Интероперабельная среда разработки интерфейса пользователя «Live universalInterface»

Руководство программиста

Оглавление

Введение

Описываемый программный продукт родился и развивался в недрах большой, долгоживущей, тиражной системы для автоматизации деятельности предприятий связи – ACP Fastcom.

Нас, разработчиков этой Автоматизированной Системы Расчётов, не мог не беспокоить вопрос конструирования и применения единого принципа реализации интерфейса пользователя. Самое естественное было бы выбрать самую подходящую из имеющихся на рынке систем построения интерфейса. В большинстве случаев так и поступают – выбирают систему, с помощью которой быстрее всего достигаются поставленные цели. Но в этих случаях цель, как правило, выполнение одного конкретного контракта. В нашем случае цель гораздо шире: Выполнение множества контрактов с разнообразнейшими требованиями, меняющимися со временем согласно конъюнктуре бизнеса и изменению конкурентной среды. Если конкретизировать эти требования, то получится следующее:

1. Интерфейс должен легко адаптироваться для отображения на любых существующих и появляющихся в будущем интерактивных системах взаимодействия с пользователем;
2. Интерфейс должен быть гибким, легко настраиваемым под потребности конкретного заказчика и, более того, настраиваемым под меняющиеся со временем потребности конкретного заказчика;
3. Интерфейс должен быть адаптивным к данным, то есть уметь автоматически изменяться в зависимости от отображаемых данных;
4. Разработчики бизнес-логики должны пользоваться таким инструментом построения интерфейсных форм, который не зависел от какой бы то ни было системы взаимодействия с пользователем.

Основная идея, проходящая красной нитью через всю концепцию universalInterface – это чёткое разграничение деятельности программиста прикладной системы и разработчика средства отображения интерфейса («движок»). Второй интерпретирует результаты деятельности первого. Обмен происходит на уровне абстрактных понятий, универсальных для любого средства разработки и отображения интерфейса. Разработчик прикладной системы только обозначает – какую функциональность он ожидает от интерфейса, а как и какими средствами эта функциональность будет реализована – не его забота. Таким образом, разработка прикладной системы оказывается полностью отделённой от средства (среды и языка) разработки «движка» или нескольких «движков», которыми осуществляется и будет осуществляться в будущем интерпретация труда программиста.

Как же достигается поставленная цель?

Первое – как уже сказано, абстрагирование свойств элементов интерфейса. Для примера: чтобы визуально выделить элемент среди других элементов интерфейса разработчик прикладной системы не оперирует такими свойствами, как шрифт (вдруг «движок» алфавитно-цифровой?), цвет (вдруг «движок» монохромный?), фон и т.п. Он оперирует ролями, о которых существует «договорённость» с разработчиком «движка»: «Внимание», «Опасность», «Незаметный» и т.п. А уже разработчик конкретного движка на свой вкус и, исходя их предоставленных ему инструментария, интерпретирует эти роли.

При существенных расхождениях во мнении об интерфейсных реализациях свойств элементов интерфейса разработчик «движка» реализует средства кастомизации (настройки) под конкретный Проект или даже под любого Пользователя внутри Проекта.

Второе – это уход при разработке прикладного кода от понятия «Событие». При описании интерфейса нельзя мыслить в категории событий и триггеров, так как они являются очень специфическими для конкретного «движка», разработанного в конкретной среде. Продвинутые средства поддерживают множество типов событий, а другие – не так много, но зато берут обработку части событий на себя. В одних средах под одним и тем же событием подразумевается одно, а в других – другое. Одни считают, что при каком-то событии надо произвести такой набор стандартных действий, а другие – совсем иной.

Поэтому, в universalInterface многообразие отношений интерфейсных элементов реализуется не описанием обработчиков событий, а декларативными динамическими свойствами. То есть свойства элементов интерфейса могут изменяться как при синтезе интерфейсных форм движком, так и уже в процессе их функционирования – в зависимости от входных параметров, значений контекстных элементов, параметров контента, изменяемых пользователем элементов, а также результатов запроса к прикладной системе (бизнес-логика).

Для реализации данной концепции вполне достаточно декларативного языка – SQL, но также подойдёт и процедурный язык – PL/SQL. В universalInterface описание динамических свойств производится так:

Динамикой является всё, что заключено в фигурные скобки. Их наличие – указание интерпретатору, что в некоторый момент (какой – решает «движок») содержание фигурных скобок должно быть проинтерпретировано, вычислено и заменено (включая и сами скобки) на результат вычисления.

Пример динамического свойства – заголовка таблички с объектами договора:

```
Объекты договор{Select NVL(Max('а №'||CONTR_NO), 'ов') from CONTRACTS where ID=0{CONTRID}}
```

Это свойство содержит динамический фрагмент в виде запроса к базе данных, опирающийся на входной параметр CONTRID. Ноль перед параметром страхует от отсутствия значения параметра CONTRID.

Динамические фрагменты могут быть любой глубины вложенности друг в друга. Определение момента вычисления и перевычисления динамических фрагментов – задача «движка». Отслеживание зависимостей динамических фрагментов от ссылок на меняющиеся элементы – тоже задача разработчика «движка». Разработка «движка» - вообще, довольно сложная работа, так как с одной стороны нельзя упустить момент изменения данных, от которых зависят значения с динамическими фрагментами, с другой стороны необходимо минимизировать трафик от сервера к клиенту и не делать лишних перевычислений динамических фрагментов.

Базовые принципы universalInterface

1. Алгоритмы изменений интерфейсных элементов в процессе работы форм описываются декларациями динамических свойств элементов форм, а не процедурами обработки событий в формах. Движок обеспечивает адекватность отображения текущего значения свойства каждого элемента в любой момент времени.
2. Набор свойств элемента фиксирован и согласован. То есть, не должно быть таких свойств элемента, значения которых могут противоречить друг другу или быть несовместимыми.
3. Элементы и их свойства должны быть достаточно абстрактными, чтобы не привязываться к конкретной реализации интерфейсного движка. Иными словами, свойства оперируют понятиями, а не физическими устройствами.
4. Программы на языке, обеспечивающем декларации динамических фрагментов, не должны возбуждать необработанных исключений и ошибок. В любой момент времени вычисление динамического фрагмента должно возвращать какое-то значение без возникновения ошибки.
5. Динамические фрагменты должны вычисляться только по мере необходимости и только непосредственно в момент их востребованности.
6. При обработке клиентского события единожды вычисленный динамический фрагмент не перевычисляется при повторных использованиях свойствах, где указан этот или текстуально равный динамический фрагмент.
7. Движок при реакции на событие должен передавать на клиента команды по изменению интерфейсных элементов, минимизируя трафик – то есть, отправляться должны только новые или реально изменившиеся свойства.

Концепции universalInterface

Интерфейс умеет отображать ***Формы*** трёх типов:

- ***Списки***;
- ***Бланки***;
- ***Иерархические структуры***.

Формы могут принимать набор ***Входных параметров***, которые влияют на внешний вид и логику работы формы. Для некоторых входных параметров можно указать значение по умолчанию, которое будет присвоено параметру, если родительская форма при вызове не укажет значение этого параметра.

Каждая форма содержит набор визуальных элементов:

- ***Строк списка***;
- ***Заголовков столбцов***;
- ***Ячеек строк***;
- ***Полей ввода*** (разного типа);
- ***Кнопочек***.

Все визуальные элементы могут группироваться при помощи ***Групп***, которые, впрочем, тоже могут группироваться в Группы более высокого уровня.

Каждая форма может активировать ***Действия*** различными способами:

- Клик левой клавишей мыши на интерфейсной кнопочке;
- Выбор пункта ***Локального меню***;
- Ролевая клавиша клавиатуры: ***Enter***, ***Insert***, ***Delete***;
- Двойной клик левой клавишей мыши;
- Попытка завершения работы формы;
- Тик таймера;
- Автоматическое выполнение.

Посредством действий может не только выполняться функционал бизнес-логики, но и вызываться дочерние формы. Все вызываемые формы открываются в модальном (по отношению к вызывающей форме) режиме – то есть, пока дочерняя форма не завершится, в родительской форме не обрабатываются никакие события. Исключение – формы, открываемые автоматически выполняемым действием. В этом случае форма открывается не в эксклюзивном режиме – то есть, в вызывающей форме всё ещё обрабатываются любые события.

Действия могут иметь набор ***Выходных параметров***, значения которых становятся значениями входных параметров вызываемой формы.

В иерархических структурах некоторые Действия могут порождать вложенные элементы иерархической структуры. Именно таким образом в иерархических структурах возникает иерархия.

Любой описываемый элемент формы (столбец, поле, группа, действие, параметр), так же, как и сама форма, имеют специфический набор ***Свойств***, совокупное значение которых определяет внешний вид и способ функционирования формы.

Все наборы заданных значений свойств одной формы могут объединяться в некоторой системной ***Конфигурации***. Системные конфигурации формы могут выстраиваться в иерархию. При этом у любой формы должна обязательно быть одна конфигурация верхнего уровня с кодом DEFAULT. Не заданные значения свойств в некоторой конфигурации наследуют значение этого свойства у родительской конфигурации, а при её отсутствии – принимают значение свойства по умолчанию.

Значения свойств могут быть

- Статическими
- Динамическими

В первом случае значение свойства не меняется на протяжении всего функционирования формы, во втором случае значение свойства меняется в соответствии с декларацией и всегда поддерживается «движком» в актуальном состоянии.

Динамическое значение свойства элемента, помимо возможной статической части, содержит минимум одну динамическую составляющую (***Динамический элемент***), которая задаётся одним из следующих способов:

1. {*PARAM*} – заменяется текущим значением входного параметра с кодом `PARAM`
2. {*COL*} – заменяется текущим значением ячейки столбца с кодом `COL` из текущей (контекстной) строки Списка
3. {*FIELD*} – заменяется текущим значением поля с кодом `FIELD` бланка
4. {OLD.*FIELD*} – заменяется стартовым значением поля с кодом `FIELD` бланка
5. {Property:*PropCode*} – заменяется текущим значением другого свойства с кодом `PropCode` этого же элемента (или его родительского элемента или формы). Перечень свойств см. «Оперативные свойства Списка и его элементов» на стр.14 и «Оперативные свойства Бланка и его элементов» на стр.22.
6. {Property:Var *Variable*} – заменяется текущим значением технологической переменной `Variable` текущей формы. Перечень переменных см. «Системные переменные рабочей области Списка» на стр.14 и «Системные переменные рабочей области Бланка» на стр.22.
7. {Select *Expession* …} – заменяется на результат выполнения SQL-запроса. Если запрос возвращает более одной строки, то все `Expession` объединяются в одну строку. Запрос, не вернувший ни одной строки заменяется пусто.
8. {Begin … :*BIND* … end;} – заменяется на результирующее значение :`BIND` – переменной анонимного PL/SQL блока. BIND-переменная должна быть одна и только одна. Если динамика одни, то перед очередным вычислением такого динамического элемента BIND-переменная заполняется предыдущим значением свойства элемента.
9. {LOV:*Select* …} – из запроса формируется список, который отображается пользователю на экране в модальном режиме. Вся конструкция заменяется на значение первого выражения SQL-запроса из выбранной пользователем строки. Правила формирования LOV см. ниже.
10. {Counter:*Cnt*} – заменяется на число - увеличенное на единицу значение, полученное при предыдущем парсинге этого динамического элемента с именем `Cnt`.

Динамические элементы могут вкладываться друг в друга. В этом случае вычисление (***парсинг*** динамики) идёт, начиная с вложенных элементов. Порядок парсинга элементов одного уровня (рядом расположенных) – не определён и не гарантируется. Результатом вычисления динамического элемента может быть другой динамический элемент – он тоже будет пропарсен (рекурсия). Для маскирования парсинга результата вычисления динамического SQL и PL/SQL используется функция `UI_P_VIRTUAL.DeBraces()`.

Динамические элементы значений свойств формы вычисляются тогда и только в тот момент, когда они востребованы «движком». Если свойство действует постоянно (например, отображение текста заголовка формы), то перевычисление динамического значения будет производиться всегда в тот момент, когда и если изменятся опорные элементы, которые использовались при последнем парсинге динамического элемента (значения ячеек, полей, свойств и переменных формы).

В universalInterface существует понятие ***Идентификатора момента*** – SCN. Это число, которое меняется при каждом событии на клиенте (клик мышкой, нажатие клавиши, тик таймера). Все вычисления динамических элементов, необходимые для обработки некоторого события на клиенте происходят в контексте одного SCN. Текстуально совпадающие динамические элементы в рамках одного SCN вычисляются только один раз!

Формы universalInterface

Формы располагаются на окнах (или страницах). Все окна открываются в модальном режиме. То есть, фокус ввода находятся всегда в самой верхней форме. В одном окне может быть несколько форм разного типа, активных одновременно.

Ниже следует подробное описание форм, их свойств и ожидаемого поведения.

Список

Основное предназначение Списков – поиск и отображение данных, полученных на основании SQL-***Запроса*** при выполнении которого заполняется таблица (***Grid***). Каждый ***Столбец*** Grid является элементом интерфейса со своими свойствами. Новые строки таблицы ***подфетчиваются*** (поступают в форму из источника выполнения SQL-запроса) по мере листания списка пользователем. Количество выфетченных строк не ограниченно. Соблюдается целостность чтения данных в списке в рамках одного запроса (данные соответствуют состоянию на момент начала выполнения запроса). Строки Grid не являются элементами интерфейса, но существует понятие контекстной строки. Текущим контекстом является набор значений ***Полей*** (***Ячеек***) текущей строки, которая явно выделяется строковым курсором.

Столбцы списка могут группироваться в ***Группы***, которые имеют некоторый тип: Static, Overflow или Virtual. Столбцы из ***Static*** групп располагаются вначале Grid и не прокручиваются по горизонтали. Столбцы из ***Overflow*** групп отображаются в виде отдельных полей в области справа от Grid. Любая группа может объединять столбцы общим заголовком. Не допускается вложенность групп друг в друга. Несколько групп Overflow образуют группу закладок справа от Grid.

Пользователь может в форме-списке перейти в режим ***QBE*** (Query by Example). В этом режиме для разных полей может быть доступен перечень возможных или типичных значений. Строки Grid, отображаемые после выполнения QBE, удовлетворяют заданным условиям. Поля в Overflow группах также могут участвовать в формировании QBE.

Под таблицей (Grid) может располагаться строка с ***Итоговыми полями***. Формула вычисления итогового значения работает над всеми строками в таблице (даже формально не выфетченными), с учётом отбора, накладываемых условиями QBE.

Ячейки имеют свойство – «роль». Разные роли умеют выделять ячейки цветом, шрифтом или другими стилями.

Пользователь может перевести форму-список в режим ***многострочного выделения*** (явно или кликом мышкой с нажатым Ctrl). В этом режиме можно выделить несколько строк Grid, выделить все строки (даже не выфетченные, в этом случае подфетчиваться будут строки уже в состоянии выделения), снять все выделения и инвертировать выделения строк. В режиме текущей строки она всегда является единственной выделенной строкой.

Пользователь может выполнять ***Действия*** в форме-списке. Действия могут относиться ко всему списку («Добавить», «Найти»…), к текущей строке («Редактировать», «Показать детали»…), к нескольким строкам («Удалить», «Обработать»…) или к текущей ячейке. Активизация действия может производиться различными способами: выбором пункта из локального иерархического меню, нажатием кнопки с пиктограммой на ToolBar или в ячейке, нажатием кнопки с текстом внизу формы... Кроме того, действия могут быть привязаны и выполняться автоматически при событиях: нажатие ролевой клавиши (Insert, Delete, Enter…), перемещении по строкам, закрытии формы, по таймеру и т.п.

Действия, работающие в отношении нескольких строк в режиме многострочного выделения, могут работать и в режиме текущей строки. Например, «Удалить».

У любого действия есть признак ***доступности*** – свойство, как правило, имеющее динамическое значение, опирающееся на входные параметры и контекстные значения полей. При активизации пользователем конкретного действия (всё равно – каким способом) непосредственно перед его выполнением снова проводится контроль применимости действия (динамическое значение применимости перевычисляется).

Действия в форме-списке могут вызывать программу на процедурном языке и/или команду вызова другой формы. После выполнения Действия форма может перезапросить данные в текущей ячейке, во всей текущей строке или обновить Grid целиком (полный перезапрос). Действия могут иметь ***выходные параметры***, которые определяют значения ***входных параметров*** в вызываемой этим действием форме.

Действие в отношении ячейки может иметь специальное назначение – применение изменения текста в ячейке. В этом случае ячейка прямо в Grid является модифицируемой. При попытке уйти из ячейки выполняется действие, применяющее модифицированный текст.

Пользователь может менять ***порядок сортировки*** строк в Grid списках. Этот порядок вместе с условиями отбора в QBE может быть сохранён пользователем как персональная конфигурация списка под некоторым именем. Перечень сохранённых конфигураций всегда легкодоступен в списковой форме. Сохранённые пользовательские конфигурации могут стать доступными в основном меню, а одна из них может быть назначена как конфигурация по умолчанию и вызываться сразу при открытии формы.

Пользователи могут модифицировать ***интерфейс*** списков в разумных пределах: менять видимость, порядок и ширину столбцов, размер всего окна и пропорции областей внутри него. Эти изменения привязываются к текущей конфигурации (базовой или пользовательской) и действуют при последующих открытиях формы.

Форма-список может иметь ***входные параметры***. Ссылки на них используются в свойствах элементов формы, как правило – в динамических значениях. Входные параметры могут иметь значения по умолчанию, которые действую в случае, если форма вызывается без указания этого входного параметра. Значения по умолчанию могут быть динамическими и ссылаться на другие входные параметры.

★ Общие свойства Списка

QUERY_TEXT – ***SQL–выражение***. Основной SQL-запрос, формирующий Grid Списка. Вместо столбцов запроса необходимо указывать *, обязательно обрамлённую пробелами! Не следует указывать в запросе хинты и ORDER BY - для этого есть отдельные свойства. При выполнении этого запроса вместо звёздочки будут подставлены выражения, определяемые в каждом столбце Списка.

QUERY_HINT – ***Oracle Hint***. Пподсказка интерпретатору запросов. Наиболее часто использующиеся значения: CHOOSE, RULE, FIRST_ROWS, ALL_ROWS, FIRST_ROWS. Значение по умолчанию: FIRST_ROWS(10)

ON_LOAD – ***Код, выполняющийся при загрузке списка***. Сюда может быть помещён программный код на процедурном языке (PL/SQL), который добавляет, удаляет или модифицирует свойства элементов Списка, загруженного из метаданных.

