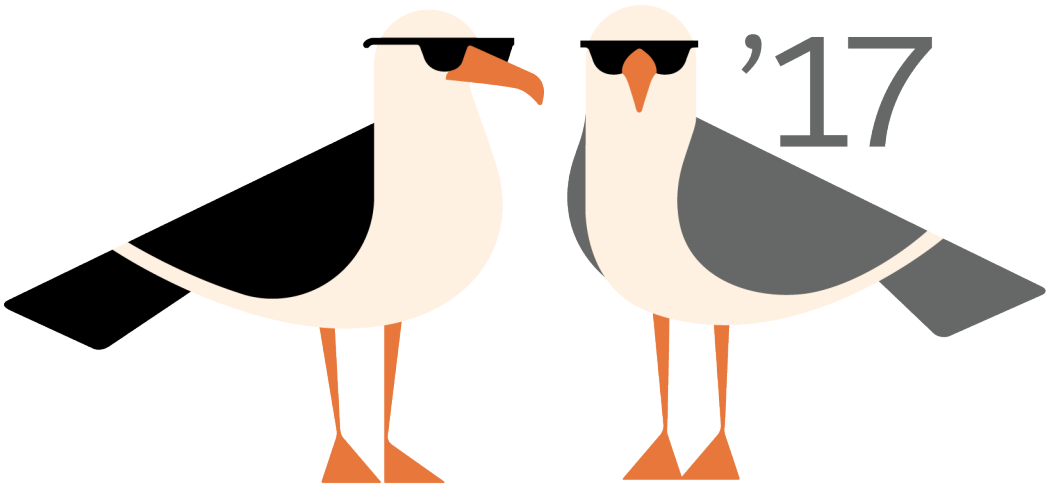
A Guide to Sharing Architecture

Salesforce, Summer '17



@salesforcedocs

Last updated: July 12, 2017

CONTENTS

A GUIDE TO SHARING ARCHITECTURE

Introduction

The Salesforce sharing model is an essential element in your organization's ability to provide secure application data access. Therefore, it's crucial to architect your sharing model correctly to meet your current and future data access requirements. In this document, we'll review data accessibility components, sharing model use cases, and real customer sharing solutions, and we'll provide some troubleshooting guidelines.

Intended Audience and Prerequisites

This document is intended for advanced system administrators and architects. To understand the concepts, you must have a working knowledge of the Salesforce security and sharing model. Prerequisites to this guide are:

- *Security Implementation Guide*
- Force.com Platform Fundamentals
- http://wiki.developerforce.com/page/An_Overview_of_Force.com_Security

Data Access Not Covered

The topics not covered in Data Accessibility Architecture:

- Folder access
- Content access
- Chatter access
- Knowledge Base access
- Ideas, Questions/Answers access
- Salesforce2Salesforce
- Mobile data accessibility

Types of Data Access

Record-level security lets you give users access to some object records, but not others. As with most applications, data access begins with a user. The application needs to know who you are before it provides access. For Salesforce, there are different types of users and, sometimes, the level of access is different by type. Instead of reviewing every attribute of every license type, we're going to focus on those interesting attributes that have significant impact on data access. Record ownership and full access are synonymous and interchangeable and provide the user with the highest level of access to a record.

Licenses

Full Sharing Model Usage Users/Licenses

Most Standard Salesforce license types take full advantage of the sharing model components. The license might not make a module accessible, or even some objects accessible. For example, the Force.com Free edition can't access any CRM objects. However, the sharing entities, and functionality, still exists and is ready when and if the module ever does become active.

