PROJECT <24>
IT-ГРУППА СПЕЦИАЛЬНОГО НАЗНАЧЕНИЯ

<t> (+7 495) 730 52-26
<@> HELLO@P-24.RU

<address> 115054, Г. МОСКВА,
3-Й МОНЕТЧИКОВСКИЙ ПЕР. 11 СТР. 1

Требования к написанию и оформлению программного кода 1С

PROJECT <24>
IT-ГРУППА СПЕЦИАЛЬНОГО НАЗНАЧЕНИЯ

<t> (+7 495) 730 52-26
<@> HELLO@P-24.RU

<address> 115054, Г. МОСКВА,
3-Й МОНЕТЧИКОВСКИЙ ПЕР. 11 СТР. 1

Содержание

PROJECT <24>
IT-ГРУППА СПЕЦИАЛЬНОГО НАЗНАЧЕНИЯ

<t> (+7 495) 730 52-26
<@> HELLO@P-24.RU

<address> 115054, Г. МОСКВА,
3-Й МОНЕТЧИКОВСКИЙ ПЕР. 11 СТР. 1

1. Общие требования к конфигурации

Для типизированных объектов метаданных, хранящихся в информационной базе, настоятельно рекомендуется не использовать тип ЛюбаяСсылка. Состав типов того или иного типизированного объекта должен определяться явным образом.
Объекты метаданных верхнего уровня, такие как Справочники, Документы и т.д., рекомендуется сортировать в дереве метаданных по имени. Исключение составляют объекты с префиксом «Удалить», которые допустимо размещать в конце соответствующей ветки метаданных.
Для типизированных объектов метаданных строкового типа рекомендуется использовать переменную длину строки. Свойство «Фиксированная длина» может устанавливаться только в тех случаях, когда действительно необходимо при манипуляции этими данными иметь гарантию, что строка имеет определенную длину, даже несмотря на наличие концевых пробелов.
Подчиненные объекты метаданных, такие как реквизиты, измерения, формы, располагаются в дереве метаданных в соответствии с проектной логикой.
Для оптимизации тех или иных отчетов или для оптимизации выполнения отбора и сортировки в формах списков возможно использование индексирования. При этом индексирование следует использовать сдержанно, так как увеличение числа индексов приводит к дополнительной нагрузке на систему при записи данных и увеличивает объем базы данных.

2. Имя, синоним комментарий

Для подчиненных объектов метаданных, таких как реквизиты, измерения, ресурсы, рекомендуется не использовать имена, совпадающие с именами объектов-владельцев. Также рекомендуется не использовать имена, которые применяются при именовании таблиц языка запросов (например, «Документ», «Справочник», «РегистрСведений» и т.д.)

3. Многократное выполнение запросов

Рекомендуется получать все необходимые однотипные данные одним запросом вместо выполнения серии запросов.

4. Проверка на пустой результат выполнения запросов

Проверку того, что результат выполнения запроса не содержит строк, следует выполнять с помощью метода Пустой(). Поскольку на получение выборки из результата запроса (выгрузка его в таблицу значений) будет затрачиваться дополнительное время.

5. Оформление текстов запросов

1. Все ключевые слова языка запросов пишутся заглавными буквами.
2. Рекомендуется указывать и необязательные конструкции запроса, прежде всего — явно назначать псевдонимы полям в целях повышения наглядности текста запроса и «устойчивости» использующего его кода. Например, если в алгоритме используется запрос с полем, объявленным как Касса.Валюта, при изменении имени реквизита нужно будет также изменить и код, осуществляющий обращение по имени свойства «Валюта» к выборке из результата запроса. Если же поле будет объявлено как Касса.Валюта КАК Валюта, то изменение имени реквизита приведет только к изменению текста запроса.