HEADER – ***Заголовок списка***. Если список создаёт своё окно, то данный текст отображается как заголовок окна. Если это не первая форма в окне, то такой текст отображается как подзаголовок

START_MODE – ***Начальный режим выделения строк***. Каждый Список может находиться в одном из двух режимов выделения строк: SINGLE_ROW – «Выделена только текущая строка» и MULTI_ROW – «Многострочное выделение». Начальный режим выделения строк в процессе работы с формой может быть изменён пользователем. Значение по умолчанию - SINGLE_ROW.

ADD_INFO – ***Информация о записи***. Текст, который будет помещён в информационное многострочное поле под списком и который будет меняться при навигации по строкам Grid. Таким образом, для каждой строки Списка может быть определён свой дополнительный текст.

FORM_NOTES – ***Пояснения в Help***. Текст описания текущей конфигурации Списка для отображения в Help и генерации документации.

FEATURE – ***Код привязки***. Это рабочий код Списка. Используется для ассоциации с ним пользовательских интерфейсных предпочтений, а также прав доступа к элементам для групп пользователей. По умолчанию код привязки совпадает с кодом метаданных, из которых был загружен Список

★ Свойство входных параметров Списка

В Списке могут быть описаны входные параметры с единственной целью – указать значение параметров по умолчанию, так как при динамической ссылке на неопределённый входной параметр её значение останется неизвестным. Для ссылки на значение входного параметра укажите {КОД}, где КОД – это код самостоятельного элемента Списка типа PARAM - «Входной параметр».

NAME – ***Код входного параметра***. Если не пуст, то переопределяет код элемента типа PARAM. Рекомендуется всегда оставлять пустым.

VALUE – ***Значение по умолчанию***. Если вызывающая Список форма не укажет значение параметра, то при ссылке на данный входной параметр в динамических свойствах будет подставляться указанное тут зна-

чение. По умолчанию – пустое значение. Если вызывающая форма переопределила значение по умолчанию входного параметра, то указанная здесь динамика вычисляться никогда не будет.

DESCRIPTION – ***Описание***. Комментарий разработчика о входном параметре и о том, на что он влияет.

★ Свойства столбцов Списка

В Списке должны быть определены столбцы – минимум один столбец. Каждое событие в Списке обрабатывается в контексте текущей строки (если строки вообще есть). Совокупность значений ячеек текущей строки и есть контекст. Динамические свойства элементов Списка могут ссылаться на значение ячейки столбца из контекстной строки. Для этого укажите {КОД}, где КОД – это код элемента типа «Столбец». Если контекст не определён, то все ссылки на столбцы Списка возвращают пусто.

DEFINITION – ***SQL–выражение***. Выражение, которое будет добавлено в основной SQL-запрос (вместо звёздочки) через запятую, наряду с SQL-выражениями других столбцов. Значение этого выражения даёт значение ячейки для каждой строки Grid.

DATA_TYPE – ***Тип данных***. Тип данных влияет на правильность сортировки и учитывается в запросах QBE. Ссылка из динамических свойств элементов принимает текстуальное значение из соответствующей ячейки, независимо от типа данных. Возможные типы:

- CHAR – Текстовая строка (по умолчанию);
- DATE – Дата/Время;
- NUMBER – Число;
- TEXT – Многоязычный текст;
- TIMESTAMP_TZ – Точное зоновое время.

Любое другое значение интерпретируется как CHAR.

FORMAT – ***Формат данных***. Указание формата обязательно для полей с типом данных DATE и возможно для полей с типом данных NUMBER.

FOR_UNIQUE_KEY – ***Входит в уникальный ключ строки***? В любом Списке должен быть определён набор столбцов, составляющий уникальный ключ данного списка. Значения, порождённые из столбцов, входящих в уникальный ключ, используются при перезапросах (одной записи или всего списка). Возможные значения:

- Y – Столбец входит в уникальный ключ;
- Любое другое значение (включая пустое) - Столбец не входит в уникальный ключ.

По умолчанию значение N.

CAPTION – ***Заголовок столбца***. Текст над колонкой или полем (если столбец в области Overflow).

ALIGNMENT – ***Выравнивание***. Выравнивание текста внутри ячейки:

- LEFT – По левому краю (по умолчанию);
- CENTER – По центру;
- RIGHT – По правому краю.

Любое другое значение (включая пустое) эквивалентно значению LEFT.

VISIBLE – ***Доступен***? Признак доступности столбца пользователю:

- Y – столбец доступен (по умолчанию). Даже если он имеет изначально не видим, пользователь может сделать его видимым;
- Любое другое значение (включая пустое) – столбец недоступен и пользователь не видит его при настройке отображения, и не может сделать видимым.

WIDTH – ***Признак видимости столбца***. Отображать ли столбец при первом открытии Списка. Значение 0 (ноль) говорит о невидимости столбца. Любое другое значение (включая пустое) означает видимость столбца. Пользователь в процессе работы с формой может изменять видимость столбцов, а также устанавливать им ширину. По умолчанию значение 0.

VISUALIZATION – ***Код набора свойств отображения***. Способ визуально выделить ячейку с данными. По сути это – имя класса для стилевого оформления ячейки. Автоматически поддерживаются следующие варианты:

- FONT_SELECTED – Выделен текст;

- BACKGROUND_SELECTED – Выделен фон;
- ALL_SELECTED – Выделен текст и фон;
- LOWERED – слабозаметный;
- DANGER – Опасность!;
- ATTENTION – Внимание!;
- OK – Успех;
- INHERIT – Унаследованое значение;
- SECTION – Подраздел;
- START – Включить/Работает;
- INTERIM – Приостановить/Переключается;
- STOP – Остановить/Выключено.

Можно добавлять свои собственные классы визуализации.

HINT – ***Подсказка***. Текст, отображающийся в качестве пояснения к столбцу или значению ячейки столбца в конкретной записи. Рекомендуется вместить пояснения в одну ёмкую, но предельно точную фразу.

SORT – ***Порядок сортировки***. Число, отражающее порядок столбца в общей сортировке данных в Grid. Пусто - поле не участвует в сортировке. Отрицательное число - обратная сортировка. Лидирующий ноль - пустые значения в начале.

VARIANTS – ***Варианты для QBE***. Предлагает пользователю список типичных значений для данного поля в режиме задания условия Query by Example. Здесь можно указать:

- SQL - запрос. Будут отображаться только столбцы с алиасами в двойных кавычках. Выбранный результат – значение из первого столбца;
- Коллекцию вариантов в формате Значение1:Пояснение;Значение2:Пояснение;... Выбранный результат – значение выбранного варианта

Запрос и коллекцию не нужно заключать в фигурные скобки, ибо это не динамика, а, собственно, значение свойства, что не исключает использование динамики для генерации запроса или коллекции

TOTAL – ***Итоговое значение***. Значение, отображающееся в итоговой строке списка, идущей сразу за Grid вне вертикальной прокрутки. Здесь надо указать групповую SQL-функцию, например SUM(). Аргументами могут выступать коды столбцов списка (без обрамляющих скобок). Итоговая строка отображается, только если есть хоть один столбец с непустым TOTAL. По умолчанию автоматическое вычисление значений итоговой строки не гарантируются. Необходимо сделать двойной клик мышью в любой ячейке итоговой строки. Для гарантии автоматического вычисления значения итоговой строки добавьте в основной запрос (свойство DEFINITION) в любое место текст `/* CALCULATE_TOTAL_IMMEDIATE */`

ITEM_NOTES – ***Пояснения в Help***. Текст описания столбца Списка для отображения в Help и генерации документации. Следует помещать сюда всё, что не уместилось в одну фразу в свойстве HINT.

★ Группировка столбцов Списка

Столбцы списка могут объединяться в группы с различными целями. Самая простая цель – указать для группы столбцов общий заголовок. Пользователь в процессе использования Списка может самостоятельно перемещать столбцы из группы в группу. Эти перемещения запоминаются и автоматически привязываются к пользовательской конфигурации.

Свойства групп Списка:

GROUP_TYPE – ***Тип группировки***. Возможны варианты:

- FIXED – Столбцы внутри группы не прокручиваются по горизонтали – всегда видны в начале Grid. У группы возможен общий заголовок.
- OVERFLOW – Столбцы внутри группы отображаются как поля вне Grid права от него. При наличии нескольких групп типа OVERFLOW, они образуют блок закладок.
- Любое другое значение, включая пустое – Виртуальная группа обычных столбцов, создаваемая исключительно ради отображения общего заголовка.

Значение по умолчанию – Virtual.

TITLE – ***Общий заголовок***. Заголовок, объединяющий входящие в группу столбцы. Для групп типа OVERFLOW – это надпись на закладке в области переполнения (справа от Grid).

POSITION – ***Положение подписей***. Расположение подписей к полям внутри группы Owerflow:

- AUTO (по умолчанию) – решает движок, как считает более подходящим. В настоящий момент – сверху от полей;
- LEFT – слева от всех полей;
- UP – сверху от всех полей.

  Любое другое значение эквивалентно значению UP.

GROUP_NOTES – ***Пояснения в Help***. Текст описания группы столбцов для отображения в Help и генерации документации.

★ Свойства действий Списка

Действия, каким бы способом они ни активировались, выполняются в контексте текущей формы. Если только этим и ограничивается, то действие производится в отношении всего Списка (добавить новый элемент списка, изменить условия отображения всего списка, удалить, подтвердить обработать все строки и т.п.). Действия могут выполняться в контексте текущей строки (удалить строку, изменить строку, показать детали строки и т.п.). Контекст может быть расширен на несколько (выделенных) строк. Действия могут быть локализованы внутри одной ячейки некоторой строки.

PROMPT – ***Наименование действия***. Заголовок действия в локальном меню, отображаемом по нажатию правой кнопки мыши. Если PROMPT пуст, то соответствующего пункта локального меню не будет.

EXPLANATION – ***Подсказка***. Текст, отображающийся в качестве пояснения к действию. Рекомендуется вместить пояснения в одну ёмкую, но предельно точную фразу.

ACTIVE – ***Признак активности***. Это свойство проверяется как минимум 2 раза. Первый – при отображении элемента управления, активирующего данное действие (активно/не активно), второй – непосредственно перед выполнением Возможные значения:

- Y – Действие доступно для выполнения любым возможным способом активации (по умолчанию);
- EXIST – Действие применимо только к текущей записи (записям). Не применимо к списку в целом;
- Любое другое значение (включая пустое) – Действие не доступно для выполнения, каким бы способом оно ни активировалось;

DISPLAY – ***Признак отображения в локальном меню***. Можно ли активировать действие из локального меню, отображаемого по нажатию правой кнопки мыши. Если действие не активно, но отображается в меню, то текст пункта меню будет серым цветом. Возможные значения:

- Y - Действие отображается в локальном меню (при наличии непустого PROMPT);
- EXIST – Действие отображается только если существует текущая запись (есть выделенные записи);
- Любое другое значение (включая пустое) – Действие не отображается в локальном меню.

Значение по умолчанию синхронизировано со свойством ACTIVE (ссылается на значение этого свойства: `{Property: ACTIVE}`). Действия из одного десятка по порядку отделяются в локальном меню от других действий сепаратором. Действия ячеек всегда отображаются вначале меню.

В качестве значения тут можно указать запрос (без фигурных скобок), в котором алиасы столбцов должны совпадать с кодами свойства действия. Для каждой строки, порождаемой запросом, будет образовываться отдельный пункт меню (следствие: столбец PROMPT обязательно должно быть в запросе). Выбор некоторого пункта меню, порождённого этим запросом, выполняет это действие с переопределёнными запросом свойствами.

ICON – ***Иконка***. Другой способ активации действия – нажатие на иконку. Для глобальных действий иконка отображается в виде кнопочки на Tool-Bar над Grid, для действий в ячейках иконка отображается прямо в ячейке. Если данное свойство пусто, то никакая иконка не отображается и место для действия для неё резервируется. Для нескольких действий с одинаковой иконкой она отображается только в одном экземпляре. При клике на иконке выполняется первое по порядку активное действие с данной иконкой.

BUTTON – ***Текст на кнопке***. При непустом значении этого свойства внизу формы отображается кнопка с текстом, активирующая данное действие.

ROLE – ***Способы активации***. Альтернативные способы активации действия:

- DBLCLICK – Двойной клик левой кнопкой мыши или нажатие клавиши `Enter`;
- ADD – Нажатие клавиши `Insert`. По смыслу – действие добавляет строки в список;

- DELETE – Нажатие клавиши `Delete`. По смыслу – действие удаляет строки из списка;
- CLOSE – Нажатие клавиши `Esc` или закрытие окна. Выполняется также при попытке закрытия формы, каким бы способом это закрытие не выполнялось;
- AUTO – Действие, выполняющееся при каждом изменении значения любого его выходного параметра;
- TIME – Действие, выполняющееся ежесекундно по таймеру (в периоды активности формы);
- WIZARD – Действие, являющееся очередным шагом визарда (вперёд или назад);
- POSTEDIT – Действие в ячейке, выполняющееся после изменения данных в ней.

Каждый способ активации выполняет первое по порядку активное действие, которому назначен данный способ активации. Способы активации можно комбинировать, указывая их через пробел и/или запятую.

APPLY_TO – ***Применимо к***… Указание каким образом будет производиться обработка строк при выполнении действия:

- N – Действие будет применяться только к одной записи – текущей. Или же ни к какой строке, а к списку в целом. В действии возможны ссылки на текущие значения любых столбцов;
- PROGRESS – Действие будет применяться к текущей (в режиме однострочного выделения) или к выбранным (в режиме многострочного выделения) записям. В действии возможны ссылки только на столбцы из уникального ключа. Процедуры из PROC_INITIALY и PROC_FINALY будут выполняться для каждой выделенной строки, если код из них отличается раз от раза;
- Y – Действие будет применяется к нескольким записям. Ссылки на текущие значения столбцов бессмысленны. Для перебора выбранных строк используется конструкция `Select Column_Value from Table(UI_P_List.SelectedRows('{Property:FormID}'));`

Любое другое значение (включая пустое) эквивалентно значению N. По умолчанию свойство имеет значение N.

QUESTION – ***Текст предупреждения***. Если свойство не пусто, то его значение будет выдано как предупреждения с двумя кнопками. Если текст заканчивается знаком вопроса, то на кнопках будут надписи «Да» и «Нет». Иначе на кнопках отобразится «Продолжить» и «Отменить». Выбор первой кнопки позволяет продолжить выполнение действия. Выбор второй кнопки – приводит к прекращению выполнения действия и откату к началу его выполнения.

LOCK – ***Накладывать ли блокировку***. Перед выполнением действия в отношении текущей строки (APPLY_TO = N, ACTIVE = EXIST) может быть произведена попытка наложения на неё блокировки посредством выполнения основного запроса списка с дополнительными условиями на значения столбцов, входящих в уникальный ключ списка. Блокировка накладывается на таблицу, содержащую столбцы, входящие в уникальный ключ списка (FOR_UNIQUE_KEY = Y). При неудаче производится попытка наложения блокировки на записи всех таблиц из основного запроса. Если и в этом случае наложить блокировку не получилось, то блокировка не накладывается без сообщения об ошибке. Варианты:

- Y – выполнять попытку блокирования текущей записи (по умолчанию);
- Любое другое значение (включая пустое) – не выполнять попытку блокировки.

PROC_INITIALLY – ***Программный код действия***. Анонимный PL/SQL-блок, выполняющийся при активации данной команды перед (или вместо) вызовом формы, отчёта, команды OS и т.п. В случае APPLY_TO = PROGRESS данное действие выполняется для каждой выделенной строки. Программный код не нужно заключать в фигурные скобки, так как это не динамика, а, собственно, само значение свойства, что не исключает использование динамики для генерации всего кода или его части. Необходимо избегать использования тут команды Commit.

COMMAND_TYPE – ***Тип вызываемого действия***. Возможны следующие значения:

- FUNCTIONAL – Вызов функционала. COMMAND - код функционала;
- LIST – Вызов формы универсального интерфейса. COMMAND – код списка или бланка из метаданных;
- BLANC – Вызов формы универсального интерфейса. COMMAND – код списка или бланка из метаданных;
- TREE - Вызов иерархической структуры. COMMAND – код списка из метаданных, который образует элементы первого уровня иерархической структуры;
- REPGROUP – Открытие группы отчётов для выбора и отображения отчёта из этой группы. COMMAND – код группы отчётов;
- REPORT – Отображение отчёта. COMMAND – код отчёта;
- HOST – Вызов внешней команды OS. COMMAND – команда OS;

- LINK – Вызов WEB-страницы. COMMAND – URL страницы;
- UPLOAD – Загрузка файла с выбором его из файловой структуры клиентского компьютера. COMMAND – место загрузки и, возможно, имя файла;
- DOWNLOAD – Выгрузка файла на клиентский компьютер. COMMAND – место хранения и имя файла.

COMMAND – ***Объект действия***. Вызываемый объект, соответствующий типу действия, указанному в свойстве COMMAND_TYPE. Помимо традиционного способа передачи параметров (отдельными подчинёнными элементами действия типа PARAM), возможно перечисление параметров в коллекции, указанной в круглых скобках сразу после объекта - идентификатора действия: `Объект(Параметр1:Значение;Параметр2:Значение;...;)`. Даже при обработке действием нескольких строк вызов указанного тут объекта выполняется только 1 раз.

PROC_FINALLY – ***Программный код после действия***. Этот код выполняется уже после команды. В случае APPLY_TO = PROGRESS данное действие выполняется для каждой выделенной строки (если, конечно, он текстуально отличается для разных строк). При необходимости завершить транзакцию не используйте `Commit`, а используйте `{Content:COMMIT}`, так как вызывающая форма посредством свойства CHILD_COMMIT может запретить завершать транзакцию.

Если вызываемая командой COMMAND форма (Список, Бланк или Иерархическая структура) закрылась в режиме CANCEL (действие в вызываемой форме, выполняющееся по закрытию, имеет свойство AFTER = CANCEL), то код из PROC_FINALLY не выполняется, действие отменяется, а транзакция откатывается к началу действия.

CHILD_COMMIT – ***Завершать транзакцию в вызываемой форме?*** Если в вызываемой форме и всех последующих вместо Commit в коде используется `{Content:COMMIT}`, то при значении этого свойства N реальные коммиты выполняться не будут, то есть транзакции останутся не завершёнными. Завершение транзакции в этом случае берёт на себя вызывающая форма. Возможные значения:

- N – в вызываемой форме не будет завершаться транзакция (все commit не будут выполняться). Рекомендуется использовать в действиях – шагах визарда вперёд;
- Y – в вызываемой форме все commit всегда безусловно выполняются. Использовать это значение не рекомендуется без крайней на то необходимости;
- Любое другое значение (включая пустое) – режим завершения транзакций не меняется (наследуется из текущей формы);

  По умолчанию значение – пусто.

