# **Given a set of requirements, determine, create, and access the appropriate data model including objects, fields, and relationships**

Data Modeling and Management

* *After studying this topic, you should be able to:*
* *Define the data model and describe how it is used in Salesforce*
* *Describe how standard and custom objects and fields are used in Salesforce*
* *List the main types of relationships in Salesforce*
* *Describe how lookup, master-detail and many-to-many relationships are used to relate objects*
* *Describe the use cases of lookup, master-detail and many-to-many relationships in Salesforce*
* *Determine the appropriate data model to use in a given scenario*

Introduction

This topic explains some major **considerations** for choosing an appropriate **data mode**l in Salesforce and provides several **examples** about choosing an appropriate data model.

Creating a data model refers to the use of objects and fields for the **purpose** of setting up an **application** in Salesforce. It can comprisestandard and custom **objects, fields,** and different **relationship types**.

Salesforce Data Model

**Data Model**: Objects, fields, and relationships can be created in Salesforce to represent the data in the database.

**Standard Objects**: Standard objects such as Account, Contact, Lead, and Opportunity are included in Salesforce by default.

**Custom Objects**: Custom objects can be created to store specific information that cannot be stored in standard objects

Standard Fields: Standard fields such as Name, ID and CreatedDate are created by default.

**Custom Fields**: Create custom fields of different data types to store additional information.

**Lookup Relationships**: Two different objects can be linked such that there is no effect on deletion or security of records

**Many-to-Many Relationships**: A record of one object can be related to multiple records of another object and vice versa using a junction object.

Data Modeling

A data model can be created using standard objects, custom objects, and relationships. Schema Builder can be used for viewing and setting up the data model.

**STANDARD OBJECTS**: Standard objects such as account, contact, case, and opportunity are provided by Salesforce.

**CUSTOM OBJECTS**: Custom objects can be used to store information specific to a company’s requirement, such as properties.

**SCHEMA BUILDER**: Schema Builder allows viewing and setting up the data model quickly and efficiently.

**RELATIONSHIPS**: Different **relationship** **types** can be used to set up the data model, such as **lookup** and **master-detail**.

Data Model

A data model is a way to represent data in a database consisting of tables with columns and rows.

(Mô hình dữ liệu là 1 cách biểu diễn dữ liệu trong cơ sở dữ liệu bao gồm các bảng với các cột và các hàng)

**Objects, Fields & Record**s: Objects represent tables, fields represent columns, and records represent rows in Salesforce.

**Relationships**: Specific relationships can be defined among the objects in an app to create the data model.

**Apps:** A data model can represent a collection of objects and fields in an app and how they are related to one another.  
**Schema Builder**: The data model can be examined bynavigating to ‘Schema Builder’in Setup. Choosing a Data Model

Custom Application: A data model is typically **created for a custom application** in Salesforce.

Relationship Types: Relationships, such as lookup,master-detail and many-to-manyrelationships, can be utilized in a datamodel.

Composition: A data model can comprise **standard** and **custom objects**, **custom fields**, and **relationships.**