<t> (+7 495) 730 52-26
<@> HELLO@P-24.RU
<address> 115054, Г. МОСКВА,
3-Й МОНЕТЧИКОВСКИЙ ПЕР. 11 СТР. 1

3. Особенно внимательно следует относиться к автоматически присваиваемым псевдонимам для полей – реквизитов других полей, типа «… Касса.Валюта.Наименование…». В приведенном выше примере поле получит автоматический псевдоним «ВалютаНаименование», а не «Наименование».
4. Следует обязательно указывать ключевое слово «КАК» перед псевдонимом поля источника.
5. Текст запроса должен быть структурирован, не следует писать запрос в одну строку, даже короткий. Текст запроса должен быть нагляден, поскольку это существенно улучшает его понимание другими разработчиками.
6. В запросы, сложные для понимания, в которых используются вложенные запросы, объединения или соединения, рекомендуется вставлять комментарии. Комментарии, например, могут объяснять, для получения каких данных используется та или иная таблица в соединении или объединении.
7. При создании объекта Запрос рекомендуется указывать комментарии, для получения какой информации или каких иных целей будет использован данный запрос.
8. При программной «сборке» текста запроса рекомендуется комментировать все этапы его сборки.

6. Использование ключевых слов в запросах «Объединить» и «Объединить все» в запросах

В общем случае, при объединении в запросе результатов нескольких запросов следует использовать конструкцию «ОБЪЕДИНИТЬ ВСЕ», а не «ОБЪЕДИНИТЬ».
Поскольку во втором варианте при объединении запросов полностью одинаковые строки заменяются одной, на что затрачивается дополнительное время, даже в случаях, когда одинаковых строк в запросах заведомо быть не может.
Исключением являются ситуации, когда выполнение замены нескольких одинаковых строк одной является необходимым условием выполнения запроса.

7. Использование строк неограниченной длины

Для хранения строк, максимально возможная длина которых заведомо известна, используются строковые реквизиты с длиной, равной максимально возможной.
Когда максимально возможная длина строки неизвестна, для хранения используются строковые реквизиты неограниченной длины.
При этом следует помнить о некоторых ограничениях, возникающих при использовании полей неограниченной длины в языке запросов.
В частности, при вычислении выражений в запросах такие реквизиты необходимо выражать как строку определенной длины — такой, чтобы выражение было вычислено верно.

8. Программное управление формой

Обращение к процедурам, функциям, реквизитам, свойствам и методам, доступным для формы, из модуля этой формы происходит напрямую, без использования объекта «ЭтаФорма», кроме случаев, когда такое обращение не может быть выполнено иначе.
В разделе инициализации модуля формы запрещается открывать другие формы или диалоги (например, операторами Вопрос(), Предупреждение() и т. д.).

<t> (+7 495) 730 52-26
<@> HELLO@P-24.RU
<address> 115054, Г. МОСКВА,
3-Й МОНЕТЧИКОВСКИЙ ПЕР. 11 СТР. 1

Программное управление формой из других модулей производится через присвоение её реквизитам (свойствам) значений и через вызов её методов или экспортных процедур (функций). Не допускается делать предположения о свойствах реквизитов формы.
Не допускается обращение к элементам формы не из модуля этой формы: ни непосредственно, ни при помощи перебора коллекции «ЭлементыФормы», ни каким-либо другим способом. Например, вполне корректно предполагать, что у формы элемента справочника есть свойство «ПараметрОснование», однако предположение о наличии у «ПараметрОснование» свойства «Дата» уже недопустимо.

9. Обращение к данным информационной базы в обработчиках часто вызываемых событий

Следует минимизировать обращение к данным информационной базы в обработчиках событий, приведенных ниже, поскольку это может существенно замедлить интерактивную работу.
События формы:

- ОбновлениеОтображения().

События табличного поля:

- ПриВыводеСтроки(),
- ПриАктивацииСтроки().

В качестве средств минимизации в зависимости от ситуации могут быть:

- использование переменных модуля формы для кеширования данных;
- перенос обработки данных в обработчики других событий;
- для таблиц значений — получение необходимых данных на этапе заполнения;
- любые другие методы.

10. Обращение к свойству «ТекущаяСтрока» табличного поля

Запрещается использовать свойство «ТекущаяСтрока» для получения значений полей строки табличного поля.
Обращение к данным значениям должно выполняться через «ТекущиеДанные» или «ДанныеСтроки».
Правильно:

```
ИнформационнаяНадписьАдрес = "Адрес: " + Элемент.ТекущиеДанные.Адрес;
```

Неправильно:

```
ИнформационнаяНадписьАдрес = "Адрес: " + Элемент.ТекущаяСтрока.Адрес;
```

При этом следует учитывать, что для динамических списков обращения к значениям полей с помощью свойства «ТекущиеДанные» зависит от видимости соответствующих колонок в списке. Поэтому необходимо явно добавлять данные колонки в источник данных табличного поля перед открытием формы, например:

```
СправочникСписок.Колонки.Добавить("Адрес");
```

Данное правило не относится к полям, необходимым для функционирования динамических списков и расширений табличного поля (т.н. системные поля, например: «ПометкаУдаления», «ЭтоГруппа», «Дата» и т.д.). Такие поля являются всегда доступными и не удаляются табличным полем из коллекции колонок динамического списка при изменении видимости или удалении колонок табличного поля.

