Data Modeling

Monday, August 08, 2016 10:21 AM

Object Overview

Data is stored in records of the object.
Standard v. Custom Objects

System Fields:

CreatedDate - the Date and time when the object was created
CreatedById - the ID of the User who created the object.
LastModifiedById - the ID of the User who last modified the object
LastModifiedDate - the date and time when the object was last modified by a User
SystemModStamp - the date and time when the object was last modified by a User OR a Process.

Name Field - A human-readable identifier for a record. Can either be a test string or an auto-number field.

Custom Field:
Auto Number - system-generated read-only sequence number, analogous to the SQL identity type.

- Can be used to provide a unique ID that is independent of the internal object ID (SAP ID).
- Not used in creating object relationships.

Checkbox - for Boolean Data
Date or Date/Time
Number
Email, Phone and URL - format-validated email, phone and URL string representations.
Picklists and Multi-select picklists
Text and Text Areas
Currency
Formula - a read-only field holding data generated from a formula expression
Geolocation - allows you to identify locations by their latitude and longitude and calculate distances between locations

Relationship Fields:
Lookup - Links one object to another object. When created, you have the option to create a related list on the associated object's page layouts.
Master-Detail - used to denote a tight-binding between objects.

- The master object can also contain rollup summary fields - fields that store values aggregated from the child records in the relationship.

Other Features of Objects

Here is a partial list of features available with all objects.

- **Formulas**—Formulas can be used in many places, such as setting up validation rules, creating workflow rule criteria, and even to define a field. A formula field type behaves much like a spreadsheet formula—it reflects some calculation based on other fields and operations on those fields. The formula language is a rich expression language that lets you perform calculations and manipulate strings, dates, numbers and regular expressions.
- **Validation**—Validation rules help improve data quality by preventing users from saving incorrect data. These rules use the same formula syntax as found in formula field types to define a formula which is evaluated each time a record is saved. If the formula for a validation rule evaluates as “True”, the save is aborted and an error message displayed. You can define one or more validation rules that consist of an error condition and corresponding error message. For example, you can create a validation rule to ensure that a number falls within a particular range.
- **Triggers**—Triggers, written in the Apex language, are pieces of code that can fire before or after a record is saved, updated or deleted.
- **Labels**—Every object and record has a label and can include a description, for help, which is automatically included in the user interface.
- **Notes and Attachments**—You can create, view, and edit notes and add attachments for any record in an object that has this functionality enabled. This provides users of the object the ability to easily add arbitrary text notes, and upload associated documents, for each record.
- **Track Field History**—Certain fields on your objects can be configured to track their history. Any time a user modifies any of the field data whose history is set to be tracked, a new entry is added to an automatically created History related list. This History list tracks the date, time, nature of the change, and who made the change.
- **Security**—Database services provide a flexible security model that you can use to control who has access to objects, records and/or fields.

From <https://developer.salesforce.com/trailhead/force_com_dev_beginner/data_modeling/objects_intro>

Create a Many-to-Many Relationship

You can use master-detail relationships to model many-to-many relationships between any two objects. A many-to-many relationship allows each record of one object to be linked to multiple records for another object, and vice versa. For example, suppose your recruiting app has a custom object called Website that stores information about various employment websites. You want to track which open positions are posted to those sites. Use a many-to-many relationship because:

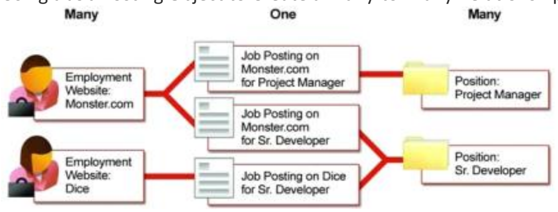
- One position could be posted on many employment websites.
- One employment website could list many positions.

Instead of creating a relationship field on the Position object that directly links to the Website object, we can link them using a junction object. A junction object is a custom object with two master-detail relationships, and is the key to making a many-to-many relationship. To create a many-to-many relationship, you first create the junction object, then create the two master-detail relationships for it.
Let's look at a typical scenario in the recruiting app. There are open positions for a Project Manager and a Sr. Developer. The Project Manager position is only posted on Monster.com, but the Sr. Developer position is harder to fill, so it's posted on both Monster.com and Dice. Every time a position is posted, a job posting record tracks the post.
As you can see in the following diagram, one position can be posted many times, and both positions can be posted to the same employment website. In relational database terms, each job posting record is a row in the Job Posting table. Each row consists of a foreign key to a position record, and a foreign key to a website record.

So, to define a many-to-many relationship between the Position and Employment Website objects, you'd create a Job Posting object with the following fields:

- A Position master-detail relationship
- A Website master-detail relationship

Using a Job Posting Object to Create a Many-to-Many Relationship Between Positions and Employment Websites


Many
One
Many
Employment Website: Monster.com
Employment Website: Dice
Job Posting on Monster.com for Project Manager
Job Posting on Monster.com for Sr. Developer
Job Posting on Dice for Sr. Developer
Position: Project Manager
Position: Sr. Developer

? To handle this scenario, you can create a junction object called Job Posting. A Job Posting record represents a posting about a single position on a single employment website. In essence, the Job Posting object has many-to-one relationships with both the Position and Website objects. Through those many-to-one relationships, the Job Posting object also creates a many-to-many relationship between the Position and Website objects.

