THE EXPERT'S VOICE® IN CLOUD TECHNOLOGY

Practical Salesforce.com Development Without Code

Customizing Salesforce on the Force.com Platform

Philip Weinmeister

Apress®

Contents at a Glance

Introduction

If you are scratching your head in response to the book's title, Development without Code, you will be happily surprised to discover that this is not at all a paradox. While typical development requires the knowledge of a particular programming language, the creation of business applications within Salesforce.com does not. Some very intelligent employees at Salesforce.com have spent thousands of hours working away to build a highly configurable interface that is both user friendly and intuitive. I am neither a part of that group, nor an employee of Salesforce.com, but I can say without equivocation that building solutions within this CRM application for the last few years has been extremely enjoyable and quite fruitful, both for clients and my own organizations. In this book, I will share tips, tricks, lessons learned, and the application of Salesforce.com features and functionality that do not require the utilization of code.

If you are a traditional developer, keep reading! The content of this book will serve as the foundation to essentially unlimited customization of the platform. I will not cover Apex, Visualforce, or any other languages; however, it's important to understand that a more traditional approach to development in Salesforce.com is almost always accompanied by "declarative" elements that are implemented via configuration, or "clicks," instead of code. There are numerous occasions in which a particular configuration setting will be preferable over a solution built from scratch. Why reinvent the wheel when you can simply input your specifications and order one up? While the content directly applies to administrators, consultants, and analysts, there is also significant value for any Salesforce.com developer looking to better understand the platform and produce reliable, extensible applications.

Why Develop Without Code?

Regardless of your role, you'll need to be able to make wise decisions when it comes to delivering functionality for your organization or client. Inevitably, you will encounter scenarios in which a solution can be delivered either with or without the use of code. There are unique pros and cons in every situation, but do consider a few key points that may support the decision to avoid using code:

- The need to consider Salesforce.com limits and parameters is significantly reduced or, at times, eliminated completely when building solutions using declarative means.
- Modifications are often more straightforward, as they may only require a change to a configuration setting, not to a line of code.
- No unit testing is required (Apex test classes, in contrast, must be written for custom Apex code).
- Knowledge transfer burdens are reduced, since an understanding of the particular feature or function is typically sufficient to quickly determine what a specific application is intended to do.
- Future maintenance is simplified. If, for example, an individual who built custom applications for you leaves abruptly, picking up the pieces is much simpler if the work was done declaratively.

Don't get me wrong—there are numerous scenarios that do warrant development *with* code. However, developing with code in Salesforce.com is often done based on need; many organizations want to extend the platform to support functionality that simply does not exist "out of the box." Make sure you have a solid justification for developing with code if your business need can be met through configuration.

Code Sightings

I would like to comment on the "no code" aspect of this book. First, you *will* find a few snippets of code sprinkled throughout the book. For example, with formula functions, a few occasions arise that require manual input. Here are two examples from Chapter 2:

```
BEGINS(City, "Ph")

AND(
     Escalated,
     Subject = "AS400 is no longer functioning"
)
```

In both examples, you can see text strings that have been manually typed (e.g., "Ph" and "AS400 . . ."). While the values were not set via configuration or "clicks," I do not consider this code in the traditional sense. It is expected that some scenarios will emerge that require you to manually input a value.

Second, you will encounter traditional code in Chapter 11. In that chapter, I cover development with the Web-to-Lead tool, for which some HTML-editing capability will come in handy. However, HTML-editing skills are not required to understand the chapter and apply the principles being taught.

In both cases, "code" takes a back seat to declarative development.

Glossary

Let's clarify a few terms that will assist you as you walk through the book. You will also likely encounter these terms often when living in the world of Salesforce.com.

- **Org**: An "org" represents a particular instance of Salesforce.com, whether in production or in a test environment. An org has a unique set of configuration settings and data that formulates the overall user experience.
- **Sandbox**: A *sandbox* is a Salesforce.com-specific test org. There are different kinds of sandboxes, but one thing is common to all of them: they never exist in a production environment. Salesforce.com allows deployments both from and to sandboxes from other sandboxes or even a production environment.
- **Declarative**: This term describes a method for configuration or development that does not require coding or programming but, rather, the utilization of UI-based components to set up an org.
- **Customization**: In this book, *customization* is used to describe the utilization of the Force.com platform to deliver solutions or functionality that are not available to users upon initial org activation. Note that, in the world of Salesforce.com development, some individuals refer to traditional, code-based development as "customization."
- **Out of the box**: This term means different things to different people, so I want to be as clear as possible. For purposes of this book, *out of the box* signifies a feature, tool, or function that is ready for use without significant declarative development and definitely without any custom code. For example, workflow rules are supported out of the box, but a complex, custom-built solution that employs workflow rules would not be considered out of the box. Additionally, anything that requires customization via Apex or Visualforce is not considered out of the box.

Salesforce.com Editions

It is important to understand that a number of different editions of Salesforce.com exist. The edition that you are on depends on the licensing agreement that your organization shares with Salesforce.com. Certain features and tools are only available for particular editions, so make sure you are familiar with your edition and the corresponding functionality. You can find an overview of the different editions of Salesforce.com at `http://www.salesforce.com/crm/editions-pricing.jsp`.

Developer Org/Environment

If you don't already have a Salesforce.com org in which you can follow my examples and build similar solutions, you'll want to obtain one. The good news is that Salesforce.com development environments are free and are provisioned very quickly. Navigate to `https://developer.salesforce.com/signup` and sign up for your own org today. I highly recommend that you follow along and try to create solutions based on what you are learning. Reading is a great thing, but obtaining hands-on experience is what will bring you the most long-term benefit.

If at first you don't succeed, try, try again. There was a starting point for all of us in the development ecosystem, so stay encouraged and motivated. If you are struggling to put a particular takeaway into practice, continue to study and apply what you have learned and you'll find that things will eventually "click." Pun intended . . . now go have some fun!

CHAPTER 1

The Salesforce.com Data Model: Objects, Fields, and Relationships

If you have any familiarity with Salesforce.com, you know that it has never had a shortage of rich features and functionality. One facet of its broad development platform clearly stands out as a prime starting point for developing without code in Salesforce.com and will serve as the foundation for the rest of this book. That element is the *Salesforce.com data model*, which consists of *objects*, *fields*, and *relationships*. The data model serves as a framework for almost everything else you can do within the application. Once you understand objects, fields, and relationships within Salesforce.com, the other areas I will cover in this book will start to come into focus.

Because Salesforce.com has a wide variety of features and functionality, readers will vary greatly in terms of knowledge, development experience, business analysis/comprehension, and technical aptitude related to the program. While some of you may see this chapter as a refresher before digging deeper, others will see it as more of a critical starting point.

Throughout this book, you will find step-by-step instructions to guide you through building out the platform yourself. I highly recommend taking a hands-on approach as you make your way through each chapter since it will allow you to recall the details more vividly and possibly answer some of your own questions along the way.

By the end of this chapter, you will:

- understand standard and custom fields, including the corresponding differences
- be familiar with field types and when to use each of them
- know how to effectively create custom fields step by step
- understand standard objects and their purpose
- have learned about custom objects, including how to create your own custom object
- understand how to effectively build relationships between different objects

Salesforce.com Fields

When you want to get familiar with a new system or application, it can help to have a frame of reference. To give you one, I might suggest that the fields in Salesforce.com are similar to the columns that you would find in a relational database table. Like database columns, Salesforce.com fields contain data of a specific type. But whereas in a database table, a column and row intersect to give you a field that contains the actual data, in Salesforce.com, a field is present on a record and can be populated with data.

However, it is important to point out where the comparison between database columns and Salesforce.com fields falls short. To be truly nontechnical, I might refer to a Salesforce.com field as a database column on steroids. A field in Salesforce.com has a number of related attributes and properties that make it much more robust than a simple database column.

■ **Note** A **database column** may also be referred to as a field. Regardless of how it is labeled, a column/field in a database has some significant differences from a Salesforce.com field.

To effectively meet your company's needs within Salesforce.com, having the right fields is an absolute must. As you will see in subsequent chapters, fields will be your most basic building block throughout the development process. You will not only need to identify what fields will be meaningful and valuable within your org, but also to build them with the appropriate attributes. In this chapter, I will review Salesforce.com fields in detail and provide you with a foundation for success when developing without code on the Force.com platform.