<t> (+7 495) 730 52-26
<@> HELLO@P-24.RU

<address> 115054, Г. МОСКВА,
3-Й МОНЕТЧИКОВСКИЙ ПЕР. 11 СТР. 1

11. Требования по локализации модулей

Если в модулях конфигурации встречаются строки, предназначенные для пользовательского интерфейса (сообщения пользователю, надписи в формах, названия и подсказки команд и т.п.), необходимо обеспечить возможность локализации таких строк.

Для этого необходимо применять функцию НСтр() вместо прямого использования строковых литералов. Иное использование строк, предназначенных для пользовательского интерфейса, не допускается.

В том случае, если строка является составной и включает в себя части, зависящие от тех или иных условий, настоятельно рекомендуется использовать логически завершенные, целостные фразы. Для формирования переменной составляющей строки при этом необходимо применять замену подстрок по определенным правилам. При этом можно использовать как функцию СтрЗаменить, так и предусмотреть в конфигурации специально предназначенную для этого функцию.

Неправильно:

```
СообщениеОНехватке = "Не хватает товара " + НаименованиеТовара + " на складе " + НаименованиеСклада + "!";
```

Правильно:

```
СообщениеОНехватке = НСтр("ru='Не хватает товара %Товар% на складе %Склад%!'")

СтрЗаменить(СообщениеОНехватке, "%Товар%", НаименованиеТовара);

СтрЗаменить(СообщениеОНехватке, "%Склад%", НаименованиеСклада);
```

В функции НСтр() строка ограничивается символами одинарных кавычек.

Такое требование обусловлено частым использованием двойных кавычек в строковых литералах.

Правильно:

```
Сообщить(НСтр("ru='Переменная типа ""Строка"""'));
```

Неправильно:

```
Сообщить(НСтр("ru=Переменная типа ""Строка"""));

Сообщить(НСтр("ru=""Переменная типа ""Строка"""""));
```

В том случае, если все же применяется не замена строк в строке-шаблоне, а сложение строк, то неязыковые символы (пробелы, табуляция и пр.) в начале и конце строк необходимо выделять в отдельные строковые литералы.

Правильно:

```
ТекстСообщения = НСтр("ru = 'Остаток выданных подотчетому лицу денежных средств:'") + " " +
ВыборкаИзРезультатаЗапроса.ОстатокУПодотчетногоЛица;
```

Неправильно:

```
ТекстСообщения = НСтр("ru = 'Остаток выданных подотчетому лицу денежных средств: '") +
ВыборкаИзРезультатаЗапроса.ОстатокУПодотчетногоЛица;
```

В редких случаях строковые литералы из текстов запросов также могут оказаться частью пользовательского интерфейса. В таких случаях строковые литералы необходимо выносить из текста запроса в параметры.

Правильно:

```
ЗапросПоВерсиям = Новый Запрос("
|ВЫБРАТЬ
|Версии.Ссылка,
|ВЫБОР КОГДА Версии.Выпущена = ИСТИНА
|    ТОГДА &amp;ТекстВыпущеннойВерсии
|    ИНАЧЕ &amp;ТекстНеВыпущеннойВерсии
|КОНЕЦ КАК ТекстПояснения
| ИЗ
```

<t> (+7 495) 730 52-26
<@> HELLO@P-24.RU
<address> 115054, Г. МОСКВА,
3-Й МОНЕТЧИКОВСКИЙ ПЕР. 11 СТР. 1

```
|    Справочник.Версии КАК Версии");
ЗапросПоВерсиям.УстановитьПараметр("&amp;ТекстВыпущеннойВерсии", НСтр("ru='(выпущена)'"));
ЗапросПоВерсиям.УстановитьПараметр("&amp;ТекстНеВыпущеннойВерсии", НСтр("ru='(в разработке)'"));
```

Неправильно:

```
ЗапросПоВерсиям = Новый Запрос("
|ВЫБРАТЬ
|Версии.Ссылка,
|ВЫБОР КОГДА Версии.Выпущена = ИСТИНА
|    ТОГДА ""(выпущена)""
|    ИНАЧЕ ""(в разработке)""
|КОНЕЦ КАК ТекстПояснения
| ИЗ
|    Справочник.Версии КАК Версии");
```

12. Тексты модулей

Тексты модулей должны быть написаны на русском языке.