Create the Junction Object

1. From Setup, enter Objects in the Quick Find box, then select **Objects**.
2. Click **New Custom Object**.
3. In the custom object wizard, consider these tips specifically for junction objects:
   - Name the object with a label that indicates its purpose, such asJobPosting.
   - For the Record Name field, we recommend using the auto-number data type.
   - Leave the "Launch New Custom Tab Wizard after saving this custom object" box unchecked before clicking **Save**. Junction objects don't need tabs.

Create the Two Master-Detail Relationships

1. On the junction object, create the first master-detail relationship field. In the custom field wizard:
   a. Choose Master-Detail Relationship as the field type.
   b. Select one of the objects to relate to your junction object. For example, select Position.
   The first master-detail relationship you create on your junction object becomes the primary relationship.
   c. Give your master-detail relationship a name.
   d. Select a Sharing Setting option. For master-detail relationship fields, the Sharing Setting attribute determines what access users must have to a master record to create, edit, or delete its associated detail records.
   e. Optionally, add your new field to the page layout.
   f. For the Related List Label that displays on the page layout of the master object, don't accept the default. Change the label to the name of the other master object in your many-to-many relationship. For example, change it to Websites so users see a Websites related list on the Position detail page.

   Note
   We haven't created a custom object called Website as part of this module. To try out this step, create a custom Website object using the steps from the Creating Custom Objects and Fields unit.
2. On the junction object, create the second master-detail relationship. In the custom field wizard:
   a. Choose Master-Detail Relationship as the field type.
   b. Select the other desired master object to relate to your junction object. For example, select Website.
   The second master-detail relationship you create on your junction object becomes the secondary relationship. If you delete the primary master-detail relationship or convert it to a lookup relationship, the secondary master object becomes primary.
   c. Select a Sharing Setting option.
   d. For the Related List Label that displays on the page layout of the master object, do't accept the default. Change the label to use the name of the other master object in your many-to-many relationship. For example, change it to Positions so users see a Positions related list on the Website detail page.

For a many-to-many relationship, each master object record displays a related list of the associated junction object records. For a seamless user experience, change the name of the junction object related list on the master object page layouts to the other master object's name. For example, change the JobPosting related list to Positions on the websites page layout and to Websites on the positions page layout. You can further customize these related lists to display fields from the other master object.

Tell Me More...

You can change the parent record for a child record in a master-detail relationship. To enable this option, select the Allow reparenting option in the master-detail relationship field definition. By default, you cannot "reparent" records in master-detail relationship.

A lookup relationship field is optional but you have the option to make it required. When a lookup field is optional, you can specify one of three actions to take place on dependent lookup fields when someone deletes a referenced lookup record.

- **Clear the value of this field:** This is the default option. Setting a lookup field to null is appropriate when the field does not have to contain a value.
- **Don't allow deletion of the lookup record that's part of a lookup relationship**: This option restricts the deletion of a lookup record that has dependencies, such as a workflow rule built on the relationship.
- **Delete this record also:** This option cascades the deletion of a referenced lookup record to dependent records. It is available only for lookup fields in a custom object; however, the lookup field can reference either a standard or custom object. Choose this option for tightly-coupled record relationships when you want to completely delete related data in one operation

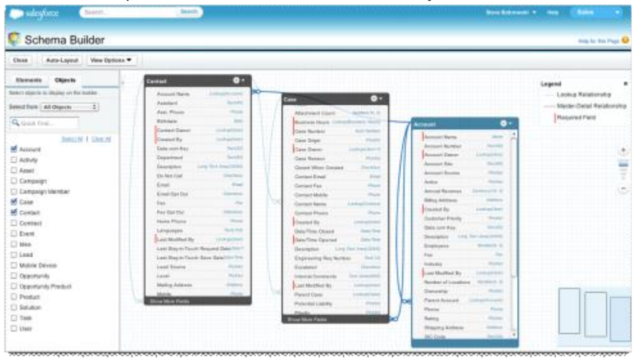
From <https://developer.salesforce.com/trailhead/force_com_dev_beginner/data_modeling/object_relationships>

Schema Builder

Schema Builder provides a dynamic environment for viewing and modifying all the objects and relationships in your app. This greatly simplifies the task of designing, implementing, and modifying your data model, or schema.

You can view your existing schema and interactively add new custom objects, custom fields, and relationships, simply by dragging and dropping. Schema Builder automatically

implements the changes and saves the layout of your schema any time you move an object. This eliminates the need to click from page to page to find the details of a relationship or to add a new custom field to an object in your schema.
Schema Builder provides details like the field values, required fields, and how objects are related by displaying lookup and master-detail relationships. You can view the fields and relationships for both standard and custom objects.



