
Функция НайтиОбъект(Наименование)
	ЦвСсылка = Справочники.Цвета.НайтиПоНаименованию(Наименование);
	Если ЦвСсылка = Справочники.Цвета.ПустаяСсылка() Тогда
		Возврат("Цвет не найден");
	Иначе
		Возврат(ЦвСсылка.Наименование + ": " + ЦвСсылка.Код);
	КонецЕсли;
КонецФункции
```

P.s. На клиент нельзя вернуть объект. Можно при желании вернуть структуру, или например ссылку, которую можно использовать в других подпрограммах:

```
&НаКлиенте
Процедура ПолучитьОбъектПоНаименованию(Команда)
	ССылкаНаЦвет = НайтиОбъект(Объект.Наименование);
	Если ССылкаНаЦвет = Неопределено Тогда
		Сообщить("Цвет не найден");
	Иначе
		ВывестиПоСсылке(ССылкаНаЦвет);
	КонецЕсли;
КонецПроцедуры

&НаСервереБезКонтекста
Функция НайтиОбъект(Наименование)
	ЦвСсылка = Справочники.Цвета.НайтиПоНаименованию(Наименование);
	Если ЦвСсылка = Справочники.Цвета.ПустаяСсылка() Тогда
		Возврат(Неопределено);
	Иначе
		Возврат(ЦвСсылка);
	КонецЕсли;
КонецФункции

&НаСервереБезКонтекста
Процедура ВывестиПоСсылке(ЦветСсылка)
	Сообщить(ЦветСсылка.Код);
КонецПроцедуры
```

Создание групп или элементов.

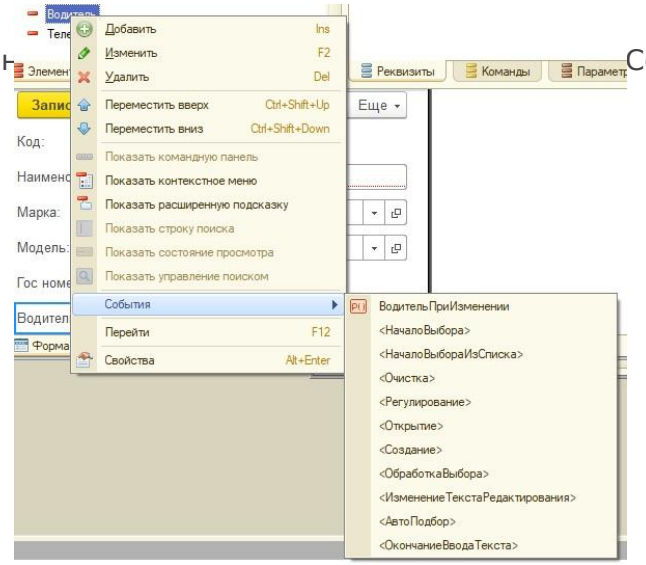
```
НовыйЭлемент = Справочники.Номенклатура.СоздатьЭлемент();
НовыйЭлемент.Реквизит1 = "КакоеТоЗначение";
```

```
НовыйЭлемент.Записать();
```

Выборка элементов

```
Выборка = Справочники.Номенклатура.Выбрать();
Пока Выборка.Следующий() Цикл
    ТекСсылка = Выборка.Ссылка;
    Сообщить(ТекСсылка.Наименование);
КонецЦикла;
```

Автоматическое заполнение реквизитов. Например есть справочник Сотрудники, у которого есть реквизит Рабочий телефон. Есть справочник Автомобили, у которого есть реквизит Водитель и Рабочий телефон. Нужно, чтобы при выборе водителя, происходило заполнение рабочего телефона.

Вариант 1. В редактирован 

События - ПриИзменении

```
&НаКлиенте
Процедура ВодительПриИзменении(Элемент)
	Объект.ТелефонВодителя = ПолучитьТелефон(Объект.Водитель);
КонецПроцедуры

&НаСервереБезКонтекста
Функция ПолучитьТелефон(ССылкаНаВодителя);
	Возврат ССылкаНаВодителя.РабочийТелефон;
КонецФункции
```

Однако, при изменении в справочнике Сотрудники внутреннего телефона, обновление не происходит.

Вариант 2. Обновление будет при создании связанных справочников Сотрудники - ТелефоныСотрудников и т. д. Обработчик, который добавляет первый указанный телефон пользователя

```
&НаКлиенте
Процедура ВодительПриИзменении(Элемент)
	Перем Телефон;
	НайтиТелефоныВодителя(Объект.Водитель, Телефон);
	Если Телефон <> Неопределено Тогда
		Объект.ТелефонВодителя = Телефон;
	КонецЕсли;
КонецПроцедуры

&НаСервереБезКонтекста
Процедура НайтиТелефоныВодителя(ССылкаНаСотрудника, ССылкаНаТелефон)
	Запрос = Новый Запрос(
		"ВЫБРАТЬ
		|	Телефоны.Ссылка как ТелефонВодителя
		|ИЗ
		|	Справочник.ТелефоныСотрудников КАК Телефоны
		|ГДЕ
		|	Телефоны.Владелец = &Ссылка");
	Запрос.УстановитьПараметр("Ссылка", ССылкаНаСотрудника);
	Результат = Запрос.Выполнить();
	Если не Результат.Пустой() Тогда
		ВыборкаДетальныеЗаписи = Результат.Выбрать();
		ВыборкаДетальныеЗаписи.Следующий();
		ССылкаНаТелефон = ВыборкаДетальныеЗаписи.ТелефонВодителя.Ссылка;
	КонецЕсли;
КонецПроцедуры
```

Можно доработать процедуру, возвращая массив ссылок, на клиенте сохраняя первую и сообщая остальные в виде всплывающего сообщения.

Есть косяк: При изменении владельца номера телефона, в справочнике Автомобили остается старый Водитель и не меняется ссылка на номер телефона. Возможно, необходима обработка При изменении на ТелефоныСотрудников. Типа уведомления о том, что нужно обновить данные в таком-то справочнике для такой-то записи.

Обработка события удаления элементов

Справочник Менеджеры подчинен справочнику Прайс-лист. В модуле объекта справочника Прайс-лист обработка удаления элемента.

```
Процедура ПередУдалением(Отказ)
	НайтиПодчиненныеЭлементы(ЭтотОбъект.Ссылка, Отказ);
КонецПроцедуры

&НаСервере
Процедура НайтиПодчиненныеЭлементы(ССылкаНаОбъект, Отказ)
    Запрос = Новый Запрос(
        "ВЫБРАТЬ
        |   Менеджеры.Ссылка.Наименование как ИмяМенеджера
        |ИЗ
		|   Справочник." + Метаданные.Справочники["Менеджеры"].Имя + " КАК Менеджеры
        |ГДЕ
        |   Менеджеры.Владелец = &Ссылка");
    Запрос.УстановитьПараметр("Ссылка", ССылкаНаОбъект);
	Результат = Запрос.Выполнить();
	Если Результат.Пустой() Тогда
		Отказ = Ложь;
	Иначе
		Сообщить("Нельзя удалить элемент, так как он владелец элементов.");
		ВыборкаДетальныеЗаписи = Результат.Выбрать();
		СписокЗависимостей = "";
		Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
			СписокЗависимостей = СписокЗависимостей + " " + ВыборкаДетальныеЗаписи.ИмяМенеджера;
		КонецЦикла;
		Сообщить(СписокЗависимостей);
        Отказ = Истина;
    КонецЕсли;
КонецПроцедуры
```

Группа или элемент

Метод «ЭтоГруппа()» возвращает 1 в случае группы и 0 в случае элемента.

Доступ к табличной части