Размер табуляции стандартный (4 символа).

Программные модули не должны иметь неиспользуемых процедур и функций.

Программные модули не должны иметь закомментированных фрагментов кода.

Тексты модулей оформляются по принципу «один оператор в одной строке». Наличие нескольких операторов допускается только для «однотипных» операторов присваивания, например:

```
А = 0; Б = 0; С = 0;
```

Текст модуля должен быть оформлен синтаксическим отступом. Для синтаксического отступа используется табуляция.

С крайней левой позиции должны начинаться только:

- операторы Процедура, КонецПроцедуры, Функция, КонецФункции;
- операторы предварительного объявления процедур и функций;
- заголовки(описания) процедур и функций;
- объявление переменных модуля;
- операторы «раздела основной программы» (с учетом синтаксического отступа).

Процедуры НачатьТранзакцию() и ЗафиксироватьТранзакцию() не являются операторными скобками, поэтому текст внутри этих процедур не сдвигается.

При длине строки более 120 символов следует использовать переносы. Строки более 120 символов делать не рекомендуется, за исключением тех случаев, когда перенос невозможен.

Тексты модулей должны содержать комментарии.

Небольшие комментарии пишутся в конце строки, которую комментируют, например:

```
НайденныеОшибки.Колонки.Добавить("Номер"); // для совместимости
```

Большие комментарии или комментарии к фрагменту кода пишутся перед комментируемым кодом в отдельной строке.

Комментарии записываются по правилам русского языка, то есть должны начинаться с большой буквы и заканчиваться точкой. Текст выравнивается по левой границе комментируемого фрагмента.

Между символами комментария «//» и текстом комментария должен быть пробел.

```
// Инициализируем переменные для выполнения расчетов.
ТекущаяДата    = ОбщегоНазначения.ПолучитьРабочуюДату();
ТекущийГод     = Год(ТекущаяДата);
```

<t> (+7 495) 730 52-26
<@> HELLO@P-24.RU
<address> 115054, Г. МОСКВА,
3-Й МОНЕТЧИКОВСКИЙ ПЕР. 11 СТР. 1

13. Структура модулей

В программном модуле в общем случае могут присутствовать следующие разделы в приведенной ниже последовательности:

- заголовок модуля;
- раздел описания переменных;
- процедуры и функции модуля;
- обработчики событий элементов формы;
- обработчики событий;
- раздел инициализации.

Некоторые разделы могут присутствовать только в модулях определенного вида. Например, обработчики событий элементов форм могут присутствовать только в модулях форм, а раздел описания переменных и раздел инициализации не могут быть определены в неглобальных общих модулях, модулях менеджеров объектов, наборов записей, значений констант и модуле сеанса. Допускается располагать процедуры и функции в особом порядке, если такое расположение лучше иллюстрирует логику работы модуля. Например, если некоторая процедура вызывается из обработчика события элемента формы, то правильным будет расположить их одну за другой, а не «разносить» в разные разделы.

Заголовок модуля

Заголовок модуля представляет собой комментарий в самом начале модуля.
В заголовке модуля приводится его краткое описание и условия применения.
Для общих модулей заголовок является обязательным.

Раздел описания переменных

Переменным модуля обычного приложения, модуля управляемого приложения и внешнего соединения назначается префикс «гл».
Экспортные переменные модуля должны быть снабжены комментарием, достаточным для понимания их назначения. Для не экспортных переменных наличие комментария желательно, но не обязательно.
Комментарий рекомендуется размещать в той же строке, где объявляется переменная.
Пример:

```
Перем глВалютаУчета Экспорт;    // Валюта, в которой ведется учет
Перем глАдресПоддержки Экспорт; // Адрес электронной почты, куда направляются сообщения об ошибках
```

Процедуры и функции модуля

Процедуры и функции, которые не являются обработчиками событий, размещаются сразу же после описания переменных. Процедуры и функции, связанные между собой по характеру работы или логике работы, рекомендуется располагать вместе.