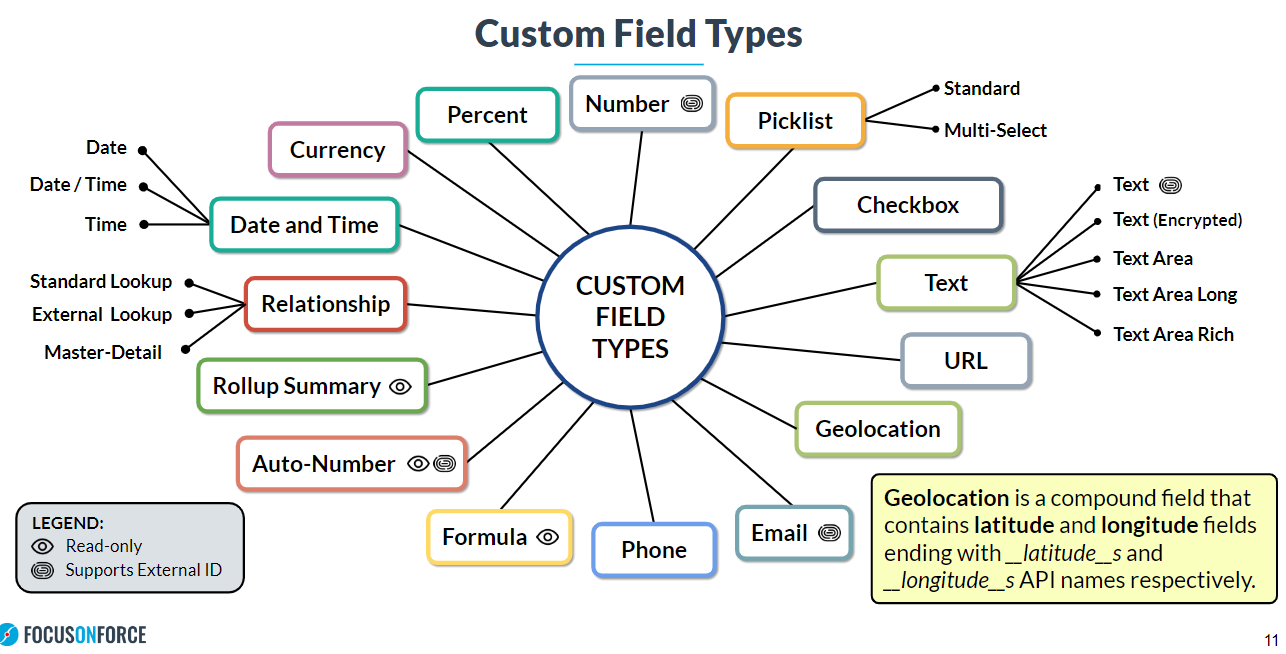
Using the Schema Builder: Schema Builder can be used to **view** thedata model as well **as create objects,fields, and relationships** quickly.

Business Objects: **Business objects**, including Account,Contact, Opportunity, and Case, are **provided by Salesforce**, which means thatcustom objects are not required for them.

Custom Fields: Custom fields can be created on both standard and custom objects to store **additional data.**

Custom Objects: Custom objects are used to **store information specific to the company or organization**, such as properties or employees.

Relationships: Two objects can be linked to each otherusing a relationship field. The type ofrelationship depends on factors such as **object type, security, and deletion**.



Lookup Relationships

A **lookup relationshi**p can be used to link two objects together loosely.

RECORD SECURITY: Parent and child objects have **independent** sharing settings. Each object has an **owner** field.

DELETING PARENT OPTIONS: The parent can be **prevented** from being **deleted** if children exist. Children can also bedeleted when parent is deleted (**requires** Salesforce support).

RELATED LIST: Records of the **child** object will appear in a related list on the **parent record**.

REQUIRED OR OPTIONAL: A lookup relationship field can be made required or optional in the **field level** configuration. It can also be made required on a **page layout**.

LOOKUP FILTER: A lookup filter **can be created** to define which records can be used as parent records.

SELF RELATIONSHIP: A lookup relationship field can be used to **link** an object with **itself**.

Many-to-Many Relationships

A **many-to-many relationship** can be created to link each record of an object to multiple records of another object and vice versa.

JUNCTION OBJECT: A **custom junction object** is created with two master-detail relationships, each for one of thetwo objects that need to be linked together.

RECORD ACCESS: Access to a junction object record is determinedby a user’s sharing access to both masterrecords.

DELETION: Junction object records are **deleted wheneither associated master record is deleted. If both master records are deleted**, the junctionobject record is deleted **permanently**.

ROLL-UP SUMMARY: **Roll-up summary fields** can be created on both master objects to summarize data from the junction object.