```
&НаСервереБезКонтекста
Процедура ПолучитьТабличнуюЧастьНаСервере(КанцелярияСсылка)
	Набор = КанцелярияСсылка.СоставНабора;
```

```
Если Набор.Количество() > 0 Тогда
Сообщить("Состав набора:");
Иначе
Сообщить(КанцелярияСсылка.Наименование + " не набор.");
КонецЕсли;
Для каждого СтрокаТаблицы из Набор Цикл
Сообщить(СтрокаТаблицы.Канцелярия.Наименование + ", " + СтрокаТаблицы.Количество + " шт.");
КонецЦикла;
КонецПроцедуры
```

Класс Документы и Журналы документов

Стандартные реквизиты: Номер документа и Дата документа.

Заполнение реквизита значением по умолчанию. В свойствах реквизита есть параметр Значение заполнения. Для простых типов оно заполняется вручную. Для типов СправочникСсылка список значений берется из предопределенных данных справочника.

Ввод на основании: В разделе настройки "Ввод на основании" добавляется документ, который является основой для ввода и в модуле объекта создается процедура ОбработкаОснования.

```
Процедура ОбработкаЗаполнения(ДанныеЗаполнения, СтандартнаяОбработка)
	//{{__КОНСТРУКТОР_ВВОД_НА_ОСНОВАНИИ
	// Данный фрагмент построен конструктором.
	// При повторном использовании конструктора, внесенные вручную изменения будут утеряны!!!
	Если ТипЗнч(ДанныеЗаполнения) = Тип("ДокументСсылка.ПрибытиеВГараж") Тогда
		// Заполнение шапки
		Автомобиль = ДанныеЗаполнения.Автомобиль;
		Гараж = ДанныеЗаполнения.Гараж;
	КонецЕсли;
	//}}__КОНСТРУКТОР_ВВОД_НА_ОСНОВАНИИ
КонецПроцедуры
```

Обработка событий формы. Управление отображением и событиями платформы

Обработка табличной части документа:

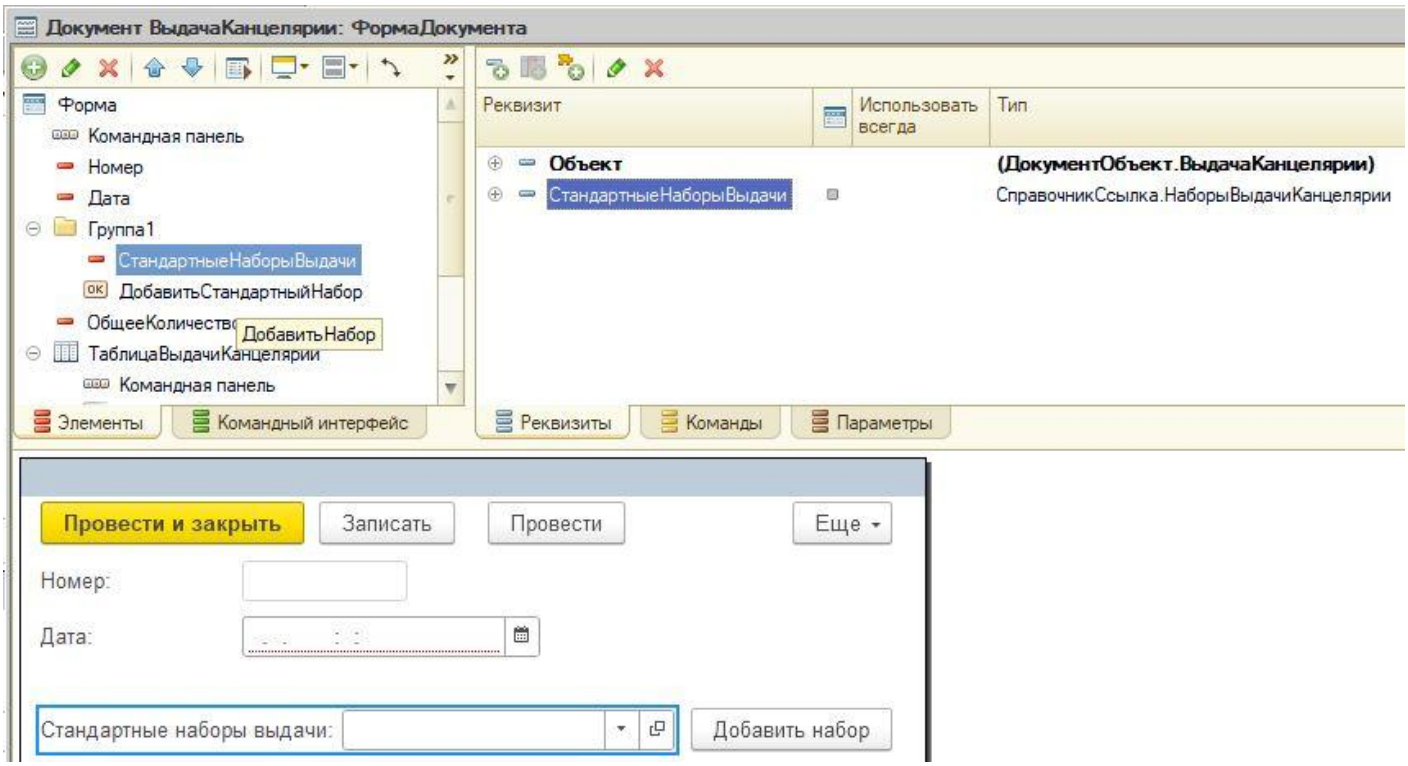
```
&НаКлиенте
Процедура ПересчитатьИтоговуюСумму()
	ТабЧасть = Объект.ПереченьТоваров;
	ФиналСумма = 0; 
	Для каждого Товар из ТабЧасть Цикл
		ФиналСумма = ФиналСумма + Товар.Сумма;
	КонецЦикла;
    Объект.СуммаПоДокументу = ФиналСумма;
```

```
КонецПроцедуры

&НаКлиенте
Процедура ПереченьТоваровКоличествоПриИзменении(Элемент)
СтрТабЧасти = Элементы.ПереченьТоваров.ТекущиеДанные;
СтрТабЧасти.Сумма = СтрТабЧасти.Цена * СтрТабЧасти.Количество;
ПересчитатьИтоговуюСумму();
КонецПроцедуры

&НаКлиенте
Процедура ПереченьТоваровЦенаПриИзменении(Элемент)
СтрТабЧасти = Элементы.ПереченьТоваров.ТекущиеДанные;
СтрТабЧасти.Сумма = СтрТабЧасти.Цена * СтрТабЧасти.Количество;
ПересчитатьИтоговуюСумму();
КонецПроцедуры
```

Добавление данных в табличную часть Есть справочники Канцелярия и НаборыВыдачиКанцелярии и документ ВыдачаКанцелярии. Нужно добавить опцию заполнения табличной части документа стандартными наборами. В документе нет реквизита НаборыВыдачиКанцелярии, так как наборы могут меняться, в документ может быть добавлено несколько наборов, но само название набора не важно, а фактически важен добавленный состав. На форму был добавлен реквизит СтандартныеНаборыВыдачи и кнопка Добавить набор.