Обработчики событий элементов формы

После процедур и функций в модуле формы располагают обработчики событий элементов формы. Рекомендуется обработчики одного элемента формы располагать вместе, придерживаясь при этом порядка их следования в описании встроенного языка.
У каждого события должен быть свой обработчик. Если одинаковые действия должны выполняться при возникновении событий в разных элементах формы, следует:

- создать отдельную процедуру (функцию), выполняющую необходимые действия;
- для каждого элемента формы создать отдельный обработчик с именем, назначаемым по умолчанию;
- из каждого обработчика вызвать требуемую процедуру (функцию).

<t> (+7 495) 730 52-26
<@> HELLO@P-24.RU
<address> 115054, Г. МОСКВА,
3-Й МОНЕТЧИКОВСКИЙ ПЕР. 11 СТР. 1

Обработчики событий

Последними из процедур располагаются обработчики событий модуля (формы, объекта, менеджера объекта и т.д.). Для них также рекомендуется придерживаться порядка следования, приведенного в описании встроенного языка.

Раздел инициализации

Раздел инициализации содержит операторы, инициализирующие переменные модуля или объект (форму).

14. Описание процедур и функций

Процедуры и функции рекомендуется комментировать.
Обязательного комментирования требуют экспортные процедуры и функции.
Прочие процедуры и функции, в том числе обработчики событий, рекомендуется комментировать, если требуется пояснить назначение процедуры (функции) или особенности её работы. Если процедура (функция) не сложна для понимания и ее назначение и порядок работы следуют из ее названия и имен формальных параметров, комментарий можно не писать. Следует избегать комментариев, не дающих дополнительных пояснений о работе процедуры (функции).
Комментарий размещается перед объявлением процедуры(функции) и имеет следующий формат:
Секция «**Описание**»
Содержит словесное краткое описание назначения и/или принципов работы процедуры(функции).
Секция «**Параметры**»
Описывает параметры процедуры (функции). Если их нет, секция пропускается. Предваряется строкой «Параметры:».
Секция «**Возвращаемое значение**»
Описывает тип и содержание возвращаемого значения функции. Предваряется строкой «Возвращаемое значение:». Для процедур эта секция отсутствует.
Пример:

```
// Проверяет существование файлов - архивов данных в переданном каталоге.
//
// Параметры:
// ПутьККаталогу - Строка - путь к каталогу, который необходимо
// просмотреть на наличие файлов
//
// Возвращаемое значение:
// Строка.
// Полный путь к файлам на диске ИТС. Если строка пустая,
// значит файлов на диске ИТС не найдено.
//
Функция ПроверитьНаличиеФайловНаДискеИТС (знач ПутьКДискуИТС) Экспорт
```

Следует стремиться к тому, чтобы имена процедур и функций и имена формальных параметров были "говорящими" (документировали сами себя).
Правильно:

```
Функция ПроверитьТипРеквизитаОбъекта(Объект, ИмяРеквизита, ТипЗначения)
```

Неправильно:

```
Функция ВыполнитьПроверку(Параметр1, Рекв, ТЗ)
```

Имена процедур и функций в общем случае, следует образовывать от неопределенной формы глагола, например:
Правильно:

```
Процедура ЗагрузитьКонтрагента()
...
```

<t> (+7 495) 730 52-26
<@> HELLO@P-24.RU
<address> 115054, Г. МОСКВА,
3-Й МОНЕТЧИКОВСКИЙ ПЕР. 11 СТР. 1

```
Функция ПолучитьПутьКФайлуОписаниюДанных()
```
Неправильно:
```
Процедура ЗагрузкаКонтрагента()
...
Функция ПутьКФайлуОписаниюДанных()
...
Функция НамПодходит()
```
Исключение составляют функции, которые предназначены только для проверки истинности некоторого факта и которые возвращают в качестве результата проверки значение типа Булево. Имена таких функций образуются из написания проверяемого факта.
Например, если функция должна проверить, что в переданной строке присутствуют только цифры, то она может называться ТолькоЦифрыВСтроке().
Описание процедур и функций должны отделятся друг от друга в тексте модуля пустыми строками.

