Scalable Security Modeling with Microsoft Dynamics CRM

VERSION: 1.1

AUTHOR: Roger Gilchrist
COMPANY: Microsoft Corporation

RELEASED: September 2015
Updated: March 2016
Applies to: Microsoft Dynamics CRM 2015 and Microsoft Dynamics CRM 2016

Microsoft Dynamics CRM

Feedback

To send comments or suggestions about this document, please click the following link and type your feedback in the message body: Send Feedback

Important: The subject-line information is used to route your feedback. If you remove or modify the subject line, we may be unable to process your feedback.

Table of Contents

Introduction **6**

Common business scenarios **7**

Access control options **11**

Concepts 11
- Security principals 11
- Access control components 12
- Privileges 13
- Access control mechanisms 22

Sharing 25
- Sharing with users 25
- Sharing with teams 27
- Access Teams 27

Record ownership 28
- User ownership 29
- Team ownership 29
- Owner teams 30

Business unit privileges 30
- Business unit local privileges 31
- Business unit deep privileges 32

Organization privileges 34

Hierarchies 34
- Hierarchy levels 37
- Hierarchy types 37

Access control to fields 38

Scalability characteristics of Microsoft Dynamics CRM elements **40**

Privilege types 40

SQL View access 40

Initial access: caching 41
- Initial user access caching 41
- Cache flushing 42

Sharing access checks 42
- Sharing records 42
- Single record sharing access check 46
- Multiple record sharing access check 46
- Cascading sharing 47
- Team sharing 47
- Removing sharing 48
- Sharing implications 48

Access team lifecycles 48
- Design considerations of using access teams 50

Ownership access checks 51
- Accessing an individual record 52
- Accessing a view or performing a RetrieveMultiple 53
- Ownership implications 54

Business unit access checks 55
- Accessing an individual record 56
- Accessing a view or performing a RetrieveMultiple 57

Business unit privilege implications 58
Organization-wide privileges 59
Organization-wide access implications 59
Hierarchy access 59
Hierarchy levels 60
Combinations of access types 62
Trade off with granular access 63
Comparison 64
Alignment with real world usage 65
Usage patterns 65
Active involvement 66
Management involvement 67
Design considerations 69
Understanding business needs and scenarios 69
User types and usage patterns 69
Granularity of access 69
Design patterns 70
Separating and optimizing different usage patterns 70
Customizing the security model for different business areas 71
Customizing the security model to account for exceptions 72
Separating historical data and active data 73
Modelling security walls rather than the organizational hierarchy 75
Providing separate reporting 75
Controlling versus filtering 76
Modelling data along security lines 77
Security role versus privilege 78
Controlling security through automation 79
Optimization example 80
Original approach 81
Alternative sharing approach 82
Use owner teams for sales, access teams for management 84
Consider different role needs 86
Use business units for senior managers and oversight roles 88
Use reporting 89
Archive historical data 91
Overall assessment 92
Summary 92

Introduction

Microsoft Dynamics CRM 2015 and Microsoft Dynamics CRM 2016 offer a wide range of security modeling features, and it is important to choose the most appropriate approach to implementing a particular solution. Each feature offers a combination of characteristics that provide a balance between granularity of access control, administrative ease, and impact on scalability. Having an understanding of the underlying mechanisms supporting each security modeling feature can be useful when selecting the best approach to solving a particular challenge, especially when planning to develop a large volume system.

Granting access for a user to the system can be broken out into:

- **Authentication**: Determining who the user is and confirming that they are who they say they are
- **Authorization**: Determining whether the authenticated user is entitled to access the system and what they're permitted to see or do in the system

Authentication in Dynamics CRM is handled using platform features such as Integrated Windows Authentication or claims-based authentication with an identity provider such as Active Directory Federation Services. These all determine the identity of the user who is requesting access to the system. The deployment and scalability of the technologies supporting authentication is best described by resources focused specifically on those technologies and, therefore, is out of the scope of this document.

After a user has been identified, information recorded about the user in the Dynamics CRM system, such as their security roles and team memberships, is used to determine whether they are allowed to use the system and what they are allowed to see and do in the system, or in other words, what they are authorized to do.

This paper describes how these security modeling features in Microsoft Dynamics CRM for authorization work at scale, the implications associated with these features functioning at high volumes, and guidance on common and recommended usage patterns for modeling Dynamics CRM security at scale, incorporating teams as appropriate.

Important: For additional information about scalable security modeling in Microsoft Dynamics CRM 2016, see Security concepts for Microsoft Dynamics CRM.

Common business scenarios

In most CRM implementations, access to information is either provided openly within the organization or it's limited by a combination of the role and the business area or group in which a user works or operates. In many organizations, people perform multiple roles concurrently. Sometimes, there are also requirements for exceptional circumstances in which individuals require access to information that is outside of their normal job demands and perhaps information that wouldn't normally be exposed to them.

While there is no one-size-fits-all model and different businesses and industries follow varying approaches, common user access patterns do emerge, particularly regarding alternative perspectives on relationship management. The reason these common patterns occur is that often the approach to interact with a client and the way that client expects to be treated by the organization are the same particularly when the importance of that interaction to the business is equivalent, even though the actual content of the conversations are very different.

Typically encountered user access patterns are described in the following table.

Usage pattern	Description
Active involvement	▪ Regular, significant involvement directly with the customer/deal ▪ Informed, with existing knowledge of the customer/deal and current related activity, and personal actions based on a direct relationship with the people involved
Secondary involvement	▪ Informed involvement, maintaining active knowledge of activity but not directly participating or acting on the deal or with the customer, such as providing cover for absence of actively involved staff ▪ Support others who have a personal relationship with customer such as providing advice or support to the people actively involved, providing specialist knowledge to a specific deal or customer
Transactional interaction	▪ Specific activity oriented involvement, for example, receiving and acting on a request to update a customer's address ▪ No personal or on-going engagement, such as in a contact center
Management oversight	▪ Managerial or governance responsibility across a business or geographical area ▪ Viewing and directing involvement of others rather than specific involvement
Reporting	▪ Aggregated business reporting ▪ Data organized to preserve anonymity rather providing direct access to customers/deals
Compliance	▪ Oversight read-only access to all records for a business area

In a CRM system, an important concept to understand and model is the nature of the active relationship to individual customers, including aspects such as:

- How often the organization and customer interact
- Who initiates each interaction
- Whether or not there is interaction even when no active business is taking place at that moment
- Who within the organization may be involved in an interaction with the customer

How each of these interaction characteristics is exhibited for an organization can vary depending on the type of service the organization delivers and the size and type of customer base they work with to be able to deliver an effective working model. This interaction in a relationship often can be viewed based on the value of the relationship with a customer; the higher the value, the more personalized and actively managed the relationship

becomes. In this context, value can be measured from a variety of perspectives, including financial, influence, sensitivity, or risk, depending on the specific business in question.

Characteristics of the different values of customer interactions are shown in the following table.

Value of interaction	Characteristics
Low	▪ Minimal investment of time ▪ Transactional relationship ▪ No personal relationship ▪ Wide access
Medium	▪ Proactive management ▪ Perception of personalized approach ▪ Group relationship/access
High	▪ Large investment of time ▪ Personalized approach ▪ Personal relationship ▪ Privacy/controlled Access

In some industries, particularly in financial services and professional services, users typically work more on the basis of individual opportunities or cases. With higher value services, such as investment banking and legal services in which large sums of money are involved, a common requirement is to provide access to information only when a person needs to work on individual deals or cases. This requirement may arise for a number of reasons, such as legal restrictions, privacy, competitive detail, or data sensitivity.

In these scenarios, people from different parts of the business work together in teams on each opportunity or case. Often, there isn't a specific pattern for allocating people particular work items, but instead work is allocated based on criteria such as specialist skills and availability. In these types of scenarios, it's important to only grant permissions to individual records or sets of records (such as a case and all the supporting activities related to the case) to the specific people who will be involved.

This determination of restricted access is important to define. In many cases, while there is a need to assign individual responsibility, there is no requirement to prevent other users from seeing the information. The preceding examples contain many cases in which it is important to control access, but the additional checks and controls that are required add complexity to the solution implementation and to the processing required of the system. It is therefore a valuable exercise to determine if the extra security controls are genuinely required to address the business need. For situations that don't require the extra controls, it makes sense to determine this early on, as broad access needs for secondary usage patterns may contradict an initial perception that tight controls to primary owners are required. This is particularly important for environments in which individual owners are supported by much broader call center support teams that need access to the same data to assist the customer.

For situations in which it is important to control access to individuals directly involved in a deal or case, team ownership can become an effective way to model access to information. This approach is typically combined with access granted at a more general level for specialist roles or users who need access to a wide ranging set of information. This type of access can be required for roles such as Compliance Officer or General Manager that work across all the information in an area.

In addition to considering options for granting users access to data, it can be equally important to manage situations in which users should no longer have access to information, such as when an employee leaves a job or

changes roles. In these cases, their access must be revoked. As a result, be sure to carefully consider the lifetime of information access permissions.

Access control options

Microsoft Dynamics CRM offers a variety of ways to model user access to information. Understanding the options available is vital to determine the most appropriate solution for a particular scenario. The following sections describe a range of approaches for providing users with access to information in Dynamics CRM.

Concepts

This section will describe:

- The objects to which security access can be granted.
- The components used to grant security access.

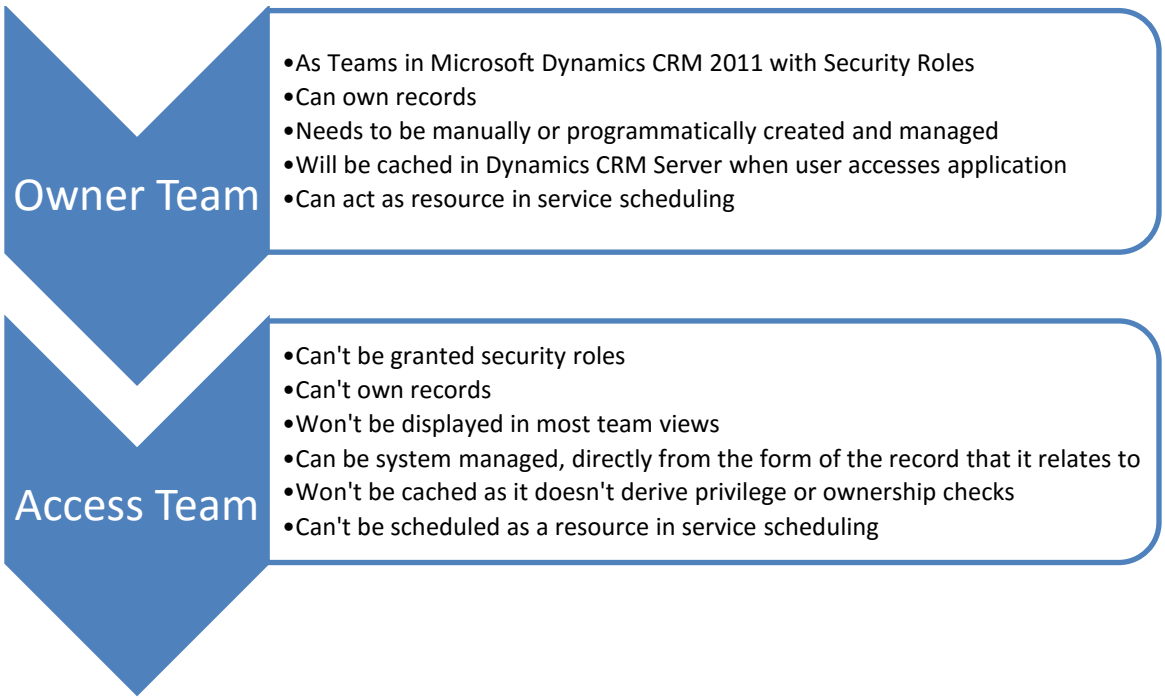
Security principals

Access to data and actions is provided by granting privileges to security principals.

Security principals in Dynamics CRM represent:

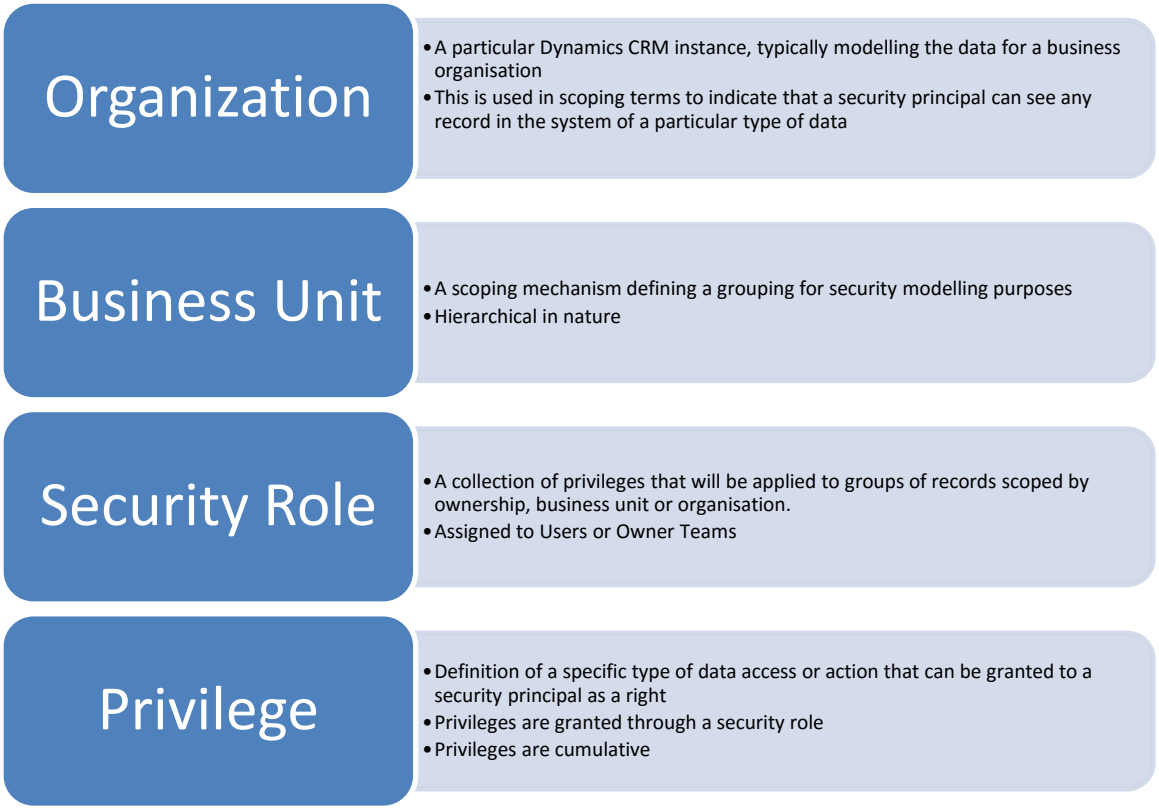
- System users:
    - Represents a user of the system and has a distinct identity for access to Dynamics CRM.
    - Representing either a physical human being or a logical identity such as to provide integration to another application.
    - Is defined as a member of a business unit.
- Teams
    - Can't define an identity to access Dynamics CRM.
    - Is a logical entity in Dynamics CRM used to model common groupings for security modelling.
    - Is defined as a member of a business unit.
    - Can contain zero or more users as members.
    - Have two subtypes:
        - Access Teams
        - Owner Teams

Introduced in Dynamics CRM 2013, these subtypes of teams allow modelling a range of optimized scenarios. A summary of the difference between the types of teams is shown in the following diagram.