```
&НаСервере
Процедура ПолучитьСоставНабора()
ТекСоставНабора = СтандартныеНаборыВыдачи.СоставНабора;
Для Каждого ЭлемНабора из ТекСоставНабора цикл
НовыйЭлем = Объект.ТаблицаВыдачиКанцелярии.Добавить();
НовыйЭлем.НазваниеПредмета = ЭлемНабора.Канцелярия;
КонецЦикла;
КонецПроцедуры

&НаКлиенте
Процедура ДобавитьНабор(Команда)
ПолучитьСоставНабора();
КонецПроцедуры
```

Создание документа

```
&НаКлиенте
Процедура СоздатьНовыйДокумент(Команда)
    Если СоздатьНовыйДокументНаСервере() = 0 Тогда
        Сообщить("Не удалось создать новый документ");
    КонецЕсли;
КонецПроцедуры

&НаСервере
Функция СоздатьНовыйДокументНаСервере()
    НоваяРасходнаяНакладная=ДокРасходнаяНакладная.СоздатьДокумент();
    НоваяРасходнаяНакладная.Дата=  ТекущаяДата();
    НоваяРасходнаяНакладная.Фирма =Справочники.Фирмы.ОсновнаяФирма;
    НоваяРасходнаяНакладная.Контрагент=Справочники.Контрагенты.НайтиПоКоду("000000001");
    НоваяРасходнаяНакладная.Склад=Справочники.Склады.ОсновнойСклад;
    СтрокаТЧ=НоваяРасходнаяНакладная.ТЧТовары.Добавить();
    СтрокаТЧ.Товар=Справочники.Номенклатура.НайтиПоКоду("000000002");
    СтрокаТЧ.Цена=СтрокаТЧ.Товар.РозничнаяЦена;
    СтрокаТЧ.Количество= 2;
    СтрокаТЧ.Сумма=СтрокаТЧ.Цена*СтрокаТЧ.Количество;

    Попытка
        НоваяРасходнаяНакладная.Записать();
        Возврат 1;
    Исключение
```