AFTER – ***Поведение Списка после выполнения действия***. После успешного выполнения действия в случае, если вызывающая форма не закрылась в режиме CANCEL, текущий список поведёт себя в соответствии с указанными тут вариантами:

- NONE – Ничего не произойдёт, даже выделенные строки не сбросятся;
- FIELD – Обновится значение текущего поля текущей строки Списка (в котором стоит курсор);
- CURRENT – Обновится вся текущая строка Списка;
- ALL – Обновятся все строки Списка, то есть сделается перезапрос Grid. Текущая строка попытается сохраниться, но это не гарантируется. Все выделенные строки сбросятся;
- REFRESH – Переинициировать Список, то есть, как бы выйти и войти с теми же значениями входных параметров. Код из свойства ON_LOAD выполнится повторно;
- EXIT – Закрыть Список и передать управление в вызывающую форму. В случае визарда закроется весь визард;
- CANCEL – Отменить вызов Списка. В вызывающей форме не будут выполняться POST-действия и перезапросы. В случае визарда откатится весь визард. Для отката только на предыдущий шаг визарда данное действие должно иметь ROLE = WIZARD.

ACTION_NOTES – ***Пояснения в Help***. Текст описания действия Списка для отображения в Help и генерации документации. Следует помещать сюда всё, что не уместилось в одну фразу в свойстве EXPLANATION.

★ Свойства выходных параметров действий Списка

Подчинённые действию элементы Списка типа PARAM – это выходные параметры действия Списка, передающие в вызываемую форму значения, которые становятся значениями её входных параметров. Если в вызываемой форме описано значение по умолчанию входного параметра, то при совпадении кода (свойство NAME выходного параметра) это значение будет переопределено тем, которое указано в выходном параметре действия вызывающей формы.

Все выходные параметры, независимо от того – описаны они как входящие в вызываемой форме, будут доступны в ней по коду, указанному в свойстве NAME выходного параметра вызывающей формы.

В случае если набор выходных параметров действия не может быть заранее определён и становится известным только в момент выполнения действия, существует альтернативный способ передачи параметров: в виде коллекции, указанной в круглых скобках сразу после кода формы в свойстве COMMAND: `КОД_ФОРМЫ(Параметр1:Значение;Параметр2:Значение;...;)`. Возможен комбинированный вариант задания выходных параметров – и посредством подчинённых элементов действия типа PARAM и коллекцией параметров в свойстве действия COMMAND. Параметры, определённые отдельными элементами, имеют приоритет.

NAME – ***Код параметра***. Реальный код выходного параметра. Именно по нему заполняется значение входного параметра в вызываемой форме. Если код пустой, то данный выходной параметр не работает (ничего в вызываемую форму передаваться не будет).

Выходной параметр с кодом CONFIG открывает вызываемую форму в указанной тут конфигурации. При отсутствии его в выходных параметрах, вызываемая форма открывается в той же конфигурации, что и вызывающая. Если в вызываемой форме нет нужной конфигурации, она открывается в конфигурации DEFAULT.

VALUE – ***Значение выходного параметра***. При использовании динамики её значение вычисляется непосредственно в момент выполнения действия. При совпадении по коду с входным параметром вызываемой формы его значение по умолчанию переопределяется данным значением выходного параметра.

DESCRIPTION – ***Комментарий***. Пояснения разработчика о значении выходного параметра.

★ Оперативные свойства Списка и его элементов

- **{Property:FormID}** – Уникальный текстовый идентификатор рабочей области, куда загружен Список. Не изменяется. Доступно в любом контексте.
- **{Property:ListCode}** – Код метаданных, откуда загружался Список в рабочую область. Для чисто программного Списка это значение совпадает с {Property:FormID}. Не изменяется. Доступно в любом контексте.
- **{Property:Code}** – Идентифицирующий код формы, определяющий её сущность. В частности, к этому коду привязываются пользовательские настройки. При загрузке Списка из метаданных свойство {Property:Code} получается из общего свойства Списка FEATURE. При создании Списка программным способом это значение указывается параметром в процедуре создания Списка. . Не изменяется. Доступно в свойствах Списка.
- **{Property:Code}** – Код текущего элемента (в свойствах элемента). Не изменяется. Доступно в свойствах элементов.
- **{Property:ConfigCode}** – Код конфигурации метаданных из которой загружался Список. «VIRTUAL», если Список чисто программный. Не изменяется. Доступно в любом контексте.
- **{Property:ConfigName}** – Наименование конфигурации метаданных из которой загружался Список (в текущем языке). «Виртуальная конфигурация», если Список чисто программный. Не изменяется. Доступно в любом контексте.
- **{Property:FType}** – Тип формы = LIST. Не изменяется. Доступно в любом контексте.
- **{Property:EType}** – Тип текущего элемента (в свойствах элемента): ITEM, ACTION, PARAM или GROUP. Не изменяется. Доступно в свойствах элементов.
- **{Property:SelectionMode}** – Режим выделения строк в Списке: Single или Multi. Доступно в любом контексте.
- **{Property:Value}** – Текущее значение ячейки столбца. Доступно в свойствах столбцов и подчинённых им элементов (локальных действий и их параметров).
- **{Property:Parent}** – Код родительского элемента. Для самостоятельных элементов тут значение «Form». При некоторых обстоятельствах может изменяться! Доступно в свойствах элементов

★ Системные переменные рабочей области Списка

- **{Property:Var TransitParam}** – Коллекция значений параметров, указанных при загрузке метаданных в рабочую область. То есть это те параметры, с которыми открывалась форма, разработанная в метаданных

- **{Property:Var ColCode}** – Код столбца, к которому относится текущая ячейка в текущей строке Списка
- **{Property:Var CurrentRow}** – Номер по порядку текущей строки Списка
- **{Property:Var FetchedRows}** – Количество выфетченных строк в гриде Списка на текущий момент.
- **{Property:Var Inversion}** – Режим инверсии выделения строк: Y – включён, все вновь выфетчевыемые строки изначально уже выделены. Любое другое значение (включая пустое) – выключен.
- **{Property:Var CountSelected}** – Количество «особых» строк – Явно выделенных вне инверсии выделения или явно развыделенных в режиме инверсии выделения
- **{Property:Var IsLastPage}** – Признак полноты полученных данных в Grod Списка: Y – все строки Списка уже выфетчены. Любое другое значение (включая пустое) – на сервере, возможно, есть ещё строки, не полученные в Grid.
- **{Property:Var Is_QBE_Mode}** – Режим Query by Example: Y – Grid Списка находится в режиме ввода условий QBE. Любое другое значение (включая пустое) – основной режим Grid.
- **{Property:Var MainQuery}** – Текст основного запроса Списка (с учётом условий QBE)
- **{Property:Var CurrentAction}** – Код текущего выполняющегося действия.
- **{Property:Var ParentWorkID}** – ID рабочей области вызывающей формы.
- **{Property:Var UserConfigName}** – Наименование текущей пользовательской конфигураци

Бланк

Форма-бланк – это набор полей ввода данных, над которыми могут производиться некоторые действия. Каждое поле бланка является отдельным элементом формы со своими свойствами.

Поля в бланках могут объединяться в группы, которые в свою очередь могут включаться в другие группы. Теоретически глубина иерархии группировки элементов формы-бланка не ограничена. Порядок следования полей и групп одного уровня вложенности определяется только их номером по порядку. Группы имеют свои свойства, основное из которых – тип. Тип группы определяет способ её отображения и расположение элементов внутри неё: Блок закладок, сворачивающаяся и разворачивающаяся область, элементы, объединённые общим заголовком и т.п.

Каждое поле бланка имеет статическое или вычисляемое значение по умолчанию. Помимо этого, для поля можно указать процедуру автоматического обновления значения. Автоматическое обновление значения поля происходит при каждом изменении любого поля, от значения которого оно зависит.

Контекстом для вычисления любых динамических свойств любых элементов бланка являются текущие значения всех полей. Перевычисление некоторого динамического свойства происходит в момент изменения значения поля. Например, после заполнения некоторого поля может открыться новая закладка или появится дополнительные поля или измениться список допустимых значений другого поля, стать активной кнопка и т.п.

Значения полей (если это позволено) могут меняться пользователем разными способами: свободный ввод с клавиатуры, выбор из списка допустимых или типичных значений, переключателями и т.п. Проверка корректности данных в процессе редактирования полей может производиться, но не должно вызывать ошибок и исключений. Окончательная проверка может быть произведена только перед или в процессе выполнения некоторого действия в бланке, например, нажатия кнопки.

Поля бланков могут поддерживать полнофункциональный редактор с форматированием текста и шрифта, вставкой мультимедиа объектов.

Действия в формах-бланках практически эквивалентны действиям в формах-списках, только с иным контекстом (там – значения полей меняющейся текущей строки, тут – изменяемые значения всех полей). Действия, как и в списках, могут активироваться различными способами: выбором пункта из локального меню, нажатием кнопки с текстом, нажатием кнопки с пиктограммой в поле... Кроме того, действия могут быть привязаны и выполняться автоматически при событиях: нажатие ролевой клавиши (такой, как Enter…), соответствие текста в поле регулярному выражению, закрытии формы, по таймеру и т.п.

Динамические свойства при вычислении элементов формы-бланка могут опираться на публичные индикаторы и флаги формы, например: Код текущего поля, код выполняемого действия, признак наличия изменений в поле или во всём бланке и т.п.

★ Общие свойства Бланка

QUERY_TEXT - ***Select заполнения***. SQL-запрос, выполняющийся при построении бланка на основе метаданных, который опирается только на значения входных параметров бланка. Если запрос возвращает одну строку, то значения столбцов первой строки станут либо значениями по умолчанию перечисленных полей бланка (если алиас столбца совпадает с кодом соответствующего поля), либо значениями виртуальных параметров бланка, на которые могут ссылаться все свойства всех элементов бланка (если алиас столбца не совпадает ни с одним кодом элемента бланка).

Если запрос не возвращает ни одной строки, то будут действовать значения по умолчанию полей, указанные в их свойствах DEFINITION.

Если запрос возвращает более одной строки, то поля, для которых все значения в возвращаемых строках совпадают, заполнятся этим значением, а поля, для которых значения различаются – становятся пустыми (`{Property:Value}` – пусто), но зато свойство `{Property:Diff}` становится равным Y. На основании этого можно выделять стилями поля, значения которых не определено.

ON_LOAD – ***Код, выполняющийся при загрузке Бланка***. Этот программный код может добавлять, удалять или модифицировать свойства элементов Бланка, загружаемого из метаданных.

HEADER – ***Заголовок Бланка***. Если бланк создаёт своё окно, то данный текст отображается как заголовок окна. Если это не первая форма в окне, то такой текст отображается как подзаголовок

FORM_NOTES – ***Пояснения в Help***. Текст описания текущей конфигурации Бланка для отображения в Help и генерации документации.

FEATURE – ***Код привязки***. Это рабочий код Бланка. Используется для ассоциации с ним пользовательских интерфейсных предпочтений, а также прав доступа к элементам для групп пользователей. По умолчанию код привязки совпадает с кодом метаданных, из которых был загружен Бланк

★ Свойства входных параметров Бланка

В Бланке могут быть описаны входные параметры с единственной целью – указать значение параметров по умолчанию, так как при динамической ссылке на неопределённый входной параметр её значение останется неизвестным. Для ссылки на значение входного параметра укажите `{КОД}`, где КОД – это код самостоятельного элемента Бланка типа PARAM - «Входной параметр».

NAME – ***Код входного параметра***. Если не пуст, то переопределяет код элемента типа PARAM. Рекомендуется всегда оставлять пустым.

VALUE – ***Значение по умолчанию***. Если вызывающая Бланк форма не укажет значение параметра, то при ссылке на данный входной параметр в динамических свойствах будет подставляться указанное тут значение. По умолчанию – пустое значение. Если вызывающая форма переопределила значение по умолчанию входного параметра, то указанная здесь динамика вычисляться никогда не будет.

DESCRIPTION – ***Описание***. Комментарий разработчика о входном параметре и о том, на что он влияет.

★ Свойства полей Бланка

Поля – это области формы, где отображается и/или может быть введена или изменена информация. У каждого поля может быть надпись, кратко поясняющая суть информации, а также набор локальных действий, доступных только в этом поле. Информация может меняться различными способами: вводом с клавиатуры, выбором значения из списка предложенных вариантов, последовательным перебором допустимых значений, посредством выполнения некоторого действия. Данные в полях могут ещё обновляться автоматически (так же как и любые свойства любых элементов), если указать алгоритм пересчёта, зависящий от других полей того же Бланка. В конечном итоге, значения полей используются в некотором действии – для DML-операций СУБД или для передачи их в другие формы интерфейса.

DEFINITION – ***Значение по умолчанию***. Это значение (статическое или вычисленное динамическое) помещается в поле в момент открытия бланка только в том случае, если SQL-запрос из свойства QUERY_TEXT бланка не возвращает строк или в нём нет столбца с алиасом, совпадающим с кодом поля. В противном случае значение этого свойства будет проигнорировано, а значением по умолчанию станет значение столбца с алиасом, совпадающим с кодом этого поля Бланка.

RENOVATION – ***Перевычисление поля***. Здесь в фигурных скобках указывается алгоритм автоматического изменения значения поля. Указанная динамика срабатывает и автоматически заменят значение поля

только в том случае, если изменится какое-то поле, явно используемое в динамическом выражении. По умолчанию (для совместимости) используется свойство DEFINITION.

DATA_TYPE – ***Тип данных поля***. Указанный тип используется в процедуре конвертации полученных значений поля по умолчанию при выполнении SQL-запроса из свойства QUERY_TEXT. Кроме того, указанный ип используется для автокоррекции введённого пользователем значения и приведения его к указанному типу по указанному формату. Помимо этого, при выполнении действия может производиться проверка на соответствие значения поля указанному типу и указанному формату (свойство FORMAT). Возможные значения:

- CHAR – Текстовая строка (значение по умолчанию). Формат (значение свойства FORMAT) интерпретируется как код алфавита, на соответствие которому будет производиться проверка вводимого текста;
- NUMBER – Число. Возможно использование формата данных (свойство FORMAT);
- DATE – Дата и/или время. Обязательно использование формата данных (свойство FORMAT);
- TIMESTAMP_TZ – Точное зоновое время. Обязательно использование формата данных (свойство FORMAT);
- TEXT – Многоязычный текст. Для каждого используемого языка будет отображено отдельное поле;
- FILE – значение поля будет выбрано из множества имён и путей к файлам, доступным с клиентского компьютера;
- RICHTEXT – Форматированный текст. В поле можно будет вставлять мультимедиа элементы, а также форматировать текст посредством текстового редактора;

  Любое другое значение (включая пустое) интерпретируется также как и CHAR.

FORMAT – ***Формат поля***. Указанный формат может использоваться при проверке и автокоррекции введённых данных, а также при заполнении значения посредством SQL-запроса из свойства QUERY_TEXT Бланка.

- Если DATA_TYPE = CHAR, тут можно указать код алфавита, на соответствие которому будет проверяться вводимый текст;
- В случае DATA_TYPE = NUMBER или DATE, формат указывается согласно документации по СУБД;
- Для DATA_TYPE = FILE здесь указывается шаблон выбираемых имён файлов в формате `Пояснительный текст|Шаблон|`. Например: `SQL-скрипты (*.SQL)|*.sql|`

CAPTION – ***Подпись к полю***. Текстовая надпись слева или сверху от поля, отражающая суть информации в данном поле.

VISIBLE – ***Количество отображаемых строк***. Возможные значения:

- 1 (или Y) - Однострочное поле;
- 2 и более - многострочное поле. Возможно добавление перевода строк;
- Любое другое значение (включая пустое) означает невидимое (неотображаемое) поле.

VISUALIZATION – ***Код набора свойств отображения***. Способ визуально выделить поле. По сути это – имя класса для стилевого оформления поля. Автоматически поддерживаются следующие варианты:

- FONT_SELECTED – Выделен текст;
- BACKGROUND_SELECTED – Выделен фон;
- ALL_SELECTED – Выделен текст и фон;
- LOWERED – слабозаметный;
- DANGER – Опасность!;
- ATTENTION – Внимание!;
- OK – Успех;
- INHERIT – Унаследованое значение;
- SECTION – Подраздел;
- START – Включить/Работает;
- INTERIM – Приостановить/Переключается;
- STOP – Остановить/Выключено.

HINT – ***Подсказка***. Текст, поясняющий суть информации в поле одной ёмкой, но предельно точной фразой.

VARIANTS – ***Набор возможных значений***. Свойство служит для получения перечня возможных или типичных значений данного поля. Здесь можно указать:

SQL-запрос. При выборе отображаются только столбцы с алиасами в двойных кавычках. Значением поля станет текст из первого столбца выбранной строки запроса;

Коллекцию значений в формате Код1:Пояснение;Код2:Пояснение;... Значением поля станет код выбранного варианта

Запрос и коллекцию не нужно заключать в фигурные скобки, ибо это не динамика, а, собственно, значение поля, что не исключает использование динамики для генерации запроса или коллекции

PATTERN – ***Условие автоперехода***. Здесь ожидается шаблон регулярного выражения. Ввод пользователя постоянно проверяется на соответствие этому шаблону. Автопереход на следующее поле случается при первом же соответствии введённых данных заданному здесь шаблону. Автопереход можно подменить выполнением некоего локального действия этого поля, у которого свойство ROLE = MATCH.

CHANGING – ***Способ изменения данных***. Способ ввода информации в поле. Возможны следующие значения:

- Y – Произвольный текст (значение по умолчанию). Пользователь может вводить данные с клавиатуры;
- C – Список значений. Пользователь может делать выбор только из списка доступных значений из свойства VARIANTS;
- U – Текст в верхнем регистре. Вводимый с клавиатуры текст сразу вводится в верхнем регистре;
- H - Скрытый текст для ввода паролей. Вводимые символы не отображаются;
- B – Check Box. Пользователь переключает значения из списка, определённых свойством VARIANTS;
- Любое другое значение (включая пустое) интерпретируется как неизменяемое поле;

Если при значениях Y и U определено свойство VARIANTS, то возможен выбор из наиболее типичных значений.

SHOW – ***Расшифровка значения***. При CHANGING равном C и N данное свойство подменяет реальное значение поля всегда. Во всех других случаях - только когда поле не активно. Если свойство не указано (пусто), то работает стандартный механизм вычисления расшифровки Read-Only и LOV-Only полей, а именно - Расшифровка вычисляется исходя из множества значений, указанном в свойстве VARIANTS

OPTIONAL – ***Можно не заполнять?*** Возможны следующие значения:

- N – в поле должно отображаться непустое значение;
- Любое другое значение свойства (включая пустое) - поле можно не заполнять.