Access control components

There are a number of components used in security modelling as shown in this diagram.



Privileges

The concept of a privilege in Dynamics CRM is an important one as privileges are used to define the entitlement to perform an action as well as the range of records against which that privilege applies.

The privilege itself defines either:

- A type of record and the type of action that can be performed on that record, such as write permission to the Account entity.
- Or an action the user can perform in Dynamics CRM, such as actions in a CRM mobile app.

The level of a privilege defines the scope of that privilege.

- Basic: Applies to records owned by the user or team that the privilege is linked to.
- Local: Applies to records in the same business unit as the security principal that the privilege is linked to.
- Deep: Applies to records in the same business unit as the security principal that the privilege is linked to or any of the child business units.
- Organization: Applies to all records in the organization.

Security roles are used to manage collections of related privileges, which can then be assigned collectively to a security principal. The security principals are either a user or a team.

Security roles, therefore, are intended to assist with the process of administration and granting of privileges, not to grant rights to users directly. A scenario that often comes up in customization design is how to check whether a user is entitled to perform a particular action, perhaps a custom action. While it may be tempting to check a user for a security role, an approach that is better aligned with the way that the platform works is to check whether the user has a particular privilege granted by using a security role. As will be shown later, privileges granted to a user are cached. There are many standard mechanisms of the product that allow a privilege to be directly checked. Using privileges rather than roles to control a particular action is a more efficient way to manage a user's activity in the application.

The privileges for a security role apply to the security principal they are linked to.

- If actions and the related privileges directly apply to or on behalf of a user, the user needs those privileges granted directly. For example:
    - When logging onto the system it is the user that is logging on and therefore the user that needs the initial privileges for access.
    - Providing owner privileges to a team security role doesn't apply to the records the user owns even if the user is a member of that team.
- If the security role is linked to a team, that privilege applies to the team, and is therefore applied in the context of that team.

This becomes especially important to differentiate where providing owner privileges to a team security role doesn't give access to the records the user owns, even if the user is a member of that team.

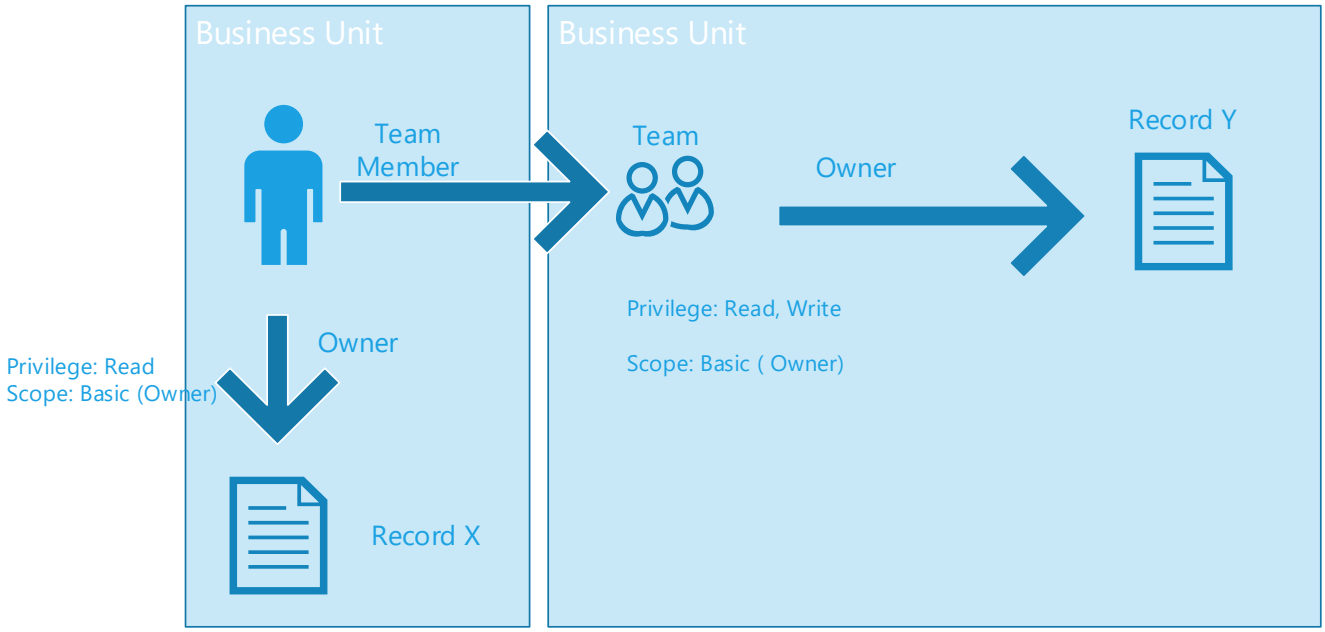
- Where the privileges can equally apply to the user or the team, but having one does not infer the other:
    - Create privileges at a basic scope for a team means that records can be created with the team as the owner.
    - But for the user to also be able to create records they own, that user must have Create privileges at a basic scope through their own security roles. The Team basic scope creation privilege to create

records owned by the team does not infer the User basic scope creation privilege to create records owned by the user.

- Where the privileges do not apply equally to a user or team. For example, only a user can sign in to the system or access views in the UI:
    - Privileges for these user actions need to be held in a security role linked to the user, not in a team security role, where the actions are not applicable to the team.

In the following example, although the user is granted read and write privileges as part of their team membership, this only applies to records that the team itself can access through ownership by the team. This therefore applies to Record Y and means the user can read or write Record Y.

This doesn't, however, also automatically grant privileges directly to the user to records they can access directly. The user can therefore access Record X through their own direct read privileges, but this doesn't mean they can write to that record because they don't have write privileges directly as the user and they don't inherit write privileges to records they own themselves from team assigned privileges.



We will examine some scenarios that highlight these concepts but also clarify some of the considerations needed when defining privilege models.

Minimum privileges for signing in

Before the user can act on any broader data, as part of signing in a minimum set of permissions are required for the user to be simply able to access the system itself. These privileges are required to be granted directly to the user through a security role assigned to the user. They can't be inherited from a team security role because it's the user, and not the team, who is accessing the system.

These minimum privileges for access are:

- When signing in to Microsoft Dynamics CRM:
    - To render the home page, assign the following privileges on the Customization tab: Read Web Resource, Read Customizations.

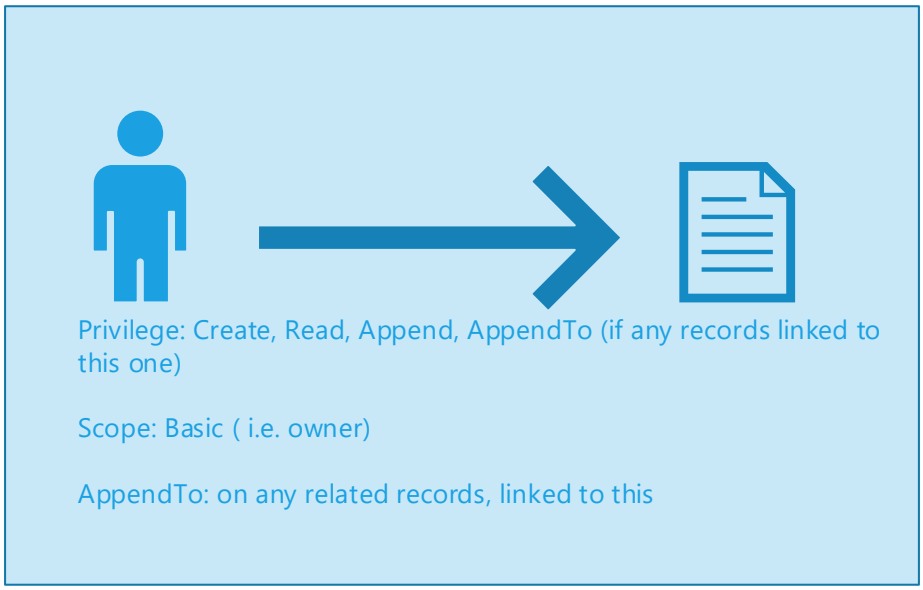
- To render an entity grid (that is, to view lists of records and other data): Read privilege on the entity, Read User Settings on the Business Management tab, and Read View on the Customization tab.
- To view single entities in detail: Read privilege on the entity, Read System Form on the Customization tab, Create and Read User Entity UI Settings on the Core Records tab.

- When signing in to Microsoft Dynamics CRM for Outlook:
  - To render navigation for Dynamics CRM and all Dynamics CRM buttons: Read Entity and Read View on the Customizations tab.
  - To render an entity grid: Read privilege on the entity, Read Customizations and Read Web Resource on the Customization tab, and Read Saved View on the Core Records tab.
  - To render entities: Read privilege on the entity, Read System Form on the Customization tab, and Create, Read, and Write User Entity UI Settings on the Core Records tab.

These minimum privileges are documented in the TechNet article, Create or edit a security role.

Basic creation assigned to the creating user

Creating a record assigned to the current user highlights the core creation privileges required for creation as shown in the following diagram.



To create a record a user needs, at least, the following privileges.

- Definitely required
  - Create, Read: Basic scope
    - Need the ability to create and view the created record.
    - Because the record will be owned by the creating user only, Basic (or Owner) scope is required for these privileges. If a user has a deeper scope of privilege, such as Local/Deep/Organization, this would also work. However, at least Basic level of scope is required.
- May be required
  - Append: Basic scope
    - If this record is linked to other records, Append privilege is also needed.

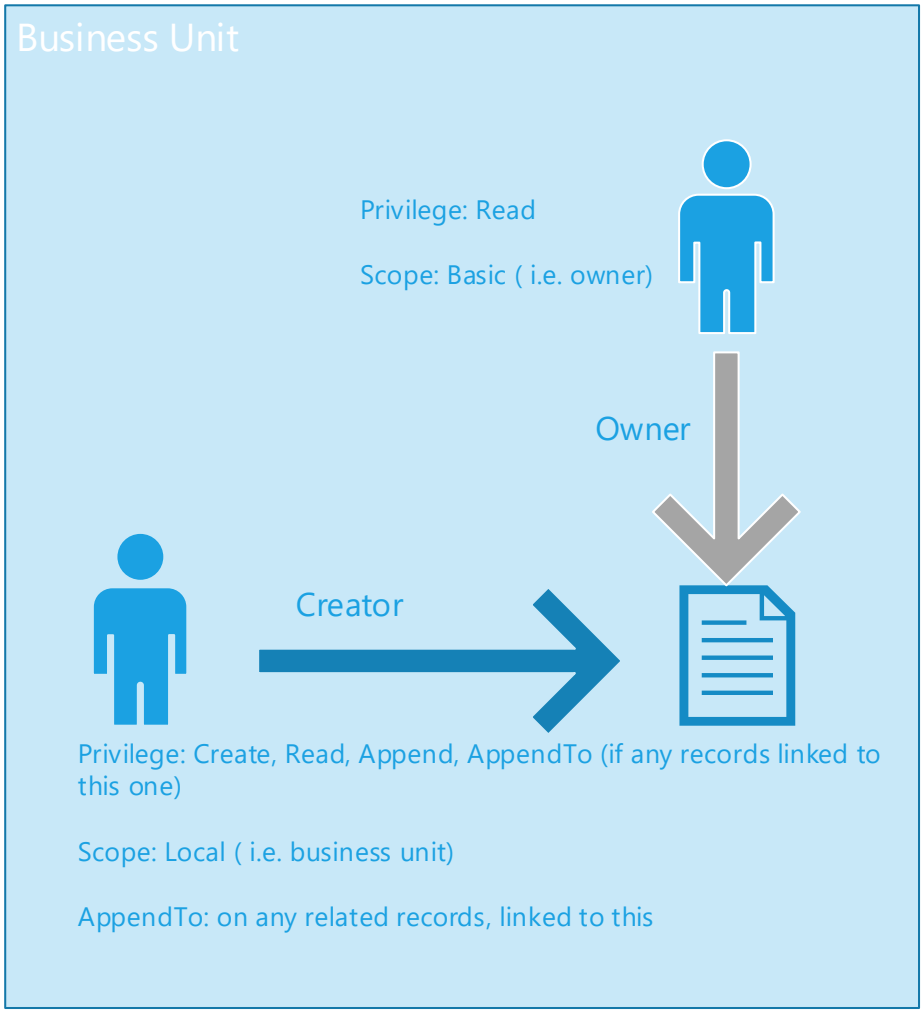
- Append To:
    - If other records are linked to this one, AppendTo is also required.
    - The scope would depend where the other records exist. If the other records are also owned by the same user, Basic scope would suffice. If the records are in another business unit, wider permissions are needed, such as Organization scope, or via a team.

Creation of a record assigned to another owner

Creating a record assigned to a different user highlights the need to consider these privileges:

1. The rights to create a record.
2. The ownership privileges in cases where the record needs to be assigned to a different user.

The following diagram shows the privileges needed by both the creating user and the user set as the owner.



Creating a record for another user requires privileges for both:

- The creator: confirming what and where they can create data.
- The person who is set as the owner: confirming they are entitled to own this data.