High Volume Customer Portal License

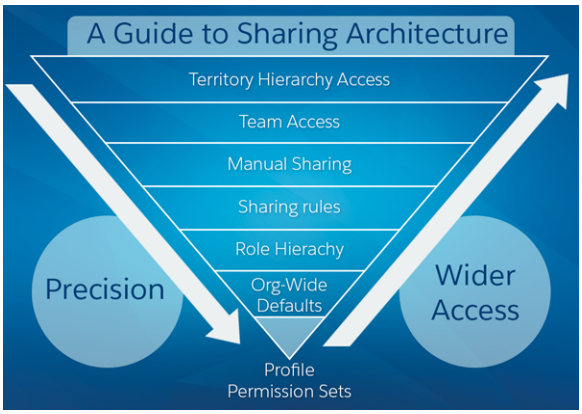
High Volume Customer Portal (HVPU) license users (including Community and Service Cloud license users) do not utilize the sharing model. HVPU licenses have their own sharing model that works by foreign key match between the portal user (holding the license) and the data on Account and Contact lookups. HVPU license is only used for the Customer Portal and not the Partner Portal.

Chatter Free License

The Chatter Free license doesn't follow the standard sharing model. Chatter Free is a collaboration-only license with the following features: Chatter, Profile, People, Groups, Files, Chatter Desktop, and limited Salesforce1 app access. The license doesn't have access to CRM records (standard or custom objects) and Content functionality, and therefore, there is no sharing.

For the remainder of this document, we are assuming a Salesforce user type utilizing a full sharing model. See User Licenses Overview for more information about each available license type.

Components



Profiles and Permission Sets

Profiles and permission sets provide object-level security by determining what types of data users see and whether they can edit, create, or delete records. For each object, the "View All" and "Modify All" permissions ignore sharing rules and settings, allowing administrators to quickly grant access to records associated with a given object across the organization. These permissions are often preferable alternatives to the "View All Data" and "Modify All Data" administrative permissions.

Profiles and permission sets also control field-level security, which determines the fields within every object that users can access. For example, an object may have 20 fields, but field-level security can be set up to prevent the users from seeing five of the 20 fields.

Record Ownership and Queues

Every record must be owned by a single user or a queue. The owner has access to the record, based on the Object Settings for the owner's profile. For example, if the owner's profile has `Create` and `Read` permission on an object, but not `Edit` or `Delete` permission, the owner can create a record for the object and see the new record. However, the owner won't be able to edit or delete the record. Users higher in a hierarchy (role or territory) inherit the same data access as their subordinates for standard objects. Managers gain as much access as their subordinates. If the subordinate has read-only access, so will the manager. This access applies to records owned by users, as well as records shared with them.

Queues help you prioritize, distribute, and assign records to teams who share workloads. Queue members and users higher in a role hierarchy can access queues from list views and take ownership of records in a queue.

If a single user owns more than 10,000 records, as a best practice:

- The user record of the owner should not hold a role in the role hierarchy.
- If the owner's user record must hold a role, the role should be at the top of the hierarchy in its own branch of the role hierarchy.

Organization-Wide Defaults

Organization-wide sharing settings specify the default level of access users have to each others' records. You use organization-wide sharing settings to lock down your data to the most restrictive level, and then use the other record-level security and sharing tools to selectively give access to other users. For example, let's say users have object-level permissions to read and edit opportunities, and the organization-wide sharing setting is Read-Only. By default, those users can read all opportunity records, but can't edit any unless they own the record or are granted additional permissions.. Organization-wide defaults are the **only way** to restrict user access to a record.

Organization-wide default settings can be changed from one setting to another (`private` to `controlled by parent`, then back to `private`); however, these changes require sharing recalculation and depending on volume could result in very long processing times.

For custom objects only, use the Grant Access Using Hierarchies setting, which if unchecked (default is checked), prevents managers from inheriting access. This setting is found in the organization-wide default settings.

Role Hierarchy

A role hierarchy represents a level of data access that a user or group of users needs. The role hierarchy ensures that managers always have access to the same data as their employees, regardless of the organization-wide default settings. Role hierarchies don't have to match your organization chart exactly. Instead, each role in the hierarchy should represent a level of data access that a user or group of users needs.