По умолчанию значение Y. Надо понимать, что на заполненность проверяется значение свойства SHOW, так как именно это значение видит пользователь в поле.

ITEM_NOTES – ***Пояснения в Help***. Текст описания поля Бланка для отображения в Help и генерации документации. Следует помещать сюда всё, что не уместилось в одну фразу в свойстве HINT.

★ Свойства действий Бланка

Любое действие бланка выполняется в контексте текущих значений полей бланка. Корректность данных в полях жёстко проверяется только непосредственно перед- или в процессе выполнения действия. По результатам проверки может быть возбуждена ошибка с откатом состояния к началу выполнения действия.

Действия могут активироваться нажатием кнопок, выбором пункта из локального меню поля, кликом на иконке справа от поля. Также некоторые действия могут выполняться автоматически или по событию (Enter, Esc, закрытие формы и т.п.)

PROMPT – ***Наименование действия***. Этот текст отображается или на кнопке (для глобального действия) или в локальном меню (для локального действия поля), если, конечно, PROMPT не пустой.

EXPLANATION – ***Подсказка***. Текст, поясняющий суть действия одной ёмкой, но предельно точной фразой. Используется в виде всплывающей подсказки у управляющего элемента, активирующего действие.

ACTIVE – ***Признак активности***. Это свойство проверяется как минимум 2 раза. Первый – при отображении элемента управления, активирующего данное действие (активно/не активно), второй – непосредственно перед выполнением Возможные значения:

- Y – Действие доступно для выполнения любым возможным способом активации (по умолчанию);
- Любое другое значение (включая пустое) – Действие не доступно для выполнения, каким бы способом оно ни активировалось;

DISPLAY – ***Отображать кнопку?***. Создавать ли кнопочку для активации глобального действия. Если действие не активно, но создаёт кнопку, то эта кнопка будет не активна. Возможные значения:

- Y - Действие порождает кнопку (при наличии непустого PROMPT);
- Любое другое значение (включая пустое) – Действие не порождает кнопку.

Значение по умолчанию синхронизировано со свойством ACTIVE (ссылается на значение этого свойства: `{Property: ACTIVE}`).

ICON – ***Иконка***. Имя иконки, активирующей локальное действие (подчинённое полю). Иконка всегда будет видимой. Для сокрытия иконки сделайте значение данного свойства динамическим. Активность иконки изменяется в соответствии со свойством ACTIVE. Если несколько локальных действий имеют одну иконку, то она будет отображаться только один раз. При нажатии на эту иконку из них сработает первое активное действие.

У поля могут появляться системные локальные действия со своими иконками:

- Calendar – Если поле типа «Дата/Время» (DATA_TYPE = DATE)
- LOV – Если свойство поля VARIANTS содержит SQL-запрос для отображения List of Value
- DropDown – Если свойство поля VARIANTS содержит коллекцию вариантов для выпадающего списка
- edit – Если свойство поля DATA_TYPE = RICHTEXT

Согласно правилам вызова действий при нажатии иконки любое локальное действие, если его иконка совпадает с системной, может переопределить вызов соответствующего стандартного действия при нажатии на эту иконку.

ROLE – ***Способы активации***. Альтернативные способы активации действия:

- DBLCLICK – выполняется при нажатии `Enter` в любом однострочном поле бланка;
- CLOSE – Нажатие клавиши `Esc` или закрытие окна. Выполняется также при попытке закрытия формы, каким бы способом это закрытие не выполнялось;
- AUTO – Действие, выполняющееся при каждом изменении значения любого его выходного параметра;
- TIME – Действие, выполняющееся ежесекундно по таймеру (в периоды активности формы);
- WIZARD – Действие, являющееся очередным шагом визарда (вперёд или назад);
- MATCH – Локальное действие поля, выполняющееся при соответствии значения шаблону из свойства PATTERN этого поля.

Каждый способ активации выполняет первое по порядку активное действие, которому назначен данный способ активации. Способы активации можно комбинировать, указывая их через пробел и/или запятую.

QUESTION – ***Текст предупреждения***. Если свойство не пусто, то его значение будет выдано как предупреждения с двумя кнопками. Если текст заканчивается знаком вопроса, то на кнопках будут надписи «Да» и «Нет». Иначе на кнопках отобразится «Продолжить» и «Отменить». Выбор первой кнопки позволяет продолжить выполнение действия. Выбор второй кнопки – приводит к прекращению выполнения действия и откату к началу его выполнения.

VERIFICATION – ***Контролировать правильность значений?*** Некоторую проверку корректности ввода данных в редактируемых видимых полях Бланка может взять на себя интерфейс, а именно: заполненность обязательных полей и соответствие значений полей типу и формату данных:

- Y (по умолчанию) – проверять поля на заполненность и соответствие типу и формату;
- Любое другое значение (включая пустое) – не производить проверку. Например, при закрытии формы без сохранения данных.

Более сложную проверку корректности данных в поля должен брать на себя код из свойства PROC_INITIALLY.

PROC_INITIALLY – ***Программный код действия***. Анонимный PL/SQL-блок, выполняющийся при активации данной команды перед (или вместо) вызовом формы, отчёта, команды OS и т.п. Программный код не нужно заключать в фигурные скобки, так как это не динамика, а, собственно, само значение свойства, что не исключает использование динамики для генерации всего кода или его части. Необходимо избегать использования тут команды Commit.

COMMAND_TYPE – ***Тип вызываемого действия***. Возможны следующие значения:

- FUNCTIONAL – Вызов функционала. COMMAND - код функционала;
- LIST – Вызов формы универсального интерфейса. COMMAND – код списка или бланка из метаданных;

- BLANC – Вызов формы универсального интерфейса. COMMAND – код списка или бланка из метаданных;
- TREE - Вызов иерархической структуры. COMMAND – код списка из метаданных, который образует элементы первого уровня иерархической структуры;
- REPGROUP – Открытие группы отчётов для выбора и отображения отчёта из этой группы. COMMAND – код группы отчётов;
- REPORT – Отображение отчёта. COMMAND – код отчёта;
- HOST – Вызов внешней команды OS. COMMAND – команда OS;
- LINK – Вызов WEB-страницы. COMMAND – URL страницы;
- UPLOAD – Загрузка файла с выбором его из файловой структуры клиентского компьютера. COMMAND – место загрузки и, возможно, имя файла;
- DOWNLOAD – Выгрузка файла на клиентский компьютер. COMMAND – место хранения и имя файла.

COMMAND – ***Объект действия***. Вызываемый объект, соответствующий типу действия, указанному в свойстве COMMAND_TYPE. Помимо традиционного способа передачи параметров (отдельными подчинёнными элементами действия типа PARAM), возможно перечисление параметров в коллекции, указанной в круглых скобках сразу после объекта - идентификатора действия: `Объект(Параметр1:Значение;Параметр2:Значение;...;)`. Даже при обработке действием нескольких строк вызов указанного тут объекта выполняется только 1 раз.

PROC_FINALLY – ***Программный код после действия***. Этот код выполняется уже после команды. При необходимости завершить транзакцию не используйте `Commit`, а используйте `{Content:COMMIT}`, так как вызывающая форма посредством свойства CHILD_COMMIT может запретить завершать транзакцию.

Если вызываемая командой COMMAND форма (Список, Бланк или Иерархическая структура) закрылась в режиме CANCEL (действие в вызываемой форме, выполняющееся по закрытию, имеет свойство AFTER = CANCEL), то код из PROC_FINALLY не выполняется, действие отменяется, а транзакция откатывается к началу действия.

CHILD_COMMIT – ***Завершать транзакцию в вызываемой форме?*** Если в вызываемой форме и всех последующих вместо Commit в коде используется `{Content:COMMIT}`, то при значении этого свойства N реальные коммиты выполняться не будут, то есть транзакции останутся не завершёнными. Завершение транзакции в этом случае берёт на себя вызывающая форма. Возможные значения:

- N – в вызываемой форме не будет завершаться транзакция (все commit не будут выполняться). Рекомендуется использовать в действиях – шагах визарда вперёд;
- Y – в вызываемой форме все commit всегда безусловно выполняются. Использовать это значение не рекомендуется без крайней на то необходимости;
- Любое другое значение (включая пустое) – режим завершения транзакций не меняется (наследуется из текущей формы);

По умолчанию значение – пусто.

AFTER – ***Поведение Бланка после выполнения действия***. После успешного выполнения действия можно, в случае, если вызывающая форма не закрылась в режиме CANCEL, текущий список поведёт себя в соответствии с указанными тут вариантами:

- NONE – Ничего не делать;
- FIELD –Текущее поле Бланка обновиться значением из свойства DEFINITION;
- CURRENT – Обновятся все поля той группы, к которой относится действие;
- ALL – Обновятся все поля Бланка;
- REFRESH – Переинициировать Бланк, то есть, как бы выйти и войти с теми же значениями входных параметров. Код из свойства ON_LOAD выполнится повторно;
- EXIT – Закрыть Бланк и передать управление в вызывающую форму. В случае визарда закроется весь визард;
- CANCEL – Отменить вызов Бланка. В вызывающей форме не будут выполняться POST-действия и перезапросы. В случае визарда откатится весь визард. Для отката только на предыдущий шаг визарда данное действие должно иметь ROLE = WIZARD.

ACTION_NOTES – ***Пояснения в Help***. Текст описания действия Бланка для отображения в Help и генерации документации. Следует помещать сюда всё, что не уместилось в одну фразу в свойстве EXPLANATION.

★ Свойства выходных параметров действий Бланка

Подчинённые действию элементы Бланка типа PARAM – это выходные параметры действия Бланка, передающие в вызываемую форму значения, которые становятся значениями её входных параметров.

В случае если набор выходных параметров действия не может быть заранее определён и становится известным только в момент выполнения действия, существует альтернативный способ передачи параметров: в виде коллекции, указанной в круглых скобках сразу после кода формы в свойстве COMMAND: `КОД_ФОРМЫ(Параметр1:Значение;Параметр2:Значение;...;)`. Возможен комбинированный вариант задания выходных параметров – и посредством подчинённых элементов действия типа PARAM и коллекцией параметров в свойстве действия COMMAND. Параметры, определённые отдельными элементами, имеют приоритет.

NAME – ***Код параметра***. Реальный код выходного параметра. Именно по нему заполняется значение входного параметра в вызываемой форме. Если код пустой, то данный выходной параметр не работает (ничего в вызываемую форму передаваться не будет).

Выходной параметр с кодом CONFIG открывает вызываемую форму в указанной тут конфигурации. При отсутствии его в выходных параметрах, вызываемая форма открывается в той же конфигурации, что и вызывающая. Если в вызываемой форме нет нужной конфигурации, она открывается в конфигурации DEFAULT.

VALUE – ***Значение выходного параметра***. При использовании динамики её значение вычисляется непосредственно в момент выполнения действия. При совпадении по коду с входным параметром вызываемой формы его значение по умолчанию переопределяется данным значением выходного параметра.

DESCRIPTION – ***Комментарий***. Пояснения разработчика о значении выходного параметра.

★ Группировка полей и действий Бланка

Поля и кнопки Бланка могут группироваться по смыслу. Сами группы, наряду с другими полями и кнопками, также могут включаться в группы более высокого порядка. Глубина вложенности теоретически не ограничена.

Свойства групп Бланка:

GROUP_TYPE – ***Тип группировки***. Поддерживаются следующие типы:

- TAB – Закладка или вкладка;
- BUNCH – Раскрывающаяся/свёрнутая группа;
- SIDE_BY_SIDE – Поля внутри группы будут располагаться горизонтально, друг за другом;
- Любое другое значение, включая пустое – Виртуальная группа, объединяющая элементы общим заголовком.

Значение по умолчанию – Virtual.

TITLE –***Заголовок группы***. Заголовок, объединяющий входящие в группу элементы. Для групп типа TAB – это надпись на закладке.

SEEABLE – ***Признак видимости***. Значения:

- Y (по умолчанию) – Группа видима
- Любое другое значение (включая пустое) – группа невидима и потому недоступны все вложенные а неё элементы;

POSITION – ***Положение подписей***. Расположение подписей к полям внутри группы относительно самих полей. Варианты:

- AUTO (по умолчанию) – решает движок, как считает более подходящим. В настоящий момент – слева от полей;
- LEFT – слева от всех полей;
- UP – сверху от всех полей.

Любое другое значение эквивалентно значению LEFT.

GROUP_NOTES – ***Пояснения в Help***. Текст описания группы элементов для отображения в Help и генерации документации.

★ Оперативные свойства Бланка и его элементов

- **{Property:FormID}** – Уникальный текстовый идентификатор рабочей области, куда загружен Бланк. Не изменяется. Доступно в любом контексте.
- **{Property:ListCode}** – Код метаданных, откуда загружался Бланк в рабочую область. Для чисто программного Бланка это значение совпадает с {Property:FormID}. Не изменяется. Доступно в любом контексте.
- **{Property:Code}** – Идентифицирующий код формы, определяющий её сущность. В частности, к этому коду привязываются пользовательские настройки. При загрузке Бланка из метаданных свойство {Property:Code} получается из общего свойства Бланка FEATURE. При создании Бланка программным способом это значение указывается параметром в процедуре создания Бланка. Не изменяется. Доступно в свойствах Бланка.
- **{Property:Code}** – Код текущего элемента (в свойствах элемента). Не изменяется. Доступно в свойствах элементов.
- **{Property:ConfigCode}** – Код конфигурации метаданных из которой загружался Бланк. Пусто, если Бланк чисто программный. Не изменяется. Доступно в любом контексте.
- **{Property:ConfigName}** – Наименование конфигурации метаданных из которой загружался Бланк (в текущем языке). Пусто, если Бланк чисто программный. Не изменяется. Доступно в любом контексте.
- **{Property:FType}** – Тип формы = BLANC. Не изменяется. Доступно в любом контексте.
- **{Property:EType}** – Тип текущего элемента: FIELD, ACTION, PARAM или GROUP. Не изменяется. Доступно в свойствах элементов.
- **{Property:Value}** – Текущее значение поля бланка. Доступно в свойствах полей и подчинённых им элементов (локальных действий и их параметров).
- **{Property:Show}** – Текст, видимый пользователем как значение поля. Доступно в свойствах полей и подчинённых им элементов.
- **{Property:UserChanged}** – Y - Признак изменённости поля пользователем. При автоматическом перевычислении значения поля и только если значение изменилось, данный признак сбрасывается. Доступно в свойствах полей и подчинённых им элементов.
- **{Property:Diff}** – Y – Признак неопределённости значения поля. После изменения данных в поле это свойство сбрасывается. Доступно в свойствах полей и подчинённых им элементов.
- **{Property:Parent}** – Код родительского элемента. Для самостоятельных элементов тут значение «Form». При некоторых обстоятельствах может изменяться! Доступно в свойствах элементов

★ Системные переменные рабочей области Бланка

- **{Property:Var TransitParam}** – Коллекция значений параметров, указанных при загрузке метаданных в рабочую область. То есть это те параметры, с которыми открывалась форма, разработанная в метаданных.
- **{Property:Var CurrentField}** – Код текущего поля Бланка.
- **{Property:Var CurrentAction}** – Код текущего выполняющегося действия.
- **{Property:Var ParentWorkID}** – ID рабочей области вызывающей формы.
- **{Property:Var Changed}** – Y – Признак наличия изменений в Бланке, сделанных пользователем.

Иерархическая структура

В universalInterface для отображения иерархии нет отдельного типа форм. Для построения иерархии используются специальные свойства уже существующих элементов в формах-списках. Иерархия выстраивается посредством действий Списков, вызывающих другие Списки. Таким образом, для отображения иерархической структуры достаточно при её вызове специальным способом открыть Список, все строки которого породят узлы верхнего уровня. Действия этого списка породят узлы второго уровня, а их раскрытие вызовет отображение Списка, вызываемого действием, в виде всё тех же узлов уже третьего порядка и т.д. Возможна укороченная схема иерархии, когда раскрытие узла-строки некоторого Списка отображает не действия над этой строкой, а сразу узлы-строки другого Списка.

Узлы, помимо текста, могут снабжаться пиктограммами и выделяться стилями. Существует понятие текущего узла, определяющего контекст – значения всех ячеек строки Списка, породившего этот узел. Пользователь может выделять несколько узлов на одном уровне (кликая мышкой с нажатым Ctrl).

Действия могут выполняться как над текущим узлом, так и над всеми выделенными узлами. После завершения действия может быть обновлён текущий узел или перечитан все узлы текущего уровня.

★ Особенности свойств уровня иерархии

Ниже описаны свойства Списка, участвующего в построении некоего уровня в иерархической структуре, в части, отличающейся от свойств Grid (см. «Общие свойства Списка» на стр.8)

QUERY_TEXT – ***SQL–выражение***. Основной SQL-запрос, формирующий множество элементов раскрываемого уровня иерархии. Множество значений столбцов станут контекстом порождаемого узла

HEADER – ***Текст родительского узла***. Он заменит текст раскрываемого узла. Если Список используется для порождения элементов самого верхнего уровня иерархии, HEADER становится заголовком окна формы-иерархии. Если это не первая форма в окне, то такой текст отображается как подзаголовок. Возможно, что один и тот же Список будет использоваться и для отображения Grid и для отображения Иерархии. В этом случае рекомендуется в каждой ситуации пользоваться разными конфигурациями Списка.

PORTION – ***Выбирать группами по***... Размер порции записей, отображающихся при раскрытии узла дерева. Если все порождаемые узлы не поместились в первую порцию, то последним станет узел специального типа с текстом в виде троеточия. Любое событие с этим узлом приведёт к отображению следующей порции узлов вместо этого специального узла.

NODE_TEXT – ***Текст узла дерева***. Текст подписи к узлу дерева. Обычно здесь используются ссылки на столбцы.

NODE_ICON – ***Иконка для каждого узла***. Имя иконки для отображения в узлах дерева.

ADD_INFO – ***Информация о записи***. Подсказка к каждому узлу порождаемого уровня иерархии. Обычно здесь используются ссылки на столбцы.

Свойство Списка START_MODE в режиме отображения иерархии не используется. Уровень иерархии всегда находится в многострочном режиме. Выделение нового узла без снятия уже существующих выделений производится кликом с нажатой клавишей Ctrl. Одновременно выделить можно узлы только одного уровня иерархии. Инверсию выделений включить нельзя.