If we look at those two cases separately, each user would need:

- Creator: To create a record a user needs the same privileges as when creating the record for themselves, but differently scoped.

        - Instead of only “Basic” scoped data, the user needs “Local” scoped data to reflect that the data will be created in the same business unit rather than for them as the owner.
        - Definitely required
            - Create, Read: Local scope.
        - May be required
            - Append: Local scope.
            - Append To: Scope dependent on record location.
- Owner:
    - To own data the privileges the owner requires are:
        - Read: Basic scope (as they will be the owner).

Creation of a record assigned to another owner in a different business unit

When creating a record in another business unit, the creating user needs privileges to the other business unit. This can be achieved through one of these mechanisms:

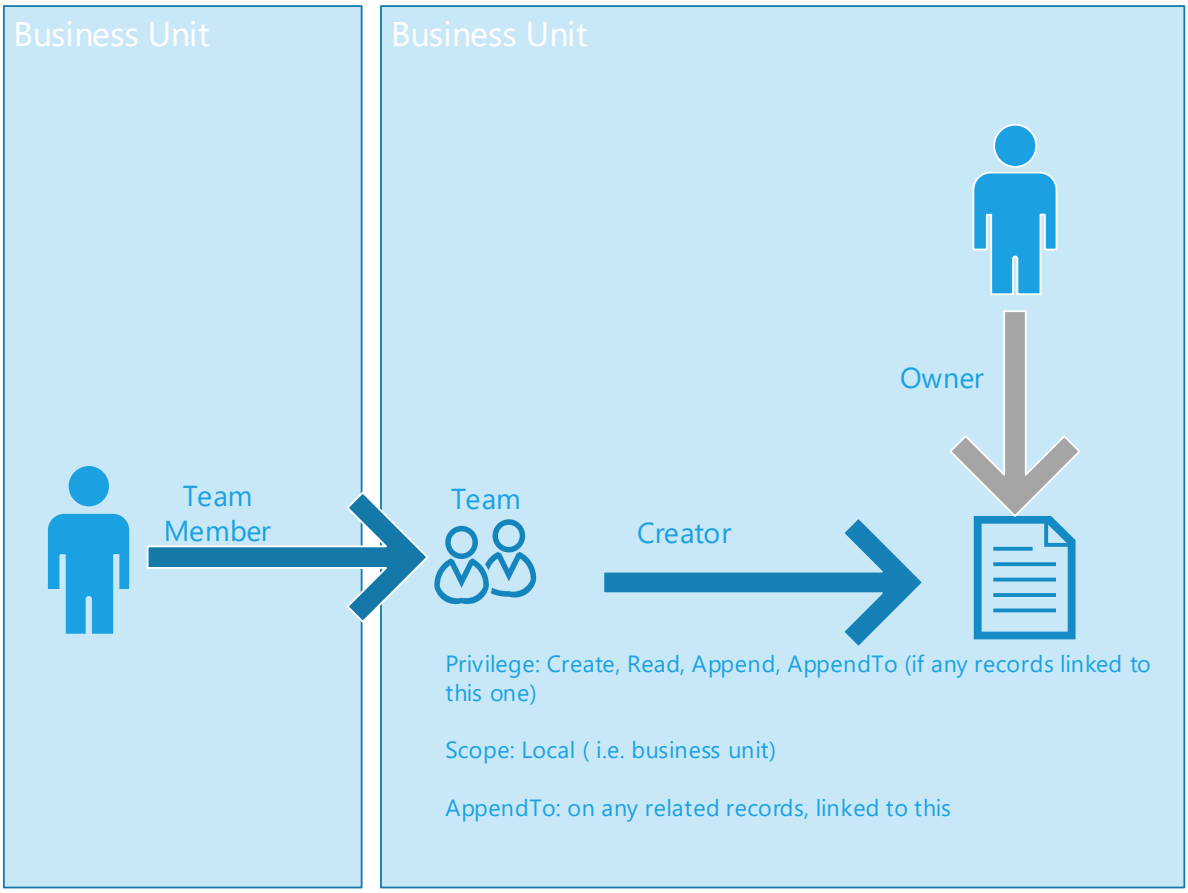
- User Organization level privileges: this grants the user the right to create a record anywhere in the system.
- User Deep privileges: this grants the user the right to create a record in either their own business unit or a child business unit of the unit they are in. So if the target user exists in a child business unit of the creator’s business unit, then deep privileges would enable this.
- Team privileges: because a user can only exist in one business unit, if the target owner is in a business unit that cannot be accessed through either Organization, Local or Deep privileges of the user, then making the user a member of a team in another business unit and granting a security role to the team allows the user to be indirectly granted rights to create records there. For example:
    - Only records that the team itself owns through Basic scope.
    - Records owned by any security principal in that business unit through Local or Deep scope.

In most scenarios, a user is granted privileges in a different business unit through a team in that business unit with a security role:

- Because, in this scenario, we’re granting access to a user from outside of that business unit, the default business unit team can’t be used because it contains only users in the business unit as members.
- So a separate team has to be created and assigned a security role to grant privileges.
- Providing Organization scope privileges to a user is often too open an approach as the user can then create records anywhere, and Deep privileges are constrained to a hierarchical structure. Using a team allows more granular control as to which business units a user can create records in.

Creating a record for another user requires privileges for:

- The creator: confirming what and where they can create data.
- The person who is set as the owner: confirming they are entitled to own this data.



Assuming that the creating user is a member of a team that is used to provide the creation privileges, each user requires the following privileges:

- Creator: Needs no direct privileges, although the user would need to be in a team that has the create privileges.
- Creator team: To create a record, a team needs the same privileges as a user would need to create the record.
    - Definitely required
        - Create, Read: Local scope.
    - May be required
        - Append: Local scope.
        - Append To: Scope dependent on record location.
- Owner (user or team)
    - To own data the privileges the owner requires are:
        - Read: Basic scope (as they will be the owner).

Append and Append To privileges

When creating a record in isolation, the only privileges required relate to the record itself. But in a relational system it is rare that an isolated record is created.

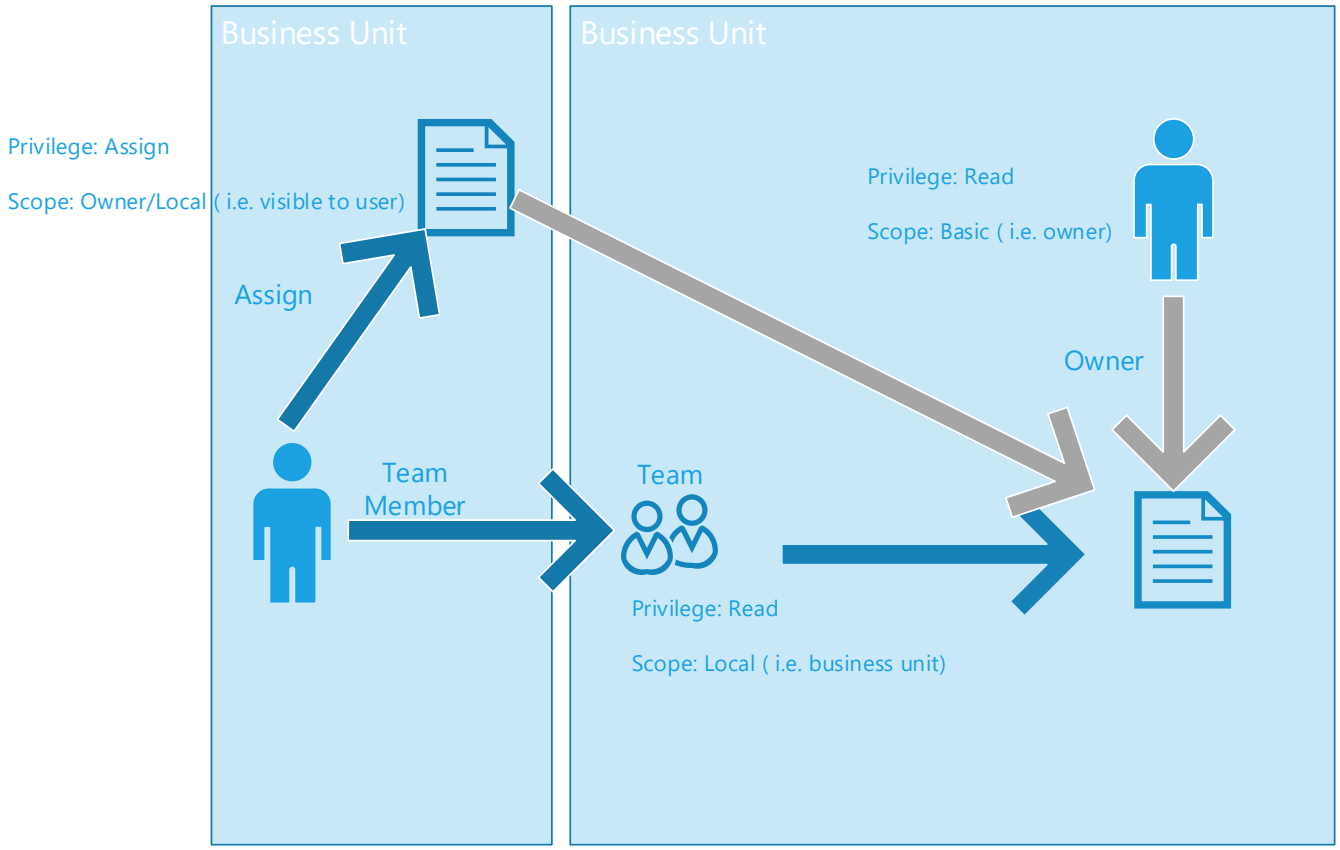
In most cases, relationships to other records are also set up as part of the creation of the record. In this case there are other privileges that are required indicating the ability to set up relationships between certain kinds of records.

In these cases, the appropriate level of Append and Append To privileges are required to any related records.

- Append:
  - The same scope of privilege is required as to create the record.
  - If the created record is owned by the user or in the same business unit, Basic or Local privileges for the user may apply.
  - If the created record is in another business unit, either user Organization level privileges are required or the creating team will need the Append privilege.
- Append To
  - The scope of privilege is required for wherever the related record is owned.
  - So Append To is required for each record type to be linked to. And the scope must cover the location of the record to be linked to.

Assign

Assigning a record requires the Assign privilege against the record to be assigned in its original business unit. Assigning a record also requires Read privilege to where the record will be assigned to. This allows control to ensure that users can only move ownership of records to areas where they are authorized to act.



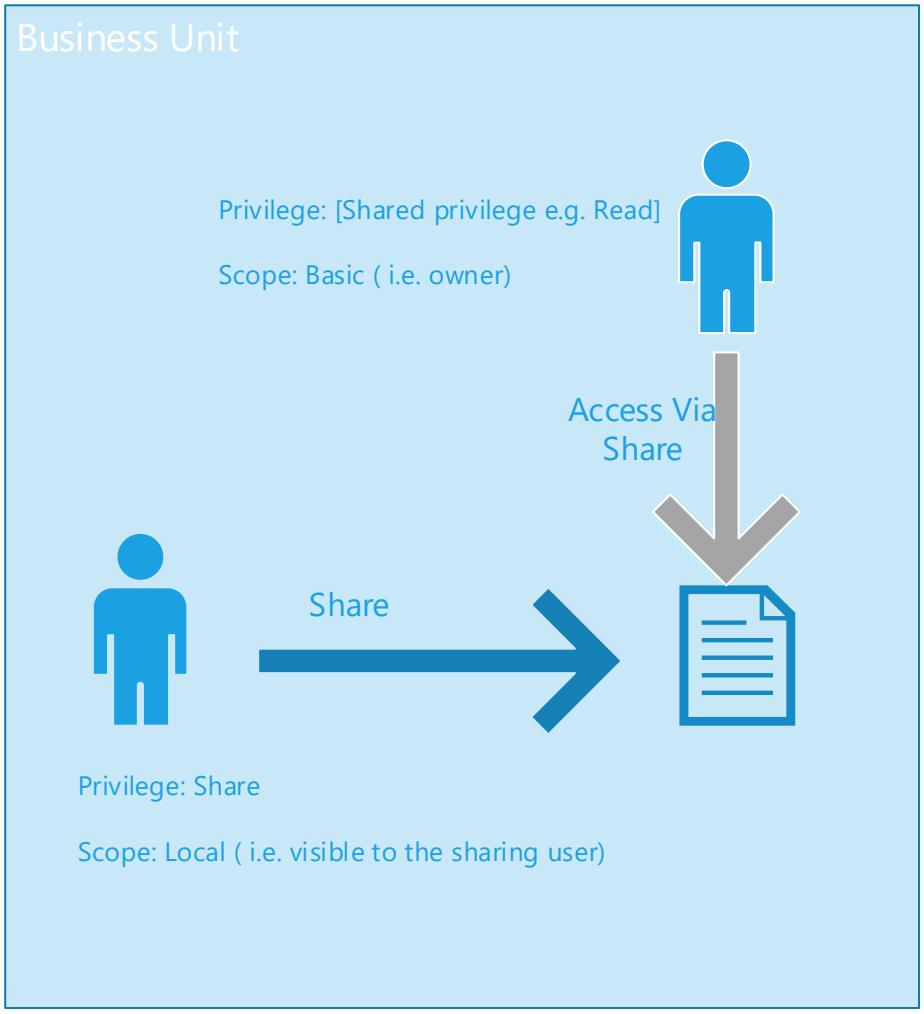
The specific privileges required as a minimum for both the assigning user and the new owner are:

- Assigning user
  - Assign privilege: scoped to access the current record. For example, Basic scope if owned by the user, Local scope if in the same business unit.
  - Read privilege: scoped to cover the destination locale of the record. For example:

        - Local if in the same business unit.
        - Through a team if in a different business unit.
- New owner
    - Read privilege.
    - Scope: at least Basic.

Sharing

Sharing a record requires the sharing user to be authorized to share and the target user or team to be authorized to receive the share. This again highlights the control available and consideration of privileges that need to be allowed for.



