# Relationships

Last Modified on 21/02/2020 8:46 am AEDT

## What is a Relationship?

Relationships allow records to be associated with each other.

Depending on context, the word relationship might refer to:

- a type of relationship
    - that defines a particular type of association between two objects
    - For example: there is an "Assigned tasks" relationship defined between the Person object and the Task object.
- an instance of a relationship
    - that represents a concrete association, of a particular relationship type, between two particular records
    - For example: 'Susan has been assigned the Q1 2020 Procurement Risk Assessment task'.

## To-one and To-many relationships

Every relationship type has a *cardinality*, which simply describes whether each record can relate to one or multiple related records.

Relationships types that only allow a record to point to a single related record are referred to as a *lookup*.

The following relationship cardinalities are possible:

- one-to-many
- many-to-one
- many-to-many
- one-to-one

### One-to-Many relationships

For a one-to-many relationship, each record on the first side of the relationship can be related to multiple records on the seconds side. However, in the other direction, this is an exclusive relationship, and the second record can only be related from one record.

For example: a "Building has Rooms" relationship would be one-to-many.

To define a one-to many relationship: add a relationship control to a form. Refer to Relationship Control for more details.

## Many-to-One Relationships

Each record on the first side relates to a single record on the second side. For example: an "Employee is in a Department".

You will notice this is equivalent to a one-to-many relationship, but defined in the other direction. The platform does not distinguish between the two, so use which ever feels most natural in context.

To define a many-to-one relationship: add a lookup control to a form. Refer to Lookup Control for details.

## Many-to-Many Relationships

Records can be associated with multiple records in either direction.

For example: "Employee is invited to Meeting".

To define a many-to-many relationship: add a relationship control to a form, then change the **Relationship Type** setting to relate to many records in both directions.

## One-to-One Relationships

Each record, on either side, can only associated to only a single record. The relationship is exclusive in both directions. For example, "Person has Payroll Details".

It is uncommon to require this type of relationship, but it can be useful to secure access to certain information, such as allowing Person records to be visible to certain users without also making Payroll Details accessible.

To define a one-to-one relationship:  add a lookup control to a form and then change the

**Relationship Type** settings to relate to one record in both directions.

## Relationships are Bi-Directional

Relationships have meaning in both directions and can be used in both directions.

For example, consider a relationship between the Employee object and the Employee object that represents the employee-manager relationships.

- in one direction it may simply be called "Manager", and is considered as being a lookup. That is, each employee has (at most) one manager.
- in the other direction it may be called "Direct Reports", and in this direction is considered a to-many relationship.

Any association made between records in one direction is immediately present in the other direction as well.

When a new Relationship or Lookup is first added to a form, the control will initially only appear on that form. However, the relationship is now available to use in all application building features, such as forms, reports, workflows, and calculations, in both directions. That is to say, from either of the two objects that the relationship connects.