An organization is allowed 500 roles; however, this number can be increased by Salesforce. As a best practice, keep the number of non-portal roles to 25,000 and the number of portal roles to 100,000.

As a best practice, keep the role hierarchy to no more than 10 levels of branches in the hierarchy.

When a user's role changes, any relevant sharing rules are evaluated to correct access as necessary. Peers within the same role don't guarantee them access to each other's data.

Modeling the role hierarchy begins with understanding how the organization is structured. This is usually built from understanding a manager's scope, starting from the top. The CEO oversees the entire company. The CEO usually has direct reports that can then be segmented by Business Unit (Sales or Support) or geographical region (EMEA, APAC). That person then has direct reports that could be further segmented, and so on. Although this sounds very much like an HR organizational chart, and we have said they might be very much alike, keep in mind, when modeling data access, focus on data accessibility with a consideration to HR reporting.

Overlays are always the tricky part of the hierarchy. If they're in their own branch, they'll require either sharing rules, teams, or territory management to gain needed access. If they are folded into the hierarchy, there might be reporting implications.

It's important to spend the time setting up the role hierarchy because it's the foundation for the entire sharing model.

Role Hierarchy Use Cases
Management access – the ability for managers to be able to see and do whatever their subordinates can see and do.
Management reporting – the ability for reporting to roll up in a hierarchical fashion so that anyone higher in the hierarchy sees more data than those below them.
Segregation between organizational branches – different business units don't need to see each other's data; having a hierarchy in which you can define separate branches allows you to segregate visibility within business units, while still rolling visibility up to the executive levels above those units.

Role Hierarchy Use Cases

Segregation within a role – in many organizations/applications, people who all play the same role should not necessarily see each other's data. Having hierarchical roles allows you to define a "leaf" node in which all data is essentially private, and still roll that data up to a parent role that can see all.

Public Groups

A public group (not Chatter group) is a collection of individual users, roles, territories, and so on, that all have a function in common. Public groups can consist of:

- Users
- Customer Portal Users
- Partner Users
- Roles
- Roles and Internal Subordinates
- Roles, Internal and Portal Subordinates
- Portal Roles
- Portal Roles and Subordinates
- Territories
- Territories and Subordinates
- Other public groups (nesting)

Groups can be nested (Group A nested into Group B), however don't nest more than five levels. Nesting has an impact on group maintenance and performance due to group membership calculation. As a best practice, keep the total number of public groups for an organization to 100,000.

Public Groups Use Cases

When you need to provide access to an arbitrary group of people, you create a public group to collect them, and then use other sharing tools to give the group the necessary access. Group membership alone doesn't provide data access.

Groups can also be nested inside each other, therefore, allowing a nested group to gain the same access as the group in which it is contained. This allows the creation of smaller, ad-hoc hierarchies that don't necessarily interact with the role or territory hierarchies. If Group A is a member of Group B, then the members of Group A will have access to data shared to Group B at the same access level as the members of Group B.

Groups also have the ability to protect data shared in the group from being made accessible to people in the role hierarchy above the group members. This (and dealing with the access of record owners and their management hierarchy) allows the creation of groups in which very highly confidential information can be shared—the data will be accessible ONLY to group members, and nobody else in the organization. This is accomplished by using the Grant Access Using Hierarchies setting.

Ownership-based Sharing Rules

Ownership-based sharing rules allow for exceptions to organization-wide default settings and the role hierarchy that give additional users access to records they don't own. Ownership-based sharing rules are based on the record owner only.

Contact ownership-based sharing rules don't apply to private contacts.

As a best practice, keep the number of ownership-based sharing rules per object to 1,000.

Ownership-based Sharing Rule Use Cases
Role-based matrix management or overlay situations: a person in Service needs to be able to see some Sales data, but they live in different branches of the hierarchy, so you can create a rule that shares data between roles on different branches.
To provide data access to peers who hold the same role/territory.
To provide data access to other groupings of users (public groups, portal. roles, territories). The members of the groupings who own the records can be shared with the members of other groupings.