15. Правила образования имен переменных

Имена переменных следует образовывать от терминов предметной области таким образом, чтобы из имени переменной было понятно ее назначение.
Имена следует образовывать путем удаления пробелов между словами. При этом каждое слово в имени пишется с прописной буквы. Предлоги и местоимения из одной буквы также пишутся прописными буквами.
Пример:
```
Перем ДиалогРаботыСКаталогом;  // Диалог работы с каталогом
Перем КоличествоПачекВКоробке; // Количество пачек в коробке
```
Имена переменных запрещается начинать с подчеркивания.
Имена переменных не должны состоять из одного символа. Использование коротких имен переменных допускается только для счетчиков циклов.
Переменные, отражающие состояние некоторого флага, следует называть так, как пишется истинное значение этого флага.
Например:
```
Перем ЕстьОшибки;   // Признак наличия ошибок в процедуре.
Перем ЭтоТоварТара; // Признак, что товар относится к возвратной таре.
```

16. Перенос выражений

Длинные арифметические выражения переносятся следующим образом:

- в одной строке может находиться более одного операнда;
- при переносе знаки операции пишутся в начале строки (а не в конце предыдущей строки);
- операнды выравниваются по началу первого операнда, без учета знаков операций.

Пример
```
ВременнаяСтрока = ВременнаяСтрока
+ ", НП (в т. ч.): "
+ обФорматСумм(Спецификация.Итог("СуммаНП"));
```

При необходимости параметры процедур, функций, методов могут переноситься следующим образом:

- параметры выравниваются по началу первого параметра;

<t> (+7 495) 730 52-26
<@> HELLO@P-24.RU
<address> 115054, Г. МОСКВА,
3-Й МОНЕТЧИКОВСКИЙ ПЕР. 11 СТР. 1

- закрывающая скобка и разделитель операторов «;» пишутся в той же строке, что и последний параметр

Пример

```
СписокВидов = Новый СписокЗначений;
СписокВидов.Добавить(Метаданные.Документы.СтрокаВыпискиРасход.Имя,
Метаданные.Документы.СтрокаВыпискиРасход.Синоним);
СписокВидов.Добавить(Метаданные.Документы.РасходныйКассовыйОрдер.Имя,
Метаданные.Документы.РасходныйКассовыйОрдер.Синоним);
```

Сложные логические условия в Если…ИначеЕсли…КонецЕсли могут переноситься следующим образом:

- каждое элементарное условие может начинать новую строку:
- логические операторы И, ИЛИ ставятся в начале строки, а не в конце предыдущей строки;
- все условия выравниваются по началу первого условия, без учета логического оператора;
- ключевое слово Тогда пишется на той же строке, что и последнее условие.

Пример

```
Если (ВидОперации = Перечисления.ВидыОперацийПоступлениеМПЗ.ПоступлениеРозница)
ИЛИ (ВидОперации = Перечисления.ВидыОперацийПоступлениеМПЗ.ПоступлениеРозницаКомиссия) Тогда
Возврат Истина;
КонецЕсли;
```

17. Определение типа значения переменной

Определение типа значения переменной необходимо выполнять путем его сравнения с типом, а не каким-либо другим методом.

Правильно:

```
Если ТипЗнч(Ссылка) = Тип("ДокументСсылка.ПоступлениеТоваровУслуг") Тогда
```

Неправильно:

```
Если Ссылка.Метаданные().Имя = "ПоступлениеТоваровУслуг" Тогда
```

18. Получение метаданных объектов

Получение метаданных объекта конфигурации следует выполнять с помощью метода Метаданные() этого объекта, а не путем обращения к свойству глобального контекста «Метаданные», так как второй способ существенно более медленный.

Правильно:

```
СправочникОбъект.Метаданные()
```

Неправильно:

```
Метаданные.Справочники[ИмяСправочника]
```

19. Сортировка таблиц значений

В тех случаях, когда для таблицы значений применяется сортировка по колонкам, содержащим ссылочные значения, необходимо учитывать, что при этом для каждой из этих колонок для всех строк таблицы значений системой будет выполнено обращение к информационной базе за представлением этой ссылки.