Standard vs. Custom Fields

Once you begin developing solutions within Salesforce.com, you'll quickly become familiar with the terms *standard* and *custom*. Standard items, whether fields or objects, are elements that Salesforce.com produces for you. While you can control some of the attributes of standard items, you cannot directly control the creation of those elements. In a sense, standard elements are existing Force.com platform features. I say "in a sense" because some standard elements can be created as part of a separate process after your Salesforce.com instance is initially established. For example, when you create a custom object, standard fields are created automatically. The standard fields come along for the ride; you have no direct control over their presence.

The set of standard fields that is available to you depends on the object containing them. Standard objects each contain a unique, predefined set of fields. I will cover those later in this chapter when we dive into objects. For now, let's look at the minimal set of standard fields–those that are automatically created when an object is created:

- **Record Name**: This can be a manually entered alphanumeric value (Text field) or an "autonumber" automatically generated by Salesforce.com.
- **Owner**: With some exceptions (see the "Master-Detail Relationship" section later in this chapter), the Owner field will be automatically created. Your Owner field will be populated with a User from your system.
- **Created By**: This is a special system-populated field that captures the User that created the record along with the date and time of the creation.
- **Last Modified By**: Although this field is similar to Created By, it captures the User and Date/Time associated with the last record update instead of record creation.

■ **Note** Last Modified By is actually a concatenation of two system fields: LastModifiedById and LastModifiedDate. However, users just see the concatenated Last Modified By field.

Attributes

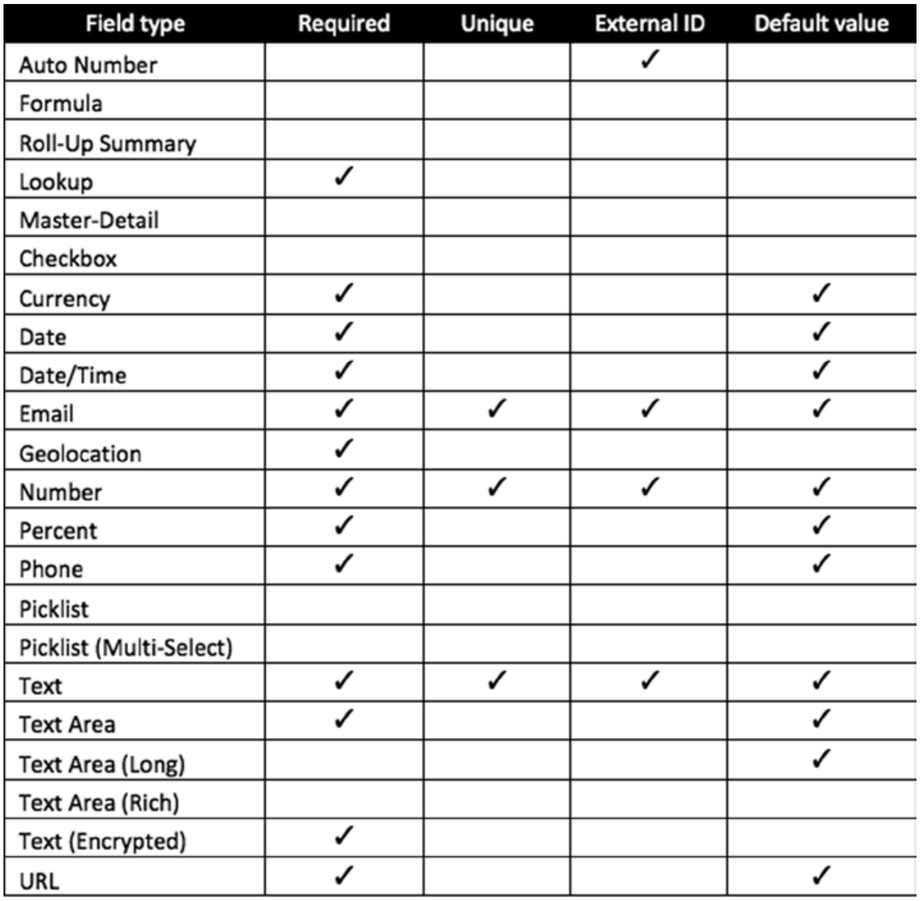
Every field that you create in Salesforce.com has certain attributes that need to be defined. There are obvious examples, such as Name and, in some cases, Length. Other attributes provide depth for fields and allow them to be used properly in a variety of scenarios. You will want to get familiar with these attributes before creating your own fields; they include:

- **Field Label**: The name of your field that is displayed to users.
- **Field Name**: The unique name of your field. Field Name is typically not shown to users, although it is possible to do so. Field Names can only contain alphanumeric characters and (nonconsecutive) underscores, must begin with a letter, and must end with a letter or number.
- **Description**: The Description field is purely for reference, used to explain the purpose and/ or context of the field you are creating. It is highly recommended that you always populate Description even if the reason you are adding a field seems obvious.
- **Help Text**: The bubble text displayed to users upon hovering over a small question mark next to the field. Populating the Help Text field is not required and is most valuable when a user might have trouble understanding how to interact with or interpret a field.
- **Required**: This Checkbox must have a legitimately formatted value present before a record can be created or saved.
- **Unique**: By selecting the Unique Checkbox, you ensure that the field on a new or existing record cannot contain a value that matches that of the same field on another record. Unique can be configured to be case sensitive or case insensitive.
- **External ID**: This Checkbox serves as a record identifier for a field in a system or application outside of Salesforce.com. External IDs have special behavior when corresponding records are imported, either via the standard import wizards or the Apex Data Loader. Note that the External ID by itself does not guarantee that the field values are unique in Salesforce.com—the unique attribute is a separate function.
- **Default Value**: By setting the Default Value on a field, you can set an initial value on every record that is created. This value can be as straightforward as a string (“New”) or a number (“5”). However, it can also be a formula that uses Salesforce.com’s built-in formula functions and can even derive values from other fields/objects.

■ **Note** The Field Name and Field API Name attributes are directly related. The API Name is referenced in code and formulas. You might have a custom field with a Field Name of “Book.” In the API Name, “__c” (for custom) is automatically appended, making the API Name “Book__c” Standard fields do not have anything appended, so the Field Name is the same as the API Name.

All fields require a Field Label and Field Name. Description and Help Text are not required but are available on every field. The remaining attributes (Required, Unique, External ID, Default Value) are available for selection on a limited number of field types. See Table 1-1.

Table 1-1. *A Matrix of Field Attributes Available by Field Type*



Field type	Required	Unique	External ID	Default value
Auto Number			✓	
Formula				
Roll-Up Summary				
Lookup	✓			
Master-Detail				
Checkbox				
Currency	✓			✓
Date	✓			✓
Date/Time	✓			✓
Email	✓	✓	✓	✓
Geolocation	✓			
Number	✓	✓	✓	✓
Percent	✓			✓
Phone	✓			✓
Picklist				
Picklist (Multi-Select)				
Text	✓	✓	✓	✓
Text Area	✓			✓
Text Area (Long)				✓
Text Area (Rich)				
Text (Encrypted)	✓			
URL	✓			✓

Custom Field Types

Every field within Salesforce.com has an associated type. When you create your own custom field in Salesforce.com, you will need to define a field type. A field's type (e.g., Number, Text, etc.) ensures that certain parameters and governing rules are enforced. It drives how users see and interact with the corresponding field. Salesforce.com provides a number of field types to choose from. The types available to you when you create a custom field are a subset of the full set of field types that exist in Salesforce.com; some types are only available via standard objects. Let's take a look at the custom field types and how they are used within the platform.

■ **Note** The Lookup relationship, Master-Detail relationship, and Roll Up Summary field types will be covered in the "Salesforce.com Relationships" section later in this chapter.

Auto Number

Auto Number fields automatically generate a value for each new record based on a simple algorithm that incrementally inserts the numeric portion of the field's value for each new record. At the most basic level, you can create an Auto Number field ("My Custom Auto Number Field") that starts with a value of 1 and increases by 1 for each subsequent record (see Table 1-2).

