Designing Record Access for Enterprise Scale

Salesforce, Summer '25

Last updated: July 11, 2025

CONTENTS

DESIGNING RECORD ACCESS FOR ENTERPRISE SCALE

Introduction

This guide introduces advanced topics in user record access, and provides recommendations on how to configure your organization to optimize access control performance.

In particular, this guide focuses on the effects of group maintenance on sharing performance and the built-in sharing behaviors that support Salesforce applications. It also:

- Addresses how to avoid common configuration traps that can drag down the performance of your record-level access system
- Introduces some key platform features that can help you speed up large-scale sharing realignments

This guide is intended for expert architects working on Salesforce implementations with complex record access requirements or large-scale sales organization realignments. However, the sharing best practices covered can be applied to orgs of any size.

This guide assumes expertise in Salesforce administration and security, and knowledge of SQL and relational database concepts. It also assumes a familiarity with the content in Record-Level Access: Under the Hood, which explains the inner workings of the flexible and powerful Salesforce record access infrastructure.

Group Membership Operations and Sharing Recalculation

The Salesforce role hierarchy, public groups, and territories are closely connected to sharing rules and other security features. Because of these relationships, seemingly simple changes to groups and group membership can sometimes involve substantial recalculations of users' access rights.

For example, when an administrator moves a user from one branch of the hierarchy to another, Salesforce performs these actions to ensure that other users have correct access to data owned by that relocated user.

- If the user:
  - Is the first member in his or her new role to own any data, Salesforce adds or removes access to the user's data for people who are above the user's new or old role in the hierarchy.
  - Owns any customer or partner accounts, Salesforce removes any child customer or partner account roles from the user's original role and adds them as children to the user's new role.

    Note:
    - Salesforce also adjusts implicit shares that provide access in the hierarchy to records owned by or shared to customer or partner users. See Implicit Sharing.
    - Salesforce must perform these tasks for every customer or partner account that the user owns.

- Salesforce also recalculates all sharing rules that include the user's old or new role in the source group. It removes all of the user's records from the scope of sharing rules where the old role is the source group and adds those records to the scope of rules where the new role is the source.

  Note: If the user owns customer or partner accounts, and there are sharing rules that use customer or partner account roles as the source group, Salesforce sometimes must recalculate those rules. Some sharing rules might no longer be valid given the user's new location in the hierarchy, in which case an administrator might need to modify or delete them.

During the user's move, the managers in the branch above the user's old role lose access to all the data that the user owns, as well as to child records shared through the managers' role settings. Managers in the branch above the user's new role gain access to the user's accounts and to child records according to their own role settings.

From this example, you can see that a lot can happen under the hood when an administrator takes what looks like a simple action, such as changing the role of a user. We chose this operation to illustrate all the possible types of sharing maintenance, but other common group and data updates can have a similar impact.

Moving a role to another branch in the hierarchy

One benefit to moving a whole role is that any customer or partner accounts simply move along with their parent role, and Salesforce doesn't have to change the related sharing. On the other hand, Salesforce must do all of the work involved in moving a single user for all users in the role being moved and for all of those users' data.

Changing the owner of a customer or partner account

The effort required for what looks like a simple data update—changing the name of the user in the Account Owner field—can be surprising. When the old and new owners are in different roles, Salesforce isn't only moving the customer and partner account roles to a new parent role but also adjusting the sharing for all the data associated with the customer or partner account.

Ownership Data Skew

Even with all of the work that Salesforce does to maintain correct access, most customers never encounter performance issues unless they're performing updates that affect many users or large amounts of data. However, there are certain common configurations that greatly increase the probability of performance problems, such as ownership data skew.

Ownership data skew is when a single user owns more than 10,000 records of an object. This situation commonly occurs when concentrating ownership of data so that a single user or queue, or all the members of a single role or public group, owns most or all records for a particular object.

For example, a customer can assign all unassigned leads to a dummy user. This practice can seem like a convenient way to park unused data. However, it can cause performance issues if those users are moved around the hierarchy, or moved in to or out of a role or group that is the source group for a sharing rule. In both cases, Salesforce must adjust a very large number of entries in the sharing tables, which can lead to a long-running recalculation of access rights.

Distributing ownership of records across a greater number of users decreases the chance of long-running updates occurring.

If you do have a compelling reason for assigning ownership to a small number of users, you can minimize possible performance impacts by not assigning the users to a role.

