Документация

Объединение (дерево)

Узел Объединение (дерево) может работать в 2-х режимах:

- *Объединение нескольких входных деревьев данных* в одно выходное дерево. По смыслу это аналог операции UNION для таблиц, но применительно к структуре дерева данных.
- *Выбор первого активного дерева*, при котором на выход узла передается лишь одно дерево из входящих.

Один из кейсов применения компонента — объединение результатов запросов к внешним сервисам с различной структурой в единое дерево для дальнейших расчетов.

У узла есть возможность активации с неактивными входными портами, как и у узла Объединение таблиц, что позволяет использовать его для объединения веток сценария, формируемых узлом Условие. Это является одним из основных способов применения режима **Выбор первого активного дерева**.

Вход

- **Главное дерево** — порт для подключения главного дерева данных, обязательный.
- **Присоединяемое дерево** — порт для подключения присоединяемого дерева данных, обязательный.
- **Добавить еще один порт** — создает новые порты входа для последующих присоединяемых деревьев, которые будут автоматически пронумерованы.

Выход

- **Выходное дерево данных** — результат объединения деревьев согласно выбранного типа операции.

Мастер настройки

Настройка объединения деревьев

Тип операции — выбор способа объединения.

Тип операции можно выбрать из двух вариантов:

- **Конкатенация всех деревьев** — если выбран данный режим, то корневой узел выходного дерева — это массив, содержащий корневые узлы деревьев из всех активных входных портов.
- **Выбор первого активного дерева** — если выбран данный режим, то выходное дерево содержит те же данные, что и дерево в первом активном входном порту.

Описание режимов работы компонента

Конкатенация всех деревьев

Логика объединения деревьев сводится к следующим правилам:

1. Объединение корневых контейнеров в массив
2. Объединение схем деревьев
3. Конфликт типов данных на одном пути
4. Учёт схем входных деревьев для неактивных портов

1. Объединение корневых контейнеров в массив

Корневые узлы входных деревьев объединяются в массив элементов. Имя корневого массива на выходе задаётся именем корневого узла дерева в порту «Главное дерево».

Если корневой узел входного дерева сам является массивом, то элементы этого массива преобразуются в отдельные элементы итогового массива.

Пример:

Входные данные:

```
*/ Главное дерево /*
☰ user
└── (ab)   tag:    "admin"

*/ Присоединяемое дерево 1 /*
☰ client
└──(ab)    tag:    "viewer"

*/ Присоединяемое дерево 2 (корневой узел — массив user) /*
├── ☰ user
│   └──(ab) tag:  "designer"
└── ☰ user
    └──(ab) tag:  "guest"
```

Результат объединения:

```
*/ Схема дерева /*

[☰] user                          // На выходе сформирован массив элементов
user
 └──(ab) tag                       // Строковый тип данных

*/ Итоговые данные /*

├── ☰ user                         // Элемент исходного главного дерева
│   └──(ab) tag: "admin"
├── ☰ user                         // Корневой узел client преобразован в
user
│   └──(ab) tag: "viewer"
├── ☰ user                         // Элемент массива из "Присоединяемое
дерево 2"
│   └──(ab) tag: "designer"
└── ☰ user                         // Элемент массива из "Присоединяемое
дерево 2"
    └──(ab) tag: "guest"
```

2. Объединение схем деревьев

На каждом уровне выходного дерева для каждого контейнера (в том числе и контейнера-массива) в итоговой схеме данных формируется общий для всех входных деревьев набор дочерних элементов.

Заполнение данными итогового дерева с общим набором элементов происходит следующим образом:

1. Отсутствующий в отдельном входном дереве дочерний элемент простого типа заполняется значением `null`.
2. Отсутствующий в отдельном входном дереве дочерний контейнер и/или массив в итоговое дерево данных не включается.

Таким образом, элементы простых типов, попавшие в схему в результате объединения, фактически ведут себя как «обязательные поля» (всегда присутствуют в данных, но могут иметь значение `null`), а контейнеры и массивы — как «необязательные» (присутствуют в итоговой схеме, но могут отсутствовать в данных итогового дерева).

Пример:

Входные данные:

```
*/ Главное дерево /*

☰ user
├──(ab)  tag:          "admin"
├──(0/1) blocked:       true
└── ☰ permissions
    └── ☰ fileStorage
        └──(0/1) read:  true


*/ Присоединяемое дерево /*

☰ user
├──(ab)  tag:          "viewer"
├──(ab)  fullName:     "Петров Н.Н."
└── ☰ permissions
    ├── ☰ fileStorage
    │   └──(0/1) write:          false
    └── ☰ planner
        └──(0/1) accessAllTasks: true
```