Table 1-2. *Auto-Numbering Sequencing*

	Record 1 value	Record 2 value	Record 3 value
My Custom Auto Number Field	1	2	3

While you cannot control the size of the increment—Auto Number fields always increase the numeric portion of the value by 1—you can control the display format and the starting number of the field. The display format consists of two parts: (1) a numeric portion that is automatically increased and (2) optional static text that supplements the number. The numeric part is identified in the field setting with curly brackets ("{}"). The starting number setting identifies the first integer to be used.

■ **Note** When you create an Auto Number field, give careful consideration to how you format the static text component to provide as much meaning as possible. For example, "ANN-{0000}" would be a sensible format for an Announcement custom object and "SCH-{0}" would work for a Schedule custom object. If possible, format the Auto Number field so that a user might understand what it represents without needing to do additional research.

Table 1-3 shows some examples of Auto Number display formats along with the corresponding field values that would actually display on a page, based on display format and starting number.

Table 1-3. *Example Values Based on Different Auto Number Formats*

Display format	Starting #	Record #	Field value
{0}	1	1	**1**
{0000}	1	1	**0001**
ABC-{0}	1	1	**ABC-1**
ABC-{0}	1	155	**ABC-155**
ABC-{0}	1000	1	**ABC-1000**
ABC-{0}	1000	155	**ABC-1154**
ABC-{0000}	1	1	**ABC-0001**
ABC-{0000}	1	11315	**ABC-11315**
ABC-{0000}	1000	1	**ABC-1000**
ABC-{0000}	1000	11315	**ABC-12314**

(*continued*)

Table 1-3. (*continued*)

Display format	Starting #	Record #	Field value
Record_{0}	1	1	**Record_1**
Record_{0}	1	87	**Record_87**
Record_{0}	100	1	**Record_100**
Record_{0}	100	87	**Record_186**
{0000}-abc	1	1	**0001-abc**
{0000}-abc	1	52	**0052-abc**

■ **Note** If you notice unexpected gaps in your Auto Number sequences (e.g., "ABC-1" to "ABC-2" to "ABC-7"), you need to know two things: First, know that you have not lost your mind. It's unlikely that a rogue user is off deleting records; this is probably a result of records being created in Apex Test Classes. Second, you can adjust this. Go to **Setup ➤ Develop ➤ Apex Test Execution ➤ Options**, select "Independent Auto Number Sequence," and click "OK."

Formula

The Formula field type allows you to construct a formula based on functions, statements, calculations, operations, user-defined values, and/or other field values that will be evaluated and return a corresponding value. Most commonly, Formula fields will involve derived values that are pulled in from other fields on the same object or from fields on related objects. For example, you may create a formula that evaluates one of the following values:

- a multiple of a currency value (e.g., amount)
- a date based on an existing date field plus a set number of days
- a true/false value derived from an IF statement that evaluates another field on the object
- text inherited from a Parent object (as is)

Formulas will be valuable to you as you build solutions within Salesforce.com. I will dedicate multiple chapters in this book to understanding how formula fields work and how to effectively create them to address your business needs.

■ **Note** Formulas will serve as one of the cornerstones for development without code in Salesforce.com and will be covered in detail in chapters 2 and 3.

Checkbox

A Checkbox field is a Boolean expression, potentially containing one of two values: True or False. Within the standard Salesforce.com User interface, a Checkbox appears with a check for True and without a check for False. You have the option of setting the default value to either checked or unchecked when defining a Checkbox field.

Numeric Fields

Number, Currency, and Percent are three fields that are very similar in behavior, so I am grouping them together here. These field types store numeric values up to a total length of 18 digits. The digits include the total number of digits to the left of the decimal point (referred to as length) and to the right of it (referred to as decimal places). For example, you can create fields from these field types with a length of 16 + 2 decimal places, a length of 10 + 8 decimal places, and so forth. A field with a length of 18 would not be allowed to have any decimal places due to the total digit limit.

It is worth noting that each of these fields stores the corresponding number without any transformation of the value itself. For example, entering 50 in a Number field, a Currency field, and a Percent field would result in a value of 50 being stored in each field in the database (i.e., not "0.5" in the Percent field). Additionally, it's critical to understand that values with precision beyond the configured number of decimal places are rounded accordingly. For example, in a Number field with a length of 2 + 3 decimal places, a submitted value of "14.1448" would be modified to "14.145." The new value is the one actually stored in the field at that point; it is not simply a shortened representation of a longer field.

■ **Note** The currency type (e.g., USD, GBP, EUR, etc.) associated with a particular Currency field is not set at the field level; it is determined by a variety of contextual factors at the record, user, and org levels.

Date Fields

Date and Date/Time fields contain date and date and time values, respectively. There are a couple key points to understand about Date and Date/Time fields: First, the format of the value will differ based on your context. In scenarios where a date or Date/Time field will be populated or edited by your users (e.g., within record detail pages, list views, reports, etc.), the format will be: MM/DD/YYYY (e.g., "9/30/2015"). You may enter the year as "YY" and it will be converted to the full four-digit date, YYYY. In other scenarios (e.g., advanced developers using Apex or system administrators using the Data Loader), the format will be: YYYY-MM-DD. See Table 1-4 and Figure 1-1 for how Date fields appear in different contexts.[1]

Table 1-4. *Display/Presentation of Date Values in Different Contexts*

Date	Salesforce.com context	Displayed value
September 30, 2015	Record detail page	**9/30/2015**
September 30, 2015	List view	**9/30/2015**
September 30, 2015	Report	**9/30/2015**
September 30, 2015	Apex trigger/class	**2015-09-30**
September 30, 2015	Data Loader	**2015-09-30**

[1]All Salesforce.com screenshots in this chapter © copyright Salesforce.com, Inc. Used with permission.

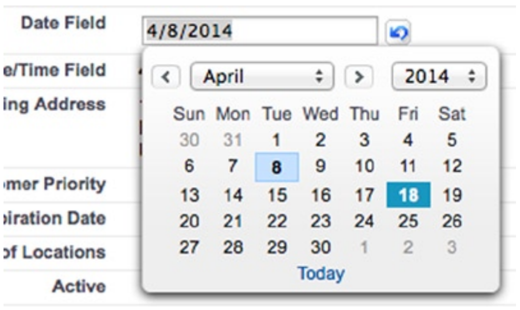


Figure 1-1. *A Date field with the automatic date picker that appears in the UI*

Date and Date/Time fields behave similarly. In the standard user scenarios previously described, Date/Time fields will need to be formatted in the UI as: MM/DD/YYYY HH:MM AM/PM. For a.m./p.m., you would enter either "AM" or "PM." Hours should be entered in the 12-hour format. Like with the Date field, users may enter YY for a Date/Time field and allow Salesforce.com to automatically convert the value into a four-digit year. In the more datacentric scenarios previously described, the format will be: YYYY-MM-DDTHH:MM:SSZ." The T and Z should be entered as is and do not need to be modified; the T stands for time and the Z stands for zone, as in time zone. By default, the value is entered in the user's local time zone. You can offset the time to a specific time zone, as shown in Table 1-5 and Figure 1-2. In that case, the Z is replaced with the time offset (e.g., "-05:00).

Table 1-5. *Date/Time Values (with Local Time and Time Zone Offset)*

Date/Time (UTC)	Time zone	Data Loader/Apex input value	As displayed in UI in the referenced time zone
September 30, 2014 10:30 AM	UTC	**2015-09-30T10:30:00Z**	**9/30/2014 10:30 AM**
September 30, 2014 10:30 AM	EST (UTC–5)	**2015-09-30T10:30:00-05:00**	**9/30/2014 5:30 AM**
September 30, 2014 10:30 AM	PST (UTC–8)	**2015-09-30T10:30:00-08:00**	**9/30/2014 2:30 AM**

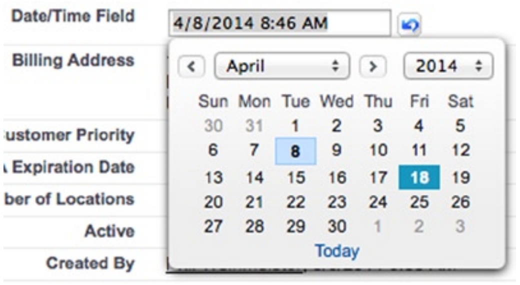


Figure 1-2. *A Date/Time field with the automatic date picker that appears in the UI*

■ **Note** The time displayed to a user depends on the time zone set on his record. So, if a Date/Time field contains a time of 1:00 PM PST, it will display as 4:00 PM for users who have the time zone set to EST.