Tip: You can take the same approach when dealing with a large amount of data that is owned by or visible to the users under a single partner or customer account—changing the owner of that account or moving those users in the hierarchy requires the system to recalculate all the sharing and inheritance for all the data under the account.

If the users must have a role to share data, we recommend that you:

- Place them in a separate role at the top of the hierarchy. (Note that this user inherits access to all data owned by or shared with users below them in the hierarchy).
- Don't move them out of that top-level role to avoid triggering sharing recalculations.
- Keep them out of public groups that can be used as the source for sharing rules.

Group Membership Locking

When updating the role hierarchy or group membership in Setup or through the API, customers can occasionally receive a "could not acquire lock" or "Group membership operation already in progress" error and have to repeat the operation. This error occurs because the sharing system locks the tables holding group membership information during updates to prevent incompatible simultaneous updates or timing issues, both of which could lead to inaccurate data about users' access rights.

Typically, these locks are held only very briefly, so most customers never see a lock conflict error. In some scenarios—such as a change in role triggering a sharing rule recalculation—locks can be held for a longer time, and conflicts can occur.

Customers who experience these locking errors are typically executing large-scale data loads or integrations with other internal systems that are making changes to role and group structure, user assignments to roles and groups, or both. When these processes are running—and an administrator tries to change a user's role, or the customer tries to provision a new external user—one of these simultaneous operations might be unable to secure the lock it requires. The most likely time for this failure to occur is during periodic organizational realignment events, such as end-of-year or end-of-quarter processing, where many account assignments and user roles are changing.

Customers can lessen the chance of locking errors by:

- Scheduling separate group maintenance processes carefully so they don't overlap
- Implementing retry logic in integrations and other automated group maintenance processes to recover from a failure to acquire a lock

Note: You can also receive locking errors if you're updating the role hierarchy or group membership while running other deployments that also update group membership information or have Apex tests that do so. If you receive locking errors, wait for the deployment operation or Apex tests to finish.

By default, granular locking is enabled, which allows some group maintenance operations to proceed simultaneously if there's no hierarchical or other relationship between the roles or groups involved in the updates. Administrators can adjust their maintenance processes and integration code to take advantage of this limited concurrency to process large-scale updates faster, all while still avoiding locking errors.

The key advantages of granular locking are that:

- Groups that are in separate hierarchies can be manipulated concurrently.
- Public groups and roles that don't include territories aren't blocked by territory operations.
- Users can be added concurrently to territories and public groups.
- User provisioning can occur in parallel.
  - External user creation requires locks only if new external roles are being created.
  - Provisioning new external users in existing accounts occurs concurrently.
- A single-long running process, such as a role delete, blocks only a small subset of operations.

This table lists all the operations that can occur in parallel. Note that certain operations, such as reparenting (moving roles within the role hierarchy), still block almost all other group updates.

Group Operation	Can be Performed Concurrently with...
Role creation	• User role change[1] • Territory reparenting • Territory deletion • Territory creation • Removal of user from territory • Addition of user to territory • User provisioning[2]
Role deletion	• Territory reparenting • Territory deletion

Group Operation	Can be Performed Concurrently with...
	• Territory insertion • Removal of user from territory • Addition of user to territory
Role reparenting (includes change of customer or partner account owner)	Territory creation
Adding user to territory	• Role deletion • Role insertion • Territory creation • Addition of user to territory • User provisioning[3]
Removing user from territory	• Role deletion • Role insertion • Territory creation • User provisioning[3]
Territory reparenting	• Role deletion • Role insertion • User provisioning[3]
Territory deletion	• Role deletion • Role insertion • User provisioning[3]
Territory creation	• Role reparenting • Role deletion • Role insertion • User role change[1] • Addition of user to territory • Removal of user from territory • User provisioning[3]
Provisioning internal user with an existing role	• Role insertion • User role change[1] • Territory reparenting • Territory deletion • Territory creation • Removal of user from territory

Group Operation	Can be Performed Concurrently with...
	• Addition of user to territory • User provisioning[3]
Changing user role (User must not own any customer or partner accounts.)	• Role insertion • Territory insertion • User provisioning[3]
Provisioning first customer or partner user under an account	• User role change[1] • Territory reparenting • Territory deletion • Territory creation • Removal of user from territory • Addition of user to territory • User provisioning[2]
Creating second customer or partner user under an account	• Role insertion • User role change[1] • Territory reparenting • Territory deletion • Territory creation • Removal of user from territory • Addition of user to territory • User provisioning[3]
Provisioning high-volume Experience Cloud site user	Any group membership operation
Changing customer or partner account owner	Territory creation
Changing role of a user who owns a customer or partner account	Territory creation

[1] The user must not own any customer or partner accounts.