Поэтому для таблиц с большим количеством (несколько сотен и тысяч) строк, особенно в алгоритмах, критических ко времени исполнения, рекомендуется сразу, на этапе заполнения, добавлять в таблицу дополнительные колонки с представлениями и сортировку выполнять уже по

<t> (+7 495) 730 52-26
<@> HELLO@P-24.RU
<address> 115054, Г. МОСКВА,
3-Й МОНЕТЧИКОВСКИЙ ПЕР. 11 СТР. 1

ним. Если, конечно, это не вызовет аналогичных многократных обращений к информационной базе.

20. Использование объекта РегистрСведенийМенеджерЗаписи

Чтение записи (набора записей) из регистра сведений без последующей модификации необходимо выполнять запросом.
Во всех остальных случаях объект РегистрСведенийМенеджерЗаписи следует применять только тогда, когда выполнение операций с регистром сведений требует использования отбора одновременно по всем измерениям. При этом менеджер записи использует для выполнения записи два набора записей, устанавливая им соответствующие значения отборов. Поэтому обработчики событий набора записей вызываются и тогда, когда для записи данных используется менеджер записи.
В остальных случаях следует использовать объект РегистрСведенийНаборЗаписей. С точки зрения производительности использование менеджера записей в некоторых случаях будет столь же эффективным, как и использование набора записей, а в некоторых — менее, так как будут выполняться лишние действия.
Правильно:

```
Набор = РегистрыСведений.ЗначенияПравПользователя.СоздатьНаборЗаписей();
Набор.Отбор.НаборПрав.Установить(ЗначениеНабораПрав);
Для Каждого СтрокаТаблицы ИЗ ТаблицаЗначенийПрав  Цикл
Запись          = Набор.Добавить();
Запись.НаборПрав = ЗначениеНабораПрав;
Запись.Право     = СтрокаТаблицы.Право;
Запись.Значение  = СтрокаТаблицы.Значение;
КонецЦикла;
Набор.Записать();
```

Неправильно:

```
Для Каждого СтрокаТаблицы ИЗ ТаблицаЗначенийПрав  Цикл
ЭлементРегистраСведений = РегистрыСведений.ЗначенияПравПользователя.СоздатьМенеджерЗаписи();
ЭлементРегистраСведений.НаборПрав = ЗначениеНабораПрав;
ЭлементРегистраСведений.Право     = СтрокаТаблицы.Право;
ЭлементРегистраСведений.Значение  = СтрокаТаблицы.Значение;
ЭлементРегистраСведений.Записать();
КонецЦикла;
```

21. Копирование строк между таблицами значений произвольной структуры

При копировании строк между различными таблицами значений (табличными частями и т.п.) со схожим составом колонок следует использовать метод глобального контекста ЗаполнитьЗначенияСвойств().
Алгоритмы, использующие данный метод, значительно эффективнее, например, многократного перебора колонок таблицы значений, выполняемого для получения их состава.
Правильно:

```
Для каждого СтрокаТаблицыИсточника Из ТаблицаИсточник Цикл
СтрокаТаблицыПриемника = ТаблицаПриемник.Добавить();
ЗаполнитьЗначенияСвойств(СтрокаТаблицыПриемника, СтрокаТаблицыИсточника);
КонецЦикла;
```

<t> (+7 495) 730 52-26
<@> HELLO@P-24.RU
<address> 115054, Г. МОСКВА,
3-Й МОНЕТЧИКОВСКИЙ ПЕР. 11 СТР. 1

Неправильно:

```
Для каждого СтрокаТаблицыИсточника Из ТаблицаИсточник Цикл
СтрокаТаблицыПриемника = ТаблицаПриемник.Добавить();
Для каждого Колонка Из ТаблицаПриемник.Колонки Цикл
КолонкаТаблицыИсточника = ТаблицаИсточник.Колонки.Найти(Колонка.Имя);
Если КолонкаТаблицыИсточника <> Неопределено Тогда
СтрокаТаблицыПриемника[Колонка.Имя] = СтрокаТаблицыИсточника[Колонка.Имя];
КонецЕсли;
КонецЦикла;
КонецЦикла;
```