Email

Email fields are essentially Text fields with specific, built-in formatting validations. Email fields must include:

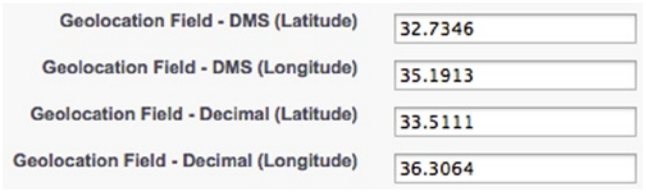
- an @ symbol
- text before the @
- a period somewhere after the @
- text immediately before and after the period
- no symbols other than underscores, hyphens, periods, characters, and @ symbols

Of course, Salesforce.com cannot fully validate an e-mail based on these conditions. Try it yourself. You can enter "1-_@1.1" in the Email field and the value will be accepted. If you do want to expand the validations on this field, you can use the REGEX function within a validation rule. Both the REGEX function and validation rules will be covered in subsequent chapters.

Email fields have a built-in mailto: link that allows you to send an e-mail to the specified address using your default e-mail application upon clicking on the address.

Geolocation

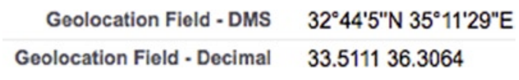
The Geolocation field type is relatively new to Salesforce.com. Geolocation actually contains two subfields: Longitude and Latitude. You can control how the value is displayed: in degrees, minutes, or seconds (DMS) or as a number with decimal places. Figure 1-3 contains four subfields related to the two Geolocation subfields. The first two fields represent the Longitude and Latitude for a DMS-formatted field; the second two fields represent a Decimal-formatted Geolocation field.



Geolocation Field - DMS (Latitude)	32.7346
Geolocation Field - DMS (Longitude)	35.1913
Geolocation Field - Decimal (Latitude)	33.5111
Geolocation Field - Decimal (Longitude)	36.3064

Figure 1-3. *How the two geodecimal fields (DMS and Decimal) appear when edited*

After saving your edit, Figure 1-4 shows what you will see in each field on the record detail page.



Geolocation Field - DMS	32°44'5"N 35°11'29"E
Geolocation Field - Decimal	33.5111 36.3064

Figure 1-4. *The two Geodecimal fields after saving*

You can set the number of decimals that are allowed in a Geolocation field. Like with other numeric fields, Salesforce.com will round a value that has more decimal places than it allows. For example, in a field with two allowed decimals places, a latitude of 35.505 would save as 35.51 and a latitude of 35.504 would save as 35.5 (note that the lagging zero is removed).

Phone

The Phone field is similar to the Email field in that it has a very specific context and use: storage of a phone number. In terms of what data is allowed in the field, it is really a simple Text field. However, Phone fields can be used for manual or automatic dialing of phone numbers within certain applications.

■ **Note** The text allowed in a Phone field goes well beyond the scope of characters/digits that would appear in real-world scenarios. I would recommend building out some validation rules to prohibit invalid entry of phone number values to maintain data integrity. Validation rules are covered in detail in Chapter 5.

Picklist

The Picklist fields, **Picklist** and **Picklist (Multi-Select)**, are used frequently within Salesforce.com and can be extremely valuable when building business processes into your application. Picklists behave most similarly to drop-down fields, although that term is completely abandoned on the Force.com platform. With single Picklists, you can define a list of one or more values that can be selected, or picked, by users through the standard UI. Figure 1-5 shows a Picklist field called Stock Exchange. This field represents the exchange on which an Account's stock trades.

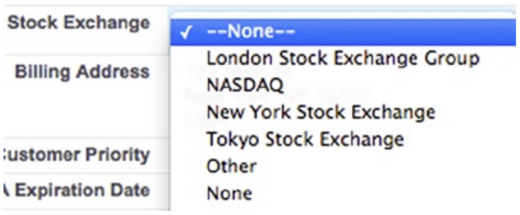


Figure 1-5. *Values from a Picklist field*

In addition to identifying the values that will be available, you will also define the sequence in which the values will appear and, optionally, the default (prepopulated) value for users.

■ **Note** Including values such as "None" and "Other" has implications in your business process that need to be considered. Including "Other" may be necessary to prevent user frustration with a finite set of values, but it could have an adverse effect on reporting granularity. "None" is most useful when you want to ensure that users consider and select a value for a Picklist field even if no value applies. Alternatively, you can simply let them avoid making a selection, but there's the risk that the user may skip the field out of hastiness.

Multi-Select Picklists allow for the selection of multiple values within a particular field. You can select zero, one, or multiple values from the defined list. Again, you can configure the order of the list. Also, you can define how tall the window is in rows. Figure 1-6 is an example of a Picklist (Multi-Select) on an Account record.

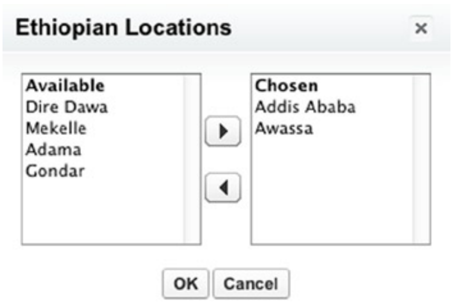


Figure 1-6. *A Multi-Select Picklist field during an edit*

■ **Note** Perhaps the primary factor you will need to consider when creating a Multi-Select Picklist field is the implication for reporting. With a standard (single) Picklist field, you can easily obtain the count of records by Picklist value. That is not the case with a Multi-Select Picklist field.

Text Fields

You can create a variety of text fields in your data model. Table 1-6 is a summary of each.

Table 1-6. *A Matrix of Text Field Types and Their Related Attributes*

Text field type	Max length	# display lines	Formatted text	Images	Links
Text	255	1	No	No	No
Text (Encrypted)	175	1	No	No	No
Text Area	N/A	1–3*	No	No	No
Text Area (Long)	32,768	2–50	No	No	No
Text Area (Rich)	32,768	2–50	Yes	Yes	Yes

**You do not have control over the number of lines displayed for a Text Area field; this is driven by the context (page) and the length of the value in the field.*

Encrypted fields allow characters to be masked and replaced with “x” or “*.” The format of the masking can be configured as well. Standard users will not be able to see encrypted fields without the “View encrypted data” permission.

URL

URL fields are text based and have a specific use and context. These fields store properly formatted hyperlinks and link to the specified URL upon the user clicking the hyperlink. Like Phone fields, URL fields have no restrictions on the amount of actual characters that can be entered. Salesforce.com will take the text input and identify whether a prefix (http://) is required. If the submitted value does *not* start with http:// or https://, Salesforce.com will add the prefix “http://” to the text value in the field. I would recommend some basic validations of this field to ensure that no invalid characters are accidentally included in the URL value.

Additional Field Types

Address and Name field types are not currently available when creating custom field types but are present on some standard objects. Each field type is actually a set of related fields and is referred to as a compound field. The Address and Name fields cannot be directly updated, but the individual field components within them can. For example, you cannot update Billing Address on the Account object, but you can update Billing Street, Billing City, Billing State, and so on. The Address field type is comprised of the following fields: **Street**, **City**, **State/Province**, **Zip/Postal Code**, and **Country**. The Name field type contains the following fields: **Salutation**, **First Name**, and **Last Name**.

Creating a Custom Field Step by Step

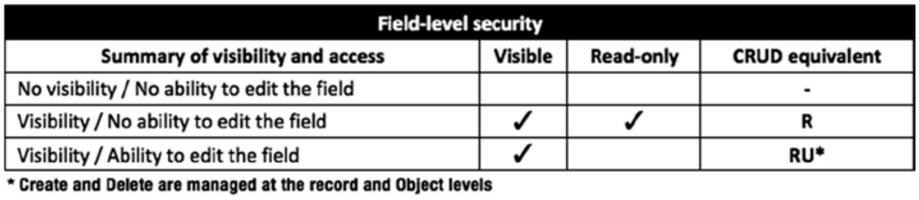
Each field type has a unique creation process. I previously described some of the elements that need to be considered when creating a corresponding field according to the field type. What follows is an overview of the creation process along with the key considerations. Keep in mind that the best way to get familiar with the process is to actually create a variety of fields with different field types.