★ Особенности свойств столбцов в уровне иерархии

Каждый столбец при формировании узла порождает значение контекста, ссылку на который (по коду столбца) можно использоваться в динамических свойствах. В частности, ссылки на значение элементов контекста необходимы для формирования текста узла уровня иерархии (свойство NODE_TEXT) и подсказок к ним (свойство ADD_INFO).

Свойства столбцов DATA_TYPE, FORMAT, CAPTION, ALIGNMENT, VISIBLE, WIDTH, VISUALIZATION, HINT, VARIANTS и TOTAL в режиме отображения иерархии не используются.

★ Особенности свойств действий в иерархии

Локальные действия (принадлежащие столбцом) в режиме отображения иерархии игнорируются. Остальные (глобальные) действия делятся на те, что порождают подчинённые узлы иерархии и те, что могут быть вызваны обычным образом, например, из локального меню (по правой кнопке мыши) или ролевыми клавишами.

Чтобы действие могло порождать подчинённые элементы иерархии, необходимы следующие условия:

- Действие должно быть глобальным (не подчинённым какому-то столбцу);
- Свойство ACTIVE = EXIST;
- Свойство APPLY_TO = N;
- Свойство COMMAND_TYPE = LIST;
- Свойство PROC_INITIALLY должно быть пустым;
- Свойство COMMAND_TYPE = LIST;
- Свойство ROLE не должно иметь значения ADD, DELETE, CLOSE, AUTO, TIME и WIZARD;

При раскрытии узла отображаются подузлы с названиями (свойство PROMPT) всех действий, удовлетворяющих вышеперечисленным параметрам. А раскрытие узлов-действий повлечёт активацию нового уровня иерархии, основанного на указанном Списке. Если же подобное действие одно для раскрываемого узла, то промежуточный уровень с названиями действий не образуется, а сразу раскрывается Список, вызываемый этим единственным действием (упрощённая схема).

Ниже описаны свойства действий Списка, участвующего в построении некоего уровня в иерархической структуре, в части, отличающейся от свойств действий в Grid (см. «Свойства действий Списка» на стр.11)

PROMPT – ***Наименование действия***. Заголовок действия при отображении в локальном меню или в качестве промежуточного уровня иерархии при раскрытии узла-строки Списка.

ACTIVE – ***Признак активности***. Надо понимать, что действия, для которых это свойство равно Y могут показываться, если текущим узлом является родительский узел уровня иерархии. Если же родительский узел в свёрнутом состоянии – это узел-строка вызывающего Списка (урощённая схема), то для него набор действий разный – в зависимости от его состояния: свёрнутый/развёрнутый. В свёрнутом состоянии – это строка родительского Списка и для неё действия порождаются родительским Списком. В развёрнутом состоянии – это заголовок дочернего (вызываемого) Списка и на него действия порождаются этим дочерним Списком.

ROLE – ***Способы активации***. Альтернативные способы активации действия:

- DBLCLICK – Двойной клик левой кнопкой мыши на узле или нажатие клавиши `Enter`;
- ADD – Нажатие клавиши `Insert`. По смыслу – действие добавляет строки на уровень иерархии;
- DELETE – Нажатие клавиши `Delete`. По смыслу – действие удаляет строки с уровня иерархии;
- CLOSE – Закрытие формы. Срабатывает только для иерархии самого верхнего уровня;
- AUTO – Действие, выполняющееся при каждом изменении значения любого его выходного параметра;
- ALLEXPAND – Автораскрытие узла-действия (и всех вложенных узлов).

AFTER – ***Поведение уровня после выполнения действия***. После успешного выполнения действия текущий уровень иерархии поведёт себя в соответствии с указанными тут вариантами:

- NONE – Ничего не произойдёт;
- FIELD – Ничего не произойдёт;
- CURRENT – Обновится текущий узел;
- ALL – Обновятся все узлы текущего уровня. Текущий узел попытается сохраниться (без гарантии);
- EXIT – Свернуть текущий уровень иерархии или закрыть иерархическую структуру для иерархии самого верхнего уровня;
- CANCEL – Отменить вызов формы – иерархической структуры. Срабатывает только для иерархии самого верхнего уровня.

Свойства ICON и BUTTON игнорируются.

У параметров действий в иерархических структурах никаких особенностей – по сравнению со Списками – нет.

★ Оперативные свойства иерархической структуры

- **{Property:FormID}** – Уникальный текстовый идентификатор рабочей области, куда загружен Список для отображения текущего уровня иерархии. Не изменяется. Доступно в любом контексте.
- **{Property:ListCode}** – Код метаданных, откуда загружался Список в рабочую область. Не изменяется. Доступно в любом контексте.
- **{Property:Code}** – Код текущего элемента (в свойствах элемента). Не изменяется. Доступно в свойствах элементов.
- **{Property:FType}** – Тип формы = TREE. Не изменяется. Доступно в любом контексте.
- **{Property:EType}** – Тип текущего элемента (в свойствах элемента): ACTION или PARAM. Не изменяется. Доступно в свойствах элементов.
- **{Property:SelectionMode}** – Режим выделения строк в уровне: Single или Multi. Доступно в любом контексте.
- **{Property:Parent}** – Код действия для выходного параметра. Для действий тут значение «Form». Доступно в свойствах элементов

★ Системные переменные рабочей области иерархической структуры

- **{Property:Var TransitParam}** – Коллекция значений параметров, указанных при загрузке метаданных в рабочую область. То есть это те параметры, с которыми открывалась форма, разработанная в метаданных.
- **{Property:Var CurrentRow}** – Номер по порядку текущего узла в уровне. Пустое значение – текущая строка сейчас не в текущем уровне.
- **{Property:Var FetchedRows}** – Количество выфетченных узлов в уровне иерархии на текущий момент.
- **{Property:Var CountSelected}** – Количество выделенных узлов внутри уровня иерархии.
- **{Property:Var IsLastPage}** – Признак полноты полученных узлов в уровне иерархии: Y – все узлы уже получены. Любое другое значение (включая пустое) – на сервере, возможно, есть ещё узлы, не полученные в данном уровне иерархии.
- **{Property:Var MainQuery}** – Текст основного запроса уровня иерархии
- **{Property:Var CurrentAction}** – Код текущего выполняющегося действия.
- **{Property:Var ParentWorkID}** – ID рабочей области вызывающей формы.
- **{Property:Var ParentNode}** – Уникальный идентификатор родительского узла.
- **{Property:Var CurrentNodeType}** – Тип текущего узла в уровне:
  ROW – узел-строка Списка
  ACTION – узел-действие
  CONTINUE – специальный узел для запроса следующей порции узлов-строк.
- **{Property:Var RootFormID}** – ID рабочей области высшего уровня иерархии.
- **{Property:Var ContextForkID}** – ID рабочей области текущего уровня иерархии (где сейчас текущая строка или выделены несколько строк). Это значение есть только у рабочей области высшего уровня иерархии.

Создание форм

Перечисленные в главе «Формы universalInterface» Списки, Бланки и Иерархические структуры нужно как-то описывать, гранить эти описания и использовать для прикладных нужд. Есть два способа делать это, применяемые как по отдельности, так и последовательно друг за другом.

Первый способ – хранение описания форм (элементов и их свойств) в специальном чётко структурированном каталоге с удобным интерфейсом по редактированию. Это способ описать вид и поведение формы в offline режиме, заранее, до начала работы прикладной системы. Такие данные, которые описывают вид и поведение формы будем называть «***метаданные***». В runtime режиме метаданные будут загружаться в рабочие области форм и сразу начинать функционировать.

Преимущества этого способа в том, что в огромном большинстве случае для создания работоспособных форм не нужно программировать. Достаточно описать свойства форм и их элементов.

Другой способ – создание элементов форм и описание их свойств «на лету», уже в процессе работы приложения, при помощи функций некоего процедурного языка. Преимущества этого способа в том, что можно порождать формы, набор элементов которых (да и свойства этих элементов) не могут быть определены заранее, вне функционирования приложения. Например, формы, элементы и свойства которых берутся из часто меняющегося справочника или порождаются исходя их оперативной обстановки.

Комбинированный способ состоит в том, что некая относительно статичная часть формы загружается из метаданных, а в программе, указанной в свойстве ON_LOAD формы (списка и бланка) можно воспользоваться функциями добавления оперативных элементов и/или коррекции свойств, загруженных из метаданных.

Редактор метаданных

Формы редактора метаданных созданы и функционируют при помощи universalInterface.

Для входа в редактор необходимо отобразить Список UI_LIST_GROUPS в режиме TREE (иерархическая структура). Отобразится список групп форм верхнего уровня. Раскрытие каждой группы отобразит список подгрупп и/или список форм типа «Список» и/или список форм типа «Бланк». Каждая форма раскрывает список конфигураций. Конфигурация позволяет увидеть список подчинённых конфигураций и иерархию элементов формы.

На каждом уровне можно добавлять и изменять группы форм, формы, конфигурации форм и элементы форм. Кроме того, навигация по списку конфигураций и по элементам форм вызывает автоматическое открытие справа в окне бланка со свойствами конфигурации или элемента.

★ Перечень Групп форм

В узле иерархической структуры для каждой группы отображается слово «Группа», код группы и через тире – Наименование группы на текущем языке.

В перечне доступны следующие локальные действия (отображаемые по правой кнопке мыши):

- ***Добавить группу*** – Доступно всегда. Вызывает бланк для добавления новой группы в текущий уровень.
- ***Переименовать группу*** – Доступно только для текущего узла-группы. Вызывает бланк, где можно изменить имя группы для разных языков и переместить группу в иерархии групп.
- ***Удалить группу*** – Доступно только для текущего узла-группы. Позволяет удалить текущую (выделенную) группу.
- ***Создать форму*** – Доступно только для текущего узла-группы и только если в групп ещё нет форм. Вызывает бланк, позволяющий создать новую форму (список или бланк) в текущей группе.
- ***Поиск*** – Доступно всегда. Открывает интерфейс, позволяющий искать формы по их кодам или по текстовым фрагментам их свойств.
- ***Изменённые формы UI*** – Доступно всегда. Открывает список форм, отсортированных по дате последнего изменения в них с указанием на место последнего изменения.

- ***Top 10 в группе*** *{КОД}* – Доступно только для текущего узла-группы. Отображает список не более 10 строк – форм в группе с кодом КОД, редактируемых последними.

Раскрытие узла-группы отображает подузлы по следующему принципу:

- **Списки группы** – Если в группе есть хотя бы одна форма типа Список, и нет ни бланков, ни подгрупп.
- **Бланки группы** – Если в группе есть хотя бы одна форма типа Бланк, и нет ни Списков, ни подгрупп
- **Список подгрупп** – Если у группы есть хотя бы одна подгруппа, и нет ни одного Списка или бланка.
- **Список узлов-действий, состоящий из**:
  - ***Списки группы {КОД}*** – если в группе есть хотя бы один Список
  - ***Бланки группы {КОД}*** – если в группе есть хотя бы один Бланк
  - ***Вложенные группы*** – если у группы есть хотя бы одна подгруппа

  Каждый их перечисленных узлов имеет локальное действие ***Открыть как список***, которое отображает содержимое узлов-действий в виде формы-Списка.

★ Перечень Форм

Существуют два перечня форм – для Списков и для Бланков, которые отображаются при раскрытии узлов-групп или узлов-действий групп (см. «Перечень Групп форм»). По сути – перечень Списков и Бланков определяется одной UI-формой, принимающей разные параметры. По этому их описание тут не разделяется.

Узлы-формы отображаются так: Слово «Список» или «Бланк» и код формы (списка или бланка). В перечне этих узлов доступны следующие локальные действия (отображаемые по правой кнопке мыши):

- ***Добавить*** – Доступно всегда. Вызывает бланк для добавления новой формы (Списка или бланка). В процессе создания новой формы можно изменить её тип со Списка ни Бланк или наоборот.
- ***Изменить*** – Доступно только для текущего узла-формы. Вызывает бланк для изменения атрибутов формы (Списка или бланка), включая код и подчинение группе.
- ***Удалить*** – Доступно для одного или нескольких выделенных узлов-форм. Удаляет ненужные формы.
- ***Кто вызывает*** *- N* – Доступно для текущего узла-формы. Отображает перечень (в виде иерархической структуры) всех форм, где есть действия, вызывающие текущую форму (код формы указан в свойстве COMMAND этого действия). Здесь N – количество таких форм.
- ***Аудит*** – Доступно для текущего узла-формы. Отображает Список с данными обо всех изменениях в текущей форме.
- ***Скрипт замены в*** *{Path}* – Доступно для одного или нескольких выделенных узлов-форм. Генерирует файлы на сервере Oracle в формате SQLPlus для полной замены форм в метаданных. Каталог (Path) для генерации файлов должен быть указан в параметре GENERATE_DESTINATION, а на сервере – разрешена запись в указанный каталог.
- ***Скрипт удаления в*** *{Path}* – Доступно для текущего узла-формы. Генерирует файл на сервере Oracle в формате SQLPlus для удаления формы из метаданных. Каталог (Path) для генерации файла должен быть указан в параметре GENERATE_DESTINATION.

Раскрытие узла-формы отображает подузлы-конфигурации верхнего уровня

★ Перечень Конфигураций формы

Конфигурации формы сами выстраиваются в иерархическую структуру. Раскрытие узла-формы отображает перечень конфигураций верхнего уровня, среди которых обязательно должна быть конфигурация с кодом DEFAULT. Текст узла-конфигурации состоит из слова «Конфигурация», кода конфигурации и через тире – названии конфигурации на текущем языке.

В перечне узлов-конфигураций доступны следующие локальные действия (отображаемые по правой кнопке мыши):

- ***Все*** [***столбцы***|***поля***] – Доступно только для текущего узла-конфигурации. Вызывает список всех столбцов (для формы типа Список) или полей (для формы типа Бланк), независимо от их расположения и подчинённости в форме.
- ***Все действия*** – Доступно только для текущего узла-конфигурации. Вызывает список всех действий в форме, независимо от их типа, способа активации и подчинённости в форме.
- ***Все параметры*** – Доступно только для текущего узла-конфигурации. Вызывает список всех параметров – как входных (для формы), так и выходных (для действий в форме).
- ***Добавить новую*** – Доступно всегда. Вызывает бланк для добавления новой конфигурации формы.
- ***Переименовать*** {КОД} – Доступно для текущего узла- конфигурации. Вызывает бланк для изменения имени конфигурации КОД.
- ***Удалить*** {КОД} – Доступно для текущего узла-конфигурации. Удаляет конфигурацию с кодом КОД
- ***Аудит*** {КОД} – Доступно для текущего узла-конфигурации. Отображает Список с данными обо всех изменениях в конфигурации формы.
- ***Скрипт замены*** – Отображает в отдельном окне скрипт в формате SQLPlus для полной замены текущей конфигурации в метаданных.
- ***Отобразить*** – Запускает текущую форму в текущей конфигурации с дефолтными значениями входных параметров

Навигация по узлам-конфигурациям отображает в том же окне справа в автоматическом режиме бланк для редактирования свойств самой формы в текущей конфигурации (см. «Общие свойства Списка» на стр. 8 и «Общие свойства Бланка» на стр. 16). Подробнее о бланке для редактирования свойств форм см. «Редактирование Свойств форм» на стр.29.

Раскрытие узла-конфигурации отображает подузлы-действия:

- ***Порождённые конфигурации*** – Отображает такой же Перечень Конфигураций, подчинённых текущей.
- ***Элементы*** [***списка***|***бланка***] – Отображает Перечень Элементов формы, с сохранением иерархии подчинения.

★ Перечень Элементов формы

Этот список выстраивается в естественную иерархию подчинённости, когда раскрытие узла-элемента сразу отображает перечень подузлов-элементов, непосредственно подчинённых раскрываемому элементу. Надо знать, что возможность манипулирования элементами (добавление, удаление, переименование) имеется только в иерархии, раскрываемой конфигурацией DEFAULT.

Элементы внутри любого уровня иерархии идут в такой последовательности:

- Сначала идут входные параметры формы (списка или бланка)
- Затем следуют вперемежку столбцы (для Списков), поля (для Бланков), группы и действия внутри группы типа SIDE_BY_SIDE в соответствии с Порядком (Seq)
- В самом конце идут действия – самостоятельные или из группы типа не SIDE_BY_SIDE

Для перечня действий возможны следующие манипуляции (доступные по правой кнопке мыши):

- ***Добавить параметр*** [***списка***|***бланка***] – Доступно для верхнего уровня элементов в конфигурации DEFAULT. Отображает бланк для добавления нового входного параметра формы.
- ***Добавить группу*** – Доступно в конфигурации DEFAULT для верхнего уровня или внутри другой группы. Отображает бланк для добавления новой группы.
- ***Добавить столбец*** – Доступно в конфигурации DEFAULT формы-Списка для верхнего уровня или внутри группы. Отображает бланк для добавления нового столбца. При этом можно выбрать код и имя добавляемого столбца – из множества столбцов таблиц, использующихся в основном запросе (свойство Списка QUERY_TEXT).

- ***Добавить поле*** – Доступно в конфигурации DEFAULT формы-Бланка для верхнего уровня или внутри группы. Отображает бланк для добавления нового поля. При этом можно выбрать код и имя добавляемого поля – из множества столбцов таблиц, использующихся в основном запросе (свойство Бланка QUERY_TEXT).
- ***Добавить действие*** – Доступно в конфигурации DEFAULT формы для верхнего уровня, внутри группы, для столбца Списка или поля Бланка. Отображает бланк для добавления нового действия. При этом можно выбрать роль действия из нескольких типовых. Роль действия определяется его кодом. На основании выбранной роли происходит частичное заполнение свойств действия характерными для этой роли значениями.
- ***Добавить параметр действия*** – Доступно в конфигурации DEFAULT формы только для некоторого действия. Отображает бланк для добавления нового выходного параметра действия.
- ***Изменить*** {КОД} – Применимо к текущему элементу в конфигурации DEFAULT формы. Позволяет изменить Код, Название, Порядок, а также подчинённость существующего элемента формы.
- ***Удалить*** [{КОД}|N ***элемент***(***а***,***ов***)] – Доступно для одного или нескольких выделенных узлов-элементов в конфигурации DEFAULT формы. Выполняет удаление элементов формы. КОД – код текущего элемента в случае выделения одного элемента, N – количество выделенных элементов, в случае выделения более одного элемента
- ***Автоматическое добавление*** [***столбцов***|***полей***] – Генерация столбцов Списка или полей Бланка на основании столбцов таблицы, выбранной из списка таблиц, использующихся в основном запросе (свойство формы QUERY_TEXT).
- ***Перенумеровать*** – Доступно для нескольких выделенных узлов-элементов в конфигурации DEFAULT формы. Позволяет перенумеровать выделенные элементы одного типа с выбранным шагом, начиная с наименьшего номера среди всех выделенных элементов.
- ***Скрипт замены*** {КОД} – Отображает в отдельном окне скрипт в формате SQLPlus для полной замены элемента КОД (с подэлементами) в метаданных.
- ***Аудит*** {КОД} – Доступно для текущего узла-элемента. Отображает Список с данными обо всех действиях с элементом КОД и его свойствами.
- ***Использования*** {КОД} – Доступно для текущего узла-элемента типа Стобец, Поле или Входной Параметр. Отображает список мест (Конфигурация + Элемент + Свойство), где есть ссылка в динамическом значении на элемент КОД.
- ***Сохранить в буфер*** – Сохраняет значения всех свойств текущего элемента в буфер для последующего их переноса в другой элемент.
- ***Копировать из*** {КОД} – Доступно для текущего элемента-узла при условии, что ране было выполнено действие «Сохранить в буфер». Переносит свойства элемента КОД в свойства текущего элемента формы.