The specific privileges needed by the involved users are:

- Sharing user
    - Share: Need share privilege scoped to include the record to be shared. For example, Basic scope if it's a record the user owns, Local scope if in the same business unit.
    - A user can only share privileges they have to the record being shared. For example, a user can't share delete privileges if the sharing user can't perform that action on the record.
- Receiving user or team

- A share becomes active only if the receiving user has at least Basic level privileges for that entity type.
- This privilege is directly checked against the security principal the sharing occurs to. So if the record is shared to a user, that user must have the equivalent Basic scoped privilege for that action type, such as Read.

There is also a less common scenario where activities are concerned and where sharing is automatically performed:

- When an activity is created, it is shared to both the creator and all participants to ensure that all the involved parties have access to this record even if someone else owns it.
- The Read privilege for the creating user is needed for this automatic share to succeed.
- Setting the DisableImplicitSharingOfCommunicationActivities option avoids this implicit share for the creator or the participants and removes the need for this share privilege. For more information about setting this option, see OrgDBOrgSettings tool for Microsoft Dynamics CRM.

Automation

There are cases where a design may call for an action requiring privileges that you don't want to grant to the requesting user.

By performing actions, such as an assign, as part of a workflow that is set to run as the owner of the workflow or in a plug-in that is registered to run as a more privileged user account, the privileges required by the end user can be reduced while still controlling access through the security model. In this case the additional privileges for the assign is required by the user who owns the workflow or that the plug-in is registered to run as.

This can be useful when assigning a record into a privilege business unit, such as one set to hold all VIP records, without granting the creating user any access to that business unit themselves.

Privileges summary

Action	Security principal	Privileges required
Create record in same BU	Creator	**Record:** Create, Read, Append (to view Create action on command bar or to connect to any other record)
Own a record	Owner	**Record:** Read
Create and assign to another principal	Creator	**Record:** Create, Read
	Owner	**Record:** Read
Create and assign to another principal in a different BU	Creator (direct user privileges)	None
	Creator Team (in different business unit)	**Record:** Create, Read

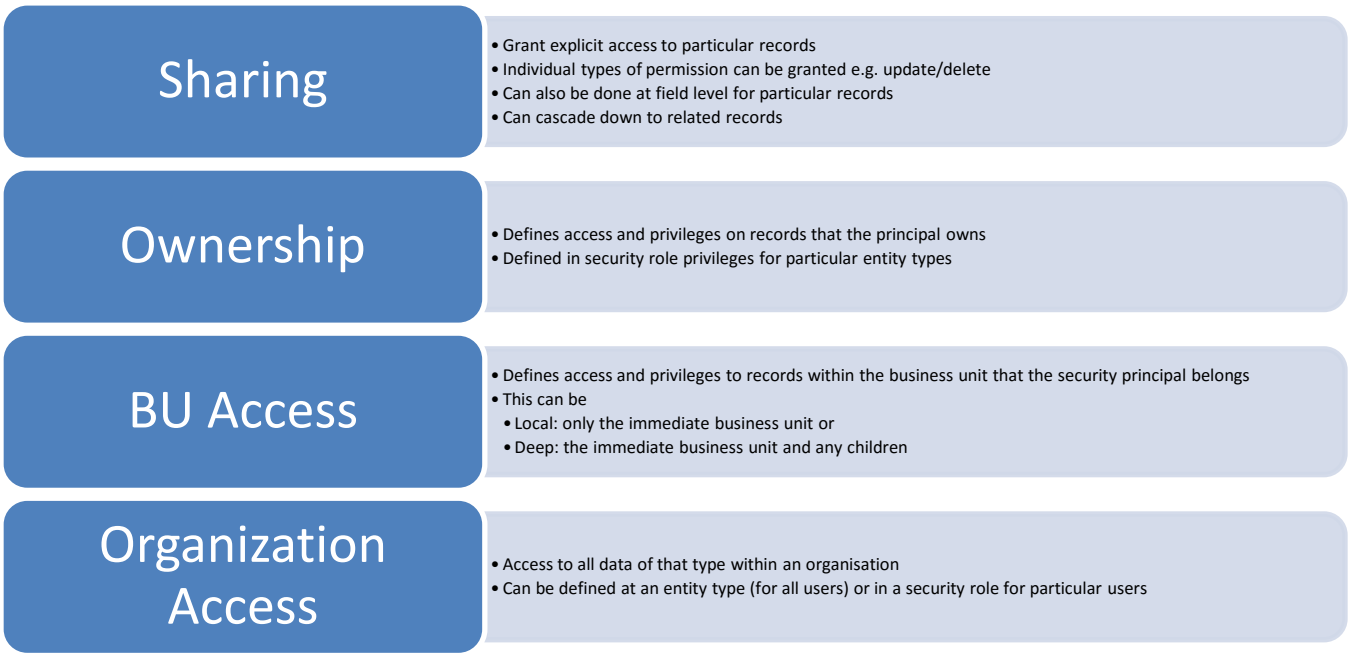
	Owner	**Record:** Read
Assign	Assignor	**Record**: Assign, Read: Target scope
	Assignee	Ownership privileges **Record Type**: Read (Basic scope or above)
Share	Sharing User	**Record:** Share
	Receiving User	**Record Type**: Shared privileges: Basic scope

Special privileges required

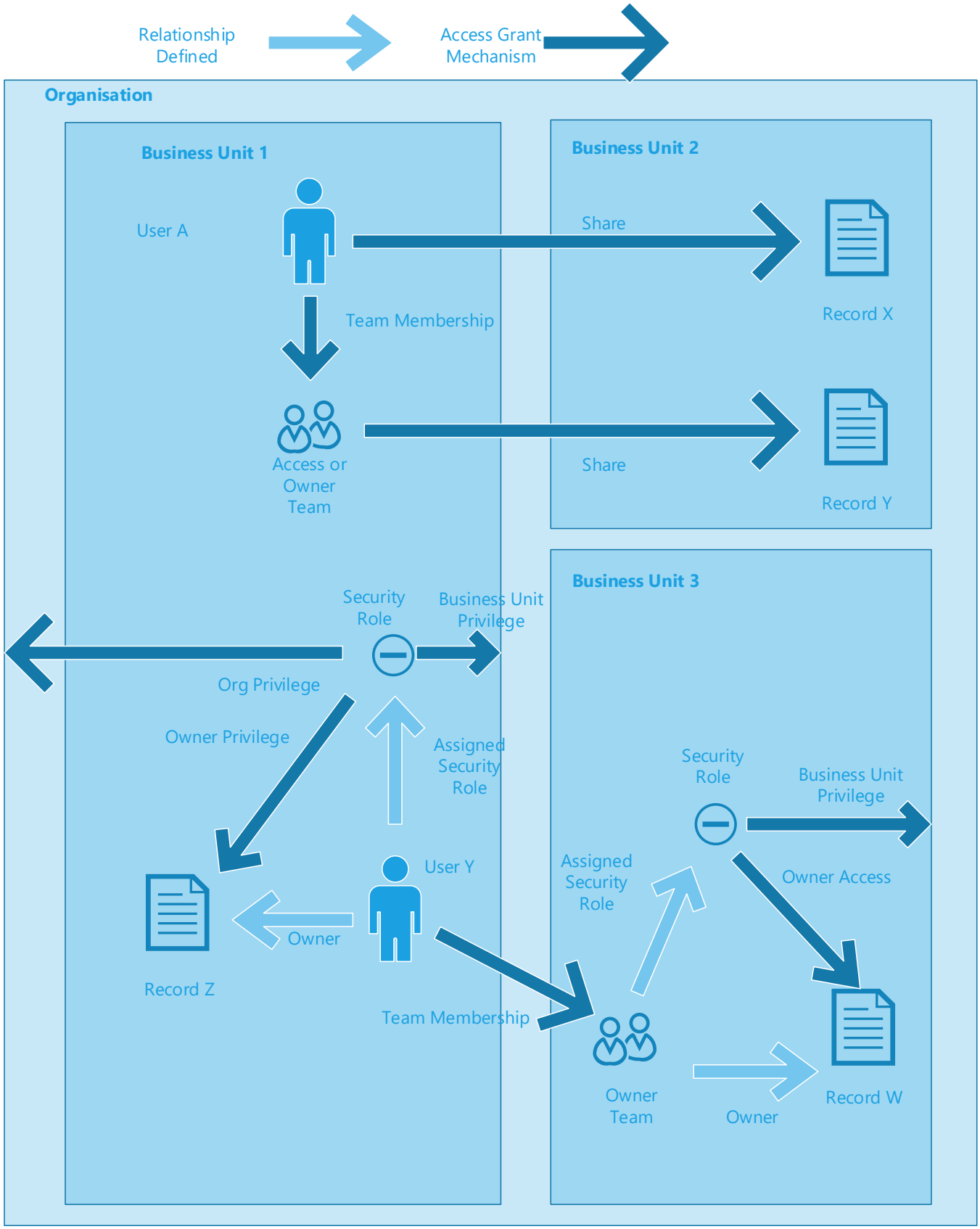
Action	Security principal	Privileges required
Link to another record	Creating User	Record: Append Linked Record: AppendTo
Activities Create, special case	Creating User	Record: Share, Write

Access control mechanisms

There are also a number of access control mechanisms that apply to these scope levels:



The following diagram provides an overview of the different access mechanisms, each will be covered in turn in the following sections.


Relationship Defined
Access Grant Mechanism
Organisation
Business Unit 1
User A
Team Membership
Access or Owner Team
Business Unit 2
Share
Record X
Share
Record Y
Business Unit 3
Security Role
Business Unit Privilege
Org Privilege
Owner Privilege
Assigned Security Role
User Y
Owner
Record Z
Team Membership
Security Role
Business Unit Privilege
Assigned Security Role
Owner Access
Owner Team
Owner
Record W

Sharing

Sharing is the direct granting of privileges for a particular record to the people who should have access to it. This grant can specify the type of access and actions that are allowed for that record, such as read and update privileges.

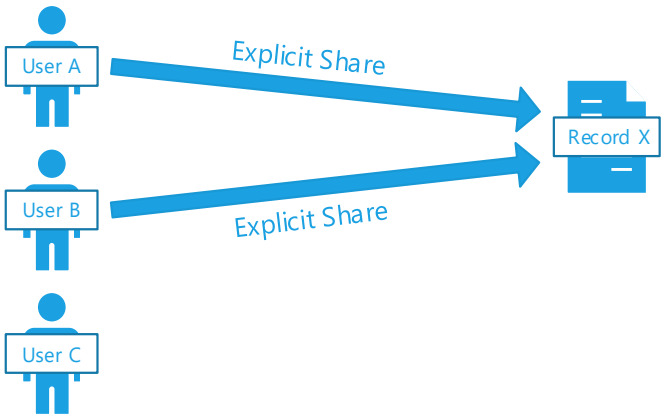
Notice that sharing a record with a user or team doesn‘t by itself allow access to that record. To make sure that inadvertent access can’t be granted through sharing to types of information a user shouldn’t be able to see, such as sharing details of a fraud investigation with a front line call center agent, the user or team being shared with needs a security role granting at least Basic (Owner) Read level privileges to that type of entity for the share to actually grant access to the record.

This privilege is required on the security principal being shared with, so if the user is being shared with then the user needs the Read privilege and if it is a team being shared with then the team needs the Read privilege.

The Basic Read level privilege grants the right to see the type of record (and by implication any records the user or team directly owns), and the individual sharing grants access to particular records of that entity type.

Sharing with users

One approach to providing individual-level access control is the extensive use of sharing of individual records with the specific users who need access. This access model requires explicit sharing of each record with each user who requires access.

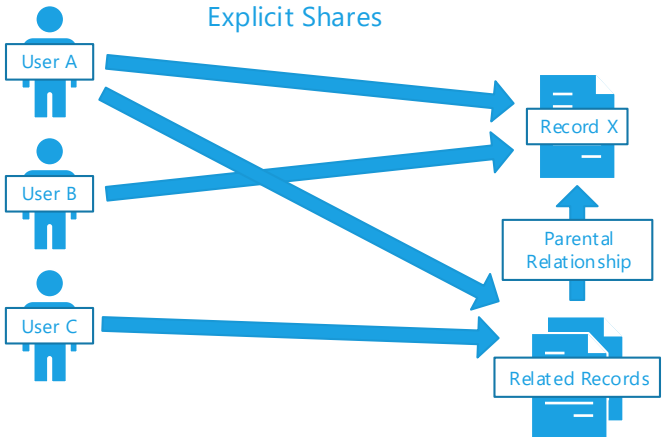


This approach works well for smaller volumes such as in smaller implementations or situations in which sharing is used to define exceptional circumstances. In larger implementations, most scenarios are handled through a more general policy, but occasionally there are specific cases that don’t follow the rules. For example, if a CEO were to take particular interest in a case because of a personal involvement, or because a particular specialist is called in to support a complex issue. In these cases, sharing can be used as a mechanism that allows for modeling these exceptional involvements without changing the overall approach for more general cases. In these situations, the infrequent nature of exceptional cases allows for defining fewer sharing rules even though the overall volume of cases may be high, which would limit the performance impact of sharing to an acceptable level given the granular control it provides.

However, as the number of database records grows, the use of explicit sharing to manage user access to data becomes an increasingly untenable approach. Using extensive sharing at high volumes has a significant performance impact because whenever a user is accessing a series of records, complex database queries result to determine the individual sharing rules that apply and must be checked before allowing access.

But there is something additional to consider. Commonly, this sort of sharing occurs at the level of a case or deal, each of which is typically linked to other related records, such as activities. Dynamics CRM includes a feature that provides the ability to “cascade” or copy changes such as ownership or sharing of a parent record to its children records, thereby maintaining consistency across the related collections of records without having to

manually apply the same change to each. As a result, for situations in which sharing is set to cascade from a parent to its children, database growth can accelerate rapidly. This occurs because the sharing records for each user are also duplicated for each child record as the sharing request is cascaded from parent to child records.