22. Получение представлений для ссылочных значений в табличном документе

При формировании табличного документа запрещено в качестве параметров ячеек с типом заполнения «Параметр» указывать ссылочные значения, поскольку в этом случае в момент вывода данных в табличный документ будет выполнено многократное обращение к базе данных для получения представлений этих значений.
Поэтому в качестве параметров следует указывать сами представления.
Исключением могут быть случаи, когда для получения представлений придется выполнять аналогичное многократное обращение к базе данных.
При этом следует иметь в виду, что при получении представлений для полей непосредственно в самом запросе (через поле Представление или функцией Представление(<Имя поля>)) выполняется неявное соединение с таблицей объекта, для которого получаются представления. Для полей составного типа — несколько соединений, для каждого из типов, входящих в состав.
Это может приводить к увеличению времени выполнения запроса (и как следствие, общего времени формирования итогового документа), а при большом количестве типов – к невозможности его выполнения в клиент-серверной версии из-за ограничения Microsoft SQL Server, по которому в запросе не может участвовать больше 256 таблиц. Такие случаи также могут быть исключением для данного правила, в них представления для ссылочных значений допускается получать в момент их вывода в табличный документ.
Поскольку однозначно рекомендовать, какой из способов получения представлений следует выбрать, нельзя, такой выбор должен делаться разработчиком самостоятельно, на основании данных, полученных экспериментально.

23. Программное создание прикладных объектов

Для программного создания прикладных объектов следует использовать методы соответствующих менеджеров (СоздатьЭлемент(), СоздатьДокумент(), СоздатьНаборЗаписей() и т.д.)
Для программного создания прикладных объектов, у которых существует соответствующие менеджеры объектов, использование конструктора (оператор встроенного языка Новый) запрещается.
Правильно:

```
ДокументПриходная = Документы.ПоступлениеТоваровУслуг.СоздатьДокумент();
```

Неправильно:

```
ДокументПриходная = Новый("ДокументОбъект.ПоступлениеТоваровУслуг");
```

<t> (+7 495) 730 52-26
<@> HELLO@P-24.RU

<address> 115054, Г. МОСКВА,
3-Й МОНЕТЧИКОВСКИЙ ПЕР. 11 СТР. 1

24. Особенности контекстного выполнения на сервере и в режиме внешнего соединения

При разработке кода общего модуля и модулей объектов, которые должны быть доступны на сервере и во внешнем соединении, следует соблюдать следующие правила:

1. Запрещено использование объектов, имеющих тип данных, недоступный на сервере и во внешнем соединении:

- ТабличныйДокумент
- ТекстовыйДокумент
- ДиалогВыбораФайла
- все другие типы, использование которых невозможно на сервере 1С:Предприятие и во внешнем соединении.

2. Запрещено использование средств, отвечающих за диалог с пользователем:

- Предупреждение()
- Вопрос()
- методы работы с формами и прочие, для которых специально указано (в документации), что они не доступны на сервере и/или во внешнем соединении.

3. Запрещается вызов экспортных процедур других общий модулей, у которых не установлен признак компиляции на сервере и/или во внешнем соединении.

4. Участки кода, в которых используются конструкции, не доступные на сервере или во внешнем соединении, должны выделяться соответствующими инструкциями препроцессору, например:

```
#Если Клиент Тогда
Сообщить(Сообщение);
#КонецЕсли
```

5. При написании кода модулей объектов, которые исполняются на сервере или доступны во внешнем соединении, недопустимо использовать переменные, процедуры и функции, которые определены в модуле обычного приложения и в модуле управляемого приложения.

6. Для сервера: Надо учитывать, что при передаче управления с клиента на сервер, а также в обратную сторону, существует ограничение на тип передаваемых параметров. Поэтому в качестве параметров процедур (функций), а также возвращаемых значений функций, выполняемых на сервере, следует использовать значения примитивных типов, ссылки на объекты базы данных, системные перечисления, уникальный идентификатор, результат запроса, хранилище значения, таблицу значений, массив, структуру и соответствие, при этом состав передаваемых коллекций также должен удовлетворять приведенным выше ограничениям.

7. Для внешнего соединения: Текст модулей объектов следует писать таким образом, чтобы при работе во внешнем соединении (в частности, при работе WEB-приложения), обеспечивалась работоспособность всей прикладной логики с учетом того, что часть объектов недоступна для использования во внешнем соединении, например, использование средств диалога с пользователем. Недопустимо размещать в общих модулях процедуры и функции, которые недоступны во внешнем соединении и без которых невозможна запланированная методика использования и работы объектов.