Sharing Architecture

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 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 utilise 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 Salesforce 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 Sales.

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 organisation. 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.

A Guide to Sharing Architecture Types of Data Access

Queues help you prioritise, 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.

Organisation-Wide Defaults

Organisation-wide sharing settings specify the default level of access users have to each others’ records. You use organisation-wide
sharing settings to lock down your data to the most restrictive level. Use the other record-level security and sharing tools to selectively give access to other users. For example, users have object-level permissions to read and edit opportunities, and the organisation-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 other permissions. Organisation-wide defaults are the only way to restrict user access to a record.

Organisation-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 organisation-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 organisation-wide default settings. Role hierarchies don't have to match your organisation chart exactly. Instead, each role in the hierarchy should represent a level of data access that a user or group of users needs.

An organisation 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.

Modelling the role hierarchy begins with understanding how the organisation 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 organisational chart, and we have said they
might be very much alike, keep in mind, when modelling 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 organisational 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.

A Guide to Sharing Architecture Types of Data Access
Role Hierarchy Use Cases
Segregation within a role – in many organisations/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 organisation 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 organisation. This is accomplished by using the Grant Access Using Hierarchies setting.

Ownership-based Sharing Rules

Ownership-based sharing rules allow for exceptions to organisation-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.

A Guide to Sharing Architecture Types of Data Access

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 don’t have access any other way. Manual
sharing isn’t automated like organisation-wide sharing settings, role hierarchies, or sharing rules. However, it gives record owners
the flexibility to share particular records with users that must 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 organisation-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.

A Guide to Sharing Architecture Types of Data Access
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. 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, 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 the original territory management feature has been enabled for an

organisation. 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.

A Guide to Sharing Architecture Types of Data Access

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 organisations. Every organisation 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 organisation. The following scenarios are common challenges when trying to architect a sharing model.

Revision #2
Created 1 September 2020 14:40:03 by SmitH
Updated 1 September 2020 14:56:31 by SmitH