■ **Note** Relationship fields (Lookup, Master-Detail) have a more detailed creation process than the four-step process described in this section.

---

1. *Select a field type*. Carefully choose your field type from the provided list. Make sure to think about the exact use case and select the field type that makes sense for your business needs.
2. *Configure the field attributes*. Once you select your field type, you will have to set the attributes for the field. Each of these is covered in detail earlier in this chapter.
3. *Set up field-level security*. Establish the permissions to your new field that should be granted to existing profiles. Once you have set the field type and configured the related attributes for your field, you will need to set the visibility of and access to the field across the existing profiles. You have three options that can be applied. Table 1-7 is a matrix showing the Visible and Read-Only columns that you edit along with the CRUD (create, read, update, delete) equivalent of the corresponding combination.

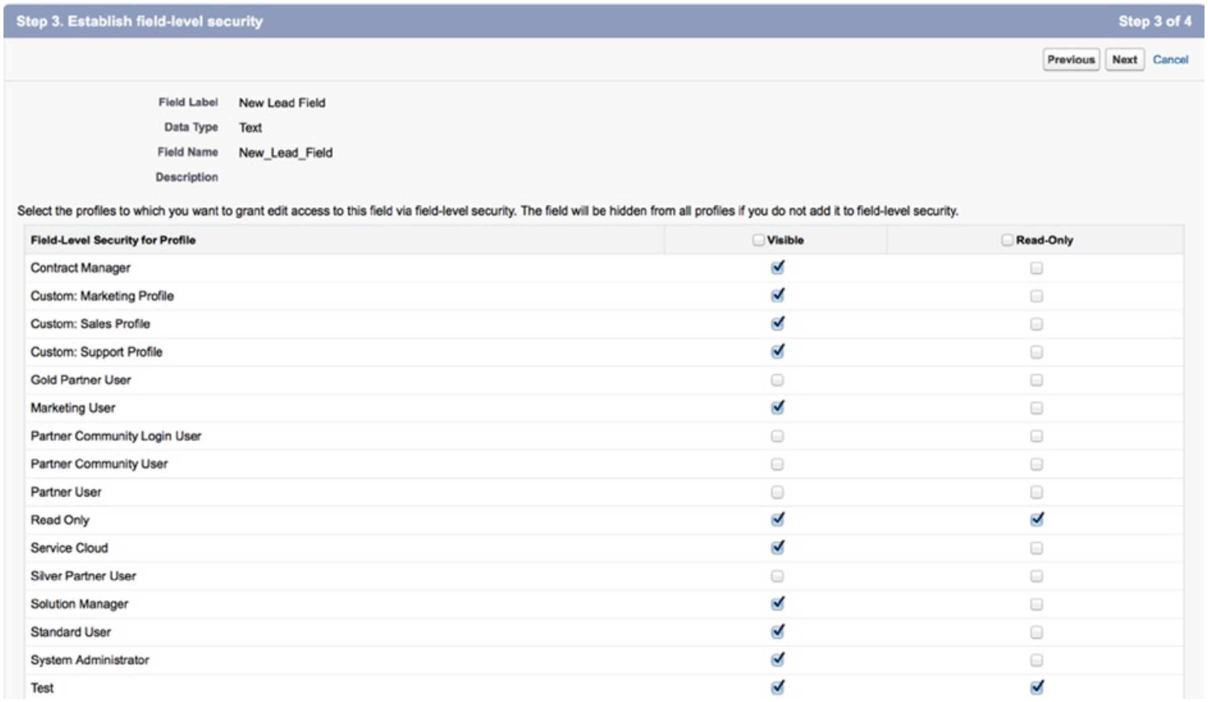
Table 1-7. *Potential Field-Level Security Settings*



Field-level security			
Summary of visibility and access	**Visible**	**Read-only**	**CRUD equivalent**
No visibility / No ability to edit the field			-
Visibility / No ability to edit the field	✓	✓	R
Visibility / Ability to edit the field	✓		RU*

* Create and Delete are managed at the record and Object levels

Keep in mind that field-level security (FLS) may or may not have a direct impact on a user's experience. If a User does not have access to a particular record at all, full access via FLS does nothing to change that—the user still will not be able to see any of the fields on the record. Or she may be granted Read/Write access via FLS but a validation rule (these will be covered later in this book) may override that access and prevent her from editing the record. FLS, whether associated with a profile or a permission set, conveys the maximum access to a field that a user might have; the user might ultimately have less access based on other settings, but she will not have more access to the field than configured via FLS. Figure 1-7 is a look at the FLS settings page during custom field creation.



Step 3. Establish field-level security Step 3 of 4

Previous Next Cancel

Field Label	New Lead Field
Data Type	Text
Field Name	New_Lead_Field
Description	

Select the profiles to which you want to grant edit access to this field via field-level security. The field will be hidden from all profiles if you do not add it to field-level security.

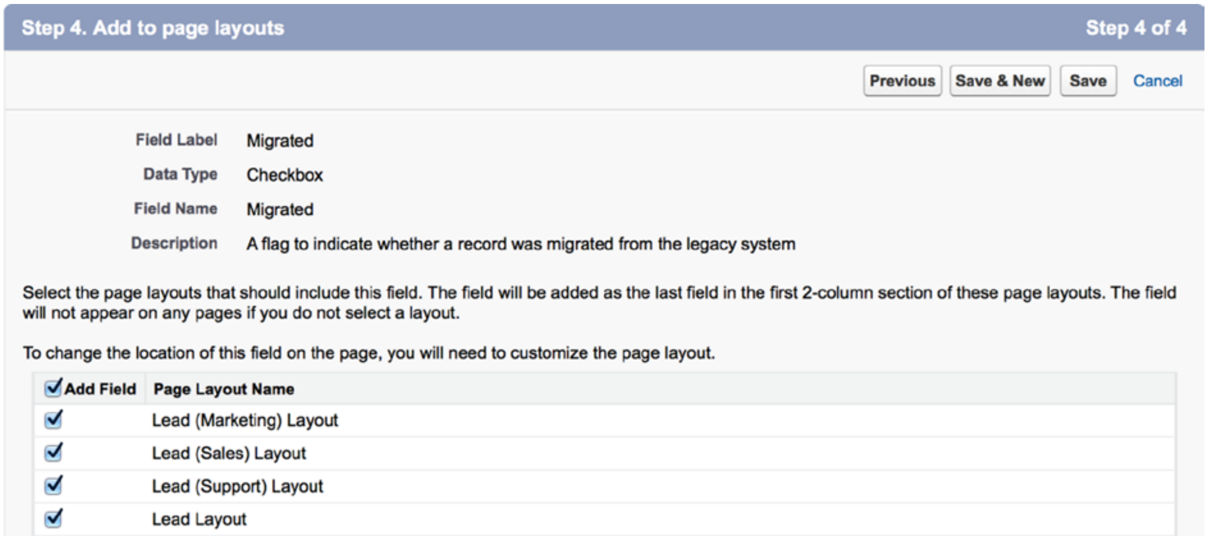
Field-Level Security for Profile	Visible	Read-Only
Contract Manager	✓	
Custom: Marketing Profile	✓	
Custom: Sales Profile	✓	
Custom: Support Profile	✓	
Gold Partner User		
Marketing User	✓	
Partner Community Login User		
Partner Community User		
Partner User		
Read Only	✓	✓
Service Cloud	✓	
Silver Partner User		
Solution Manager	✓	
Standard User	✓	
System Administrator	✓	
Test	✓	✓

Figure 1-7. *Setting the field-level security for your field*

■ **Note** Security considerations, including object and field permissions, are discussed in detail in Chapter 14.

---

4. *Add to page layouts*. The last step in the field creation process is only marginally helpful, and I'll explain why. You are given the option to automatically add the field you are creating to any of the page layouts for the corresponding object. If you are creating a Lead field and four page layouts exist for Leads, you can add the Lead field to any of those four layouts at your discretion, as shown in Figure 1-8.



Field Label	Migrated
Data Type	Checkbox
Field Name	Migrated
Description	A flag to indicate whether a record was migrated from the legacy system

Select the page layouts that should include this field. The field will be added as the last field in the first 2-column section of these page layouts. The field will not appear on any pages if you do not select a layout.