Schema Builder is enabled by default and lets you add the following to your schema:

- Custom objects
- Lookup relationships
- Master-detail relationships
- All custom fields except: Geolocation

Note

You can't export your schema from Schema Builder (for example, to use the schema in another org).

How Do I Access Schema Builder?

From Setup, enter Schema Builder in the Quick Find box, then select **Schema Builder**.
When working with Schema Builder:

- Click an object and move it to any space on the canvas. Schema Builder saves the layout of your schema any time you move an o bject.
- Click **Auto-Layout** to sort the layout of the objects in your schema.

  Important

  When you click Auto-Layout, you can't undo it.
- Click **View Options** to:
  - **Display Element Names** if you prefer system names, or **Display Element Labels** if you prefer text values.
  - **Show/Hide Relationships**
  - **Show/Hide Legend**
- The **Elements** tab lets you drag and drop new custom objects and fields onto the canvas.
- The Objects tab lets you select objects to display on the canvas.
  - Click the drop-down list in the sidebar to filter your list of objects:
    - All Objects
    - Selected Objects
    - Standard Objects
    - Custom Objects
    - System Objects

      Note

      Objects created outside of Schema Builder, such as through an app or the API, don't automatically display on the canvas. Select the checkbox for the object created outside Schema Builder to display it on the canvas.
  - To search for an object, type its name in the **Quick Find** box.
  - Hover over an object in your list of objects and click

    to find it on the canvas.
- Hover over relationship lines to show relationship details such as lookup and master-detail relationships. Click the name of the object to find it on the canvas. You can hide relationships if your schema is taking too long to load.
- To view the details of a field in a new window, right-click the element name or label and select **View Field in New Window**.
- To edit properties of a custom field, right-click the element name or label and select **Edit Field Properties**.
- To manage permissions of a custom field, click the element name or label and select **Manage Field Permissions**.
- Click

  to:
  - **Hide Object on Canvas**
  - **View Object** detail in a new window
  - **View Page Layouts** detail in a new window

- For objects with many fields (Lead or Campaign, for example), click **Show More Fields** to display all the fields.
- To zoom in, click . To zoom out, click .

  Note

  You can't save the level of zoom when closing Schema Builder.
- To collapse the sidebar, click . To expand it, click .
- The map in the lower right corner shows the overall layout of your objects on the canvas. Click the map to navigate the layou t of your objects. To pan across the schema layout while zoomed in, click and hold the canvas while moving the mouse.
- To close the Schema Builder and save the layout of your objects, click **Close**.

Important
If your schema contains many objects and fields, loading can take a long time. Click **Hide Relationships** to improve Schema Builder performance.

Create Objects with Schema Builder

To create a custom object with Schema Builder:

1. Click the **Elements** tab.
2. Click **Object** and drag it onto the canvas.
3. Enter information to define your object. For a list of object definitions, see Schema Builder Custom Object Definition.
4. Click **Save**.

Delete Custom Objects with Schema Builder

You can delete the custom objects that you no longer need by using Schema Builder.
Schema Builder displays a list of side effects when you try to delete a custom object. Be sure you're ready to accept these side effects before finalizing the deletion.

1. Click on the custom object's icon.
2. Select **Delete Object...**. A dialog box displays that explains the side effects of deleting an object. Read this information carefully.
3. If you accept the conditions, check **Yes, I want to delete the custom object.**
4. Click **Delete**.

Create Fields with Schema Builder

To create a custom field with Schema Builder:

1. Click the **Elements** tab.
2. Click a field and drag it onto an object on the canvas.
3. Enter a Field Label.

   Salesforce populates Field Name using the field label. This name can contain only underscores and alphanumeric characters, and must be unique in your org. It must begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores.
4. Enter a Description of the custom field.
5. Enter Help Text to detail the purpose and function of a custom field.
6. Enter a Default Value to automatically insert a value of a custom field when a new record is created.
7. Depending on the custom field type you choose, enter any remaining field attributes.
8. Click **Save**.

Any field you add through Schema Builder isn't automatically added to the page layout. You will need to edit the page layout to specify where the field should be displayed.

Beyond the Basics
By default, the field level security for custom fields is set to visible and editable for internal profiles. Fields that are not normally editable, such as formulas and roll-up summary fields, are visible and read-only. To manage permissions of a custom field, click the element name or label and select **Manage Field Permissions**.

Delete Custom Fields with Schema Builder

Conveniently avoid "custom field clutter" by using Schema Builder to delete custom fields that you no longer need.
Schema Builder displays a list of side effects when you try to delete a custom field. Be sure you're ready to accept these side effects before finalizing the deletion.

1. Right-click on the custom field.
2. Select **Delete Field...**. A dialog box displays that explains the side effects of deleting a custom field. Read this information carefully.
3. If you accept the conditions, check **Yes, I want to delete the custom field.**
4. Click **Delete**.

From <https://trailhead.salesforce.com/data_modeling/schema_builder>