Lookup vs Master-Detail

|  |  |  |
| --- | --- | --- |
| Lookup Relationship | Compare | Master-Detail Relationship |
| Lookup field can be **required** or **optional** | Lookup Field | Lookup field is **always required** |
| Parent and child can have **different** owners | Record Owner | Parent and child have the **same** owner |
| Parent and child have **independent** sharing settings | Record Security | Child **inherits** sharing settings of parent |
| Delete behavior can be **specified** | Parent Deletion | All child records will be automatically **deleted** |
| Lookup filter can be **created** | Lookup Filter | Lookup filter can be **created** |
| Lookup field can be **optional** or requiredon the **page layout** | Page Layout | Lookup field is **always required** on the page layout |
| Roll-up summary field is **not supported** | Roll-Up Summary | Roll-up summary field is **supported** |

Schema Namespace:Classes and Methods

A number of important classes are available in the **Schema namespace** for accessing metadatainformation.

|  |  |
| --- | --- |
| APEX CLASS | DESCRIPTION |
| DescribeSObjectResult | This class provides methods for describing **sObjects**. |
| DescribeFieldResult | This class provides methods for describing **sObject fields**. |
| DescribeTabSetResult | This class provides methods for obtaining metadata information about a **standard** or **custom app**. |
| DescribeTabResult | This class provides methods for obtaining metadata information about a **tab** in a standard or custom **app**. |
| RecordTypeInfo | This class provides methods for obtaining metadata information about **record types** of an object. |
| SObjectType | An object of this class is returned from the field describe result using the **getReferenceTo**() method, or from the sObject describe result using the **getSObjectType**() method. |
| SObjectField | An object of this class is returned from the field describe result using the **getController**() and **getSObjectField**() methods. |

The **Schema class** has certain static methods for obtaining schema describe information.

|  |  |
| --- | --- |
| APEX METHOD | DESCRIPTION |
| getGlobalDescribe() | Returns a map of all sObject namesand tokens for standard and custom objects. |
| describeSObjects(sObjectTypes) | Retrieves metadata about an individualsObject or an array of sObjects. |
| describeTabs() | Retrieves metadata about the standard and customappsavailable to the running user. |
| describeDataCategoryGroups(sObjectNames) | Returns a list of category groups related to the specified sObjects. |
| groupStructures(pairs) | Returns data category structure for one or more category groups and objects. |
| getName() | Returns the name of the object used in Apex. |
| getLabel() | Returns the text label of the object, which may or may not match the object’s name. |
| getRecordTypeInfos() | Returns a list of record types supported by the object. |
| getRecordTypeInfosByDeveloperName() | Returns a map of developer names and associated recordtypes. |
| getObjectType() | Returns the Schema.SObjectType object for the sObject. |

Scenarios and Solutions

|  |  |
| --- | --- |
| SCENARIO | SOLUTION |
| The system administrator of an organization has been given the task of creating a **custom application** in Salesforce with **a number of custom objects and fields.**  The administrator is looking for the most efficient way to set up the data model for the application. | **Schema Builder** can be used to efficiently create custom objects, fields and relationships using a single user interface.It also allows viewing how the different objects are related to one another, which can be useful to understand how the data model should be set up. |
| A real estate company needs to store information about **properties** and **sales deals** in Salesforce. | Information about properties can be stored by creating a custom object which can be named **Property**. Sales deals can be stored using the standard **Opportunity** object. |
| An organization needs to relate every employee in the company to salary plans such that the salary plans of an employee are deleted when the employee leaves. | For this particular use case, a master-detail relationship field can be created on the Salary Plan object to the Employeeobject.Deleting an employee record would delete all the related salary plans. |
| A recruiting application should consist of candidates and positions such that each candidate can apply for multiple positions and each position can have multiple candidates. | A **many-to-many relationship** can be created for this use case using a junction object between **Candidate** and **Position**.Two **master-detail relationship fields** would need to be created on the junction object to the master objects. |
| Users of a non-profit school should be able to select a contact record on the **Alumnus** object that has been created to store alumni. The contact would represent the mentor of an alumnus, but the field should be optional. | For this use case, a **lookup relationship** to the Contact object can be created on the **Alumnus** object.  The lookup relationship field can be made **optional** on the Alumnus page layout. |
| Each user's record should allow specifying the name of the manager's manager. | A **hierarchical relationship** field can be created on the User object to relate a user to another user. |
| Each account record should be related to several child shipping records, but the child records should have **distinct owners** | **Lookup Relationship** can be defined on the child object. Unlike a master-detail relationship, a lookup relationship **does not** affect the **security and ownership** of the child records. |