As mentioned earlier, activities have some special scenarios related to sharing:

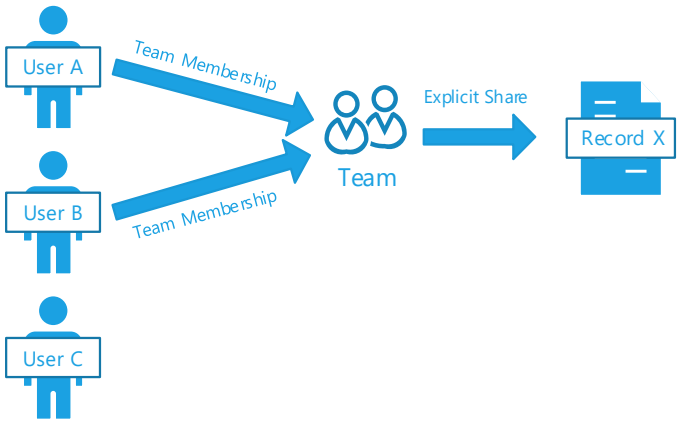
- An activity is automatically shared to both the creator and any users who are participants in the activity.
- In order for the creation to occur, the creating user needs Basic level Read privilege for the creating user for the share to work. The creation is blocked if the creating user doesn't have that privilege.
- Setting the OrgSetting DisableImplicitSharingOfCommunicationActivities option avoids this implicit share and the need for this privilege.

Sharing is implemented by creating explicitly in the database records noting the shares of users or teams to specific records with their specific sharing privileges and any inherited privileges through cascading. This level of granular approach can be valuable in managing user access to databases containing a limited number of records or in accommodating exceptional circumstances. However, using this approach also introduces several challenges, which are described in the following table.

Challenge	Description
Maintenance overhead	▪ Each time a user requires access to a record granted or removed, the administrator needs to share or unshare the record. As a result, administrators need to know exactly which information is related to a particular case or opportunity and therefore to update.
Customization	▪ Solutions that use explicit sharing to manage a database including a large number of records often require complex customizations to be set up to maintain the sharing rules.
Performance and scalability	▪ Each record shared will generate a sharing record linking that record and the individual that it is shared with to define that user's access to the information. ▪ That also typically applies to child records. ▪ An increase in the number of individual users or the records that need to be shared with those users can lead to a rapid increase in the volume of access records. This in turn can result in performance and scalability challenges when the explicit shares need to be updated and whenever a user accesses the record and the shares need to be checked.

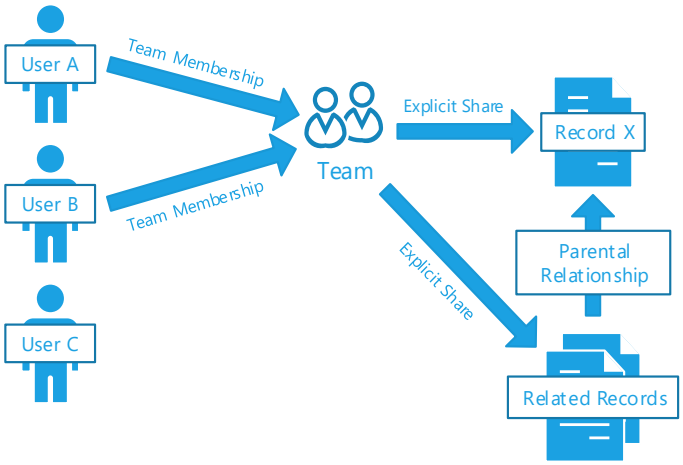
Sharing with teams

It is more efficient to use explicit sharing to manage user access to data by taking advantage of teams in Dynamics CRM. Creating a team for each group of users that have common access needs to records, such as covering a business area or particular deal types, reduces the number of shares required to grant access.



Using teams to provide users with access to records provides a number of benefits, including:

- Reducing the level of effort required to manage changes for the users covering each area.
- Limiting the number of sharing records that are created thereby reducing database storage needs.



While using this approach will reduce the overhead associated with extensive use of explicit sharing, it still requires that a sharing record be created for each record and that each record have an owner, which defines the business unit where the record belongs. In addition, solutions that employ the extensive use of explicit sharing are impacted from a scalability standpoint by the overhead associated with calculating access to each record. This impact is detailed later in this document.

Access Teams

With Dynamics CRM 2013, access teams were introduced. Access teams provide two key benefits:

- Access teams are lightweight teams aimed at high volume sharing scenarios.
- Automated creation and management of access teams and their team membership. This simplifies the administration process. For example, a scenario where individual teams are defined to manage a particular customer.

Access teams are enabled for particular entity types and a subgrid is used to manage the team membership directly in the record form. As users are added to the team through the subgrid, the system automatically creates the related team for the record and shares the record with the team based on the access team privilege template defined.

Access teams can also be created manually through the Teams view.

Where only sharing is used, access teams are an effective way to minimize the impact on the system at high volumes.

Record ownership

In Microsoft Dynamics CRM, there are many types of record ownership, as described in the following table.

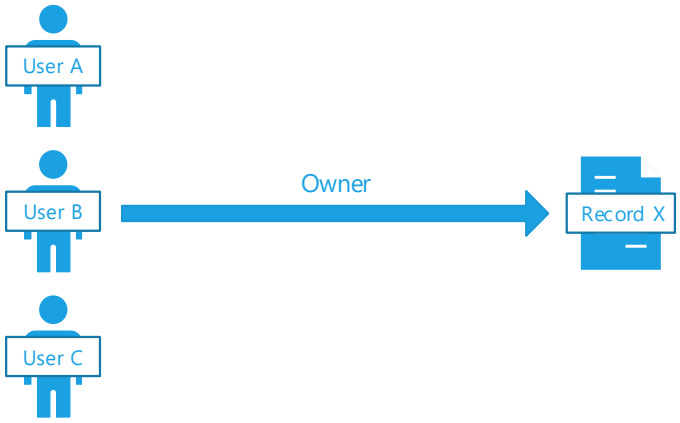
Ownership Type	
Organization owned	Contains data involving something that belongs to or that can be viewed by the whole organization. Organization-owned entities cannot be assigned or shared. For example, products are owned by the organization. These records have a field named **organizationid**.
Business owned	Entities that belong to a business unit. These records have a field named **owningbusinessunit**. For example, users and equipment are owned by the business unit.
User or team owned	Assigned to a user or team. These entities contain data that relates to customers, such as accounts or contacts. Security can be defined according to the business unit for the user or team. These records have fields named **owningteam** and **owninguser**. For team ownership, the team must be an owner team rather than an access team because access teams can't own records directly.
None	These entities are not owned by another entity but often have a parental relationship. For example, discount lists inherit the ownership from the parent discount record.

Note: While you can set the ownership for new entities that you add to the system, you can't change the ownership for pre-defined entities.

For the purposes of this discussion, we are concerned with records that are directly owned by a user or a team. Defining an entity type as organization-owned prohibits granularity of scope of access to records of that type; users have privileges to all or none of the records of that type. On the other hand, defining an entity type as user- or team-owned requires that each record be granted a specific owner, either a particular user or team.

User ownership

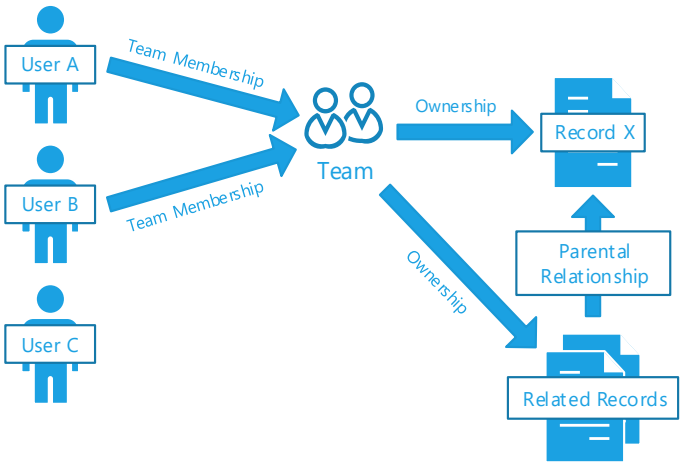
With user ownership, a particular user is recorded against a record as the owner of the record.



The user ownership model provides the granularity necessary to grant access to specific records by using security roles to define privileges that ensure that a user sees only the records that are owned by that user. For businesses in which individuals work independently on specific deals or on customer collaborations, this is a very effective and efficient model. For scenarios where multiple people need to interact with specific deals or records, this is not a viable solution. The record's business unit is derived from the owner. Therefore, an additional challenge comes when a user owning records moves roles within an organization. This potentially forces the ownership of those records to change allowing access that any other user may gain via the business unit of those records.

Team ownership

Using team ownership introduces a solution pattern that can simplify many of the challenges associated with situations where users directly own records. Instead of sharing individual records with specific users or teams, you can grant ownership of specific records to a team, which allows multiple users to gain access directly to the record through ownership. This also reduces the changes that are required when particular users need to have access granted or removed from a set of related records, which is now accomplished by adding them or removing them from the owning team rather than each record individually.



In this scenario, with records assigned directly to a team, records can be aligned with a business unit independently of the business unit to which the users accessing it belong. This then provides for access via security roles and business unit privileges, which allows the definition of rich access mechanisms.

Scenarios that require individual records to be accessed by a specific group of individuals allow for a model in which ownership is used to define the group of individuals who should be granted access. This is done by specifying a team that is allowed to view only the records it owns and then by defining the users who should have access to that record as members of the team.

Although less common, there are also situations in which groups of users act on common types of deals or cases. Having a team that owns multiple records can provide significant benefits over sharing.

It is important to remember, however, that with any modeling of direct access, whether through ownership or sharing, each record must be individually configured on creation or updated to apply the security rules to it. In addition, when access is evaluated, each team to which a user belongs must be checked for individual access, which increases the complexity of the security processing from a data and computational perspective.

Team ownership is limited to a single team owner per record, however, so that it can't be used to provide different levels of privilege to different groups of uses to the same record. Using a combination of ownership to provide general access and sharing to offer additional privileges to smaller groups of users can work well here. However, reducing the overall volume of sharing and providing an overall performance benefit.

Owner teams

To own a record, a team must be an owner team rather than an access team. Access teams are optimized as a lightweight mechanism that can't have security roles. In order to have the rights to own a record, a team must have the appropriate privileges and an access team that can't have a security role won't have those.

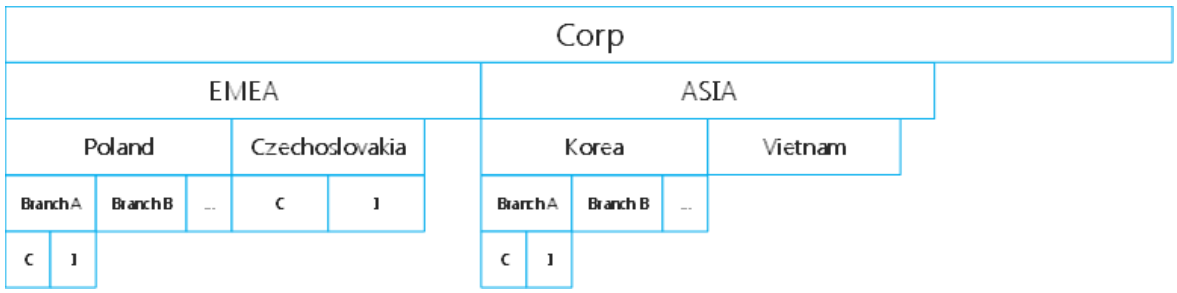
In the following sections, anywhere a team is specified that has ownership or derives access to records through security roles rather than sharing, it's referred to as an *owner team*.

Business unit privileges

As mentioned, implementing direct access controls at large volumes poses some key challenges, especially:

- Maintainability, in terms of setting up and managing access rules to each individual record.
- Scalability, in terms of adding significant processing overhead to each request for situations when individuals access a large number of records but individual access is defined.

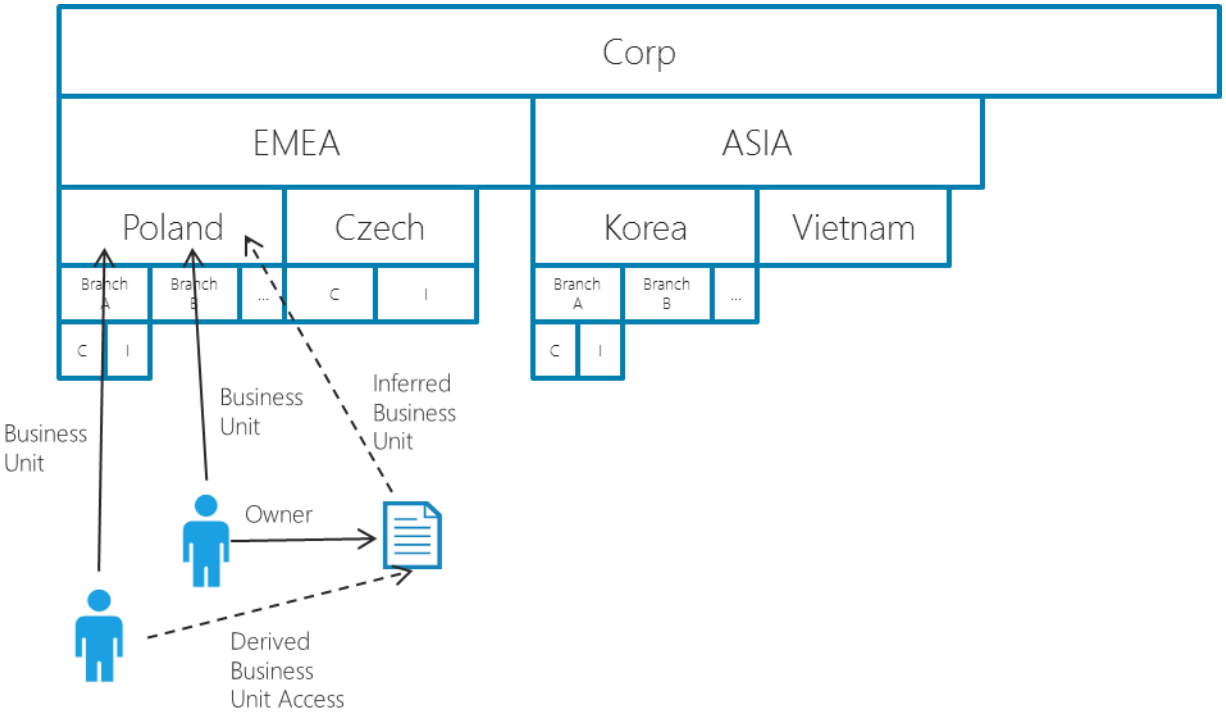
To address these challenges at scale, Dynamics CRM offers the ability to implement business units, a hierarchical structure of organizational areas. Business units manage access to large groups of records for situations when a set of records is defined as a group so that collective access is granted to multiple users. The following illustration shows how business units might be organized.