To change the location of this field on the page, you will need to customize the page layout.

✓ Add Field	Page Layout Name
✓	Lead (Marketing) Layout
✓	Lead (Sales) Layout
✓	Lead (Support) Layout
✓	Lead Layout

Figure 1-8. *Setting your page layout options*

This sounds like a nice time-saver, doesn't it? The problem is that you have no control over the field's location within the page layout, just whether it is present or not. Fields added to page layouts through this means are always added as the bottom, left-most field in the first section on the page layout, as shown in Figure 1-9.

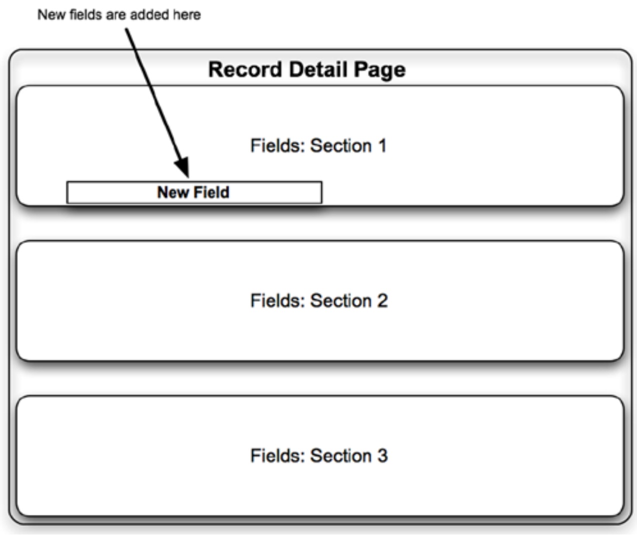


Figure 1-9. *Appearance of a new field when added during the creation process*

Except in the rare situation where you specifically want your field to appear in the bottom-left corner of the top-most field section on your page layout, you will need to go into your layout and edit it to move the field to the desired location. You may want to hold off on automatically adding a field to a page layout and just do it manually for two reasons:

- If you forget to update the corresponding page layout after creating it, your field will be in the wrong location.
- You have to update the page layout anyway and you will have additional control if you handle adding the field and moving it all at once.

That's it! You do want to spend some time thinking about your options before just breezing through the creation. You can always change your field settings later, but it's definitely easier to set up your field properly up front. Now, you're ready to move on to the world of objects.

Salesforce.com Objects

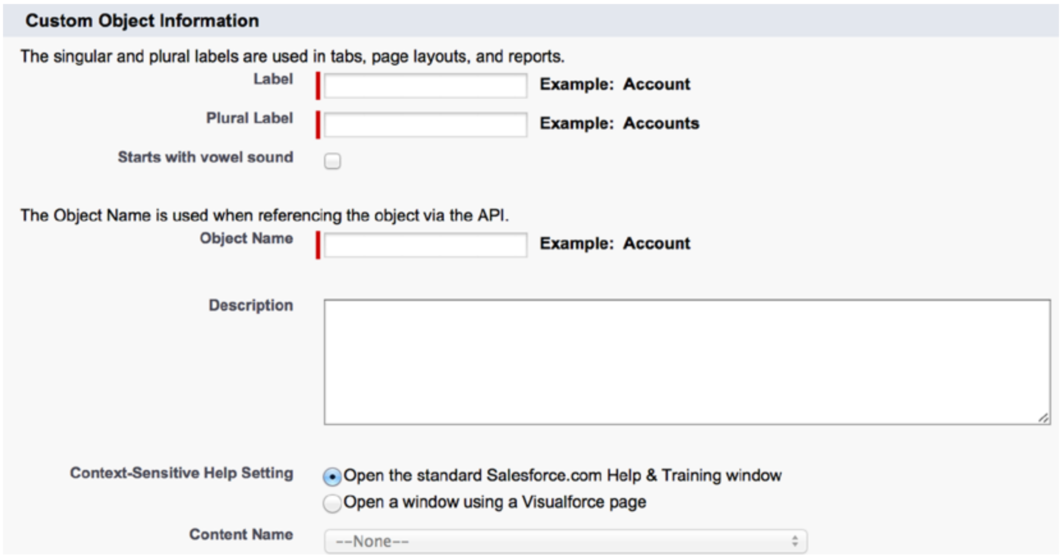
As you have seen in this chapter thus far, it is impossible to cover the breadth of Salesforce.com fields without at least touching on the objects in which those fields are contained. In this section, I will dive into the basics of objects and further prepare you to develop real-world business solutions on the Force.com platform.

Previously, I stated that fields in Salesforce.com are comparable to database columns. Along those same lines, we can compare Salesforce.com objects to database tables (again on steroids). Each object contains a set number of standard fields along with any custom fields you create. At the intersection of those fields and the records will be your field values, similar to database fields. Table 1-8 includes a couple of basic charts that have a similar structure to each other.

Table 1-8. *Comparison Between a Salesforce.com Object and a Relational Database Table*

Salesforce.com object			**Relational database table**		
	Field 1 (text)	**Field 2 (number)**		**Column 1 (text)**	**Column 2 (number)**
Record 1	My first record	100	Row 1	My first record	100
Record 2	My second record	200	Row 2	My second record	200
Record 3	My third record	300	Row 3	My third record	300

However, there is another layer entirely to objects. Like fields, objects have associated attributes and metadata that build their context and drive their use. I will now walk you through the object creation process, during which you will see the additional information that can be associated with a Salesforce.com object. Figure 1-10 is the initial screen you see when creating a custom object.



Custom Object Information

The singular and plural labels are used in tabs, page layouts, and reports.

Label		Example: Account
Plural Label		Example: Accounts
Starts with vowel sound	☐	

The Object Name is used when referencing the object via the API.

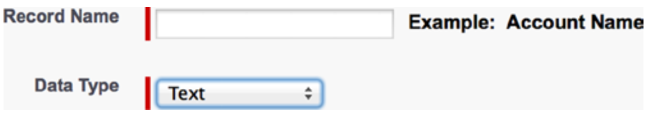
Object Name		Example: Account
Description		
Context-Sensitive Help Setting	● Open the standard Salesforce.com Help & Training window ○ Open a window using a Visualforce page	
Content Name	--None--	

Figure 1-10. *Typical custom field attributes that can be set during field creation*

The first step in creating a custom object is to define its basic, key information. You'll set your label and plural label first. Salesforce.com doesn't want to assume what your grammatical application will be. What if you wanted to make an object called "Fish"? In that case, you might want the plural label to match label. The "Starts with vowel sound" is present for a similar reason: Salesforce.com does not inherently know that the starting letter in hour and hot sound different, but it does want to apply correct grammar rules. Setting that yourself helps to achieve that goal.

Like with Salesforce.com fields, you will need to provide an object name and, optionally, a description. I would again recommend using the object name that is automatically populated, unless it is critical to your business process that you set a specific name for a business purpose. And as tempting as it is to skip entering a description, the ten seconds needed to enter one could pay off in spades. I can say from personal experience that a populated description has helped save a significant chunk of time more than once.

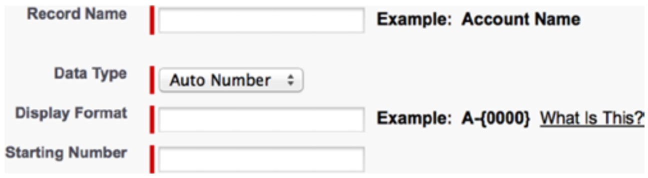
The next step is configuring the Record Name label and format for your object. Figure 1-11 shows the Record Name field, which will be the primary name field for your new records.



Record Name		Example: Account Name
Data Type	Text	

Figure 1-11. *Record Name as a text field*

As shown in Figure 1-12, you can either allow open text entry or you can automate it (without code) by using the Auto Number option. I recommend using the Auto Number option when possible, as it provides you with a built-in ID that is automatically increased, in increments, with the creation of new records. It also serves as a nice reference when looking at a list of records.



Record Name		Example: Account Name
Data Type	Auto Number	
Display Format		Example: A-{0000} What Is This?
Starting Number		

Figure 1-12. Record Name as an Auto Number field

The last setting in this section determines the source for your content. You can either use the generic custom object page that Salesforce provides or your own custom page. If you don't have a specific help page ready for use, proceed with the Salesforce.com help page.