Навигация по узлам-элементам, а также множественное выделение узлов одного типа вызывает автоматическое отображение в том же окне справа бланка для редактирования в текущей конфигурации свойств одного или нескольких выделенных элементов формы. Подробнее о бланках для редактирования свойств форм и элементов форм см.

★ Редактирование Свойств форм

При навигации по узлам-конфигурациям в немодальном режиме возникает бланк для отображения и изменения свойств формы в текущей конфигурации. Каждое свойство отображается отдельным текстовым полем. Нажатие на кнопку «Показать коды» переключает бланк в режим отображения в качестве подписей к полям кодов свойств, а не их наименований, а кнопка «Показать названия» переключает этот режим назад.

Навигация по полям-свойствам отображает в поле «Пояснения к свойству {КОД}» текст, поясняющий суть текущего свойства.

Каждое поле-свойство является редактируемым как минимум при помощи ввода текста с клавиатуры. Кроме того, для некоторых свойств возможен выбор наиболее типичных значений при помощи выпадающего списка или LOV (кнопки справа от поля или клавиша F9).

Серый фон поля-свойства говорит о том, что своего значения этого свойства форма в данной конфигурации не имеет, а наследует его из родительской конфигурации (при наличии) или использует значение свойства по умолчанию (если форма работает в конфигурации DEFAULT).

Если значение поля отображается жирным бордовым цветом, то для него имеются варианты значений на разных языках. Редактирование такого поля производится посредством локального действия «Многострочный редактор».

Для каждого поля-свойства доступны следующие локальные действия, отображаемые по клику правой кнопкой мыши в соответствующем поле:

- ***Базовое значение*** – Применимо к полям с белым фоном. Удаляет текущее значение свойства и применяет значение свойства по умолчанию. Поле при этом становится серым.
- ***Вернуть*** – Возвращает перечень всех предыдущих значений текущего свойства формы. Используются данные аудита изменений. Выбор значения из этого перечня помещает его в поле-свойство (возврат старого значения).
- ***Добавить ссылку*** – Отображает список входных параметров, столбцов (для Списков), полей (для Бланков) и некоторых системных переменных формы. Выбор из этого списка помещает ссылку на выбранный элемент в конец текущего значения свойства.
- ***Скрипт*** – Отображает в отдельном окне скрипт в формате SQLPlus для замены в метаданных текущего значения свойства в текущей конфигурации.
- ***Многоязычность*** – Доступно в полях-свойствах, не обладающих многоязычными вариантами. Позволяет задать разные варианты значений для разных языков. В этом случае свойство станет многоязычным.
- ***Многострочный редактор*** – Открывает бланк для редактирования значения свойства в многострочном поле. Для многоязычных значений многострочных полей будет столько же, сколько языков. Удаление значений не текущего языка делает значение моноязычным.

Нажатие на кнопку «Сохранить» помещает текущие изменения в хранилище метаданных. Если текущих изменений нет, то кнопка «Сохранить» будет неактивна. Уход с текущего узла-конфигурации в иерархии редактора метаданных вызывает автосохранение сделанных изменений.

При наличии текущих изменений также доступна кнопка «Вернуть». Её нажатие отменяет все текущие изменения свойств формы.

★ Редактирование Свойств элементов форм

При навигации по узлам-элементам в немодальном режиме, а также при выделении нескольких однотипных пунктов в уровне, возникает бланк для отображения и изменения свойств элементов формы в текущей конфигурации.

Каждое свойство отображается отдельным текстовым полем. В случае выделения одного элемента слева, в этих полях отображаются значения свойств этого элемента. В случае выделения более одного элемента слева, поля свойств с различающимися значениями для выделенных элементов закрыты серой сеткой. Изменение значения свойства, даже если ранее оно различалось для выделенных элементов, приводит к установлению этого значения для всех этих элементов сразу!

Нажатие на кнопку «Показать коды» переключает бланк в режим отображения в качестве подписей к полям кодов свойств, а не их наименований, а кнопка «Показать названия» переключает этот режим назад.

Навигация по полям-свойствам отображает в поле «Пояснения к свойству {КОД}» текст, поясняющий суть текущего свойства.

Каждое значение поля-свойства может меняться при помощи ввода текста с клавиатуры. Иногда возможен выбор наиболее типичных значений при помощи выпадающего списка или LOV (кнопки справа от поля или клавиша F9).

Серый фон поля-свойства говорит о том, что у всех выделенных элементов своего значения этого свойства нет, и оно наследуется из родительской конфигурации (при наличии) или используется значение свойства по умолчанию (если форма работает в конфигурации DEFAULT).

Если значение поля отображается жирным бордовым цветом, то хотя бы для одного выделенного элемента имеются варианты значений на разных языках. Редактирование такого поля производится посредством локального действия «Многострочный редактор» и только для каждого элемента по одному.

Для полей-свойств доступны следующие локальные действия, отображаемые по клику правой кнопкой мыши в соответствующем поле:

- ***Базовое значение*** – Применимо к полям с белым фоном. Удаляет текущее значение свойства и применяет значение свойства по умолчанию ко всем выделенным элементам. Поле при этом становится серым.
- ***Вернуть*** – Возвращает перечень всех предыдущих значений текущего свойства для одного текущего элемента формы. Используются данные аудита изменений. Выбор значения из этого перечня помещает его в поле-свойство (возврат старого значения).
- ***Добавить ссылку*** – Отображает список входных параметров, столбцов (для Списков), полей (для Бланков) и некоторых системных переменных формы. Выбор из этого списка помещает ссылку на выбранный элемент в конец текущего значения свойства.
- ***Скрипт*** – Отображает в отдельном окне скрипт в формате SQLPlus для замены в метаданных текущего значения свойства выделенных элементов формы в текущей конфигурации.
- ***Многоязычность*** – Доступно в полях-свойствах, не обладающих многоязычными вариантами и только при выделении одного элемента. Позволяет задать разные варианты значений для разных языков. В этом случае свойство станет многоязычным.
- ***Многострочный редактор*** – Доступно только при выделении одного элемента. Открывает бланк для редактирования значения свойства в многострочном поле. Для многоязычных значений многострочных полей будет столько же, сколько языков. Удаление значений не текущего языка делает значение моноязычным.

Кнопка «Вверх» перемещает выделенные элементы слева на одну ступень вверх по иерархии. Кнопка «Вниз» делает то же самое в обратную сторону.

Нажатие на кнопку «Сохранить» помещает текущие изменения в хранилище метаданных. Если текущих изменений нет, то кнопка «Сохранить» будет неактивна. Уход с текущего элемента в иерархии редактора метаданных или выделение/развыделение элементов вызывает автосохранение сделанных изменений.

При наличии текущих изменений также доступна кнопка «Вернуть». Её нажатие отменяет все не сохранённые изменения свойств формы.

Программное создание интерфейса

Элементы форм могут полностью или частично создаваться программным способом. Процедуры по созданию элементов интерфейса также могут использоваться для изменения всех или некоторых свойств элементов. Важно, чтобы программные манипуляции с элементами и их свойствами завершились до начала отображения формы в интерфейсе.

Чаще всего используется комбинированный способ загрузки форм: сначала применяются метаданные, а потом, в процессе загрузки, выполняется программный код из свойства ON_LOAD формы (списка или бланка). Этот код может добавить в список столбцы или в бланк поля или в любую форму действия, набор которых не может быть определён в момент разработки, а возникает посредством обращения к тем или иным ресурсам в момент загрузки формы перед её выполнением.

Однако бывают случаи, когда формы полностью порождаются чисто программным способом – от процедуры определения свойств самой формы до процедуры определения свойств каждого элемента этой формы.

★ Процедуры добавления элементов форм

Обязательные параметры помечены ***шрифтом***. Первое выполнение процедуры с некоторым набором обязательных параметров приводит к созданию формы или элемента формы. Повторное выполнение процедуры лишь заменяет некоторые свойства формы или элемента формы.

Остальные параметры – не обязательны. Если они не указаны, то при создании формы или элемента формы для них будут применяться значения по умолчанию, а при повторных вызовах процедур свойства, задаваемые неуказанными параметрами изменяться не будут.

В версии интерфейса для Oracle процедуры собраны в пакете ***UI_P_Param***.

- **Add_List** - Добавление списка или изменение его свойств (см. Общие свойства Списка на стр. 8)
  - ***FormID*** - ID рабочей области Списка
  - Title - Общий заголовок
  - SQLText - Основной запрос списка
  - SQLHint - Подсказка оптимизатору
  - FetchLine - Выводит строки группами по
  - AddText - Информация о каждой строке
  - Notes - Текст для генерации Help
  - Code - Код для привязки свойств и прав
  - SMode - Начальное состояние режима выделения строк
  - NodeText - Текст надписей на узлах дерева
  - NodeIcon - Иконки на узлах дерева

- **Add_Blanc** - Добавление бланка или изменение его свойств (см. Общие свойства Бланка на стр. 16)
  - ***FormID*** - ID рабочей области Бланка
  - Title - Общий заголовок
  - Cancel_Button - Текст на кнопке «Отказаться». Null – кнопки не будет
  - Notes - Текст для генерации Help
  - Code - Код для привязки свойств и прав
  - SQL4Fields - Запрос, заполняющий поля значениями по умолчанию

- **Add_Column** - Добавление столбца списка или изменение его свойств (см. Свойства столбцов Списка)
  - ***FormID*** - ID рабочей области Списка
  - ***Code*** - Код столбца
  - Prompt - Заголовок столбца
  - DataType - Тип данных
  - Format - Форматная маска
  - Expression - Выражение в основной запрос, вычисляющее данные для столбца
  - LOV - Набор возможных значений (коллекция или запрос)
  - Hint - Подсказка пользователю
  - Is_UK - Входит ли столбец в уникальный идентификатор строки
  - Alignment - Выравнивание текста внутри ячейки
  - Width - 0 – изначально невидимый столбец. Автоопределение ширины
  - Visual - Способ визуализации ячейки
  - Sort - Участие в сортировке
  - Total - Выражение для вычисления итогов
  - Seq - Номер столбца по порядку. Если не указать - добавление в конец
  - Notes - Текст для генерации Help
  - Available - Признак доступности столбца
  - Parent - Родительский элемент (группа)

- **Add_Field** - Добавление поля бланка или изменение его свойств (см. Свойства полей Бланка)
  - ***FormID*** - ID рабочей области Бланка
  - ***Code*** - Код поля
  - Prompt - Подпись к полю
  - DataType - Тип данных
  - Format - Формат данных
  - Def - Значение поля по умолчанию
  - Choose - Набор возможных или допустимых значений (коллекция или запрос)
  - Hint - Подсказка к полю

Upd - Способ изменения данных
Visible - Видимость/невидимость
Options - Обязательно/необзятельно
Seq - Номер поля по порядку. Если не указать - добавление в конец
Notes - Текст для генерации Help
Visual - Способ визуализации поля
Pattern - Регулярное выражение для автоперехода
Show - Что отображать в поле, когда в нём нет курсора
Parent - Принадлежность поля к группе
AutoChange - Динамика автоматической коррекции значения поля

- **Add_List_Action** - Добавление действия списка или изменение его свойств (см. Свойства действий Списка)
  FormID - ID рабочей области Списка
  Code - Код действия
  Parent - Код столбца, если действие локальное
  Prompt - Наименование действия в меню
  Visible - Признак отображения в локальном меню
  Active - Признак активности
  Icon - Имя иконки на ToolBar или в ячейке (для локальных действий)
  Hotkey - Способы активации
  Is_Multioperation - Применимо к…
  Action_Lock - Накладывать ли блокировку
  Action_SQL - Программный код действия
  Action_Type - Тип вызываемого действия
  Action_Command - Объект действия
  Action_PostSQL - Программный код после действия
  Requery - Поведение Списка после выполнения действия
  Action_Question - Текст предупреждения
  Hint - Подсказка
  Seq - Номер действия по порядку. Если не указать - добавление в конец
  Notes - Текст для генерации Help
  Close_Transaction - Завершать транзакцию в вызываемой форме?
  ButtonText - Текст на кнопке (если нужна отдельная кнопка)

- **Add_Blanc_Action** – Добавление действия бланка или изменение его свойств (см. Свойства действий Бланка)
  FormID - ID рабочей области Бланка
  Code - Код действия
  Prompt - Наименование действия
  Active - Признак активности
  Visible - Признак отображения кнопки
  Action_SQL - Программный код действия
  Action_Type - Тип вызываемого действия
  Action_Command - Объект действия
  Is_Check - Контролировать правильность значений?
  Action_Question - Текст предупреждения
  Hint - Подсказка
  Seq - Номер действия по порядку. Если не указать - добавление в конец
  Notes - Текст для генерации Help
  Role - Способы активации
  Requery - Поведение бланка после выполнения действия
  Close_Transaction - Завершать транзакцию в вызываемой форме?
  Action_PostSQL - Программный код после действия
  Action_Icon - Иконка в поле
  Parent - Код группы или код поля (для локальных действий)

- **Add_Action_Param**- Добавление выходного параметра действия формы (списка и бланка) бланка или изменение его свойств (см. Свойства выходных параметров действий Списка и Свойства выходных параметров действий Бланка)
  FormID - ID рабочей области формы

Code - Код параметра
Parent - Код действия
Call_Param_Code - Код входного параметра вызываемой формы
Param_Value - Значение параметра

- **Add_Group** - Добавление группы в форме (списке и бланке) или изменение её свойств (см. Группировка столбцов Списка и Группировка полей и действий Бланка)
  FormID - ID рабочей области формы
  Code - Код группы
  Prompt - Заголовок группы
  Visible - Признак видимости (в бланке)
  Seq - Номер элемента по порядку. Если не указать - добавление в конец
  Notes - Текст для генерации Help
  GType - Тип группировки
  Parent_Group - Код родительской группы, в которую вложена эта
  PromptPos - Положение подписей

При первом применении любой процедуры, где впервые встречается некоторое значение ***FormID***, форма с этим идентификатором создаётся автоматически, даже если это процедура добавления элемента, а не самой формы. Отсюда следствие: Элементы формы могут создаваться до вызова процедуры ***Add_List*** или ***Add_Blanc***.

★ Процедура загрузки формы из метаданных

- **Load_Metadata** - Загрузка формы из метаданных
  MetaDataCode - Код метаданных
  ConfigCode - Конфигурация метаданных (по умолчанию – DEFAULT)
  ParamsCollection - Коллекция значений входных параметров
  NewFormID - ID рабочей области формы. Null - автогенерация уникального ID
  ParentFormID - ID рабочей области вызывающей формы

Приёмы разработки форм

В этом разделе собраны типичные приёмы и методы, использующиеся при разработке пользовательских форм на universalInterface. Формат описания приёмов: декларируется цель и приводится метод или методы её достижения.

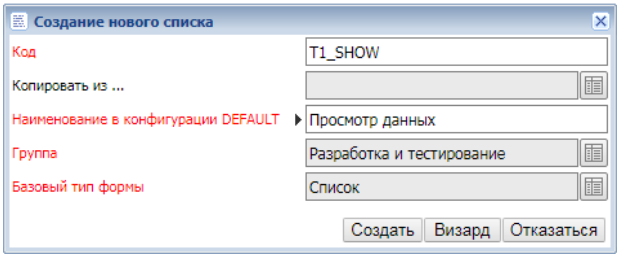
Редактор таблицы

Цель: предоставить пользователю возможность просматривать любые данные в таблице, добавлять, удалять и изменять строки таблицы так, чтобы избегать конфликтов коллективной работы.

Для достижения цели нам понадобятся две формы – Список, отображающий данные некоторой таблицы (Назовём её T1) и Бланк, для изменения данных в строке, он же – для добавления новой строки.

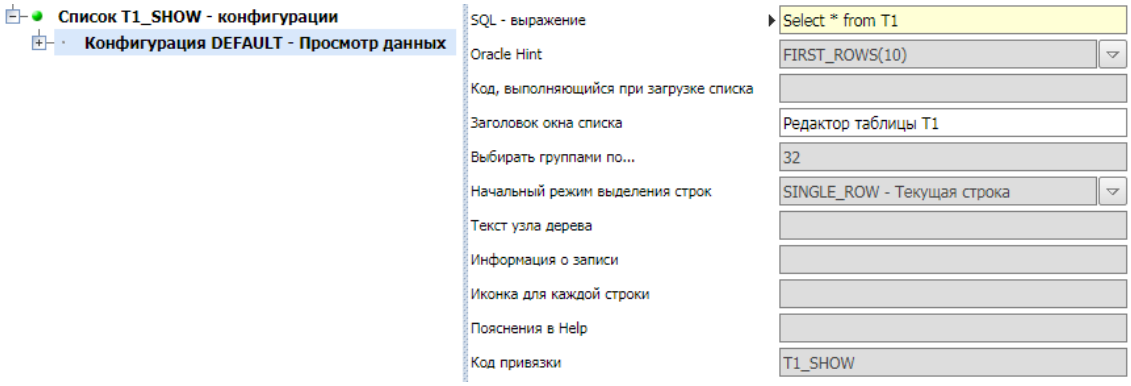
★ Список, основанный на таблице

Приходим в нужную группу, открываем список Списков этой группы (Ну, или выполняем действие «Создать форму» для совсем новой группы). Открывается бланк создания нового Списка.



Можно воспользоваться Визардом. Для достижения нашей цели это как раз самый подходящий способ. Однако мы будем создавать элементы формы сами.

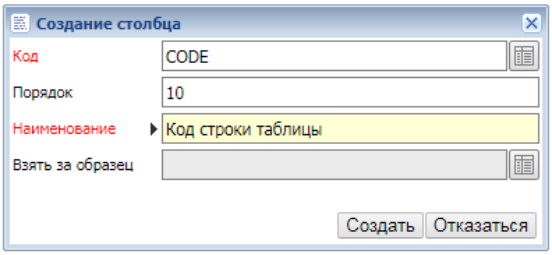
Раскрываем появившийся новый узел-Список и встаём на «Конфигурацию DEFAULT…».