Criteria-based Sharing Rules

Criteria-based sharing rules provide access to records based on the record's field values (criteria). If the criteria are met (one or many field values), then a share record is created for the rule. Record ownership is not a consideration.

As a best practice, keep the number of criteria-sharing rules per object to 50; however, this can be increased by Salesforce.

Criteria-based Sharing Rule Use Case
To provide data access to users or groups based on the value of a field on the record.

Manual Sharing

Sometimes it's impossible to define a consistent group of users who need access to a particular set of records. In those situations, record owners can use manual sharing to give read and edit permissions to users who would not have access to the record any other way. Although manual sharing isn't automated like organization-wide sharing settings, role hierarchies, or sharing rules, it gives record owners the flexibility to share particular records with users that need to see them.

Manual sharing is removed when the record owner changes or when the sharing access granted doesn't grant additional access beyond the object's organization-wide sharing default access level. This also applies to manual shares created programmatically.

Only manual share records can be created on standard objects. Manual share records are defined as share records with the row cause set to manual share.

All share records (standard and custom objects) with a row cause set to *manual share* can be edited and deleted by the **Share** button on the object's page layout, even if the share record was created programmatically.

Manual Sharing Use Case
To provide the user with the ability to give access (read only or read/write) to the current record to other users, groups, or roles.

Teams

A team is a group of users that work together on an account, sales opportunity, or case. Record owners can build a team for each record that they own. The record owner adds team members and specifies the access level each team member has to the record. Some team members can have read-only access, while others have read/write.

Only owners, people higher in the hierarchy, and administrators can add team members and provide more access to the member. A team member with read/write access can add another member who already has access to the record with which the team is associated. The team member can't provide them additional access.

Creating a team member in the app creates two records: a team record and an associated share record. If you create team members programmatically, you have to manage both the team record and associated share record. There is only one team per record (Account, Opportunity or Case). If multiple teams are needed, depending on your specific needs consider territory management or programmatic sharing

The team object is not a first-class object. You can't create custom fields, validations rules, or triggers for teams.

Teams (Account and Opportunity) Use Cases
To provide the user with the ability to give access (read-only or read/write) to a single group of users (the team).
If teams are managed externally, say through an external commission or territory management system, then integration can be used to manage the account team. There are cases when territory management in an external system can align to a team solution within Salesforce.
Multiple owners of an account can be managed by the account team.
A single group of users (team) require either read-only or read/write access to an opportunity record. (Opportunity Team)

Territory Hierarchy

The territory hierarchy is a single dimensional, additional hierarchy which can be structured by business units or any kind of segmentation in a hierarchical structure. When territory management is enabled you must manage both the role hierarchy and territory hierarchy.

Territories exist only on Account, Opportunity and master/detail children of Accounts and Opportunities. Organizations can have up to 500 territories; however, this number can be increased by Salesforce. As a best practice, keep the territory hierarchy to no more than 10 levels of branches in the hierarchy.

If the assignment rules for a territory are changed, any Account Territory sharing rules using that territory as the source will be recalculated. Likewise, if the membership of a territory changes, any ownership-based sharing rules that use the territory as the source will be recalculated.

Territory Management Use Cases
Multiple groups of people (multiple teams) require either read-only or read/write access to accounts.
An additional hierarchical structure (different from the role hierarchy) is needed.
A single user needs to hold multiple levels in the hierarchy.
Global users (GAM – global account manager) need to see everything from the global account downward.

Account Territory Sharing Rules

Account territory sharing rules become available only when Territory Management has been enabled for an organization. These sharing rules provide access to Account records that have been stamped with the Territory defined in the rule.