There are some additional attributes that you'll need to consider when creating your custom fields. Here's a summary of the remaining items:

- **Allow Reports**: Creates a custom report type for the object with which reports can be created for the new object.
- **Allow Activities**: Enables activities (tasks, events) to be associated with the record. Recommended only if activities make sense for the object. If the object does not require activities to be tracked against it, leave it unchecked. You can also enable it later.
- **Track Field History**: Allows changes made to the fields on the object to be tracked.
- **Allow Sharing**, **Allow Bulk API Access**, and **Allow Streaming API Access**: These three settings must either all be enabled or all be disabled. By default, they are enabled, making a custom object an "Enterprise Application" object. Disabling these features will convert the object to a "Light Application" object. The settings themselves allow for more advanced, robust interaction with the object.
- **Deployment Status**: Allows you to select "Deployed" if the object is ready for release to all users.
- **Add Notes and Attachments (related list to default page layout)**: Like with activities, you'll need to assess whether the notes and attachments are relevant. If so, select this option. Otherwise, leave off the clutter.
- **Launch New Custom Tab Wizard (after saving the custom object)**: Will users need to navigate to the records? Then definitely select this option. There's no need to select this option if it has more of a back-end or reporting function.

■ **Note** The Object attributes that are configured at the time of object creation can easily be modified later by navigating back to the object in the Setup menu. For example, if your reporting needs change after object deployment, you can check the Allow Reports Checkbox when needed.

Salesforce.com Relationships

While relationships in Salesforce.com technically fall under the umbrella of field types, they warrant a closer look outside the scope of other more straightforward field types. Relationship fields are the last piece of the puzzle and allow you to connect everything in your data model together.

Relationship Field Types

To fully understand how relationships within Salesforce.com work, you must first have a firm grasp on the types of relationship fields that can be created.

Lookup Relationship

Along with the Master-Detail relationship field type, which will be discussed in the next section, the Lookup relationship is vastly different from all other field types and has significant implications for your Salesforce.com org. A Lookup relationship field actually modifies the interobject schema by creating Parent/Child links between objects. To clearly see the picture, first take a look at a two disparate, unrelated objects. As illustrated in Figure 1-13, the Contact and Idea objects have no native relationship with each other.

Contact
Idea

Figure 1-13. *Two disparate objects*

Unlike the Contact and Idea objects, the Contact and Account objects are in fact related. The Contact object has an “Account Name” Lookup relationship that creates a link between the two objects. In Figure 1-14, the Account object is shown above the Contact object since it is the Parent object and the Contact object is the Child.

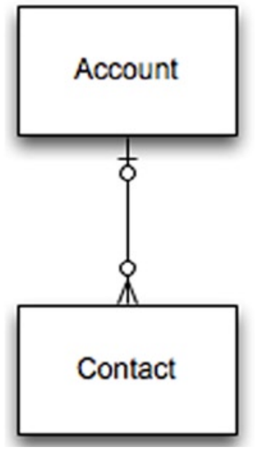
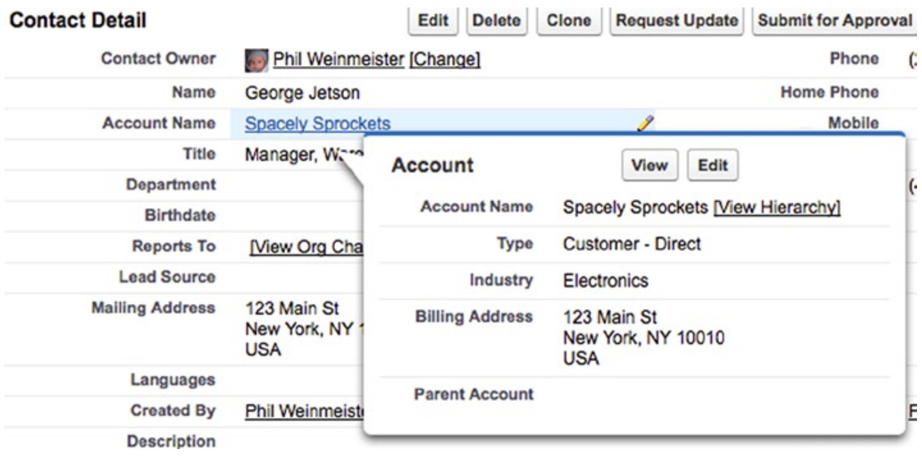


Figure 1-14. *Related Account and Contact objects, connected via a Lookup field*

Figure 1-14 shows the Crow’s Foot entity-relationship notation for the relationship between the Account and Contact. A Contact can look up zero or one Account. An Account can go from zero-to-many Contacts and is not required to have any related Contacts, but can be associated with however many you can add to it within the applicable limits of your org. An important consideration when creating a Lookup relationship field is that it is a loose relationship; deletion of the Parent record that is referenced in the Lookup field does not result in the Child record’s deletion. For example, take a look at Figure 1-15. In this image, you can see that the Contact record for George

Jetson has a Lookup relationship to the Spacely Sprockets Account. If you delete the record of the Parent, Spacely Sprockets, George Jetson's Contact record will not be deleted automatically; the Account field will simply no longer be populated.



Contact Owner: Phil Weinmeister [Change]
Name: George Jetson
Account Name: Spacely Sprockets
Title: Manager, W
Department
Birthdate
Reports To: [View Org Cha
Lead Source
Mailing Address: 123 Main St
New York, NY
USA
Languages
Created By: Phil Weinmeist
Description

Account Name: Spacely Sprockets [View Hierarchy]
Type: Customer - Direct
Industry: Electronics
Billing Address: 123 Main St
New York, NY 10010
USA
Parent Account

Figure 1-15. *A relationship field on the record detail page*

Salesforce.com does allow you to restrict deletion of a Parent record that is part of a Lookup relationship. By enabling that restriction setting, you require users to first remove or change the value in the Child record's Lookup field before deleting the Parent. In this case, you would edit George Jetson's Contact record, modify the Account Name Lookup, save the record, and then delete the Spacely Sprockets record.

Additionally, you can add a Lookup Filter to a Lookup relationship. A Lookup Filter will restrict the available records in a Lookup field based on defined criteria.

Master-Detail Relationship

The Master-Detail relationship field type is very similar to a Lookup relationship. By creating a Master-Detail field, you build a relationship between a Child record and a Parent record. However, there are some critical differences between these two relationship field types:

- A Master-Detail relationship field must be populated. In other words, you cannot have an "orphaned" Child record that has a blank Master-Detail field. This is not the case with a Lookup field unless you specify that a Parent is required in the Lookup.
- A Child in a Master-Detail relationship does not have a standard Owner field; the owner of the Parent record, via inheritance, determines the owner for this relationship.
- Cascading deletion occurs when Parent/Master records are removed. In other words, all Child/Detail records will be deleted if the Parent record linked via a Master-Detail field is deleted.
- Master-Detail fields allow you to create Roll-Up Summary fields. They will be discussed in the next section.

Although a Master-Detail field must be populated, it can be "reparented" if configured accordingly. That means that you can change the associated Parent record after the Child record is created.

Roll Up Summary

The Roll Up Summary field type does not affect object relationships, but it builds directly upon the Master-Detail relationship. A Roll Up Summary field is a great feature, which Salesforce.com makes available for creation any time you establish a Master-Detail field. The presence of an associated Master-Detail field is a requirement, since the Summary part of the field is directly related to the associated Child records. In Figure 1-16, you can see two different perspectives of how this works. On the left, you see that the Master-Detail field is created on the Detail, or Child, object. However, the Roll Up Summary is set up on the Master, or Parent, object. On the right, you can see examples of actual records that might be present in this Master-Detail relationship. The presence of the Roll Up Summary on the Master record makes sense since the summary will include a downward look across all of the children.