```
            Возврат 0;
        КонецПопытки;
КонецФункции
```

Дополнительные опции:

- Нумераторы - отвечают за сквозную нумерацию для разных документов, сначала создается нумератор, затем в разделе Нумерация нужных документов указывается нумератор.
- Последовательности - отвечают за контроль целостности данных при изменении проведенного документа. Например, при изменении документа в уже закрытом периоде. Нужно разобраться.

Журналы документов

Не создают дополнительных данных. Только для визуального объединения разных документов в одном списке.
В разделе Данные в Регистрируемые документы добавляются нужные документы, в разделе Графы - поля из документов.

Класс Регистры

Детали хранения информации в регистрах

Регистр сведений, накопления, бухгалтерии, расчета относятся к необъектным сущностям. Примеры для регистра накопления и регистра сведений относятся ко всем классам регистров. Для регистра сведений есть возможность использования без регистратора, поэтому модель его использования шире.

Модель хранения данных. В базе данных хранятся записи. Запись не является объектом базы данных, нет внутреннего идентификатора. Запись полностью описывается значениями своих полей. Т е удалив некоторую запись и записав новую с точно такими же значениями всех полей, мы получим то же состояние базы данных с точки зрения логики прикладного решения. Это принципиально отличает запись от объекта в объектных сущностях, так как объект базы данных имеет внутренний идентификатор и один и тот же объект нельзя создать дважды.

Например, если у нас есть регистр накопления, хранящий движения складских запасов товаров в разрезе товаров и складов, то каждая запись полностью определяется товаром, складом и количеством. Действительно, с точки зрения логики прикладного решения такая запись описывает движение некоторого товара по некоторому складу в некотором количестве. Можно записать вторую такую же запись, и они ничем не будут отличаться, если не отличаются значения полей.

Уникальность записей. Нельзя записывать ссылки на записи в поля базы данных, но есть уникальный ключ. Для необъектных сущностей с регистратором уникальный ключ включает регистратор и номер строки. Номер строки используется для обеспечения уникальности записей и для упорядочивания записей в пределах регистратора.

Для регистров сведений не подчиненных регистратору уникальным ключ это совокупность измерений и периода, если регистр периодический. В случае подчинения регистратору уникальность поддерживается по сочетанию полей регистратор и номер строки и по измерениям и периоду. Первое определяется его подчинением регистратору, а второе его основной прикладной логикой – хранением значений ресурсов по комбинации значений измерений и периоду. Если для регистра выбрана периодичность по позиции регистратора, то в уникальный ключ по измерениям входит и регистратор, как дополнительная детализация периода в пределах секунды.

Подчинение регистратору. Записи регистров подчиненные регистратору называются движениями. Подчинение записей регистратору определяет время жизни записей. Регистратор определяет наличие этих записей. Записи регистров создаются документом в процессе проведения. При удалении регистратора подчиненные ему записи удаляются.

Записи регистров подчинены регистратору, но не являются его частью. При считывании документа подчиненные ему записи регистров не считываются, а при записи автоматически не записываются. Обеспечивается только удаление записей при удалении регистратора. Также существует возможность автоматического удаления записей при проведении и отмене проведения. Эта возможность настраивается в свойствах документа в конфигурации и является сервисной.

Наличие записей регистров не связано с состоянием "проведен" и с пометкой удаления. Допускается наличие записей и у непроведенного документа и у помеченного на удаление. Например, это используется для создания документов предназначенных для непосредственного ввода движений регистров – ручного ввода. В этом случае пользователь непосредственно вводит записи регистров и понятие проведения бессмысленно. При непосредственном изменении полей Проведен и ПометкаУдаления (без использования методов УстановитьПометкуУдаления() и записи с режимом ОтменаПроведения) удаление движений не производится.

Поэтому организация редактирования движений документа в форме сложнее, чем организация редактирования табличной части. Наиболее простой моделью использования регистров является создание движений при проведении. В этом случае достаточно реализовать запись движений в обработчике проведения, а вся остальная логика будет поддерживаться системой автоматически.

Типы данных. Тип РегистрыНакопленияМенеджер предназначен для доступа к менеджерам конкретных регистров. К нему можно обратиться с помощью свойства глобального контекста РегистрыНакопления.
Тип РегистрНакопленияМенеджер предоставляют доступ к общим действиям, относящимся к конкретному регистру накопления. С помощью его методов выполняются действия, относящиеся к регистру, а не к его конкретным записям. Например, методы менеджера позволяют создать объект, предназначенный для манипулирования записями (НаборЗаписей), получить выборку регистра и т.д.
Менеджер регистра имеет набор методов для выполнения действий связанных с основным назначением регистра. Например, для регистра накопления это получения остатков и оборотов, а у регистра сведений получение среза первых и среза последних записей. Также данные могут быть получены виртуальными таблицами запросов, в них обеспечивается большая гибкость.