[2] Provisioning standard user or external user in an existing role

[3] Provisioning any standard or external user, including the first customer or partner user under an account

Takeaway: Tuning Group Membership for Performance

Understand the performance characteristics of the various group maintenance operations that you're performing. Always test substantial configuration changes in a full copy sandbox that's been recently refreshed so you know what to expect in production.

Here are some specific suggestions.

- Identify user and group updates that are complex, such as user role and customer or partner account ownership changes, or updates that involve a large amount of associated data. Allow for additional time to process these changes.

- When making changes to the hierarchy, process changes to the bottom (leaf) nodes first, then move upward to avoid duplicate processing.
- Limit the number of records of an object owned by a single user to 10,000.
- Tune your updates for maximum throughput by experimenting with batch sizes and using the bulk API, where possible.
- Remove redundant paths of access, such as sharing rules that provide access to people who already have it through the hierarchy.
- Schedule large group membership operations during off-peak hours.
- Run batch operations in serial mode.

Object Relationships, Bulk Loading, and Sharing Recalculation

Choices that you make when designing your data models can have a major impact on sharing performance when data is loaded, updated, or transferred between users. Understanding how Salesforce handles the relationships between objects and protects data integrity during updates can help you optimize the performance of data operations.

Implicit Sharing

The sharing capabilities of the Salesforce Platform include a wide variety of features that you can use to explicitly grant access to data for individuals and groups. In addition to these more familiar features, there are a number of sharing behaviors that are built into Salesforce. This kind of sharing is called implicit because it's defined and maintained by the system to support collaboration among members of sales teams, customer service representatives, and clients or customers.

Parent-Child Data Skew

Implicit sharing behaviors simplify the task of managing security for users. They handle the most common data access use cases without requiring additional roles, groups, and sharing rules to be configured. However, like data ownership skew, some parent-child configurations can slow the performance of large data loads and updates, and sometimes even of single-record operations.

Record-Level Locking

Many customers regularly upload large amounts of data to the service, and maintain integrations with other systems that update their data in scheduled batches or continuously in real time. Like other transactional systems, Salesforce employs record-level database locking to preserve the integrity of data during these updates.

Takeaway: Tuning Data Relationships and Updates for Performance

Understand the performance characteristics of the various maintenance operations that you're performing and always test substantial data uploads and changes to object relationships in a full copy sandbox that's been recently refreshed so you know what to expect.

Implicit Sharing

The sharing capabilities of the Salesforce Platform include a wide variety of features that you can use to explicitly grant access to data for individuals and groups. In addition to these more familiar features, there are a number of sharing behaviors that are built into Salesforce. This kind of sharing is called implicit because it's defined and maintained by the system to support collaboration among members of sales teams, customer service representatives, and clients or customers.

This table describes the different kinds of implicit sharing built into Salesforce applications and the record access that each kind provides.

Type of Sharing	Provides	Details	Example
Parent	Read-only access to the parent account for a user with access to a child case, contact, or opportunity	• Not used when sharing on the child is controlled by its parent	Henry is a Standard User with access to an opportunity through an owner-based sharing rule. He can view data about the

Type of Sharing	Provides	Details	Example
		• Expensive to maintain with many account children • When a user loses access to a child, Salesforce must check all other children to see if it can delete the implicit parent. • Not granted by the View All permission on the child object	opportunity's parent account, but he can't edit the account's data.
Child	Access to child case, contact, or opportunity records for the owner of the parent account	• Not used when sharing on the child is controlled by its parent • Controlled by child access settings for the account owner's role • Supports account sharing rules that grant child record access • Supports account team access based on team settings • When a user loses access to the parent, Salesforce removes the user's access to all children records • Not granted by the View All permission on the parent object	Henry is a Standard User who owns an account. The Salesforce admin set up Henry's role so that assigned users can view all opportunities, cases, and contacts associated with accounts they own. Henry can therefore view data for all of the child opportunities, cases, and contacts for the account he owns, but he can't edit this data.
Experience Cloud Site	Access to a customer or partner account and all associated contacts for all customer or partner users under that account. Access to a case for the customer or partner user that's the contact on the case.	Account and associated contacts shared with the lowest role under the customer or partner account	Sarah is a Customer Community Plus User added under the "Acme" account. Sarah has Read access to the "Acme" account as well as Read access to all the other contacts related to the account.
High Volume[1]	Full access to data owned by high-volume users associated with a sharing set for members of the sharing set's share group	All members of the sharing set's share group gain full access to every record owned by every high-volume user associated with that sharing set	For your site, a sharing set is created to grant high-volume users on the same account Read Only access to each other's cases. Internal users added to the sharing set's share group have full access to these cases.