Результат объединения:

```
*/ Объединённая схема /*

[☰] user                          // Массив элементов user
 ├──(ab)  tag                     // Строковый тип данных
 ├──(0/1) blocked                 // Логический тип данных
 ├──(ab)  fullName                // Строковый тип данных
 └── ☰ permissions
     ├── ☰ fileStorage
     │   ├──(0/1) read            // Логический
     │   └──(0/1) write           // Логический
     └── ☰ planner
         └──(0/1) accessAllTasks  // Логический


*/ Объединенные данные /*

├── ☰ user              // Контейнер planner отсутствует для этого user
│   ├──(ab)  tag:           "admin"
│   ├──(0/1) blocked:        true
│   ├──(ab)  fullName:       null
│   └── ☰ permissions
│       └── ☰ fileStorage
│           ├──(0/1) read:   true
│           └──(0/1) write:  null
└── ☰ user
    ├──(ab)  tag:           "viewer"
    ├──(0/1) blocked:        null
    ├──(ab)  fullName:      "Петров Н.Н."
    └── ☰ permissions
        ├── ☰ fileStorage
        │   ├──(0/1) read:    null
        │   └──(0/1) write:   false
        └── ☰ planner
            └──(0/1) accessAllTasks: true
```

В результате элементы простых типов, которые попали в схему (*tag*, *blocked*, *fullName*, *read*, *write*), всегда присутствуют в данных (иногда со значением `null`), а контейнер *planner* для первого *user* отсутствует полностью, хотя описан в схеме — это иллюстрирует различие поведения простых типов и контейнеров/массивов в Правиле 2.

3. Конфликт типов данных на одном пути

> *Примечание:* В дереве данных под путём элемента понимается его положение относительно корневого узла. Полный путь элемента — это цепочка имён родительских контейнеров и имя самого элемента. Полный путь однозначно определяет местоположение элемента в дереве данных.

Если во входных деревьях одноимённые элементы с одинаковыми путями имеют разные типы данных (в том числе свойства *Массив*, *Контейнер*), то в результирующем дереве для каждого типа создаются отдельные элементы путём добавления суффикса к имени элемента.

Различие *Меток*, Видов данных и *Назначения* не является конфликтом. Если в нескольких деревьях есть неконфликтующие узлы, у которых совпадает путь и имя, то *Метка*, *Вид данных* и *Назначение* берутся из первого по счёту дерева, в котором есть этот узел.

Заполнение значений созданных таким образом новых дочерних элементов осуществляется по Правилу 2.

Пример:

Входные данные:

```
*/ Главное дерево /*

☰ user
└──(ab)  tag: "admin"

*/ Присоединяемое дерево 1 /*

☰ user
└──(12)  tag:  100

*/ Присоединяемое дерево 2 /*

☰ user
└──(0/1) tag:  true
```

Результат объединения:

```
*/ Схема итогового дерева /*

[☰] user                        // Массив элементов user
 ├──(ab)  tag                    // Строковый
 ├──(12)  tag_1                  // Целый
 └──(0/1) tag_2                  // Логический


*/ Данные итогового дерева /*

├── ☰ user
│   ├──(ab)  tag:   "admin"
│   ├──(12)  tag_1:  null
│   └──(0/1) tag_2:  null
├── ☰ user
│   ├──(ab)  tag:    null
│   ├──(12)  tag_1:  100
│   └──(0/1) tag_2:  null
└── ☰ user
    ├──(ab)  tag:    null
    ├──(12)  tag_1:  null
    └──(0/1) tag_2:  true
```

4. Учёт схем входных деревьев для неактивных портов

Схема (структура) итогового дерева формируется как объединение схем во всех входных портах. Даже если при текущей активации узла на порт данные не поступили (порт неактивен), **описанные в его схеме элементы участвуют в формировании структуры итогового дерева**.

> **Примечание**: Неактивными порты могут быть в кейсе объединения веток сценария, образованных узлом *Условие*.

Пример (с учетом применения всех правил):

Входные данные:

```
*/ Главное дерево (порт «Главное дерево») /*

☰ user
├──(ab)  tag:               "admin"        // Строковый
├──(0/1) blocked:            true          // Логический
└── ☰ permissions
    ├──(0/1) packagePublish: true          // Логический
    └── ☰ fileStorage
        ├──(0/1) read:       true           // Логический
        └──(0/1) write:      false          // Логический


*/ Присоединяемое дерево 1 /*

☰ user
├──(12) tag:             100              // Целый
├──(ab) fullName:       "Петров Н.Н."     // Строковый
└── ☰ permissions
    ├──(0/1) packagePublish:     false    // Логический
    ├── ☰ fileStorage
    │   ├──(0/1) read:           true     // Логический
    │   └──(0/1) delete:         true     // Логический
    └── ☰ planner
        └──(0/1) accessAllTasks: true     // Логический


*/ Присоединяемое дерево 2 (неактивный порт, данные отсутствуют) /*
// в порту определена схема:

[☰] user                              // Массив элементов user
├──(0/1) tag                           // третий тип для tag
└── ☰ permissions
    └──(9.0) limit                     // дополнительный элемент
```