Account Territory Sharing Rule Use Case
To provide data access to accounts within a territory (not based on ownership) to a grouping of users. Applies only to accounts and when territory management is enabled.

Programmatic Sharing

Programmatic sharing (formally Apex-managed sharing) allows you to use code (Apex or other) to build sophisticated and dynamic sharing settings when a data access requirement cannot be met by any other means.

If you create a share record programmatically, and the out-of-box row cause (*manual share*) is used, then you can maintain this share record with the **Share** button in the app. The share record is subject to all rules with the manual share row cause such as deletion upon owner transfer.

Review https://developer.salesforce.com/page/Using_Apex_Managed_Sharing_to_Create_Custom_Record_Sharing_Logic before you consider using programmatic sharing.

Programmatic Sharing Use Cases
No other method of sharing (declarative) meets the data access needs.
There is an existing, external system of truth for user access assignments which will continue to drive access and be integrated with Salesforce.
Poor performance by using native sharing components. (Usually applies to very large data volumes)
Team functionality on custom objects.

Implicit Sharing

Implicit sharing is automatic. You can neither turn it off, nor turn it on—it is native to the application. In other words, this isn't a configurable setting; however, it's very important for any architect to understand. Parent implicit sharing is providing access to parent records (account only) when a user has access to children opportunities, cases, or contacts for that account. Salesforce has a data access policy that states if a user can see a contact (or opportunity, case, or order), then the user implicitly sees the associated account. Child implicit sharing is providing access to an account's child records to the account owner. This access is defined at the owner's role in the role hierarchy. Child implicit sharing only applies to contact, opportunity, and case objects (children of the account). The access levels that can be provided are View, Edit, and No access for each of the children objects when the role is created. By setting to View, the account owner can implicitly see the associated object records (contact, opportunity or case). By setting to Edit, the account owner can implicitly modify the associated object (contact, opportunity or case).

Implicit sharing doesn't apply to custom objects.

Customer Implementation Scenarios

There isn't a sharing model that fits all organizations. Every organization has different requirements and challenges when trying to architect the best sharing model. It's crucial to use the most appropriate data access components to fit the sharing requirements of the organization. The following scenarios are common challenges when trying to architect a sharing model.

Customer Scenario: Team Assignment Managed Externally via Customer Master System

Requirements/Challenges	Solution
Two in a box: a sales manager of one geographic coverage area also wants access to another geographic coverage area in order to assist.	Ownership-based Sharing Rule: An ownership-based sharing rule works here because these are edge cases and not the norm. It is also acceptable if the ownership-based sharing rule provides more access than is truly necessary because this is a manager – a trusted individual.

Requirements/Challenges	Solution
Country-based operations users need access to all country sales data.	Ownership-based Sharing Rule: A very common use of a sharing rule is when a different department (other than sales) needs access to sales data.
At least 80% of the time, there is a "core 4" team on an account (Account Executive, Inside Sales Rep, Sales Consultant, Technical Sales Rep). The system of record for the account team assignment is external to SFDC. There is always only one team to an account.	Teams (Account and Opportunity): Since there is always only one team per account, even if there are many different members with different roles, the account team functionality satisfies this requirement.
Managers of the team should have the same access as their subordinates.	Role Hierarchy: The role hierarchy solves this by allowing managers to have access to the data of their subordinates.
The assigned account team should not be modifiable.	Teams (Account and Opportunity): This is not actually accomplished with the account team functionality, however, it also shouldn't prevent you from still using account teams. There are multiple ways of locking down the teams, however, for this case, removing the account team page layout is used.
There needs to be "buddy" functionality so that when someone is sick or on vacation, someone without standard access to an account or opportunity can be accessed and covered during the time off.	Teams (Account and Opportunity): A "buddy" can simply be a role on the team that accomplishes this requirement. However, the challenge comes from the previous requirement where Teams should not be modifiable. The only solution is to have a set group of people who can modify teams within SFDC to create the buddy role when necessary.
When a deal requires a custom solution, additional people (who are not necessarily in the sales organization) need to have access to the deal.	Teams (Account and Opportunity): A pretty standard usage of the Opportunity Team accomplished by manually adding a new member to the Opportunity Team (via related list). Can also be accomplished via a trigger if you always know who should be added. In this case, it is opportunity by opportunity.

