Relation

Relations

By default, relations join together two datasources, for example a CUSTOMERS (source) and a ORDERS (destination) table. Through RelationItems the nature of the link between the datasources is defined, for example to link the CUSTOMERS and ORDERS table based on the customer_id column.

At runtime a relation is an object of type JSFoundSet. Relations are used in Solutions to display related data, for example using a Field , a TabPanel or a Portal or to work with related data in business logic.

Special Relations

Next to the default relations that link two datasources together, there are several other types of relations, based on how the relation is setup.

Global Relations

Relations that contain only RelationItems based on global variables on the source side, thus no dependency on the source table are considered global relations.

Global relations can be used to get a limited FoundSet, for example all active customers, by adding a relation item that specifies that the "active" flag in the CUSTOMERS table has to be equal to a certain global variable. For example, if the CUSTOMERS table (used as destination datasource) contains a column called "is_active" of type INTEGER and a default value of 1 (indicating it is an active record), the following would create a global relation that returns a FoundSet with all active customers:

Setup a global variable

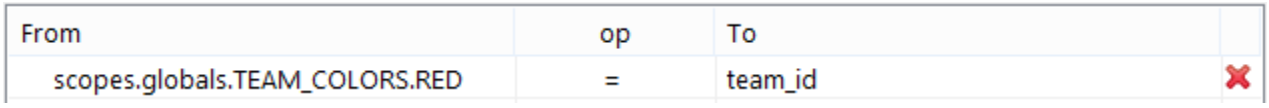
```
var ACTIVE = 1;
```

From	op	To
globals.ACTIVE	=	is_active

A global variable can be defined as an enumeration of constants, using the @enum annotation. See Annotating JavaScript Using JSDoc. The constants of the enumeration can be used in relations just as the common global variables are used.

Setup a global enumeration

```
/**
 * @enum
 */
var TEAM_COLORS = {
    RED : 1,
    GREEN : 2,
    BLUE : 3
}
```



From	op	To
scopes.globals.TEAM_COLORS.RED	=	team_id

Container Relations

If the source and destination datasources are equal and the relation does not contain any RelationItems, it becomes a Container relation. A Container relation returns it originating FoundSet. A Container relation can be used to set the FoundSet of the parent form into the Form displayed in a TabPanel.

Selfjoins

Relations can use the same datasource as source and destination datasource. Depending on the setup of the the RelationItems the relation can point back to the originating record or one or more different records.

Relation Options

Relations have several settings that influence the behavior of the Relation under different circumstances.

Join Types

Join types determine the behavior of sorting when there are no related records and the sort is performed on related fields.

Inner join : When sorting records on related dataproviders over a relation that uses the "inner join" type, if a parent record does not have any related records, the parent record is excluded from the foundset
Left outer join : When sorting records on related dataproviders over a relation that uses the "left outer join" type, if a parent record does not have any related records, the parent record remains in the foundset

Use Left outer join

In most scenario's the "Left outer join" type is the join type to use for relations

Record Creation

Allow creation of related records : This property indicates if the relation can be used to create new related records

- true: the execution of the newRecord() method on the relation (for example "customers_to_orders.newRecord()") will create a new record.
- false: the execution of the newRecord() method on the relation will throw a ServoyException.NO_RELATED_CREATE_ACCESS exception.

Record Deletion

Allow parent delete when having related records : This property indicates if the parent record can be deleted if it has related records over this relation

- true: When an attempt is made to delete the parent record, this relation **WILL NOT** be checked for the existence of records over this relation and thus it will not block the delete of the parent record
- false: When an attempt is made to delete the parent record, a check **WILL** be made for the existence of records over this relation. When related records exist over this relation, the delete of the parent record will be blocked and a ServoyException.NO_PARENT_DELETE_WITH_RELATED_RECORDS exception will be thrown.

Allow parent delete when having related records* overrules *Delete related records

The option **Allow parent delete when having related records** overrules the option **Delete related records** . This means that if the first option is set to false, the latter option is ignored. Setting the first option to false and the latter option to true will still block the delete.

All relations for the datasource checked on delete

When an attempt is made to delete a record, each relation that uses the datasource of the record (that is to be deleted) as the source datasource is checked to see if it allows the delete to proceed. If one relation is found that blocks the delete, the entire delete will fail.

Performance

If the option **Allow parent delete when having related records** is set to false on a relation, when a record is deleted from the datasource that is used as the source datasource for the relation, a check will be made to see if there are related records. If many relations exist with the option set to false, this can impact the performance of deletes.

Delete related records : This property indicates if child records should be automatically deleted when the parent record is deleted

- true: when the parent record is deleted and there are records over this relation, those related records will also get deleted.
- false: related records will remain untouched.