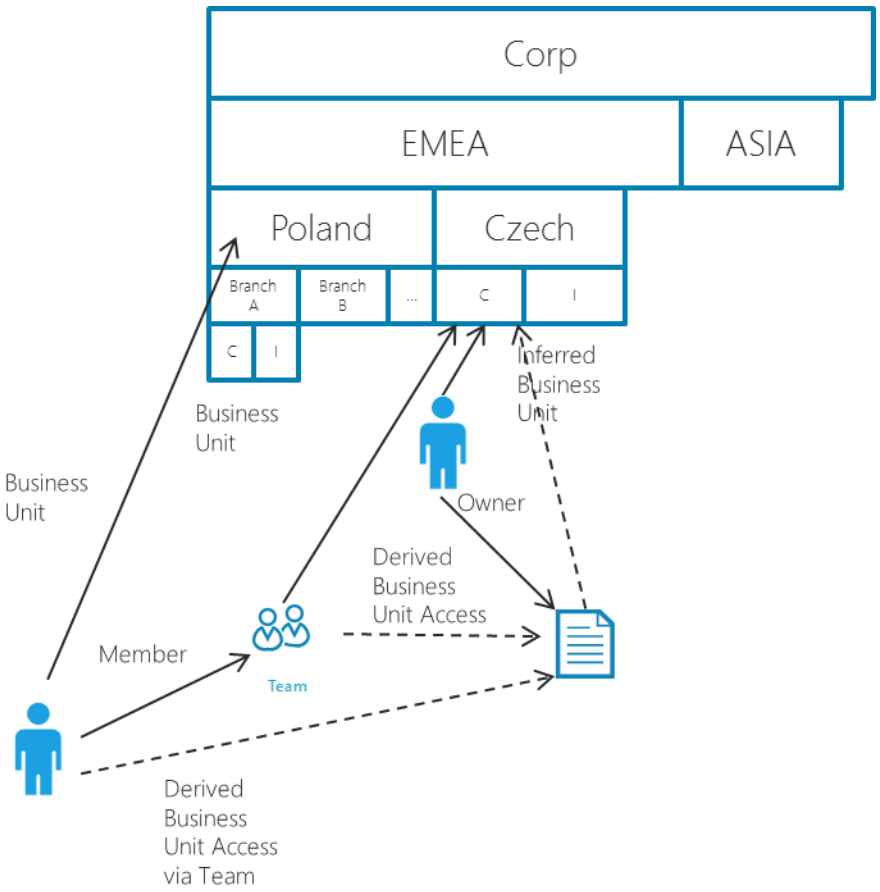
When a record is assigned an owner, either a user or a team, the record inherits the owner's business unit.

Business unit local privileges

Users or teams are assigned security roles that grant privileges at different levels, one of which is the "Business Unit" level. When privileges are granted at this level, the user or team is given the privilege (such as Read or Assign) for any records that are in the same business unit as the requesting user or team. For example, as shown in the following illustration, a user in Poland who has been given read access to any contact records in the same business unit can see any contact record owned by any other Polish user or team. As new records are created or existing records are reassigned to an owner in the Polish business unit, access is automatically be gained to these records by any other Polish user or team.



When users potentially need to access records in different areas of the business but not based on any hierarchical pattern, access can be granted by the combined use of team membership and business unit access. For example, if a user is only able to see the Polish records but should also be able to see items managed in the Czech Branch C, it would be possible to add the user to the team created in the business unit of Czech Branch C. If that business unit team is granted business unit-only level privileges, the user can see records in either the user's own Polish business unit or the business unit of the team. This access occurs without any relationship between the two business units.



Note: Though default teams are created within each business unit, they're designed to provide ownership of records by a business unit and their membership can't be amended. However, additional teams targeting different user groups can be created within the business units as needed.

Definition of access by business unit can provide significant optimization benefits to the performance of access checks, particularly when large groups of users need access to specific records. Additional detail about these benefits is provided in the section Business unit access checks.

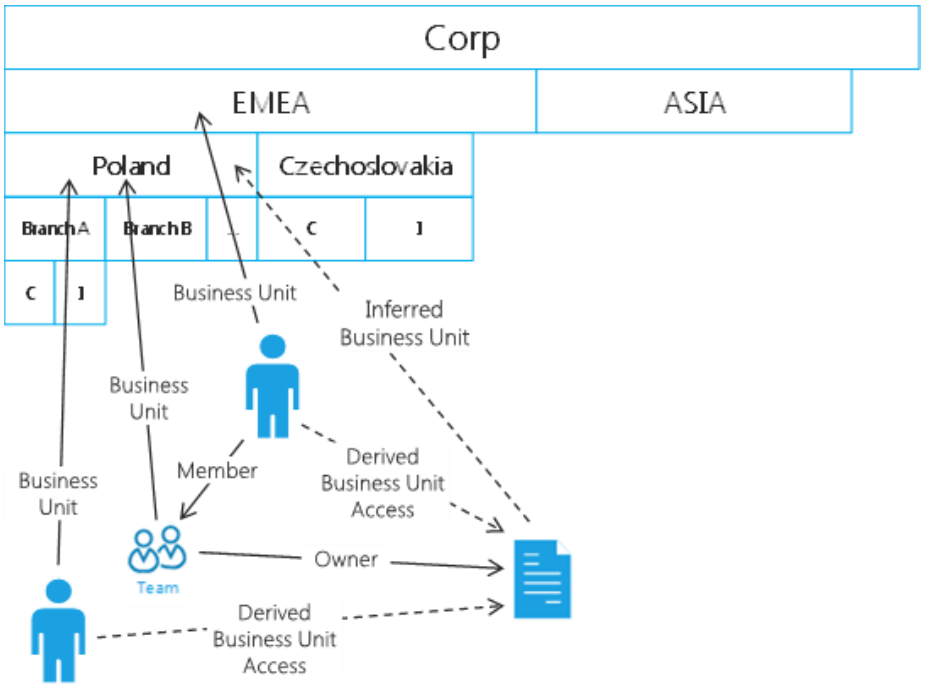
Business unit deep privileges

In addition to being able to define access within the boundaries of a particular Business Unit, it is also possible to grant users access to records in their Business Unit or child Business Units. This can be useful in hierarchical scenarios, where one user works in a particular country/region but other users cover multiple countries. In this scenario, a user who works across EMEA could see records owned by any of the subsidiary Business Units.

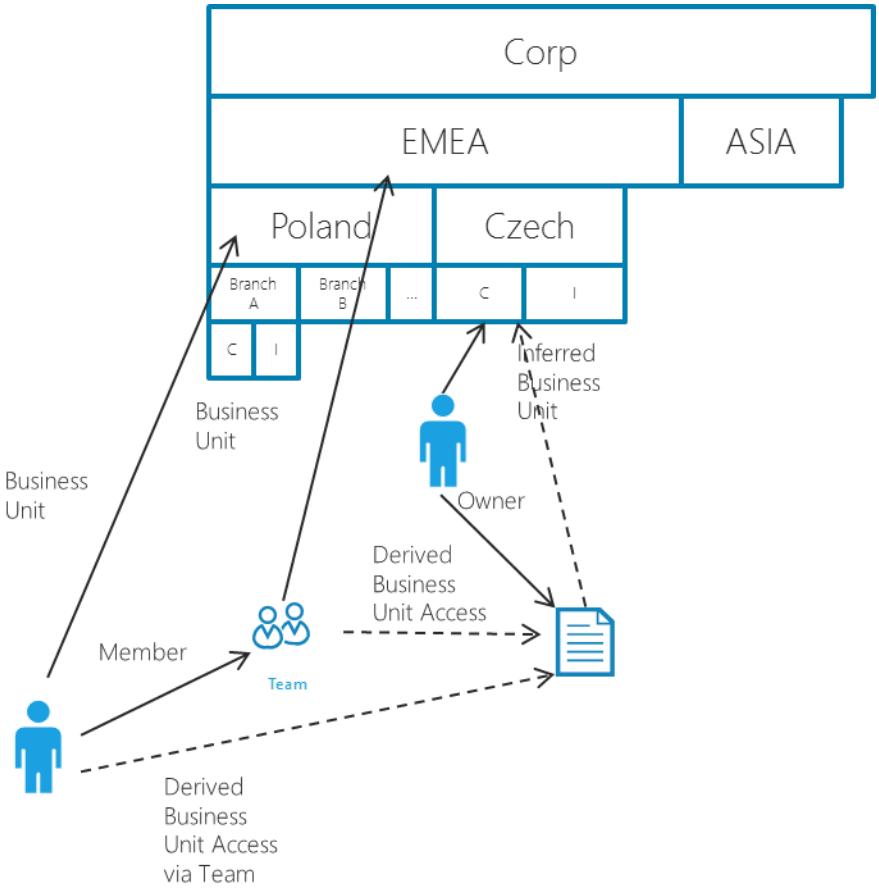
A common challenge encountered with this model is the need to accommodate users at the regional level who interact directly with accounts while other users need access bounded at a local level. To provide users regional access by taking advantage of user business unit modeling, the users must be located within the regional business unit, for example, EMEA, and be given deep privileges. However, that locates records those users own with the regional business unit, making those records outside of the scope of access to the Polish users.

In this scenario, team ownership can effectively solve this problem in one of two ways:

- Locate the user at the regional level but assign records to the team representing access to the Polish business unit rather than directly to the user.



- Locate the user in their local country/region business unit but add them to a team located in the regional level. This grants those users privileges to records owned within the region or through deep privileges to any child business units.



Organization privileges

A further option is available for records that don't need granularity of scope of access. For a scenario where all users either have or do not have access to a type of record, creating an entity type as organization-owned can simplify the management of record instances, which will not require having an owner defined. This approach also has significant performance benefits when records are accessed, because individual access checks are not required other than to verify that the user or team is allowed access to this type of record.

Hierarchies

With Dynamics CRM, security hierarchies was introduced, which layers on top of the existing capabilities. Hierarchies provide the ability for a person to inherit privileges from users they manage or otherwise have a hierarchical relationship over.

In previous versions of Dynamics CRM, this hierarchical security relationship had to be implemented through customizations.

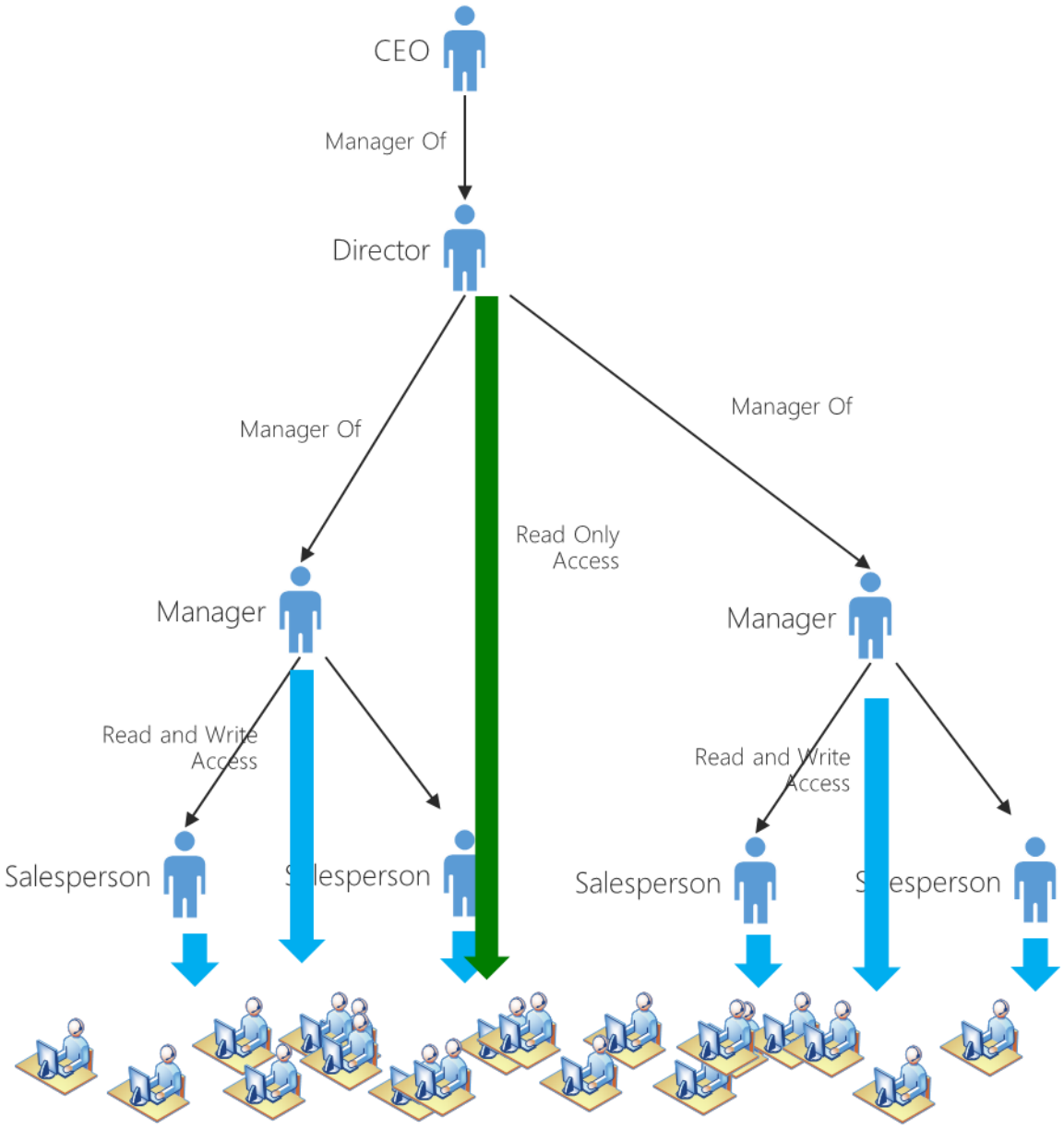
Security hierarchies give the ability to:

- Automatically roll up privileges to managers.
- Inherit privileges through multiple levels.