Тип РегистрНакопленияНаборЗаписей предназначен для чтения, модификации и удаления записей регистра. Набор записей оперирует множеством записей отбираемых по некоторому условию. Для описания условия есть свойство Отбор. Для регистров подчиненных регистратору отбор устанавливается по регистратору.
Набор записей это коллекция, которую можно прочитать и записать. Набор записей имеет только один метод для изменения данных – Записать(). Нет понятия удаления. Набор записей может быть только записан. При этом выполняется замещение всех существующих записей удовлетворяющих текущему отбору на записи, хранящиеся в наборе. Для удаления множества записей следует установить отбор и, не добавляя записей в набор, выполнить запись набора.

Можно записать набор без замещения, параметр метода Записать(). Запись без замещения выполняет добавление записей. Будет обеспечиваться корректировка номеров строк, для обеспечения последовательной нумерации записей в пределах регистратора. Обычно используется при необходимости записи большого объема движений регистров.

Набор записей необязательно считывать, чтобы выполнить запись. Не существует понятия блокировки, поэтому при записи набора могут быть изменены данные, которые были изменены другой сессией или другим набором в этой сессии уже после считывания набора записей.

Для регистров сведений не подчиненных регистратору может использоваться отбор с различными комбинациями измерений и периода. Допускается чтение и запись набора без установки отбора. В этом случае будет выполняться чтение и соответственно запись всех записей регистра. При ошибках в написании модулей (например, пропущенном вызове чтения набора или пропущенной установке отбора) запись набора может привести к полной очистке регистра, так как запись при неустановленном отборе фактически замещает весь регистр на содержимое набора записей.

Запись в наборе записей регистра накопления имеет тип РегистрНакопленияЗапись. Данный тип используется только для записи в наборе записей. Это значение не используется отдельно от набора записей.

Для регистра сведений не подчиненного регистратору кроме набора записей существует тип РегистрСведенийМенеджерЗаписи. Этот тип предназначен для чтения и записи одной записи регистра. Используется для организации редактирования одной записи в форме. Позволяет прочитать запись с определенными значениями ключевых полей, изменить поля, в том числе и ключевые, и записать. Является вспомогательным. Для работы с регистром он использует два набора записей (с отборами соответствующими соответственно ключевым полям считанной и записываемой записи). Соответственно реализация обработчиков в модуле набора записей позволят полностью контролировать изменения данных регистра, в том числе и тогда, когда они выполняются с помощью менеджера записи.

Тип РегистрНакопленияКлючЗаписи предназначен для уникальной идентификации записи регистра. Например, он используется для идентификации строки табличного поля отражающего список регистра. Это позволяет активизировать необходимую строку. Свойства ключа определяются полями образующими уникальный ключ регистра. Для регистров сведений это измерения, период и регистратор, если используется периодичность по позиции регистратора. Для других регистров это регистратор и номер строки. Ключ записи регистра может быть создан с помощью менеджера.

Тип РегистрНакопленияВыборка предназначен для динамического обхода записей регистра. Механизм выборок для необъектных таблиц не имеет существенных отличий. Следует учитывать, что выборка всегда считывает данные записей целиком (считываются все поля).

Тип РегистрНакопленияСписок предназначен для динамического просмотра записей регистра в элементе управления ТабличноеПоле. У него может быть настроен состав

считываемых полей (колонок), отбор и сортировка. Список осуществляет считывание данных порциями в процессе навигации пользователем в табличном поле.

Кроме указанных типов значений регистр определяет несколько расширений элементов управления и форм, предназначенных для интерактивного ввода и просмотра, данных регистра. Расширения не являются типами данных, а добавляют специфические свойства, методы и события к соответствующим объектам. Кроме того, расширения определяют некоторое специфическое поведение форм и элементов управления при конфигурировании и работе пользователя с системой.

Для регистров сведений кроме расширений для показа в форме списка и набора записей поддерживается расширение для редактирования в форме одной записи. В этом случае в качестве основного реквизита формы выступает менеджер записи регистра сведений.

Оборотный регистр накопления

Создается только одна виртуальная таблица "Обороты". Остатки не регистрируются. Тип настраивается во вкладке Основное.

Регистр накопления

Движения. Накапливает числовой итог в разрезе измерений по ресурсам. Остатки - только итоговое значение, обороты - состояние регистра на конкретную дату.
Необходимо указать регистраторы и затем в настройках каждого документа создать обработчик проведения. Можно через конструктор движения. Пример обработки проведения в модуле объекта документа

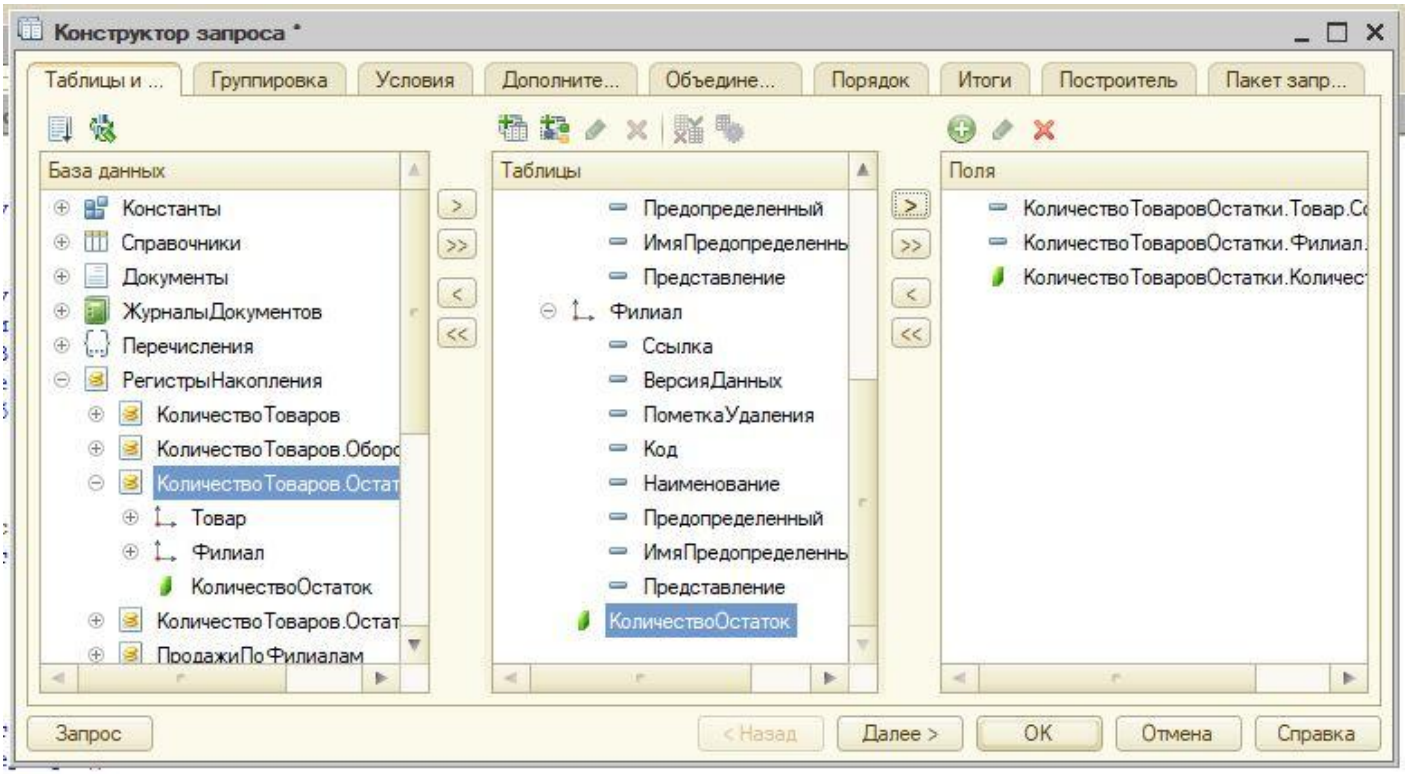
```
Процедура ОбработкаПроведения(Отказ, Режим)
	//{{__КОНСТРУКТОР_ДВИЖЕНИЙ_РЕГИСТРОВ
	// Данный фрагмент построен конструктором.
	// При повторном использовании конструктора, внесенные вручную изменения будут утеряны!!!