Cascading deletes

Deletes cascade down when encountering relations on the related records that are to be deleted that also have the **Delete related records** property set to true. This means that when a record from the CUSTOMERS datasource is deleted and the relation customer_to_orders has the **Delete related records** set to true, the order records will also be deleted. If there is a relation order_to_orderitems that also has the **Delete related records** property set to true, the orderitem records will also get deleted.

Deleting of related records can block the overall delete

When a relation has the **Delete related records** property set to true, an attempt is made to also delete all individual related records. On each individual related record the checks mentioned previous will also be performed. If any of the related records blocks the delete, the entire delete is blocked.

Example 1: A simple scenario
This example defines three relations in the typical customer > orders > orderitems setup:
Relation 1: customer_to_orders, allowParentDeleteWhenHavingRelatedRecords=true, deleteRelatedRecords=true
Relation 2: order_to_orderitems, allowParentDeleteWhenHavingRelatedRecords=true, deleteRelatedRecords=true
Relation 3: orderitem_to_product, allowParentDeleteWhenHavingRelatedRecords=true, deleteRelatedRecords=true

If an attempt is made to delete a customer record which has order and the orders have orderitems, when the relations are setup as described above the customer record will be deleted, all the orders related to the customer are deleted and all orderitems related to the orders of the customer will be deleted.

As Relation 3 has the deleteRelatedRecords property set to false, no attempt will be made to delete the product record related to the orderitems that are to be deleted.

Example 2: Implementing business logic enforcement in the data layer
This example defines four relations starting with the in the typical customer > orders > orderitems setup, followed by the following business logic enforced by the datamodel: If an order resulted in an Invoice, the id of the invoice is stored on the order record in the invoice_id column. If an order has a related invoice then it cannot be deleted
Relation 1: customer_to_orders, allowParentDeleteWhenHavingRelatedRecords=true, deleteRelatedRecords=true
Relation 2: order_to_orderitems, allowParentDeleteWhenHavingRelatedRecords=true, deleteRelatedRecords=true
Relation 3: orderitem_to_product, allowParentDeleteWhenHavingRelatedRecords=true, deleteRelatedRecords=true
Relation 4: order_to_invoice, allowParentDeleteWhenHavingRelatedRecords=false

This setup is mostly the same as example 1, except for the additional fourth relation. This relation from the orders table to the invoices defines that the order record cannot be deleted if it has a related invoice record.

In this setup, when an attempt is made to delete a customer record the delete is blocked if the customer has a related order that has a related invoice record.

Property Summary

Boolean	allowCreationRelatedRecords Flag that tells if related records can be created through this relation.
Boolean	allowParentDeleteWhenHavingRelatedRecords Flag that tells if the parent record can be deleted while it has related records.
Boolean	deleteRelatedRecords Flag that tells if related records should be deleted or not when a parent record is deleted.
String	deprecated Gets the deprecate info for this element
Number	encapsulation The encapsulation mode of this persist.
String	foreignDataSource Qualified name of the foreign data source.
String	initialSort A String which specified a set of sort options for the initial sorting of data retrieved through this relation.
Number	joinType The join type that is performed between the primary table and the foreign table.
String	name The name of the relation.
String	primaryDataSource Qualified name of the primary data source.

Property Details

allowCreationRelatedRecords

Flag that tells if related records can be created through this relation.

The default value of this flag is "false".

Returns

Boolean

allowParentDeleteWhenHavingRelatedRecords

Flag that tells if the parent record can be deleted while it has related records.

The default value of this flag is "true".

Returns

Boolean

deleteRelatedRecords

Flag that tells if related records should be deleted or not when a parent record is deleted.

The default value of this flag is "false".

Returns

Boolean

deprecated

Gets the deprecate info for this element

Returns

String - the deprecate info for this element or null if it is not deprecated

encapsulation

The encapsulation mode of this persist. The following can be used/checked:

- Public (not a separate option - if none of the below options are selected)
- Hide in scripting; Module Scope - not available in scripting from any other context except the form itself. Available in designer for the same module.
- Module Scope - available in both scripting and designer but only in the same module.
- Hide Dataproviders (checked by default)
- Hide Foundset (checked by default)
- Hide Controller (checked by default)
- Hide Elements (checked by default)

Returns

Number - the encapsulation mode/level of the persist.

foreignDataSource

Qualified name of the foreign data source. Contains both the name of the foreign server and the name of the foreign table.

Returns

String

initialSort

A String which specified a set of sort options for the initial sorting of data retrieved through this relation.

Has the form "column_name asc, another_column_name desc, ...".

Returns

String

joinType

The join type that is performed between the primary table and the foreign table. Can be "inner join" or "left outer join".

Returns

Number

name

The name of the relation.

Returns

String

primaryDataSource

Qualified name of the primary data source. Contains both the name of the primary server and the name of the primary table.

Returns

String