Type of Sharing	Provides	Details	Example
High Volume Parent	Read only access to the parent account of records shared through a sharing set's share group for members of the share group	Maintains the ability to see the parent account when users are given access to account children owned by high-volume users	For your site, the same sharing set for cases mentioned in the previous example automatically gives the internal users added to the share group read access to the cases' parent accounts.

[1]To allow external users to scale into the millions, high-volume users have a streamlined sharing model that doesn't rely on roles or groups, and functions similarly to calendar events and activities. High-volume users include the Customer Community, High Volume Customer Portal, and Authenticated Website license types.

Parent-Child Data Skew

Implicit sharing behaviors simplify the task of managing security for users. They handle the most common data access use cases without requiring additional roles, groups, and sharing rules to be configured. However, like data ownership skew, some parent-child configurations can slow the performance of large data loads and updates, and sometimes even of single-record operations.

A common configuration that can lead to poor performance is the association of a large number of child records (10,000 or more) with a single parent account. For example, a customer can have tens or hundreds of thousands of contacts generated by marketing campaigns or purchased from mailing lists—without any association to formal business accounts. If a contact is required to have an associated account, what's best for an administrator to do? It can be convenient to park all those unallocated contacts under a single dummy account until their real business value and relationship can be determined.

While this option seems reasonable, this kind of *parent-child data skew* can cause serious performance problems in the maintenance of implicit sharing.

For example, assume that you have 300,000 contacts all under the same account. A user with access to one of these contacts also has a parent implicit share in the account sharing table that gives the user access to that account. Now what happens if that user loses access to the contact?

In order to determine whether to remove the user's sharing to the account, Salesforce must scan all of the other 299,999 contacts to ensure that the user doesn't have access to them either. This practice can become expensive if Salesforce is processing many visibility changes on these highly skewed accounts.

Record-Level Locking

Many customers regularly upload large amounts of data to the service, and maintain integrations with other systems that update their data in scheduled batches or continuously in real time. Like other transactional systems, Salesforce employs record-level database locking to preserve the integrity of data during these updates.

The locks are held very briefly and don't present the same performance risks as some of the other organization locks. However, they can still cause updates to fail, so you must still be careful not to run updates to the same collections of records in multiple threads.

In addition to taking this standard precaution, developers and administrators should know that for objects with a master-detail relationship, when you update child records, the system locks the parent and the child records to prevent inconsistencies. For example, updating a child record whose parent has just been deleted in another thread. When objects being processed have a master-detail relationship, two situations in particular pose a risk of producing locking errors.

- Updates to parent records and their children are being processed simultaneously in separate threads.
- Updates to child records that have the same parent records are being processed simultaneously in separate threads.

Because Salesforce holds these locks very briefly, customers who experience a small number of locking errors might be able to handle the problem by adding retry logic to their integration code. If you experience frequent locking from integrations and mass updates, sequence batches so that the same records aren't updated in multiple threads simultaneously.

Takeaway: Tuning Data Relationships and Updates for Performance

Understand the performance characteristics of the various maintenance operations that you're performing and always test substantial data uploads and changes to object relationships in a full copy sandbox that's been recently refreshed so you know what to expect.

Here are some specific suggestions.

- Use a Public Read Only or Read/Write organization-wide default sharing model for all non-confidential data.
- To avoid creating parent implicit shares, configure child objects to be Controlled by Parent wherever this configuration meets security requirements.
- Configure parent-child relationships with no more than 10,000 children to one parent record.
- If you encounter only occasional locking errors, see if the addition of retry logic is sufficient to solve the problem.
- Sequence operations on parent and child objects by `ParentID` and ensure that different threads are operating on unique sets of records.
- Tune your updates for maximum throughput by working with batch sizes, timeout values, Bulk API 2.0, and other performance-optimizing techniques.