# **Describe the capabilities of the various relationship types and custom IDs and the implications of each on record access and development**

After studying this topic, you should be able to:

* *Describe the capabilities of lookup, master-detail and many-to-many relationships in Salesforce*
* *Describe the implications of lookup, master-detail and many-to-many relationships on record access, user interface, and reporting*
* *Describe the capabilities of external lookup and indirect lookup relationships*
* *Describe the implications related to the use of external object relationships*
* *Identify what a hierarchical relationship can be used for in Salesforce*

Introduction

This topic describes the capabilities and considerations for different relationship typesin Salesforce.

Standard and custom Salesforce objects support the use of lookup, master-detail, many-to-many, external lookup, and indirect lookuprelationships. External objects support lookup, external lookup, and indirect lookup relationships. In addition, the User object supportshierarchical relationships. The implications of these relationships on record access, user interface, and reporting are explained in this section.

Relationship Types

The types of relationships available are: Lookup, Master-Detail, Many-to-Many, External Lookup, Indirect Lookup, Hierarchical

Lookup and Master-Detail Relationships

|  |  |
| --- | --- |
| Lookup Relationships | Master-Detail Relationships |
| Link two different objects in a loose relationship where the parent and child records have independent sharing settingsand are deleted separately.While objects in a lookup relationship loosely coupled, objects in a master-detail relationship are tightly coupled. | Link two different objects in a tight relationship where the sharing settings of the child record depend on the parent’s, and deleting the parent also deletes the child record. |

Many-to-Many and Hierarchical Relationships

While a **many-to-many** relationship requires a **third** object in order to create the relationship, the **User** object is only what is needed in a **hierarchical** relationship.

|  |  |
| --- | --- |
| Many-to-Many Relationships | Hierarchical Relationships |
| Link two objects such that a record of one object can be related to multiple records of another and vice versa. | Create a lookup field to associate one user with another that does not directly or indirectly refer to itself. |

External Object Relationships

In a relationship that involves an **external object**, an **external lookup** and **indirect lookup** relationship can be created.

|  |  |
| --- | --- |
| External Lookup Relationships | Indirect Lookup Relationships |
| Link a child standard, custom, or external object to a parentexternal object using the standard External ID field. | Link a child external object to a parent standard or custom object through a custom unique, External ID field. |

# **Describe the options for and considerations when importing and exporting data into development environments**

Salesforce provides several options for importing data into the org.

Data Loader: Data Loader can be used to load large files that contain up to **5 million records**.

Data Import Wizard: The Data Import Wizard can be accessed in Setup to import up to **50,000 records.**

Data Loader CLI: Data Loader can be used through **the command-line interface** for repetitive and complex operations.

Third Party ToolsETL tools: such as **Jitterbit** can be used for enterprise-scale import jobs.

Import Knowledge Articles: The ‘Import Articles’ page in Setup can be used to import Knowledge articles into Salesforce Knowledge.Salesforce provides several options for importing data into the org.

Data Loader CLI: Data Loader can be used from the **command-line interface** (CLI) to export records.

Data Import Wizard Versus Data Loader

|  |  |
| --- | --- |
| DATA IMPORT WIZARD | DATA LOADER |
| Supports custom objects, and a few standard objects | Supports custom objects, and all standard objects |
| Can import up to 50,000 records | Can insert, update, and upsert recordsCan insert, update, upsert, and delete recordsDoes not support data exportSupports data export |
| Can insert, update, and upsert records | Can insert, update, upsert, and delete records |
| Can insert, update, upsert, and delete records | Supports data expor |
| Field mappingscannot be saved | Field mappingscan be saved |
| Can bypass workflow rules and processes | Cannot bypass workflow rules and processes |
| Does not provide a command-line interface | Provides a command-line interface (Windows only) |