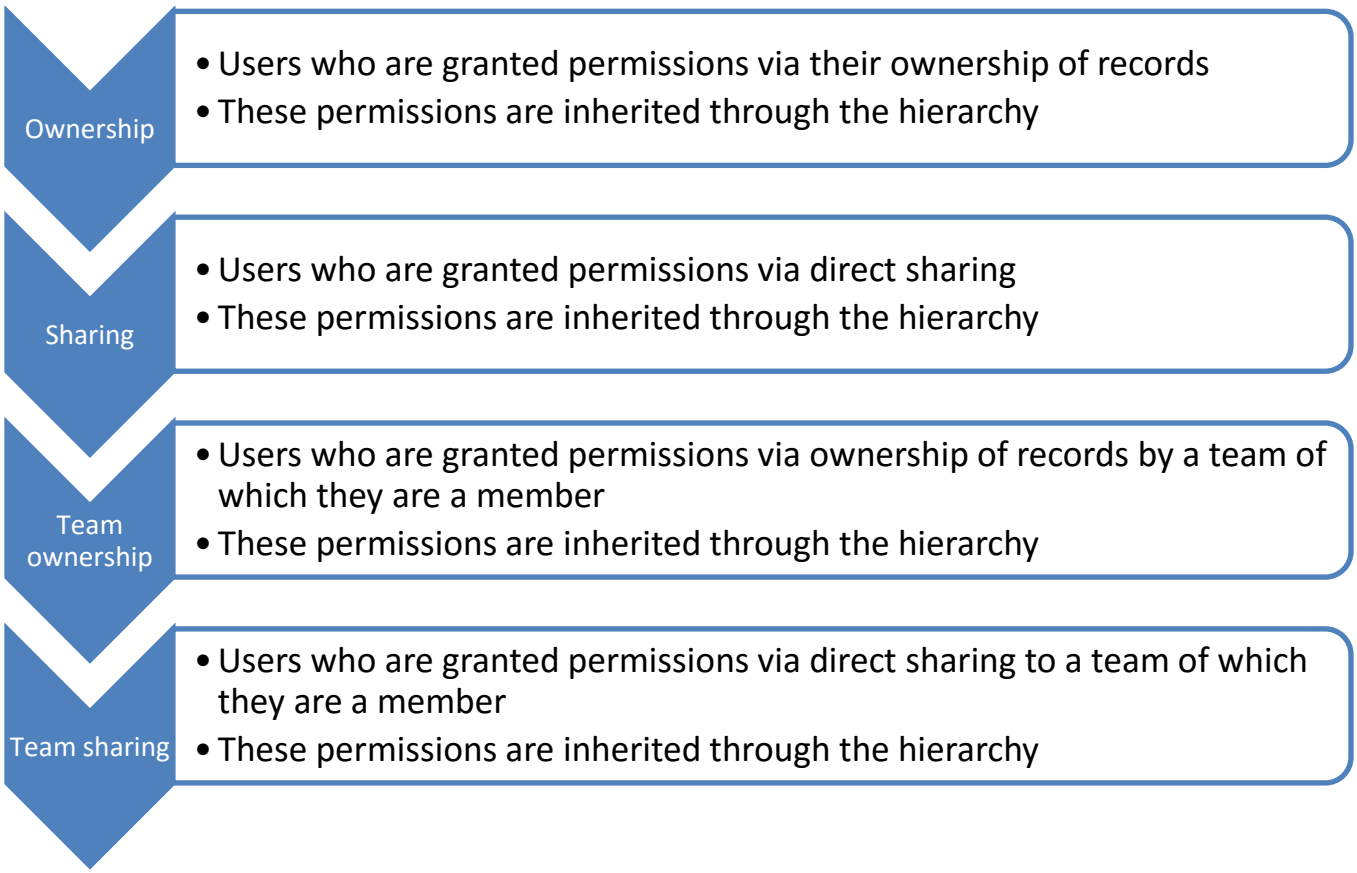
The level of privilege that is granted through the hierarchy depends how many levels in the hierarchy separate the person granted the original privilege and the inheriting manager.

- Direct manager:
    - The direct manager is granted the privileges to read and interact with records their direct reports can.
    - This lets the direct manager act on behalf of their direct reports and interact with data. For example, to cover for them in the direct report's absence.
- 2nd level manager and above:
    - Further up the management hierarchy only read privileges are inherited, giving visibility and oversight to activity of their indirect reports but not the ability to act as them.
- With this mechanism only configuration is needed. This reduces the need for customization to automate and maintain the inherited privileges.

As shown here, without direct access to records, managers in a hierarchy above a user are granted access to the same pool of records that their reports have access to.



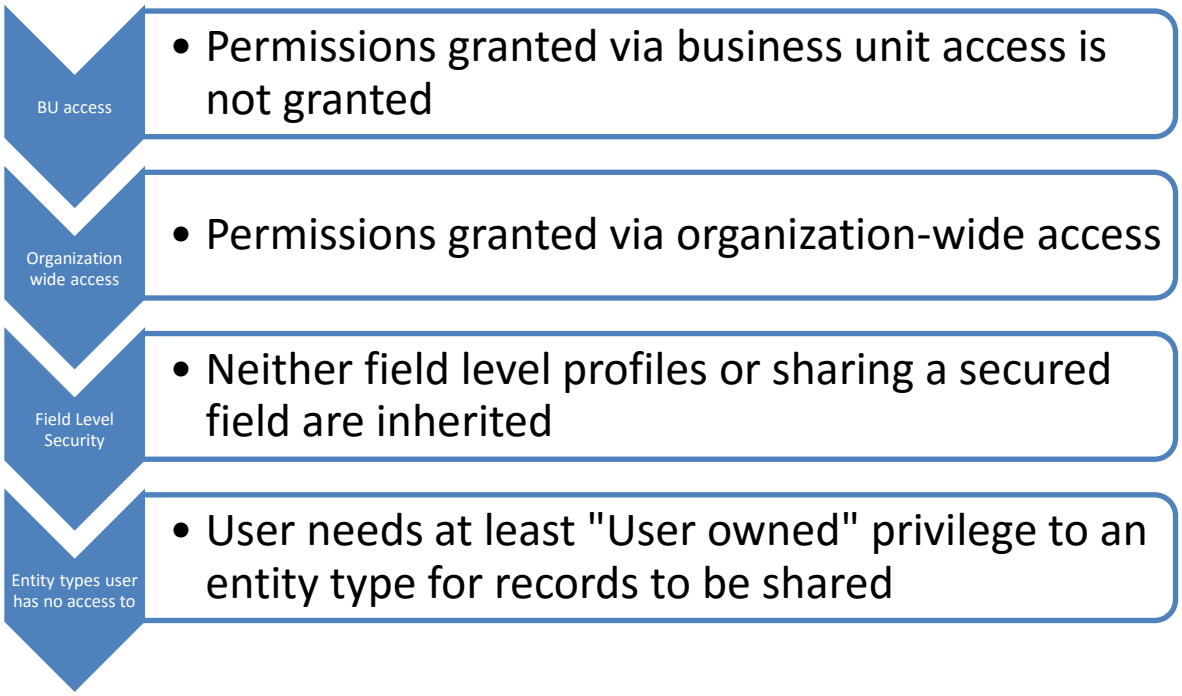
It is important to note that the exact same privileges a user has aren't inherited by their managers. At a high level, what is inherited is items that an individual can access personally. This is more specifically defined as:



The specific privileges that are inherited through the hierarchies are:

- Direct manager:
    - Read, Write, Update, Append, Append To, Share access is granted to the direct parent of the report.
- Higher manager levels:
    - Read access only.

The intention of hierarchies is to share individual level permissions so broader scope permissions, such as business unit and organization scope privileges aren't inherited. Users would likely have this access from their own security roles and business unit privileges anyway. Specific privileges rights that are not inherited are:

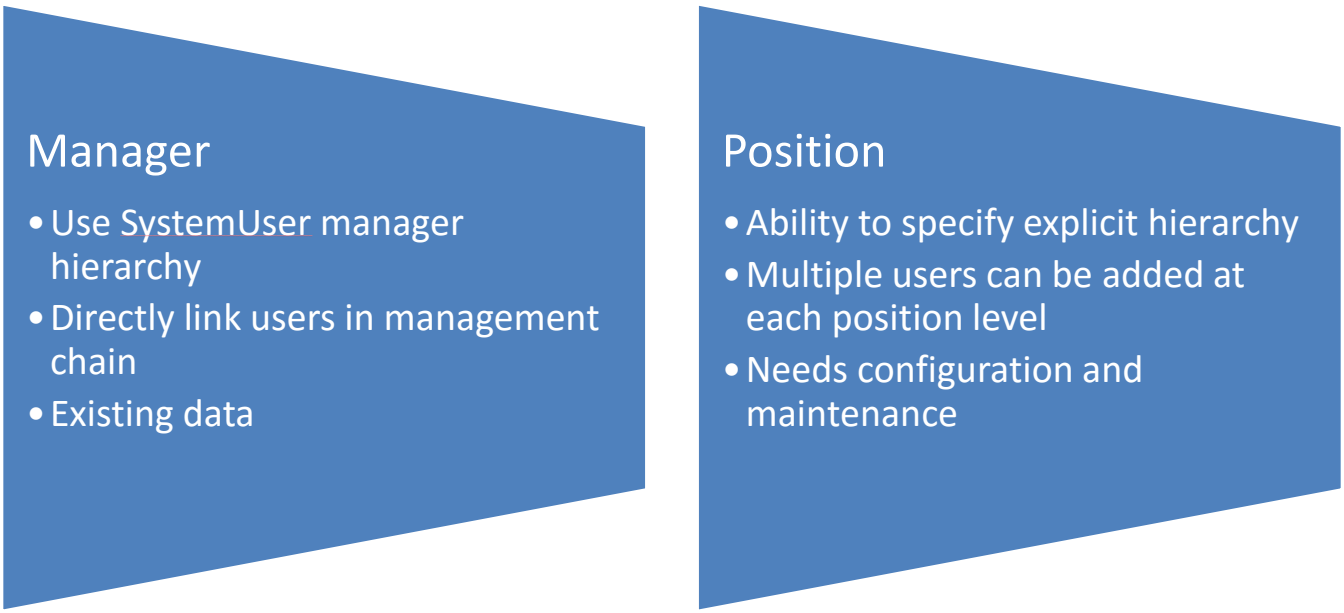


Hierarchy levels

It is possible to specify how many levels of the hierarchy inherit read access from a particular user who is directly granted access to a record. This is intended for those in direct management chain with need for detailed access to specific records and therefore is not intended to be replacement for aggregated view and reporting at higher levels where BI/BU Scope is more efficient and typically more appropriate.

Hierarchy types

There are two ways in which the hierarchy can be defined, either by using the standard manager hierarchy, or by using a more tailored position structure. The implications of each of these models are:



Access control to fields

In some scenarios, only access to certain customer fields must be locked down, rather than restricting access to a customer as a whole. In these situations, there are a number of approaches to consider.

- Break out the secure information onto separate entity types. For example, an "Annual Account" record or an annual "Account Plan" may be a valid approach. This provides the ability to secure the details of a customer separately using the mechanisms described previously but at a more granular level, possibly simplifying the access approach. For example, all users of a typical role can view this type of data for all accounts.
- Implement Field-Level Security, which is a finer-tuned approach to providing data access than entity level controls. An organization can implement Field-Level Security using one of two alternative models:
    - Users or teams can be assigned Field Security Profiles. Then, the fields that a user can access can be defined across all instances of a particular entity type. This can be a good mechanism for scenarios when the type of access control needed is universal.
    - Sharing fields, which combines sharing and field-level security. This can be implemented to allow for sharing individual fields from individual records, with specific permissions in terms of read or update access. For example, sharing fields to particular users or teams. Implementing this granularity of access, however, incurs similar performance impacts with the sharing of the records themselves. However, using this approach can provide benefits for scenarios where controlling access only to selected fields can reduce the volume of restrictions required. Thus enabling the application of a more efficient mechanism to the broader volume of records.

Field-Level Security in Dynamics CRM is extended from custom entities and attributes to include system entities and attributes. The intent is to offer Field-Level Security support for all fields for which security is applicable. Field-Level Security is not universally available for all fields, however, and there are good reasons why it is not available for certain fields or entity types:

- It is not applicable in:
    - Record primary identifier GUID fields, such as. Accountid:

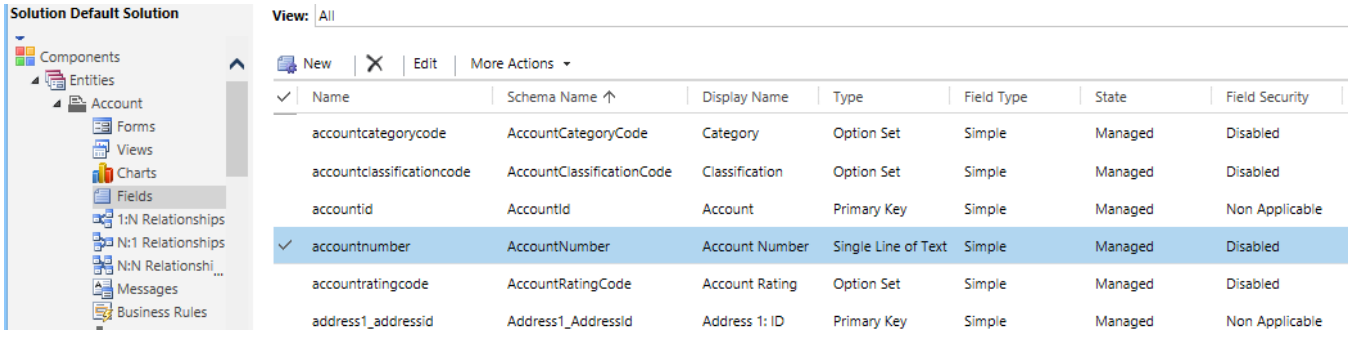
        - No real information is provided as the GUID is in and of itself a meaningless value and purely a unique pointer to a record.
        - It is meaningless for a user to be able to access a record but not to access the record identifier. It would be impossible to reference the record to load in the first place.
    - Composite fields update and create:
        - Users can't update the combined fields directly, so update and create privileges on these fields aren't relevant.
        - The system prevents the source fields being updated but composite fields not being updated. This happens when we apply different privileges just to the composite fields, which causes inconsistency.
- There are functional dependencies:
    - In some areas we have functional dependencies between fields or entity types where blocking some functionality but not all the related functionality. This can cause the system to break.
    - A key area this relates to is price lists and discounts. Allowing users to only access some of these could cause inconsistencies and errors. As a result there are certain areas that are not enabled for Field-Level Security.