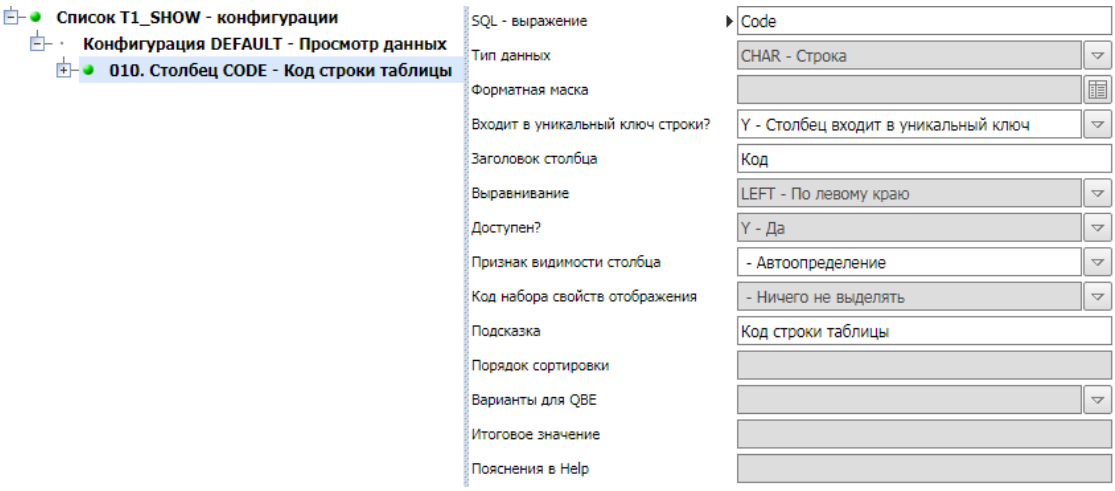
На этом этапе нам требуется только указать заголовок окна формы-Списка и ввести простой запрос, который выберет все данные из нашей таблицы T1.

Раскроем узел – конфигурацию. Нажимаем правую кнопку мыши – появляется локальное меню действий для конфигурации DEFAULT. Нам требуется добавить описание всех столбцов таблицы T1. Опять заметим, что этот процесс можно частично автоматизировать при помощи действия «Автоматическое добавление столбцов» для раскрытого узла-Конфигурации. Но мы будем добавлять столбцы по одному вручную

Выполним действие «Добавить столбец». Откроется бланк для добавления нового столбца в Список. В поле «Код» возможен выбор из списка столбцов нашей таблицы T1, который заполнит и поле «Код» и поле «Наименование»… Но мы откажемся и от этой автоматизации, так как сами знаем – какие столбцы в нашей таблице.



Итак, заполним поля для кода и наименования столбца. Возникнет первый узел-Элемент у конфигурации Списка. Это будет элемент типа «Столбец». Встанем на него – справа появятся свойства этого столбца. Нам обязательно нужно указать «SQL-выражение», которое в запросе будет заполнять ячейки нашего Столбца. В нашем простейшем случае (редактор таблицы) выражение совпадает с кодом столбца таблицы.

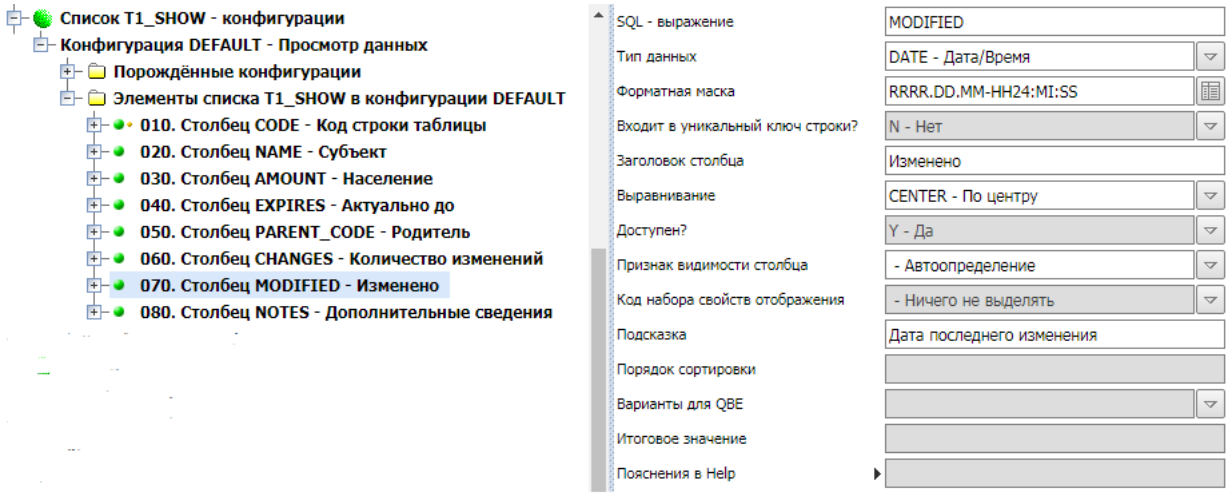


Потом, мы знаем, что столбец Code – это уникальный идентификатор записи в таблице T1. По этому выбираем для поля «Входит в уникальный ключ строки?» значение ***Столбец входит в уникальный ключ***. Поле «Заголовок столбца» автозаполнилось из названия столбца. Можно при желании его скорректировать. Так же, как поле «Подсказка».

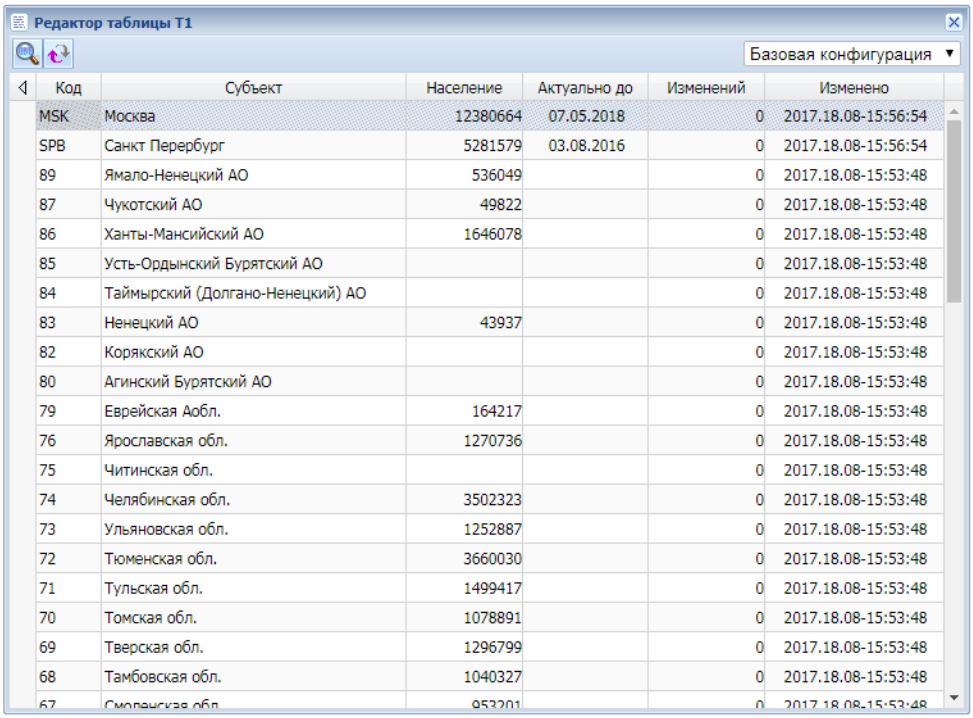
Обязательно изменим «Признак видимости столбца» с ***0*** на ***Автоопределение***. Иначе мы не увидим этот столбец в списке. Остальные свойства пока не будет заполнять или изменять.

Создадим также все остальные столбцы. Отличия могут быть такие

- Остальные столбцы не входят в уникальный ключ
- Для числовых столбцов в поле «Тип данных» укажем значение ***Число***
- Для столбцов из дат и времён укажем в поле «Тип данных» значение ***Дата/Время***. И обязательно в этом случае укажем форматную маску в поле «Форматная маска».



Форма-Список готова для отображения. Встаньте на «Конфигурацию DEFAULT», нажмите правую кнопку мыши и выберите действие «Отобразить». Если таблица T1 уде заполненна некоторой информацией (например, данными о численности населения субъектов РФ), вы получите такую форму в отдельном псевдоокне:



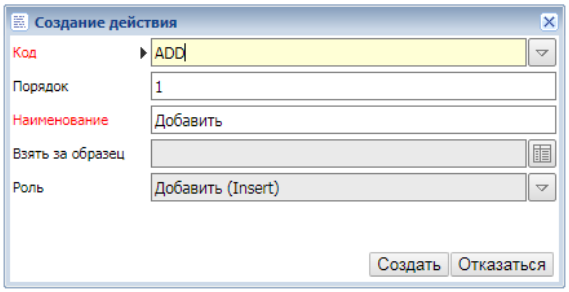
Код	Субъект	Население	Актуально до	Изменений	Изменено
MSK	Москва	12380664	07.05.2018	0	2017.18.08-15:56:54
SPB	Санкт Перербург	5281579	03.08.2016	0	2017.18.08-15:56:54
89	Ямало-Ненецкий АО	536049		0	2017.18.08-15:53:48
87	Чукотский АО	49822		0	2017.18.08-15:53:48
86	Ханты-Мансийский АО	1646078		0	2017.18.08-15:53:48
85	Усть-Ордынский Бурятский АО			0	2017.18.08-15:53:48
84	Таймырский (Долгано-Ненецкий) АО			0	2017.18.08-15:53:48
83	Ненецкий АО	43937		0	2017.18.08-15:53:48
82	Корякский АО			0	2017.18.08-15:53:48
80	Агинский Бурятский АО			0	2017.18.08-15:53:48
79	Еврейская Аобл.	164217		0	2017.18.08-15:53:48
76	Ярославская обл.	1270736		0	2017.18.08-15:53:48
75	Читинская обл.			0	2017.18.08-15:53:48
74	Челябинская обл.	3502323		0	2017.18.08-15:53:48
73	Ульяновская обл.	1252887		0	2017.18.08-15:53:48
72	Тюменская обл.	3660030		0	2017.18.08-15:53:48
71	Тульская обл.	1499417		0	2017.18.08-15:53:48
70	Томская обл.	1078891		0	2017.18.08-15:53:48
69	Тверская обл.	1296799		0	2017.18.08-15:53:48
68	Тамбовская обл.	1040327		0	2017.18.08-15:53:48
67	Смоленская обл.	953201		0	2017.18.08-15:53:48

Эта форма – уже готовый отдельный функционал. У неё такие возможности:

- Просматривать данные, прокручивая строки внутри формы. Все отображаемые данные актуальны на момент отображения первой строки;
- Ограничивать множество выводимых строк при помощи QBE – отбор строк по образцам. Поиск информации по условиям;
- Менять порядок сортировки строк, в том числе, в режиме QBE;
- Менять порядок и ширину столбцов;
- Скрывать и отображать желаемые столбцы;
- Перемещать столбцы в область переполнения справа;
- Менять размеры окна мышкой за его края и углы;
- Перетаскивать окно по экрану за заголовок;

Теперь добавим действия над Списком и строками для возможности редактирования данных в таблице T1. Для этого, стоя на раскрытом узле «Элементы списка…» нужно нажать на правую кнопку мыши и выбрать действие «Добавить действие». Откроется бланк для создания элемента Списка типа «Действие».

Выбираем в выпадающем списке поля «Код» вариант «Добавить». Это тоже разновидность автоматизации. Автоматически заполняется поле «Наименование» значением «Добавить» и поле «Роль» значением «Добавить».



После создания элемента у ему автоматически назначаются значения свойств:

- Иконка кнопки на ToolBar – addrow
- Роль (Горячая клавиша) – ADD – Добавить
- Действие после выполнения – ALL – Перечитать все строки

Нам остаётся только изменить

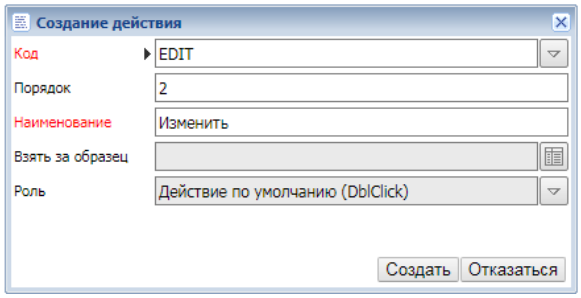
- Тип вызываемого действия – на «Бланк», выбрав его из выпадающего списка
- Объект - идентификатор действия: укажите T1_EDIT – это код бланка для ввода данных новой строки. Этот бланк нам предстоит ещё написать.

Остальные свойства оставим без изменений. Свойство «Накладывать блокировку» можно не трогать, так как действие применимо ко всему списку (свойство «Признак активности»), а не к текущей строке. В этом случае попытка блокировки так и так производиться не будет.

Добавим ещё одно действие – EDIT (Изменить), так же выбрав его код из списка типовых действий. Автоматом назначаются следующие значения свойств:

- Признак активности – EXIST – Активно только для текущей записи
- Иконка кнопки на ToolBar – edit
- Роль (Горячая клавиша) – DBLCLICK – Действие по умолчанию
- Действие после выполнения– CURRENT – Обновить текущую запись

Как и в случае действия ADD, укажем, что при его выполнении нужно вызвать всё тот же бланк T1_EDIT. Мы будем использовать один и тот же бланк и для добавления и для редактирования (хотя это вовсе не догма!). Различать, что нужно делать – добавлять запись или редактировать, бланк будет по наличию переданного в него непустого входного параметра – уникального идентификатора редактируемой строки. Есть идентификатор – редактируем строку, нет – добавляем новую.

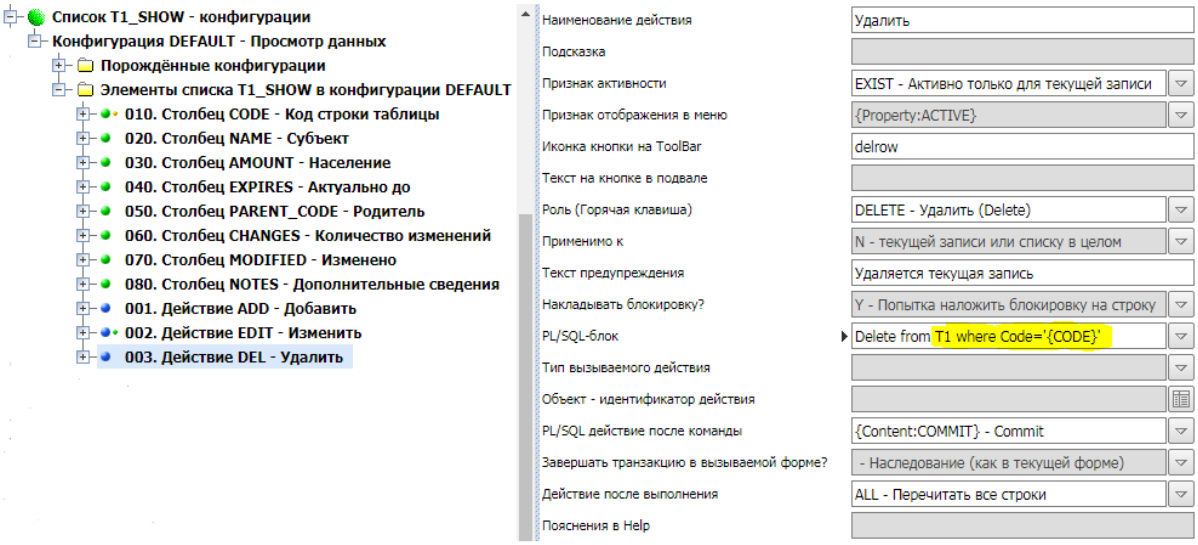


Это значит, что для действия EDIT нужно создать выходной параметр. Для этого раскроем узел «Действие EDIT – изменить». Нажмём правую кнопку мыши и выберем действие «Добавить параметр». В поле «Код» укажем значение EDIT_CODE, а в поле «Наименование» – «Код строки». После создания параметра встанем на него и справа укажем значения свойств:

- Код параметра – PCODE. Такой код будет иметь входной параметр нашего будущего бланка. Если бы бланк уже существовал, то все входные параметры можно было бы увидеть и выбрать нужный при нажатии кнопки LOV справа от этого поля.
- Значение параметра – {CODE}. Это динамическое значение. По сути – ссылка на текущее значение столбца CODE. Ссылку можно было бы выбрать из списка, если в этом поле нажать правую кнопку мыши и выбрать в локальном меню «Добавить ссылку».

Наконец, нам нужно третье действие в Списка – для удаления текущей записи из таблицы T1. В отличие от двух предыдущих действий, которые вызывают форму-Бланк, третье будет прямого действия, без вызова форм-посредников. Правда, в этом случае, пользователя хорошо бы предупредить, во избежание недоразумения.

Добавляем ещё один элемент-действие с кодом DEL (Удалить).



Тут автомат за нас практически уже всё делал. Осталось дописать только то, что помечено на картинке жёлтым маркером. Ниже идут пояснения к значениям свойств:

- Признак активности – наличие текущей строки. Если таковой нет, то и удалять нечего. При этом значении активно работает свойство «Накладывать блокировку?» - Y. То есть нельзя будет удалить заблокированную (используемую кум-то в настоящий момент) запись из таблицы T1
- Текст предупреждения можно изменить и дополнить так, чтобы пользователю была предельно понятна понятно суть операции. Действие делает Commit и потому отменено быть не может. Если это не так и операция не фатальна или может быть легко обращена, то мучить пользователя предупреждениями не стоит – не нужно заполнять это свойство. Кроме того, текст предупреждения может быть динамическим, примерно таким `{Select F('{CODE}') from dual}`, где F – функция, возвращающая текст

предупреждения, если удаление текущей строки влечёт за собой важные последствия, например «При удалении строки с кодом R12 будет удалено 2 договора, 5 начислений и 4 платежа!»