Результат объединения:

```
*/ Схема итогового дерева /*

(☰) user                          // Массив элементов user
 ├──(ab)   tag
 ├──(12)   tag_1
 ├──(0/1)  tag_2                 // добавлен из схемы неактивного порта
 ├──(0/1)  blocked
 ├──(ab)   fullName
 └── ☰ permissions
     ├──(0/1) packagePublish
     ├──(9.0) limit              // добавлен из схемы неактивного порта
     ├── ☰  fileStorage
     │   ├──(0/1) read
     │   ├──(0/1) write
     │   └──(0/1) delete
     └──☰ planner
        └──(0/1) accessAllTasks

*/ Данные итогового дерева /*

├── ☰ user                   // контейнер planner отсутствует для этого
user
│   ├──(ab)  tag:            "admin"
│   ├──(12) tag_1:            null
│   ├──(0/1) tag_2:           null    // добавлен из схемы неактивного
порта
│   ├──(0/1) blocked:         true
│   ├──(ab)  fullName:        null
│   └── ☰ permissions
│       ├──(0/1) packagePublish:  true
│       ├──(12) limit:            null  // добавлен из схемы неактивного
порта
│       └── ☰ fileStorage
│           ├──(0/1) read:    true
│           ├──(0/1) write:   false
│           └──(0/1) delete:  null
└── ☰ user
    ├──(ab)  tag:            null
    ├──(12) tag_1:           100
    ├──(0/1) tag_2:          null      // добавлен из схемы неактивного
порта
```

```
    ├──(0/1) blocked:        null
    ├──(ab)  fullName:      "Петров Н.Н."
    └── ☰ permissions
        ├──(0/1) packagePublish:  false
        ├──(12) limit:            null  // добавлен из схемы неактивного
порта
        ├── ☰ fileStorage
        │   ├──(0/1) read:    true
        │   ├──(0/1) write:   null
        │   └──(0/1) delete:  true
        └── ☰ planner
            └──(0/1) accessAllTasks: true
```

> **Важно**: Если в неактивном порту схема не задана вручную, то результат на выходе объединения может быть непредсказуем. Для получения гарантированной структуры итогового дерева рекомендуется зафиксировать схему данных, например, путем отключения автосинхронизации в порту при поданных на него данных.

Выбор первого активного дерева

В данном режиме в результирующее дерево вместо объединенных данных всех деревьев выводятся данные дерева из первого активного входного порта.

Алгоритм объединения для данного типа

1. Узел создает объединенную схему данных всех входных деревьев согласно описанным выше правилам 2, 3 и 4:
    - Объединяет наборы дочерних элементов всех деревьев (Правило 2).
    - Разделяет элементы разных типов на одном пути добавлением суффиксов (Правило 3).
    - Учитывает схемы неактивных деревьев (Правило 4).
2. Последовательно просматривает входные порты в указанном порядке — от «Главного дерева» до «Присоединяемое дерево N».
3. Находит первый активный порт (с поступившими на него данными), приводит данные этого порта к объединенной схеме в соответствии с правилами 2, 3 и выводит их в выходном порту.

Таким образом на выходе формируются данные только первого активного порта, приведенные к объединенной схеме по всем портам (включая неактивные).

Корневой узел выходного дерева является массивом в случае, если корневой узел хотя бы одного из входных деревьев является массивом (независимо от активности входных портов).

Пример:

Входные данные:

```
*/ Главное дерево (порт неактивен, определена схема данных) /*

[☰] user                       // Массив элементов user
 ├──(0/1) tag
 └── ☰ permissions
     └──(9.0) limit

*/ Присоединяемое дерево (порт активен) /*
☰ client
├──(ab)  tag:     "admin"
└──(0/1) blocked:  true

*/ Присоединяемое дерево 2 (порт активен) /*
☰ user
└──(ab)  fullName: "Петров Н.Н."
```

Результат:

```
*/ Схема итогового дерева /*
// Корневой узел становиться массивом, поскольку в одном из входных деревьев корневой узел - массив

[☰] user
 ├──(0/1)  tag        // из схемы порта без данных "Главное дерево" (Правило 2 и 4)
 ├──(ab)   tag_1      // из схемы порта "Присоединяемое дерево" (Правило 3)
 ├──☰ permissions    // из схемы порта без данных "Главное дерево" (Правило 2), необязательный
 |   └──(9.0) limit
 ├──(0/1)  blocked    // из схемы порта "Присоединяемое дерево" (Правило 2)
 └──(ab)   fullName   // из схемы порта "Присоединяемое дерево 2" (Правило 2)

*/ Данные итогового дерева /*

└── ☰ user                       // 1 элемент массива user, необязательный permissions не выводится
    ├──(0/1)  tag:       null     // элемент есть в схеме, но данных не было → null по Правилу 2
    ├──(ab)   tag_1:    "admin"
    ├──(0/1)  blocked:   true
    └──(ab)   fullName:  null     // элемент есть в схеме, но данных не было → null по Правилу 2
```