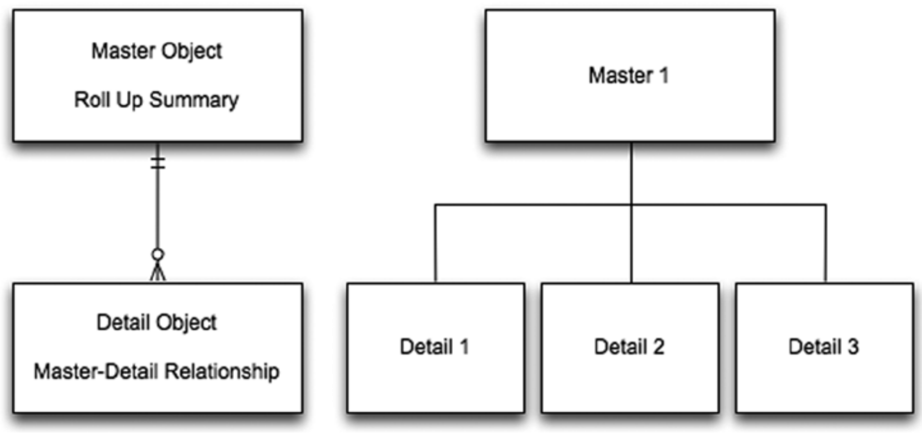


Figure 1-16. *A view of a Master-Detail relationship*

When you create a Roll Up Summary, you are given the following options:

- **Object to summarize**: Note that the objects listed will be limited to those with Master-Detail relationships with the object you're working with.
- **Roll Up type**: (COUNT, SUM, MIN, MAX) COUNT returns the number of all Child records on the selected object. SUM returns the sum of a specified Number or Currency field across Child records. MIN and MAX return the least/first or greatest/last value, respectively, from a specific Number, Currency, or Date field among Child records.
- **Filter criteria**: Allows you to specify criteria based on object fields to limit which Child records are included in the Roll Up.

It might help to look at a few specific examples. Figure 1-17 represents the Master-Detail relationship between the Opportunity and Account objects, and Table 1-9 shows sample records at the detail level.

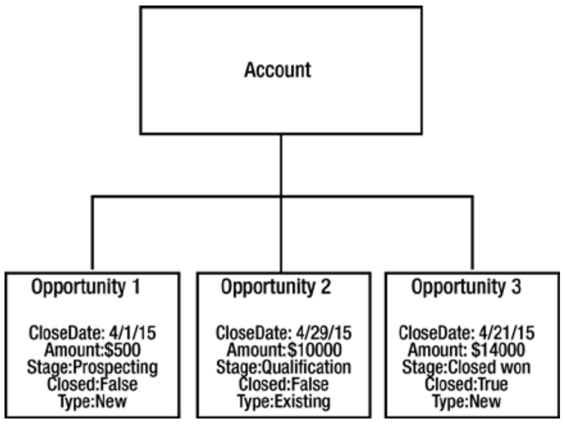


Figure 1-17. *Three Opportunities and their related Account*

Table 1-9. *Examples of Roll-Up Summary Fields and Their Values Based on the Records in Figure 1-17 Salesforce.com data model:roll up summary*

Roll-Up type	Field	Criteria	Field value
COUNT	N/A	None (All Records)	3
COUNT	N/A	Closed = False	2
COUNT	N/A	Type = Existing	1
SUM	Amount	None (All Records)	$24,500
SUM	Amount	Closed = False	$10,500
SUM	Amount	Type = Existing	$10,000
MIN	CloseDate	None (All Records)	04/01/15
MIN	CloseDate	Closed = False	04/01/15
MIN	CloseDate	Type = Existing	04/29/15
MAX	Amount	None (All Records)	$14,000
MAX	Amount	Closed = False	$10,000
MAX	Amount	Type = Existing	$10,000

Junction Objects

You just read about Lookup and Master-Detail relationships. These field types build the respective foundation for zero-to-many and one-to-many associations between Parent and Child records. What about many-to-many relationships? Can you create those in Salesforce.com, and if so, how? The answer is yes; you *can* create many-to-many relationships in Salesforce.com. I will walk you through a real-world example that requires a many-to-many relationship and explain how to appropriately configure your data model.

Let's say your firm regularly holds sweepstakes contests for marketing and branding purposes and you want to capture the specific individuals who have entered each contest. Those individuals come from your pool of existing Contact records. In this case, the standard Campaign and Campaign Member objects would likely meet your needs, but we will use custom objects for the purposes of this discussion.

You could create a Contest object and relate Contacts to it via a Lookup relationship, as shown in Figure 1-18.

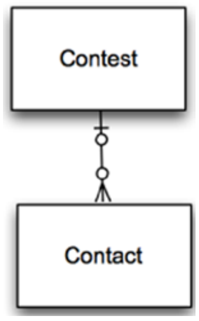


Figure 1-18. *Contact record related to a custom object "Contest"*

However, via a direct relationship to Contest, *a Contact can only be associated with one Contest*. That quickly becomes a problem as your Contacts start to attend multiple Contests. This is where a "Junction object" comes into play. Instead of directly linking your Contact to a Contest, you can create a new custom object called a Contest Registrant and add two relationship fields to it, as shown in Figure 1-19.

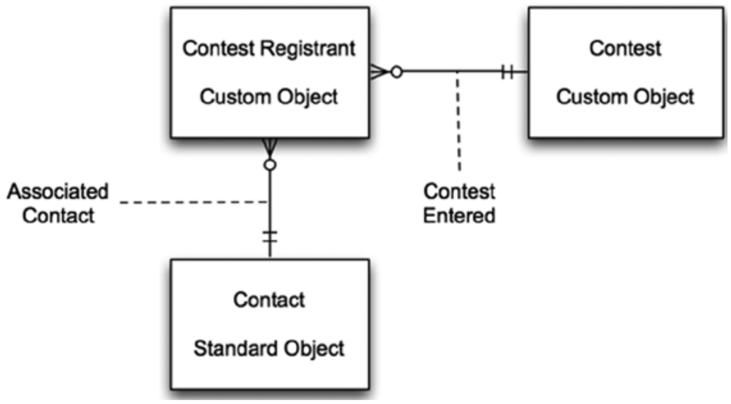


Figure 1-19. *Contest Registrant custom object serving as a Junction object*

Here is a breakdown of the various elements shown in Figure 1-19:

- **Contest Entered:** This Master-Detail field identifies a contest that the registrant is registered for. It is a Master-Detail field because a Contest Registrant cannot exist apart from a Contest.
- **Associated Contact**: This field identifies the person that registered for a particular Contest. Because you don't need a duplicate record of a person, you link to the "source" record, which is the Contact, and add any custom fields to the Contest Registrant record to provide additional context about his role as a registrant. This is a Master-Detail field because a Contest Registrant cannot exist apart from a Contact, at least in this case.

As a result, both Contacts and Contests can be associated with many Contest Registrants. That means that:

- One Contact can be associated with many Contests.
- One Contest can be associated with many Contacts.

If one Contact is registered for five Contests, you would have one Contact, five Contest Registrants, and five Contest records. If one Contest has five registrants, you would have one Contest, five Contest Registrants, and five Contacts.

Some Perspective on the Data Model

The Salesforce.com data model consists of three key parts: objects, fields, and relationships. Since Master-Detail and Lookup relationships are fields, you could make the argument that it is really just two parts, since relationships are field types. However, since the relationship field type is so fundamentally different than other field types, I find it more useful to look at it as a completely separate element.

Figure 1-20 is intended to help you see the Salesforce.com data model from a few steps back. This visual gives a basic perspective of the objects, fields, and relationships that exist within your instance of Salesforce.com. You can use Salesforce.com's "Schema Builder" tool for a detailed view of the same content.

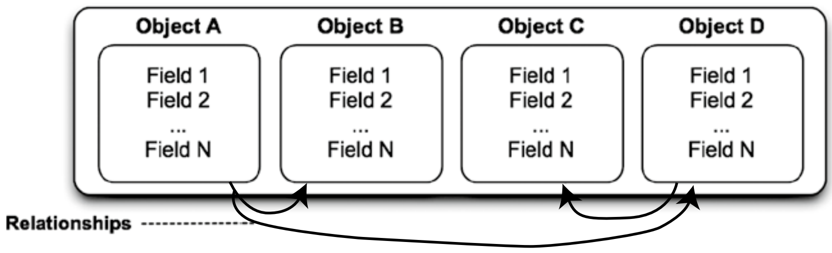


Figure 1-20. *A view of the data model*

More realistically, the creation of your data model will look like Figure 1-21.