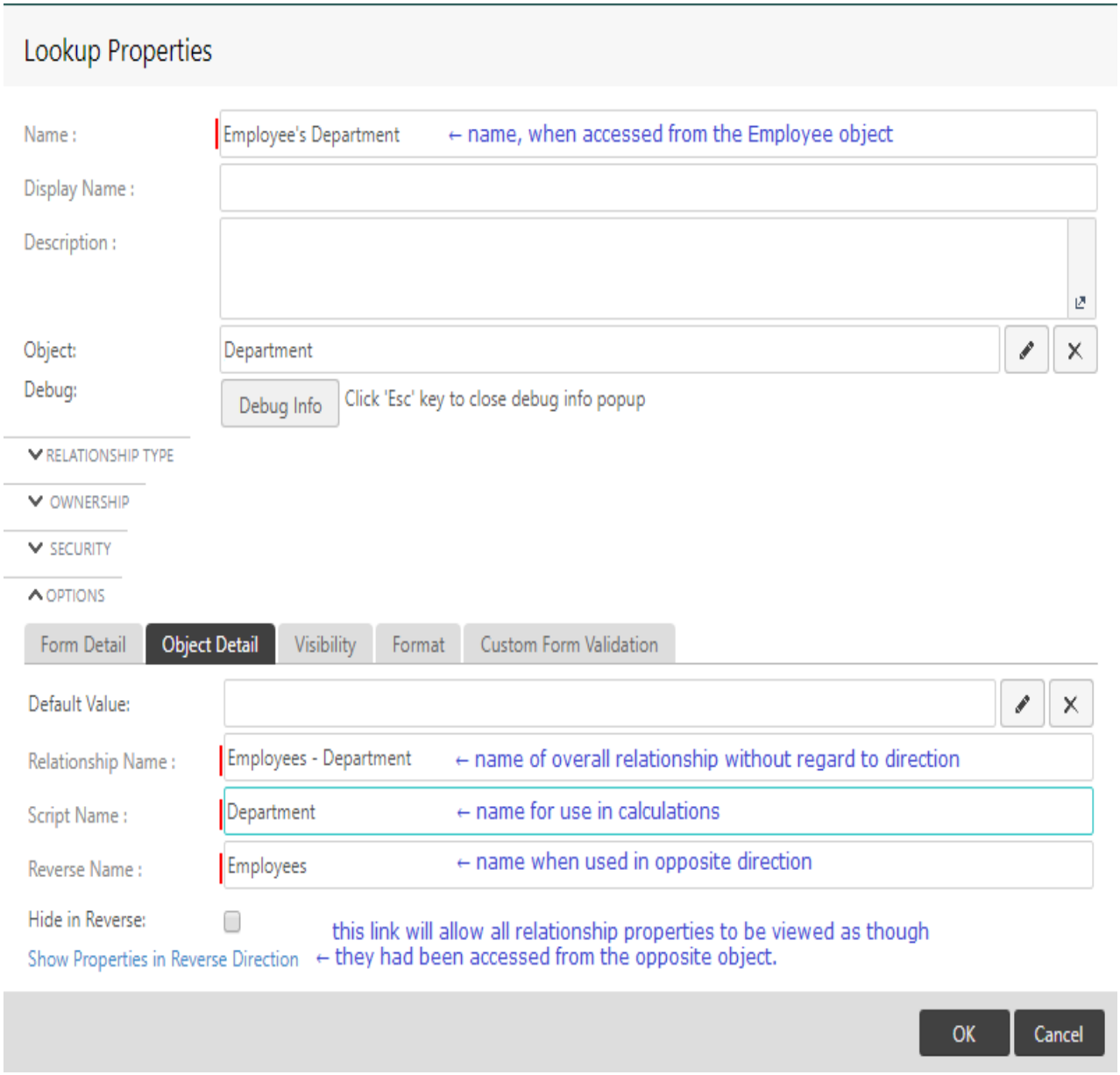
## Relationship Names

Relationship types can be named in both directions. The relationship as a whole can also be named. Typically the name that is displayed is the name that is relevant to the direction that the relationship is being followed.

For example, consider a relationship between departments and their employees.

- When viewed from a Department object, it might be called "Employees"
- When viewed from an Employee, it might be called "Employee's department"
- The overall relationship might be named "Department - Employees"

When a relationship is created, a reverse name and overall name will be automatically selected based on the selected object.




## Relationship can connect an object to itself

A relationship type can be defined so that both ends point to the same object. A relationship between Employee and Employee to represent the employee-manager relationship is a good example of this.

The direction that the relationship is being followed remains significant, and platform features such as the report builder and form builder will present both direction. It is important to always clearly name such relationships in both direction to avoid confusion.

## Relationship can form hierarchies and be followed recursively

If a relationship type connects an object to itself (or to an inherited object), then, depending on its cardinality, the relationship effectively describes a network of records as follows:

- a many-to-one (or one-to-many) relationship describes a *hierarchy* or *tree* of records
- a many-to-many relationship describes a *directed graph* of records

The ReadiNow platform has various application features for working with hierarchies and directed graphs, including:

- ability to define a hierarchy picker control
- ability to follow relationships recursively in calculations and report
- ability to filter to sub-trees in reports
- additional display options in reports
- security access control rules can be defined to take advantage of *hierarchy* and *directed graph* relationships

## Relationships and Object Inheritance

Relationships are aware of object inheritance.

If a relationship type is defined between objects A and B, then that relationship type is also automatically available from any object that (directly or indirect) inherits from A or B.

For example, consider a relationship between an Asset object and a Location object. Any hypothetical object that inherits Asset, such as Computer, or Laptop, will automatically have this relationship. This means that:

- any Asset record, any Computer record, any Laptop record, etc, can be associated with a location
- any form, report, calculations, etc, that is based on any of Asset, Computer, or Laptop can all make use of this relationship

## Relationship Ownership

Each relationship type has an *ownership* setting. This influences how various ReadiNow

features treat the relationship.

Relationship ownership, described below, can be set to one of:

- Full ownership
- Partial ownership
- No ownership

The exact options and phrasing shown in the relationship properties will vary depending on the cardinality of the relationship.

> Caution: Full and partial ownership levels enable cascade deleting of records. Take care to ensure that configurations are correct and understood to avoid unintended record deletion.

## Full Ownership Relationships

A full ownership relationship is also known as a component/subcomponent relationship. It means that the owned *child* record is a part of its *parent* record.

For example, a "Recovery Plan" record may have individual "Recovery steps" record. The steps are logically a subcomponent of the plan.

A full ownership relationship behaves as follows:

- if the parent is record is deleted, the child record will be deleted. This delete will also cascade to any subcomponents of the subcomponents and so on.
- if the parent record is cloned in a workflow, then a full (deep) clone is made of subcomponents.
- if the parent record is exported to XML, or included in an application, then child records are recursively included in full as well.

## Partial Ownership Relationships

A partial ownership relationship is also known as a dependency relationship. It means

that the child record depends on the parent record being present - but the child record isn't logically part of the parent record.

An "Employee has Performance Review" relationship might be an example of a partial ownership relationship. This would allow a workflow, for example, to create clones of Person records, without any linked performance reviews being copied to the new record. However, if the Person record gets deleted, then so does their Performance Review records.

A partial ownership relationship behaves as follows:

- a child record will be cascaded deleted when its parent is delated.
- neither record is processed in full when the other is exported or cloned, however references may be, depending on the cardinality.

### No Ownership Relationships

For a no ownership relationship:

- neither record is deleted when the other record is deleted
- neither record is processed in full when the other is exported or cloned, however references may be, depending on the cardinality.

## Security Relationships

Relationships can be configured such that either:

1. If a user has permission to a record, then they can always see the name (only) of the related record
2. or, if a user has permission to a record, then they automatically have view permission for the related record
3. or, if a user has permission to a record, then they automatically have the same permissions for the related record

Relationship security can offer significant benefits (or disadvantages if configured incorrectly) for administration convenience, and for system performance.

Refer to Security Relationships for details.