	// регистр ОстаткиКанцелярии Приход
	Движения.ОстаткиКанцелярии.Записывать = Истина;
	Для Каждого ТекСтрокаТаблицаПоступленияКанцелярии Из ТаблицаПоступленияКанцелярии Цикл
		Движение = Движения.ОстаткиКанцелярии.Добавить();
		Движение.ВидДвижения = ВидДвиженияНакопления.Приход;
		Движение.Период = Дата;
		Движение.Предмет = ТекСтрокаТаблицаПоступленияКанцелярии.НазваниеПредмета;
		Движение.Количество = ТекСтрокаТаблицаПоступленияКанцелярии.Количество;
	КонецЦикла;

	//}}__КОНСТРУКТОР_ДВИЖЕНИЙ_РЕГИСТРОВ
КонецПроцедуры
```

Получение данных через запрос. Интересный набор статей про регистр накоплений. Регистр накопления с видом "Остатки" позволяет использовать виртуальные таблицы "Обороты" и "Остатки".

Используем Конструктор запроса. Так как нужно просто получить остатки на указанную дату, выбираем регистр, таблицу и поля



Итоговая процедура:

```
&НаСервереБезКонтекста
Процедура ВзятьОстаткиНаСервере(НужнаяДата)
	Запрос = Новый Запрос;
	Запрос.Текст ="ВЫБРАТЬ
	|	КоличествоТоваровОстатки.Товар.Ссылка.Описание как Товар,
	|	КоличествоТоваровОстатки.Филиал.Ссылка.Наименование как Филиал,
	|	КоличествоТоваровОстатки.КоличествоОстаток как Колво
	|ИЗ
	|	РегистрНакопления.КоличествоТоваров.Остатки(&Дата) КАК КоличествоТоваровОстатки";
	Запрос.установитьпараметр("Дата",НужнаяДата);
	Результат = Запрос.Выполнить();
	Если не Результат.Пустой() Тогда
		ВыборкаДетальныеЗаписи = Результат.Выбрать();
		Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
```

```
		Сообщить(ВыборкаДетальныеЗаписи.Филиал + ВыборкаДетальныеЗаписи.Товар +
ВыборкаДетальныеЗаписи.Колво);
	КонецЦикла;
    КонецЕсли;
КонецПроцедуры

&НаКлиенте
Процедура ВзятьОстатки(Команда)
	Если не ЗначениеЗаполнено(ДатаФормированияОстатков) тогда
		ДатаФормированияОстатков = ТекущаяДата();
	КонецЕсли;
	ВзятьОстаткиНаСервере(ДатаФормированияОстатков);
КонецПроцедуры
```

Получение данных через менеджер.

```
&НаСервереБезКонтекста
Процедура ВзятьОстаткиНаСервере2()
	РегМенеджер = РегистрыНакопления.КоличествоТоваров;
	Остатки = РегМенеджер.Остатки();
	Для каждого ТекОст из Остатки Цикл
		Сообщить(Строка(ТекОст.Филиал) + " " + Строка(ТекОст.Товар)+ ": " + ТекОст.Количество);
	КонецЦикла;
КонецПроцедуры
```

Значения в возвращенной таблице являются ссылками, из которых еще нужно получить значения. А это дополнительные запросы к базе. В примере прокатило, т к в этих справочниках основное представление в виде наименование. Если изменить в справочнике на в виде кода, то данный пример выведет числа вместо наименований филиалов и товаров. Т е если в базе 10 филиалов, 10 товаров и представление в виде кода, то будет генериться 100 запросов к базе просто для получения имени. А это очень неэффективно. Так что этот способ использовать осознанно.

Регистры сведений

??????? ???????????? ??? ???????? ?????? (????????) ? ???????????????? ???? (? ??????? ?????????) ? ?????????????? ?????????????? ??????????? (?????????). ????????????? ???????? ?????????? ??????????? ??????, ????? ???????? ?????????? ????? ??????????. ????? ?????? ????? ???? ??????????? ??? ??????????? (???????? ????????????). ?????????????? ???????? ????????????, ?.?. ???????, ? ??????? ????????? ???????? ?????????, ?????? (???? ??????? ?????????????) ? ??????????? (???? ??????? ???????? ????????????).

?????????????? ??????????? ??????? ???????? ?????????? ?? ??????????? ?????? ????????? ???????????? ?

????? ???????????? ?????? ?????????????, ?? ????? ???????????? ????????????.

?????????? ? ???????? ???????. ????? ????? ?????????????:

- ???????? ??????? ????????????;
- ????????? ??????? ?? ?????????, ?????? (???? ?????????????) ? ???????????? (???? ???????? ????????????); ?? ???????? ????? ???????? ????? ? ???????????? ? ????????? ? ????? ??????? ?? ????????? ???? ?????.
- ?????????? ? ?????????? ???????? ????? ???????;
- ?????? ?????? ???????.

```
// Добавление записи в независимый непериодический регистр сведений
НаборЗаписей = РегистрыСведений.ВерсииПодсистем.СоздатьНаборЗаписей(); // Этап 1
НаборЗаписей.Отбор.ИмяПодсистемы.Установить(ИмяПодсистемы); // Этап 2
// Этап 3
НоваяЗапись = НаборЗаписей.Добавить();
НоваяЗапись.ИмяПодсистемы = ИмяПодсистемы;
НоваяЗапись.Версия = НомерВерсии;
НаборЗаписей.Записать(); // Этап 4
```

???? ?? ??????? ?????, ?? ????????????? ???? ???????. ? ?????? ? ???????, ????????????? ?? ????? ????????, ??????? ???? ???????.

????????, ???? ???????, ? ??????? ????????? ? ???????? - ?????????, ????????? - ??????. ???? ????? ??????:

МояСтрока	МоеЧисло	Результат
олдж	7	67
олдж	8	67
фыва	4	67

???? ????? ?????????? ???? ????? ?? ????????? ???????? "????" ? ????? ???????? ???? ??????, ?? ? ???????? ????????? 2 ?????? (???? ? ????). ?????? ????????????? ? ????? ?????? ?????????, ???????? ?? ???????? ??? ?????? ?????????. ? ? ???? ????? ??????? ????? ????, 7, ? ? ????? ?????? ????? ??????? ???????? ???? 8, ????? ??????. ?? ???? ?????? ???? ????? ????, ?? ????? ???????????? ??? ???????? ????????.

```
// Добавление записи в независимый периодический регистр сведений
НаборЗаписей = РегистрыСведений.КурсыВалют.СоздатьНаборЗаписей(); // Этап 1
// Этап 2
НаборЗаписей.Отбор.Валюта.Установить(Доллар);
НаборЗаписей.Отбор.Период.Установить(НачалоДня(ТекущаяДата()));
// Этап3
НоваяЗапись = НаборЗаписей.Добавить();
НоваяЗапись.Период = ТекущаяДата();
```

```
НоваяЗапись.Валюта = Доллар;
НоваяЗапись.Курс = 57.92;
НоваяЗапись.Кратность = 1;
НаборЗаписей.Записать(); // Этап 4
```

????????? ???????. ??? ?????? ??????? ????????? ????????? ???????? ??? ???????? ???????????? ????????. ????? ?????????????? ??????.

```
// Редактирование записей с использованием объекта НаборЗаписей
НаборЗаписей = РегистрыСведений.КурсыВалют.СоздатьНаборЗаписей(); // Этап 1
// Этап 2
НаборЗаписей.Отбор.Период.Установить(ДатаКурса);
НаборЗаписей.Отбор.Валюта.Установить(Доллар);
НаборЗаписей.Прочитать(); // Этап 3
Для Каждого Запись Из НаборЗаписей Цикл
    Запись.Курс = 57.84; // Этап 4
КонецЦикла;
НаборЗаписей.Записать(); // Этап 5
```

?????? ???????. ?????????? ????????? ????? ??????.