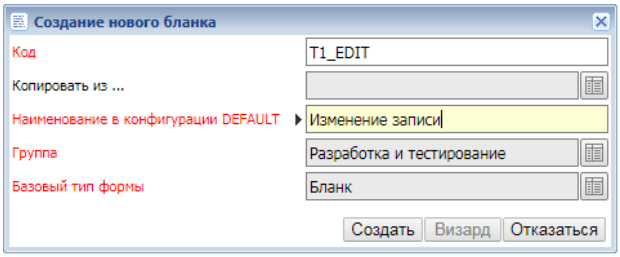
★ Бланк редактирования записи в таблице

Этот бланк предназначен для отображения и изменения значений всех ячеек одной записи таблицы T1. Использоваться он будет как при добавлении новой записи, так и при редактировании конкретной записи. Разница только в начальных значениях полей бланка.

Бланк будет иметь один входной параметр – код редактируемой записи. Если код не указан, то значит производится редактирование новой записи.

Комбинирование бланка добавления и изменения записи в одну форму позволяет экономить на разработке. По сути разница между бланками только в заполнении полей начальными данными и в действии по сохранению. Чем больше в форме редактируемых полей, тем больше унификация этого бланка.

Открываем список Бланков той же группы, где мы добавляли список T1_SHOW и выполняем действие «Добавить» (или выполняем действие «Добавить» для группы лишь с одним списком, а в открывшемся бланке меняем базовый тип формы со «Список» на «Бланк»). Открывается бланк создания нового Бланка.



При создании бланка автоматически добавится действие CANCEL, порождающее кнопку отмены, которая проверяет наличие изменений, спрашивает при их наличии пользователя подтверждения и закрывает бланк с откатом действия, его вызвавшего. Если такое действие не устраивает – можно его удалить или изменить его свойства. В нашем случае действие CANCEL нас полностью устраивает.

Добавим элемент – входной параметр: «Код» = PCODE, «Наименование» = Код строки. Свойства этого элемента не задаём – пустые значения нас полностью устраивают.

Далее, добавим по элементу-полю для каждого столбца таблицы T1:

- CODE (Код строки таблицы). Особенности: Свойство «Способ изменения данных» имеет значение `{PCODE}U`. Если входной параметр PCODE пуст, то это добавление новой записи и Способ изменения кода будет U – то есть, текст в верхнем регистре. Если PCODE будет иметь любое непустое значение, то согласно описанию двух и более символьное значение свойства «Способ изменения данных» будет интерпретироваться как N – неизменяемое поле. Что нам и нужно, так как в режиме редактирования записи значение ключевого поля менять будет нельзя. Не забываем указать, что поле обязательно к заполнению. Значение по умолчанию поля всегда совпадает со значением входного параметра. Укажем здесь ссылку на него – `{PCODE}`.
- NAME (Субъект) – Особенность: поле обязательно к заполнению
- AMOUNT (Население) – Особенность: «Тип данных» - NUMBER
- EXPIRES(Актуально до) – Особенность: «Тип данных» - DATE, «Форматная маска» - DD.MM.RRRR
- PARENT_CODE (Родитель) – По сути, это ссылка на строку в той же таблице. Чтобы избежать ошибок разрешаем изменять значение поля только посредством выбора из списка допустимых значений: «Способ изменения данных» - C (Только из списка возможных значений). В свойство «Набор возможных значений помещаем запрос, заполняющий этот список:

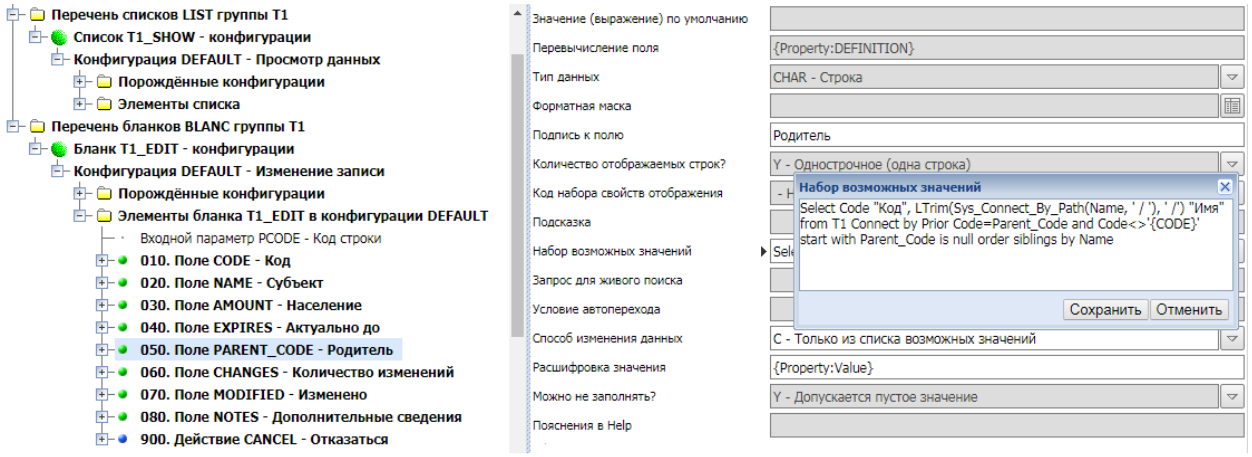
```
Select Code "Код", LTrim(Sys_Connect_By_Path(Name, ' / '), ' /') "Имя"
from T1 Connect by Prior Code=Parent_Code and Code<>'{CODE}'
start with Parent_Code is null order siblings by Name
```

Запрос выводит данные в 2 колонки. Первая колонка – код, который и будет значением поля после выбора соответствующей строки. Вторая колонка – полный путь по именам к записи с кодом Code в иерархии подчинения. Записи отображаются с сохранением иерархии с сортировкой по именам на каждом уровне иерархии. Условие соединения `Code<>'{CODE}'` гарантирует, что пользователю не отобразится в списке выбора редактируемая строка и все ей прямо и косвенно подчинённые строки – для избегания циклов подчинённости.

Ещё у поля PARENT_CODE необходимо указать в качестве расшифровки внутреннюю переменную `{Property:Value}`, так как при способе изменения «С» и наличии набора возможных значений система будет стараться показать в поле значение из второго столбца запроса набора возможных значений, найден-

ному по значению первого столбца. Нас же не интересует расшифровка, а интересует сам код родителя (то есть, значение поля). А если бы и интересовала расшифровка, то нелогично было бы выполнять столь сложный и сортированный запрос, когда нужно найти значение поля Name по ключу: `{Select Name from T1 where Code='{CODE}'}`.

- CHANGES (Количество изменений) – Особенность: неизменяемое пользователем поле. Технологическое значение, отображающее количество фактов изменения текущей записи таблицы T1. Значение в этом поле меняется программно. Поэтому, в свойстве «Способ изменения данных» укажем N, а в «Значении по умолчанию» - 0. Это значение будет помещено в данное поле только если бланк открыт для добавления новой записи в таблицу. При редактировании значение 0 будет замещено (почему – см. ниже).
- MODIFIED (Изменено) – Особенность: неизменяемое пользователем поле (Способ изменения данных = N), «Тип данных» - DATE, «Форматная маска» - RRRR.MM.DD HH24:MI:SS
- NOTES (Дополнительные сведения) – Особенность: «Количество отображаемых строк» - 3.



Теперь перейдём к редактированию свойств самого бланка. Для этого встанем на узел «Конфигурация DEFAULT» бланка T1_EDIT.

Укажем свойство «Select заполнения». Это SQL – запрос (без фигурных скобок), который будет выполняться при загрузке бланка. Если запрос вернёт одну строку, то значения, алиасы которых совпадут с кодом некоторого поля, поместятся как начальные значения в эти поля, невзирая на свойство поля «Значение (выражение) по умолчанию». Если запрос не вернёт ни одной строки, то во всех полях будут использоваться значения по умолчанию, указанные в свойстве DEFINITION (Значение (выражение) по умолчанию).

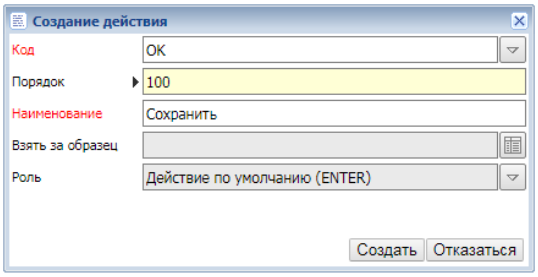
В нашем случае, если параметр PCODE не пуст, то во все поля нужно поместить значения соответствующих ячеек редактируемой строки таблицы T1:

```
Select NAME, AMOUNT, EXPIRES, PARENT_CODE, CHANGES, MODIFIED, NOTES
from T1 where Code='{PCODE}'
```

Укажем ещё заголовок бланка. Это свойство динамическое – зависит от значения входного параметра PCODE. Если параметр задан (не пуст), то заголовок должен говорить о редактировании записи, если параметр пуст, то заголовок должен обозначать добавление записи:

```
{Select NVL2('{PCODE}', 'Изменение записи {PCODE}', 'Добавление новой записи в T1') from dual}
```

Наш бланк уже умеет отображать данные и корректно ими манипулировать. Однако основное его предназначение – изменение данных. То есть, нам нужно создать действие, которое применяет сделанный пользователем изменения.



Создадим новое действие. Код выберем из списка – OK (Сохранить). Не забудем изменить порядок на число, мене 900 – чтобы кнопка «Сохранить» было ранее кнопки «Отказаться».

У этого действия все свойства уже правильные и нам нужно только указать программный код, который сохранит введённые изменения:

```
begin
  if '{PCODE}' is null then
    Insert into T1
      (CODE, NAME, AMOUNT, EXPIRES, PARENT_CODE, CHANGES, MODIFIED, NOTES)
    Values
      ('{CODE}', '{NAME}', To_Number('{AMOUNT}'), To_Date('{EXPIRES}', 'DD.MM.RRRR'),
'{PARENT_CODE}', 1,
ysDate, '{NOTES}');
  else
    Update T1 set
      NAME='{NAME}',
      AMOUNT=To_Number('{AMOUNT}'),
      EXPIRES=To_Date('{EXPIRES}', 'DD.MM.RRRR'),
      PARENT_CODE='{PARENT_CODE}',
      CHANGES=CHANGES+1,
      MODIFIED=SysDate,
      NOTES='{NOTES}'
    where Code='{PCODE}';
  end if;
  Commit;
end;
```

★ Связывание Списка и Бланка

Обычно сначала создаются список и бланк, а потом в Список добавляются действия, вызывающие бланк. Наш случай простой – у бланка только один входной параметр, но чаще бланки бывают сложнее, с множеством параметров и при наличии готового бланка подключение его вызова в Список происходит быстрее и с меньшей вероятностью ошибки.

Действия списка T1_SHOW, которые вызывают бланк T1_EDIT – это ADD (Добавить) и EDIT (Изменить). По сути действия, выполняющие некую команду по отображению другой формы (в нашем случае – бланка), являются много событийными. То есть, действие начинается по некоторому событию (например, выбор пункта меню или клик мышкой на пиктограммке), но не заканчивается. После отображения бланка управление снова передаётся клиенту. Действие заканчивается только после закрытия бланка.

Что происходит, когда выполняется действие?

Начнём с простого случая. Действие DEL (Удалить).

- Устанавливается SavePoint в СУБД.
- Действие для строки (Признак активности = EXIST) и есть просьба накладывать блокировку (Накладывать блокировку = Y). Производится попытка наложить блокировку на текущую строку Списка. Для этого используется основной запрос Списка (SQL – выражение), информация о столбцах, входящих в уникальный ключ (Входит в уникальный ключ строки? = Y) и текущие значения этих столбцов. Если запись заблокирована на протяжении более 1 секунды, то действие прекращается, происходит откат транзакции к SavePoint и отображение на экран пользователю информации о блокировке. Если попытка блокировки не удалась (например, из-за специфики основного запроса), то никаких ошибок не возникает, исключительная ситуация подавляется и действие продолжается без наложения блокировки.
- Если есть вопрос пользователю (Текст предупреждения) и он не пуст, то этот текст в модальном режиме возникает у пользователя на экране с двумя кнопками. Если текст предупреждения заканчивается знаком вопроса, то на кнопках будут надписи «Да» и «Нет». Иначе на кнопках отобразится «Продолжить» и «Отменить». В случае выбора второй кнопки действие прекращается и происходит откат транзакции к SavePoint.
- Выполнятся «PL/SQL-блок»
- Выполнятся «PL/SQL действие после команды»
- Если в процессе возникла ошибка с необработанным Exception, то действие прекращается, происходит откат транзакции к SavePoint, а на экране у пользователя возникает тест ошибки.
- Выполняется «Действие после выполнения». В нашей случае - перечитывание всех строк Списка, в которых уже не окажется удалённой записи.

  Рассмотрим как выполнятся действие с вызовом формы на примере действия EDIT (Изменить)

- Устанавливается SavePoint в СУБД.
- Накладывается блокировка на текущую строку Списка
- Если запись заблокирована на протяжении более 1 секунды, то далее бланк будет открываться в режиме «ReadOnly» - Поля не редактируются, действия недоступны.

- Открывается бланк, заполняются его значения по умолчанию.
- Если в процессе открытия бланка случается ошибка с необработанным Exception, то действие прекращается, происходит откат транзакции к SavePoint, а на экране у пользователя возникает тест ошибки.
- То же самое произойдёт, если бланк закроется действием, у которого свойство «Действие после выполнения» = CANCEL.
- Если бланк закроется действием с «Действие после выполнения» = EXIT или вообще без действия (крестиком без выполнения ассоциированного действия), то, как бы, продолжается выполнение вызвавшего бланк действия Списка – EDIT, а именно
- Выполняется «Действие после выполнения». В нашем случае - перечитывание данных текущей строки Списка, которая была модифицирована закрывшемся бланком.

Что мы имеем в итоге? Средство для согласованного просмотра всех данных в таблице, средство для поиска и отбора только необходимых данных, сгруппированных и отсортированных по некоторым признакам, средство добавления новых данных в таблицу, средство удаления и изменения данных в таблице, причём защищённое от возможных конфликтов совместного доступа.

Дополнительные функции редактора таблицы

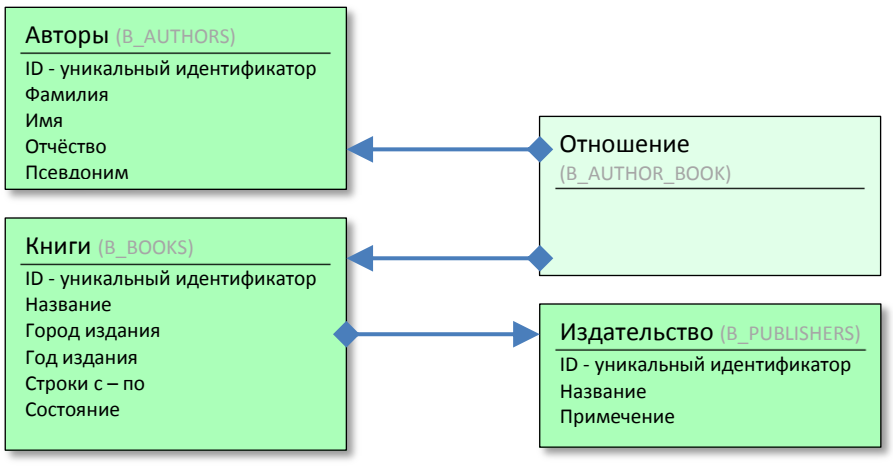
Простейший редактор таблицы создать не трудно, тем более с использованием визардов, делающих за разработчика 90% вышеописанной работы. Однако Списки и бланки редко функционируют в отрыве от Системы, и зачастую возникает потребность объединения форм с похожим функционалом или использование их в разных местах Системы для немного разных целей.

Например, Список договоров может просто отображать все договоры, заведённые в Систему, договоры одного клиента или контрагента, договоры должников, договоры управляющего ими менеджера. Понятно, что набор функций для каждого списка специфический, но обычно имеется и очень много общих функций. Чтобы не повторять общую функциональность из формы в форму хорошо бы реализовать некую конфигурацию, где учтены все теоретически возможные случаи и действия, в а каждом конкретном месте вызова формы создавать подчинённую конфигурацию, где лишняя функциональность отключена.

Для достижения желаемого рекомендуется создавать один список для отображения экземпляров сущности со всевозможными атрибутами в качестве столбцов, всевозможными действиями над экземпляром или экземплярами сущности. Вариативность может осуществляться следующими методами:

- Входными параметрами, влияющими на интерфейс Списка и его функциональность;
- Динамическими свойствами, учитывающими обстоятельства использования Списка;
- Конфигурациями Списка, отключающими или переопределяющими общую функциональность.

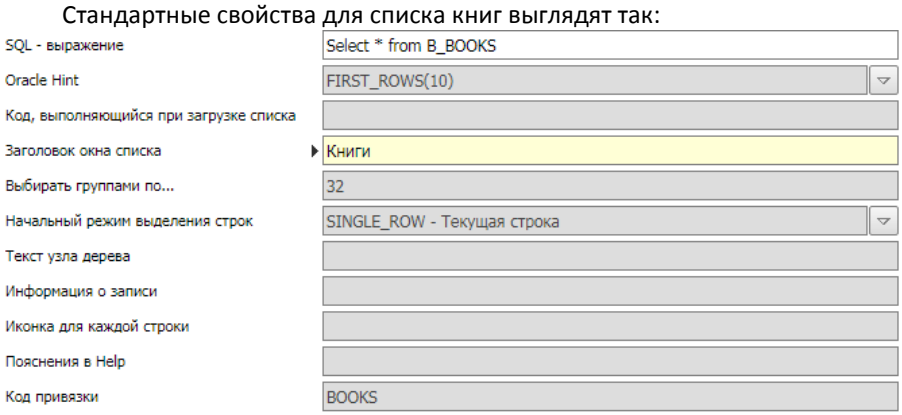
В примерах будем рассматривать такую учебную схему данных:



★ Ограничение на доступные строки

Хорошо бы создать Список, который при некоторых условиях отображал не все экземпляры сущности, а отобранные по заданному признаку. Например, по точному совпадению: «не все клиенты, а только юридические лица». Или по порогу: «Не все договоры, а только с балансом ниже заданного уровня». Или по принадлежности к конкретной мастер-записи: «Платежи указанного клиента», «Начисления на этот договор».

Стандартные свойства для списка книг выглядят так:



Допустим, что список книг будет вызываться из списка издательств. Там будет действие «Книги издательства» и нужно будет отобразить список книг, но только те, в которых есть ссылка на издательство (в поле PUBLISHER_ID).

Добавим в список книг входной параметр – P_PUBLISHER_ID. Без значения по умолчанию. Если параметр не заполнен – нам нужно отображать все книги из таблицы B_BOOKS, а если заполнен – то только те, где PUBLISHER_ID будет ранен значению параметра P_PUBLISHER_ID