Customer Scenario: Out-of-box Territory Management

Requirements/Challenges	Solution
Two different opportunity teams from two distinct business units (Retails Sales and Remarketing) need access to the same account record. They should share contacts and be aware of all activities on the account. These two business units have their own hierarchy and rollups.	Territory Management: The best way to think of this is having two branches of a hierarchy that may be structured very differently. What justifies territory management is that there are two levels of these two different branches (both levels with members = the opportunity team for that business unit) who need access to the account. Although you could have accomplished this with a Teaming concept, that was too granular. The sales segmentation was not a an account level but in a hierarchy.
There is a separate group of business developers who are assigned and need access to specific accounts for a specific opportunity team (a territory). The business developers are shared resources for the opportunity teams which mean each business developer	Territory Management: Because this is a group of users (or a team), and each business development team could be different by account, and since territory management was needed for another reason, then the likely best approach is to build out sub-territories that represent these business development teams.

Requirements/Challenges	Solution
may be assigned to one or more accounts for one or more opportunity teams.	
There are non-commission based sales supporting roles who need access to accounts on a one off basis.	Teams (Account and Opportunity): The key portion of the requirement is "one off basis". This means it is done on an account by account basis so account teams provide that natively.
The credit department needs access to all accounts for a given business unit.	Ownership-based Sharing Rule: This is a situation where across the board, for a given business unit, a group of users needs to see everything. This could be accomplished with a sharing rule for a role the group belongs to, a branch of the role hierarchy the group belongs to (role and subordinates) or even a public group.
Managers should have the same access as their subordinates.	Role Hierarchy: The role hierarchy solves this by allowing managers to have access to the data of their subordinates.

Considerations

You have exhausted all other sharing components and you still need territory management. Territory management is not reversible, so it's extremely important to know its implications.

What happens to the Role Hierarchy?

Nothing happens to the role hierarchy. You are now managing two hierarchies, which means sharing is more complex. The best practice is to flatten (or simplify) the role hierarchy as much as possible. If your organization is only an SFA organization with mostly sales data, it should be possible to flatten the role hierarchy and use the territory hierarchy as the sales hierarchy. However, if your organization has non-sales applications, like Service Cloud, an HR application, or a legal application, then you'll likely need a hefty role hierarchy as well (to satisfy those non-sales users). The rule of thumb is to make your role hierarchy your non-sales hierarchy, try to flatten the sale department branch(es), and then use the territory hierarchy as your "sales" hierarchy. We do not recommend making the role hierarchy and territory hierarchy identical because it will cause unnecessary sharing activity.

Another consideration when determining the necessity of the role hierarchy is that some functionality is only available via the role hierarchy, such as Delegated Admin, My Teams filters, and Folder-based access.

Can You Still Use Teams?

Yes. However, if you can satisfy your access requirements within the territory hierarchy (like overlays), it is better to do it there than to use teams. You are already maintaining two hierarchies (role and territory), so in trying to keep things as simple as possible, only implement teams if no other existing sharing component will satisfy the requirement.

Realignment and Reassignment

There are two types of changes that will occur – the membership of roles, teams, or territories, and the structure of the hierarchy. Membership changes can typically occur daily, even hourly. Hierarchy structural (realignments) changes generally occur less often (quarterly, semi-annually, or annually) and can be resource expensive. What needs to be considered are the volume of changes and the number of cascading changes that each change will cause. As a rule of thumb, have structural changes occur no more than quarterly and all changes of high volume (bulk or mass changes) be well planned, tested, and coordinated.