```
Запрос = Новый Запрос;
Запрос.Текст =
   "ВЫБРАТЬ
   |  КурсыВалют.Период,
   |  КурсыВалют.Валюта,
   |  КурсыВалют.Курс
   |ИЗ
   |  РегистрСведений.КурсыВалют КАК КурсыВалют";

Выборка = Запрос.Выполнить().Выбрать();
Пока Выборка.Следующий() Цикл
   // обход результата выполнения запроса
КонецЦикла;
```

??? ????????????? ????????? ???? ??????????? ??????? ????????????? (?????? ??????? ?????? ????????? ????) ? ?????????? (?????? ??????? ?????? ????????? ????).

```
// Получение записи, у которой валюта равна значению из переменной «ВыбраннаяВалюта»
// и период БОЛЬШЕ или равен значению из переменной «ВыбраннаяДата»
Запрос = Новый Запрос;
Запрос.Текст =
    "ВЫБРАТЬ
    |  КурсыВалютСрезПервых.Период,
    |  КурсыВалютСрезПервых.Валюта,
    |  КурсыВалютСрезПервых.Курс
    |ИЗ
    |  РегистрСведений.КурсыВалют.СрезПервых(&Период, Валюта = &Валюта) КАК
КурсыВалютСрезПервых";

Запрос.УстановитьПараметр("Валюта", ВыбраннаяВалюта);
Запрос.УстановитьПараметр("Период", ВыбраннаяДата);

Выборка = Запрос.Выполнить().Выбрать();
Пока Выборка.Следующий() Цикл
    // обход результата выполнения запроса
КонецЦикла;
```

???? ????? ??????? ?? ??????? ? ?????? ?? ??????? ??? ??????????? ???? ??? ????????? ?????????????:

```
Функция РозничнаяЦена(АктуальнаяДата, ЭлементНоменклатуры) Экспорт
  // Создать вспомогательный объект "Отбор".
  Отбор = Новый Структура("Номенклатура", ЭлементНоменклатуры);
  // Получить актуальные значения ресурсов регистра.
  ЗначенияРесурсов = РегистрыСведений.Цены.ПолучитьПоследнее(АктуальнаяДата, Отбор);
  Возврат ЗначенияРесурсов.Цена;
КонецФункции
```

```
// Получить текущую строку табличной части.
СтрокаТабличнойЧасти = Элементы.ПереченьНоменклатуры.ТекущиеДанные;
// Установить цену.
СтрокаТабличнойЧасти.Цена = РаботаСоСправочниками.РозничнаяЦена(
            Объект.Дата, СтрокаТабличнойЧасти.Номенклатура);
// Пересчитать сумму строки
РаботаСДокументами.РассчитатьСумму(СтрокаТабличнойЧасти)
```

Проведение одним документом в нескольких регистрах через конструктор

Справа добавить регистры и настроить сопоставление полей.

Класс Отчеты

Визуальная разработка отчета

Создание схемы компоновки данных - центральная задача при создании отчета. Открывается из раздела Основные отчета.

Схема компоновки данных (СКД).

На основе схемы компоновки и примененных к этой схеме настроек, компоновщик макета формирует макет компоновки данных. Далее этот макет обрабатывается процессором компоновки и преобразуется в результат компоновки данных. В свою очередь, результат компоновки может быть выведен в табличный документ, либо в объект – таблицу значений или дерево значений.

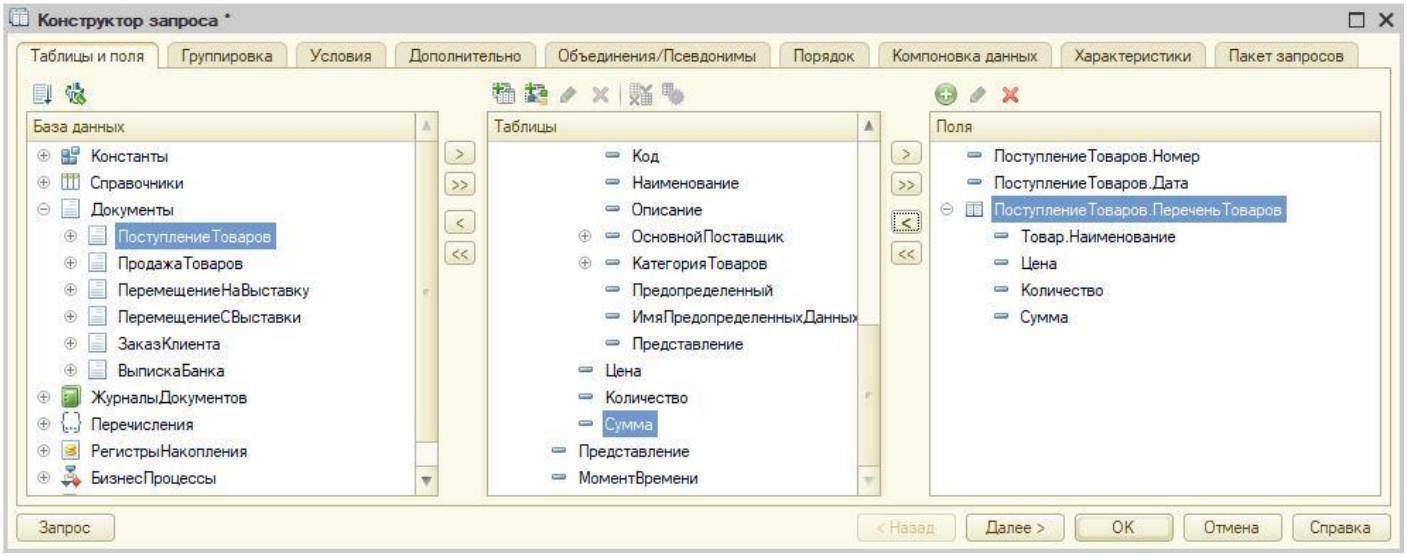
Компонент схемы	Назначение
Наборы данных	Данные для дальнейшего представления. Могут быть запросы, объекты, объединения. Объединение объединяет из нескольких наборов данных (внешнее соединение).
Связи наборов данных	Если есть несколько наборов данных и нужно использовать левое соединение.
Вычисляемые поля	Значения этих полей получаются в результате вычисления выражений, написанных разработчиком в схеме компоновки данных, или в результате выполнения функций, описанных в общих модулях конфигурации. Дополнительные столбцы.
Ресурсы	Групповые итоговые данные. Не отдельные столбцы, как в случае с вычисляемыми полями.
Параметры	Параметры могут быть явно определены в запросе, например вид номенклатуры (&ВидНоменклатуры), а могут быть параметрами виртуальных таблиц базы данных, например начало и конец отчетного периода. Как правило, параметры выводятся пользователю перед формированием отчета. Затем заданные пользователем значения параметров передаются в отчет, и отчет формируется заново, например с новым отчетным периодом
Макеты	
Вложенные схемы	Для многократного использования разработанной схемы в других отчетах, связав родительский и вложенный отчеты по общему полю.

Компонент схемы	Назначение
Настройки	Содержит 4 типа блоков: группировка, таблица, диаграмма, вложенный отчет.

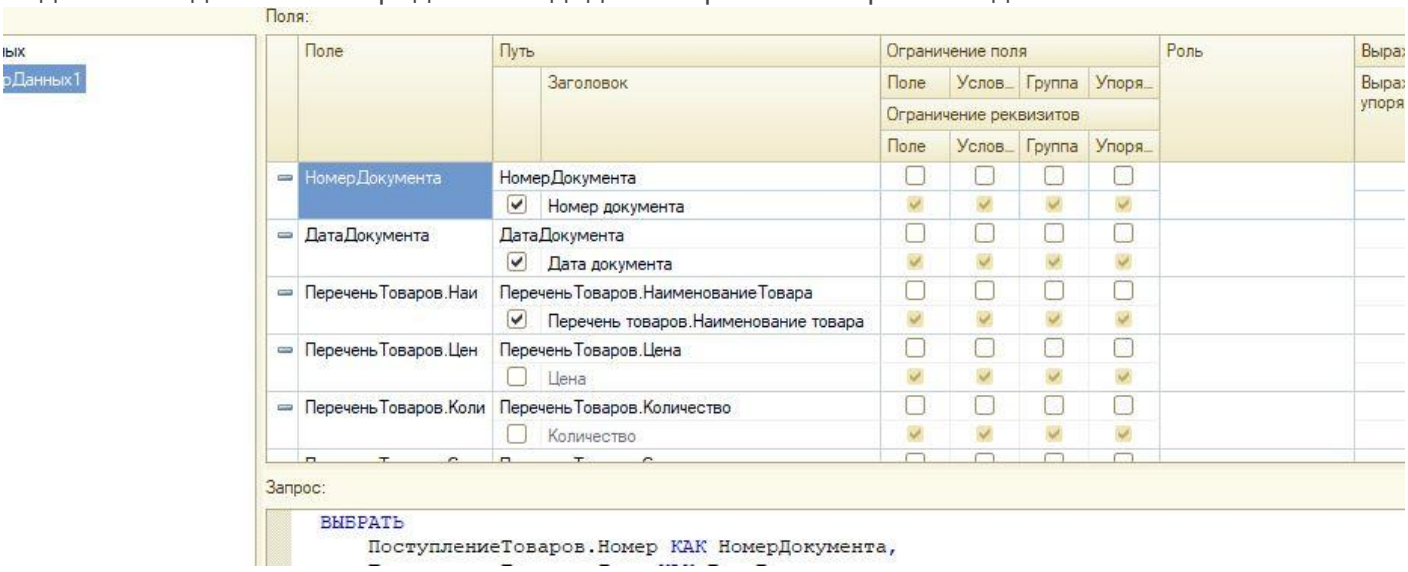
Пример создания отчета для документа с табличной частью.

Сначала заполняется наборы данных. После этого в разделе Поля появляются указанные в запросе. Варианты наборов данных: запрос, объект и объединение.

Выбираем Запрос и переносим нужные поля.



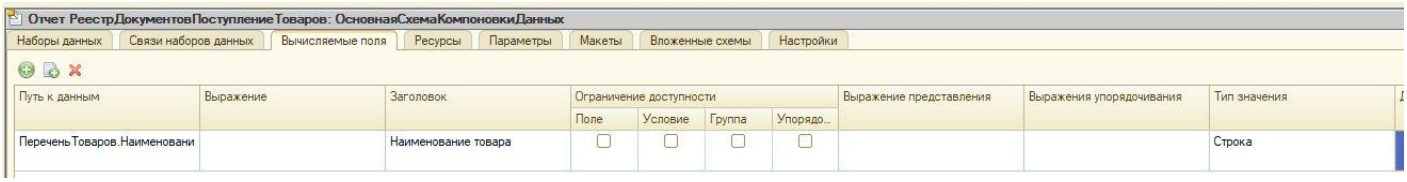
Задаем псевдонимы и порядок и создадим запрос. Из запроса создается список полей.



На вкладке Настройки Добавляем в Отчет группировку. Поле не указываем. Теперь настроим поля. На вкладку Выбранные поля и перенести. В параметрах указать оба поля даты, перейти в каждый и Включить в пользовательские настройки.

Вкладка Ресурсы СКД для добавления расчета итогов.

Для изменения имени столбца нужно добавить вычисляемое поле и заменить в нем заголовок. Учесть: флаг Группа криво объединяет, например если будет две записи с разными ценами, то объединение возьмет общее количество и первую цену.



Макеты

Присутствует во многих классах системы.

Хранит формы представления данных. Может быть текстовый документ, двоичные данные, HTML-документ или Active Document, графическую или географическую схему, схему компоновки данных или макет оформления схемы компоновки данных.

При помощи конструктора может быть создана из блока настроек класса Макеты.

Печатная форма.

Создаются области, затем при помощи встроенного языка области заполняются конкретными данными.

Добавление поля/полей: в макете выделить строки и установить имя через Таблица - Имена - Назначить имя. Для нужной строки можно назначить тип Параметр, указываем его имя. В модуле менеджера после отработки конструктора будет процедура Печать.