Tools for Large-Scale Realignments

The most demanding maintenance activity that customers perform is a large-scale realignment of sales teams, territories, and account assignments. Whether you do realignments annually, quarterly, or more frequently, the realignments typically involve extensive changes to an organization's structure and updates to large amounts of data, both of which result in many changes to record access.

At the same time, sales realignments are very time sensitive—failing to complete them quickly can adversely affect revenue. Optimizing the performance of sales realignments is naturally a key concern of many enterprise administrators, and Salesforce provides features to help with the planning and execution of realignments.

Deferred Sharing Maintenance

Performing a large number of configuration changes can lead to long sharing rule evaluations or timeouts. To avoid these issues, you can suspend sharing calculations, specifically for sharing rules and group membership, then resume calculations during an organization's maintenance period.

Takeaway: Making Realignment Smoother

Understand the pros and cons of the performance tools, and make sure they fit well with the process and timing of your realignment. Always test these tools and new realignment processes in a full copy sandbox that's been recently refreshed so you know what to expect.

Deferred Sharing Maintenance

Performing a large number of configuration changes can lead to long sharing rule evaluations or timeouts. To avoid these issues, you can suspend sharing calculations, specifically for sharing rules and group membership, then resume calculations during an organization's maintenance period.

In addition to all the technical concerns administrators must manage to perform a major realignment, they must also coordinate closely with the business to ensure that end users aren't adversely affected when access rights are being adjusted. In an enterprise environment

in which multiple systems are continually processing updates, it can be difficult to schedule an organization or sharing rule change that can take substantial time to complete. Deferred sharing maintenance can help with increasing the predictability of these kinds of updates.

Here's how deferred sharing maintenance works.

1. Based on requests from the business, an administrator identifies a number of changes to the role hierarchy and group membership, or updates to sharing rules.
2. Given best estimates of the remaining overall work, the administrator negotiates a maintenance window for completing the processing.

   Tip: This window should be modeled in a full copy sandbox to get the best estimate possible.

3. Instead of processing each separate update and waiting for it to complete, the administrator prepares all the information required to perform all updates ahead of the planned maintenance window.
4. At the start of the maintenance window, the administrator uses the deferral feature to essentially "turn off" processing of group maintenance operations, and then makes all the desired changes to role and group membership at the same time.

   Note: Sharing rule processing is also deferred at this time so the administrator can perform all sharing rule updates.

5. After the changes have completed, the administrator resumes processing group maintenance, and the system performs a recalculation to make all the role and group changes take effect.
6. At this point, the system is in a state that requires a full recalculation of all sharing rules for user access rights to be complete and accurate. The administrator can resume sharing rule processing immediately or wait to start the process at a later time. After the sharing rule recalculation has completed, all the access changes take effect.

When using the deferred sharing features, it's especially important to test the whole process in a full copy sandbox. This practice helps benchmark how long the overall recalculation is likely to take in production and smooth out any kinks in orchestrating deferred sharing maintenance. The full copy sandbox should mimic your current product environment as closely as possible. We recommend using a sandbox refreshed within the last 30 days that reflects all major changes.

Note: Deferred sharing maintenance doesn't defer the recalculation of some sharing changes in order to preserve data integrity. These calculations that can't be suspended can take significant time to process based on the org's settings and data volume.

Who's a Good Candidate for Deferred Sharing?

There are two main criteria for determining whether deferred sharing maintenance is the right tool for your organization: the size and complexity of your realignment activities, and the flexibility you have to arrange a maintenance window with your customers. If you find that organizational changes and sharing rule updates typically complete quickly enough to be scheduled into the workday and weekend downtimes in your use of the service, you're unlikely to benefit substantially from this feature. On the other hand, if you're able to negotiate downtime with your business customers and have been struggling to complete updates in a timely fashion, deferred sharing can be a good solution to your problem.

Takeaway: Making Realignment Smoother

Understand the pros and cons of the performance tools, and make sure they fit well with the process and timing of your realignment. Always test these tools and new realignment processes in a full copy sandbox that's been recently refreshed so you know what to expect.

- Consider whether it's more efficient to:
  - Set aside specific maintenance windows
  - Defer organizational or sharing rule maintenance while processing your updates

Note: While making your decision, remember that deferring organizational maintenance requires recalculating sharing rules for all objects.