Large Data Volumes

Whether you are modeling for the initial rollout or planning a realignment change, you need to make some serious consideration to the volume of data. There are thresholds where performance can become a factor, so testing out your changes in a sandbox is highly recommended before production. This will also give you a baseline for how long the change will take.

If you have more than two million accounts, and have implemented teams or Territory Management, you especially need to pay attention to performance. These are complex sharing model components that can make for a huge volume of share records and hence, long running transactions.

Defer Sharing Calculations

If you have an object that utilizes sharing and has a large volume of records (such as more than two million accounts), and you need to make a bulk change (such as a quarterly realignment requiring a hierarchy change), then there is a feature that can be enabled by Salesforce Support to defer automatic sharing calculations. Natively, every individual change to the role hierarchy, territory hierarchy, groups, sharing rules, user roles, team membership, or ownership of records can initiate automatic sharing calculations. When a bulk change is made, it causes a number of automatic sharing recalculations to begin. By suspending these temporarily, you are able to make the change and then have sharing calculations happen all at once. This is typically a more efficient and better performing method to bulk changes.

Data Skews/Ownership Skews

Data skews are defined as a few parent records with many children records. This can really hurt you when a few accounts have many contacts, opportunities, or cases. The ratio where we start seeing performance degradation is 1:10,000. As a best practice, keep the ratio as close to that as possible (lower is preferred).

Ownership skews are similar to data skews, except if we are referring to a single user, role, or group owning a large number of records for an object. As with data skews, these can also cause long running transactions, causing a performance degradation when change occurs. The recommended ratio of owner to number of records is also 1:10,000.

If a single user owns more than 10,000 records, as a best practice:

- The user record of the owner should not hold a role in the role hierarchy.
- If the owner's user record must hold a role, the role should be at the top of the hierarchy in its own branch of the role hierarchy.

The Account Hierarchies Impact on Data Access

A lot of people make a very bad assumption when they implement an account hierarchy. They assume the users who can access a parent account can also access the children accounts. The simple fact of only having a parent/child relationship between two records does not drive access. Although the role hierarchy and the territory hierarchy do work in this way, the account hierarchy does not.

Troubleshooting

Once you have completed your sharing model architecture, you will likely be challenged with why a user can or can't see a record. Typically, you won't hear when someone can see something they shouldn't, but should that arise, there is a way to see every user who has access to a record and why.

The more difficult challenge, and probably more common, is why a user can't see a record. The security layers you have architected will determine where you start. If you know the sharing model well, then you will probably know what component should have provided the access and should start there. But if you are less familiar with the sharing model, start with the role hierarchy and peel back each layer to determine which one should provide the access and make an update. Here is a troubleshooting flow.

1. Verify that the user has permissions to access to the object.
2. Identify the user's role who can't see the record and note it.
3. Identify the owner's role of the record and note it.
4. Review the role hierarchy and verify these two roles are in two different branches (they should be).
5. Now you need to review the sharing rules for the object and make sure there is no rule that will grant the user access. This can also cause you to look in public groups as well. Maybe the user just got left out of a group where there is a sharing rule, or does it make sense to create a new sharing rule to grant the user access? This depends on the architecture you are trying to maintain, and applies to both ownership-based sharing rules and criteria-based sharing rules.
6. If you are using teams, should this user be on the team for that record? How are teams maintained and how did the miss occur?

7. If manual sharing is used, the user may have lost access because the record owner changed. Manual shares are dropped when ownership changes. The manual share could also have been removed using the **Share** button.
8. If you are using territory management, is the user missing from one of the territories? Where is the membership of territories maintained and how did the miss occur? Or, maybe the record did not get stamped with the territory where the user is a member.
9. If you are creating programmatic shares and there are criteria for creating the share in code, review the code to understand why this user was omitted.

Have feedback on the Guide to Sharing Architecture? Send your feedback to ccefeedback@salesforce.com.