```
Процедура Печать(ТабДок, Ссылка) Экспорт
...
ОбластьКолВоСотрудников = Макет.ПолучитьОбласть("КоличествоСотрудников");
//формируем число
ОбластьКолВоСотрудников.Параметры.ВсегоСотрудников = КолВоСотрудников;
ТабДок.Вывести(ОбластьКолВоСотрудников);
```

Периодические задания

Расположены в Общие - Регламентные задания

В файловом варианте (8.3), задания инициирует первый запущенный клиент. Первый период пропускается.

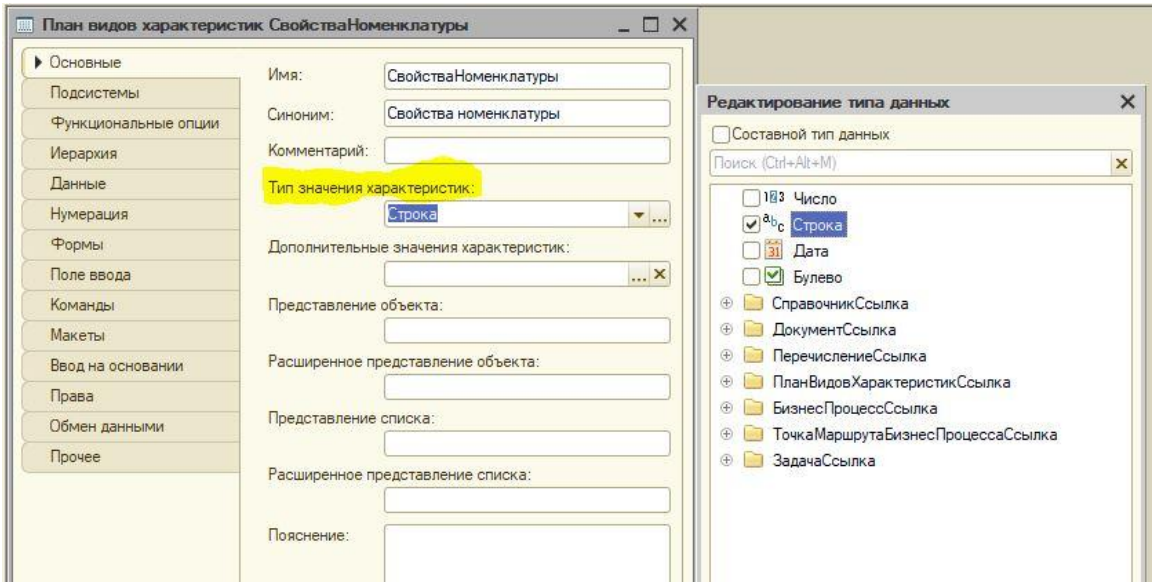
Может выполняться подпрограмма неглобального общего модуля, у общего модуля должно быть установлено свойство Сервер и Вызов сервера.

При создании задания, галочка "Предопределенное" означает запуск задания автоматически, иначе нужно запускать отдельно средствами языка. В Расписании в разделе Общие нужно выставить минимум один день, затем в разделе Дневное установить период (минимум 60 секунд для файловой версии).

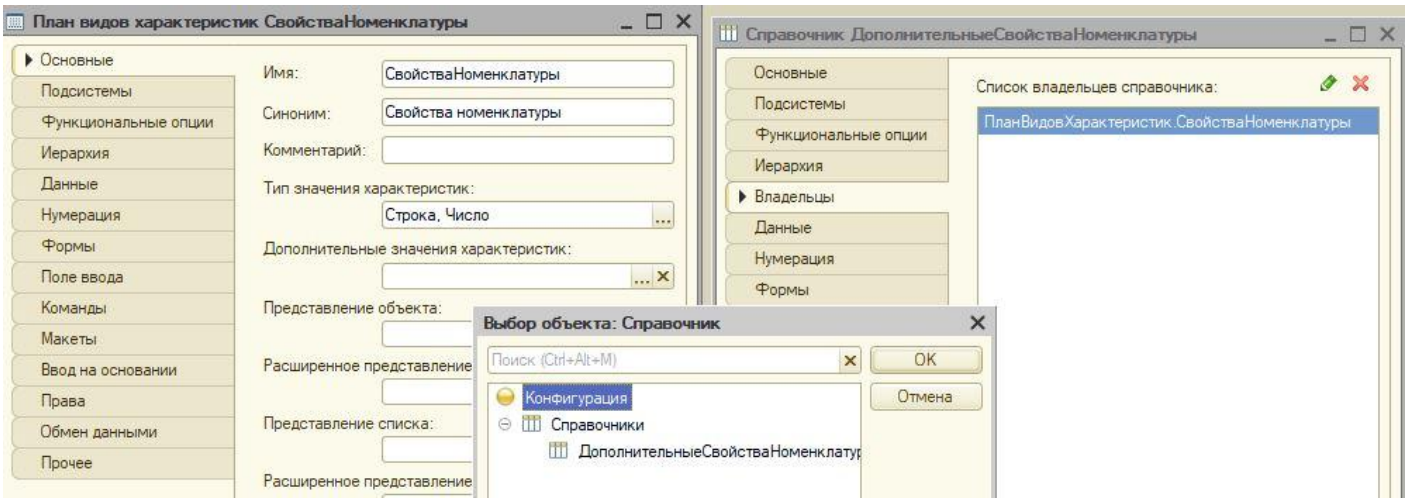
План видов характеристик

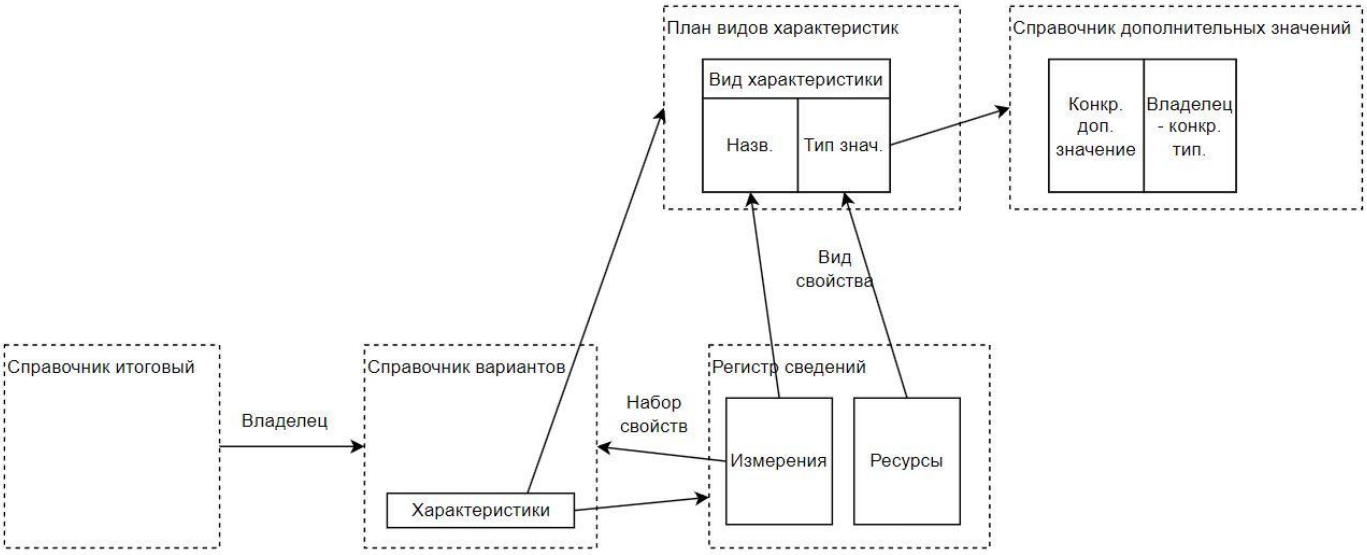
Задача элементов данного типа состоит в предоставлении пользователю возможности произвольным образом описывать наборы характеристик и вести учет в разрезе этих наборов.

Каждый вид характеристики обязательно описывается наименованием и типом значения. Все возможные типы должны быть перечислены в поле Тип значения характеристики, возможен составной тип.



Для предоставления пользователю возможности дополнительного создания характеристик, нужно создать справочник, Владельцем установить созданный план видов характеристик и указать этот справочник в Дополнительных значениях характеристик.





Пример: учет номенклатуры по заранее неизвестному набору характеристик. Под термином "Партия" будем понимать некий набор товаров с одинаковыми характеристиками, партии могут отличаться по наборам характеристик. Т е если нам пришли на склад например ручки (характеристики цвет и тип) и бумага (характеристики размер и толщина), то с точки зрения учета это две партии товара.

Для учета характеристик партий создадим справочник ВариантыНоменклатуры, подчиненный справочнику Номенклатура.

Для плана видов характеристик типы значения укажем строка, число, дата, булево, ДополнительныеСвойстваНоменклатуры.

Создадим Регистр сведений ЗначенияСвойствНоменклатуры. Измерения регистра: НаборСвойств, ведущее, тип СправочникСсылка.ВариантыНоменклатуры и ВидСвойства, тип ПланВидовХарактеристикСсылка.СвойстваНоменклатуры. Ресурс регистра: Значение, Характеристика.СвойстваНоменклатуры (берется из плана видов характеристик, может иметь значение любого типа из тех, которые описаны в типе значения плана видов характеристик). Зададим Связь по типу: Вид свойства. Это будет обеспечивать соответствие типа значений, вводимых в это поле, и типа характеристики, выбранной в поле Вид свойства. В Связях параметров выбора, также укажем Вид свойства. Это обеспечит при выборе значений, содержащихся в справочнике Дополнительные свойства номенклатуры, для выбора будут предлагаться только те значения, которые относятся к выбранной характеристике, а не все, которые есть в этом справочнике

Для справочника ВариантыНоменклатуры опишем, где хранятся свойства вариантов номенклатуры и как получить значения
этих свойств. Это описание платформа будет использовать автоматически при выполнении отчетов и при формировании различных динамических списков, в которых задействуются варианты номенклатуры. В контекстном меню справочника ВариантыНоменклатуры выберем команду Характеристики. Добавим новую запись. В качестве источника характеристик выберем план видов характеристик СвойстваНоменклатуры. Перейдем к описанию того, где и как хранятся значения

свойств. В качестве источника значений характеристик выберем регистр сведений ЗначенияСвойствНоменклатуры. Платформа автоматически определит, что в этом регистре полем объекта является измерение НаборСвойств, а полем вида – измерение ВидСвойства. Поэтому единственное, что нам останется указать самостоятельно, что значения свойств хранятся в ресурсе Значение.