But if Field-Level Security is not universally applicable, it is important to be able to determine if it is available for certain entities or attributes. This is defined in the entity metadata which can be accessed in multiple ways.

In the SDK documentation the EntityMetadata.xls file is shipped which contains the values for all the standard system entities for this. An example of this format is shown in this table.

SchemaName	Type	CanBeSecuredForCreate	CanBeSecuredForRead	CanBeSecuredForUpdate
AccountCategoryCode	Picklist	True	True	True
AccountClassificationCode	Picklist	True	True	True
AccountId	Uniqueidentifier	False	False	False
AccountNumber	String	True	True	True

The other place Field-Level Security can be viewed is from within the customization views within the Dynamics CRM application itself.



Scalability characteristics of Microsoft Dynamics CRM elements

As mentioned, Dynamics CRM offers a wide range of security modeling features that provide the right balance of flexibility and granularity to be implemented to meet the needs of a particular scenario. When attempting to determine the most appropriate way to model security in an implementation, it is important to have a clear understanding of the overall functionality of the solution to better evaluate the potential implications.

Note: Rather than offering a comprehensive review of the details associated with implementing each security model, this section summarizes the approach used for each model so that its scalability can be appreciated. Note that this information is subject to change in future releases as additional optimizations and features are added to the system.

The primary scenarios to consider as part of general access modeling include:

- Accessing the system initially.
- Accessing a single record.
- Accessing a view or retrieval of multiple records.

These scenarios potentially impact the scalability of security modeling in a number of ways depending on which security capabilities are leveraged. The following sections describe:

- How initial access is impacted by caching of user and security access items.
- How each of the different security access mechanisms work and balance granularity of control with performance and scalability, specifically:
    - **Sharing**, at the individual and team level.
    - **Ownership**, at the individual and team level.
    - **Business Unit**, at the level of Local and Deep privileges.
    - Organization ownership.

Privilege types

Though this section focuses on read access privileges, read access is applicable to the broadest set of scenarios. The characteristics described are also true for all other privilege types. In practice, the other privileges typically apply only during access of an individual record. For example, for a situation in which an update/assign/delete action is performed, even when performed on multiple records from a grid, the action would be applied on an individual basis to each record. As a result, the privileges are checked on an individual basis as described for read access to individual records below. Only when performing reads will multiple records genuinely be queried and acted upon as a group at a technical level. As a result, the approach for accessing multiple records concurrently, as described later, only applies to read privileges.

SQL View access

The subsequent sections focus on the way that application servers apply security principals. As well as the more commonly considered access to information via the web services, to ensure completeness of the security model it is necessary to also consider querying data directly from the database server. This can be via Filtered Views in SQL Server, which applies only in an on-premises implementation, and with functionality provided as part of Microsoft Dynamics CRM for Outlook that allows querying via the filtered views on the local offline database.

Though the technical implementation is slightly different, with the query being defined as part of the view definition rather than by code in the application server, the principles of access are the same as from the application server. As a result, this paper doesn't cover the topic separately.

Initial access: caching

To optimize many processes in Dynamics CRM, a number of items are cached, which can have an impact on security modeling optimization. This is especially with regards to:

- **Metadata**: the definition of entities including whether the entity is organization owned.
- **User**: security roles and business unit of the user.
- **Teams**: security roles and business unit of owner teams.
- **Team memberships**: the owner teams a user is a member of.

When this information is first needed, each application server caches the detail locally in memory. Caching this information optimizes subsequent access to the data within the system.

Note that particularly for user access, requests from a specific user that are load balanced across multiple web servers require that each web server load its cache for access to that user's details. This can have a number of consequences for performance and scalability. As a result, be sure to keep in mind the following considerations.

- For scenarios in which the amount of data held about particular users is high, for example situations in which users are members of several owner teams:
    - Loading that information can impact the user's initial request to each Dynamics CRM application server that occurs after the process starts.
    - The data must be loaded and cached multiple times. Each application server will host at least one worker process and each worker process will hold its own cache, which can impact the overall scalability of a solution.
- A significant number of users signing in at the same time can impact overall scalability as well as initial access times for individual users.
- For scenarios in which owner team memberships or user details are changed on a regularly basis, the application servers will be notified with each change to flush the related information from the cache.
    - The impact occurs during the subsequent request for the data, which must be reloaded into the cache. This potentially impacts both the request and overall server performance.
    - Making regular changes to user records. For example, recording a metric that tracks the last time the user accessed the system not only incurs the write impact to the record, but also triggers the application server caches to flush that user's information on each update. Thus, forcing it to be reloaded on the next request. This is a very expensive pattern that should be avoided if possible because of the resource implications it incurs as a result of the cache reloading.
    - This is another scenario where access teams can benefit. Regularly changing access team memberships has much lower system impact because they're not cached. They are primarily used directly within the database to calculate sharing and therefore don't benefit from being cached.

Initial user access caching

When a user initially accesses the system, the cache is loaded with certain elements of information for that user including both user information and related access information about teams, as follows:

- For the user:

- Details about the user.
- Security roles and therefore privileges for the user.
- Owner team memberships for that user.

- For owner teams that the user belongs to:
  - Security roles and therefore privileges for the team.

When a user initially accesses an application server, loading the cache for these values involves the following.

1. **Retrieve User:**
   a. Retrieve user details.
   b. Retrieve user's security roles.
   c. Retrieve user's team memberships.
2. **Retrieve Owner Teams** – For each owner team that a user belongs to:
   a. Retrieve team details.
   b. Retrieve security roles for team.
3. **Retrieve Security Roles** – For each security role which the user and any team which the user is a member of has been granted:
   a. Retrieve the parent root role for each role.
   b. Retrieve privileges from the role template for the parent role.

The system is therefore making multiple requests for each team membership; one to retrieve:

- Each owner team (if the team is not already cached).
- Each role per owner team.
- The parent root role per team role.

Cache flushing

Understanding when a user's cache entries can be flushed and reloaded can also be a significant factor in large system scalability.

The web servers will be notified to flush their cache of a user's details when one of the following events occurs:

- The system user record is updated or changes state.
- The security roles for a user change.
- A user is added or removed from an owner team.
- The security roles are changed for an owner team that a user is a member of.
- The user has been inactive, and hasn't made a request of the system for 20 minutes.

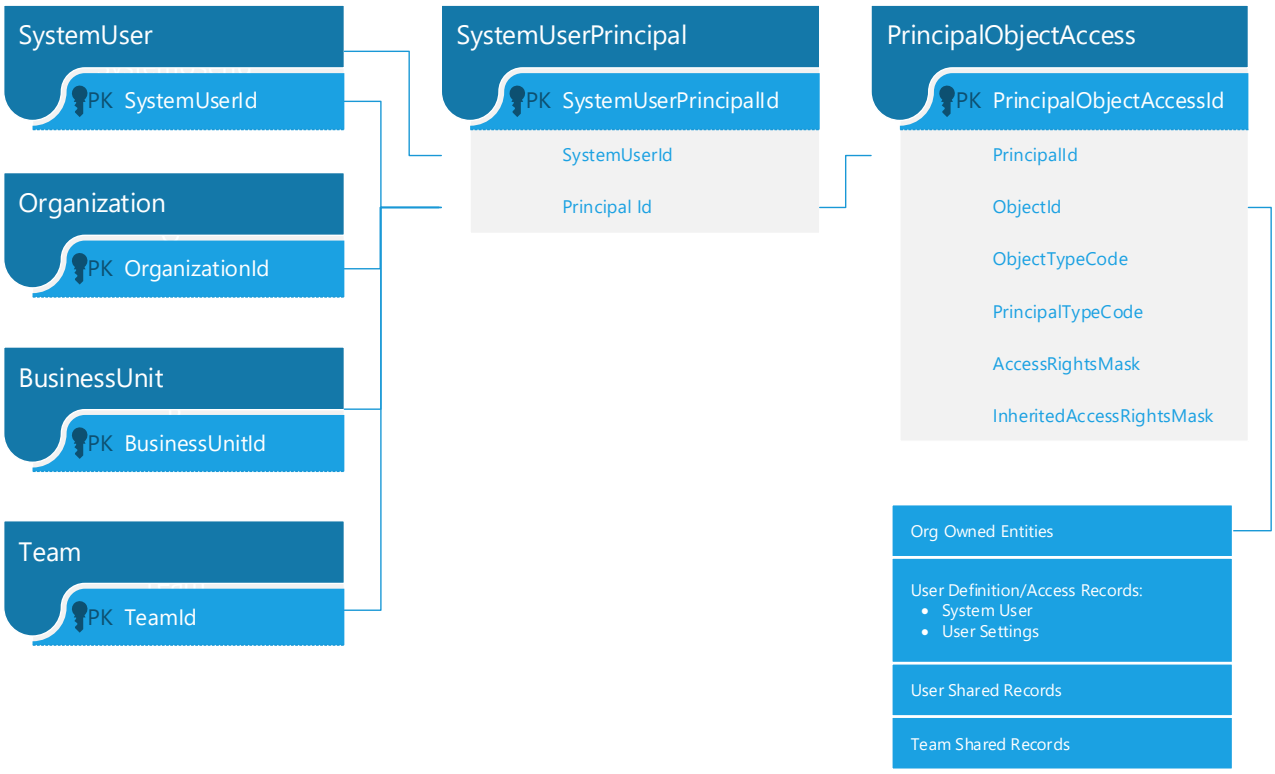
Designs that regularly update a system user record, for example to record the last request, will have a significant impact on system performance and scalability. This is because the cache will constantly be flushed and reloaded.

Sharing access checks

Understanding how sharing is implemented in Dynamics CRM is key to determining the scalability characteristics of the sharing mechanism.

Sharing records

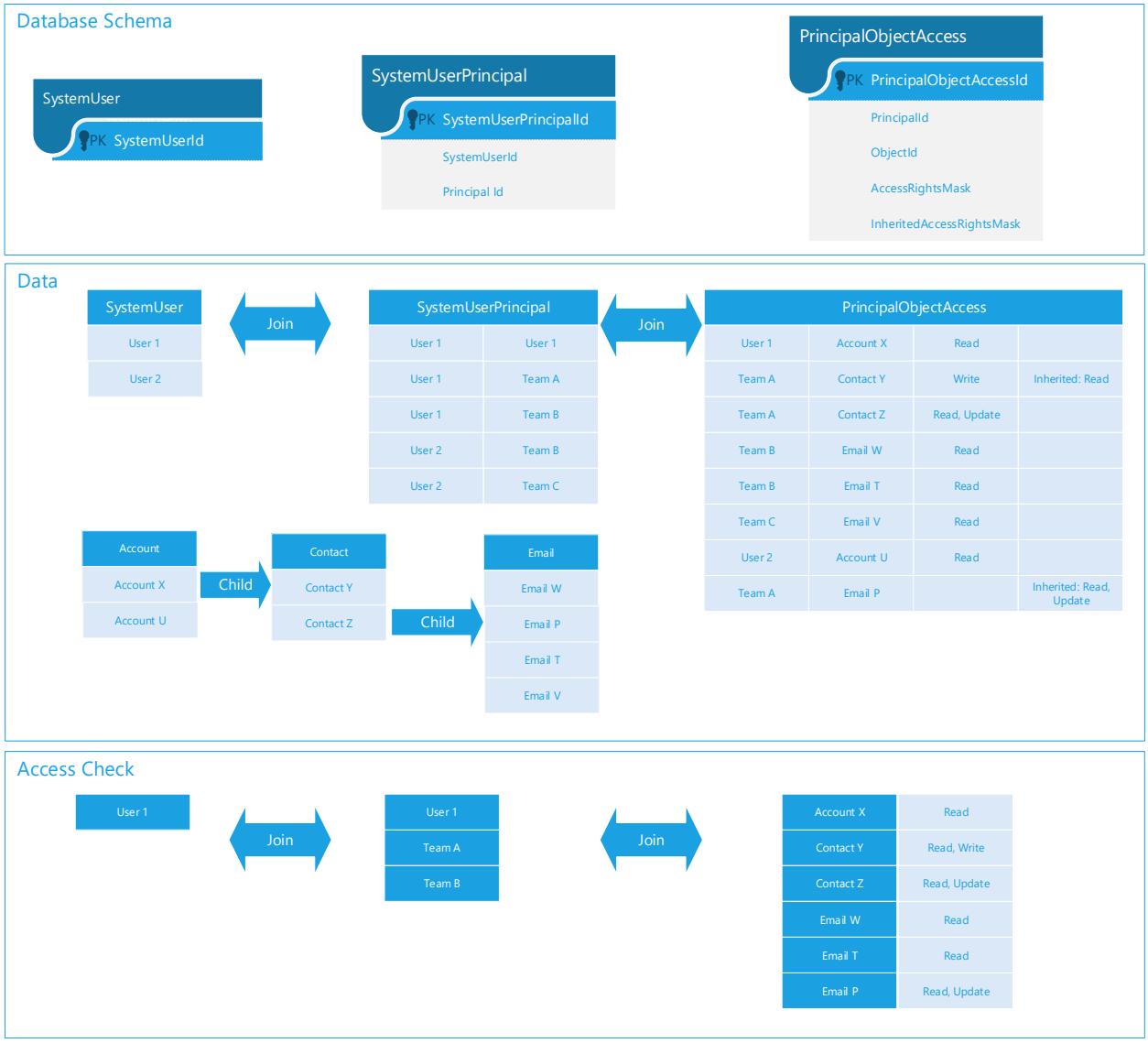
Of initial importance is an understanding of how the sharing rules are recorded in the system. The sharing model is shown in this diagram.


SystemUser
PK SystemUserId
Organization
PK OrganizationId
BusinessUnit
PK BusinessUnitId
Team
PK TeamId
SystemUserPrincipal
PK SystemUserPrincipalId
SystemUserId
Principal Id
PrincipalObjectAccess
PK PrincipalObjectAccessId
PrincipalId
ObjectId
ObjectTypeCode
PrincipalTypeCode
AccessRightsMask
InheritedAccessRightsMask
Org Owned Entities
User Definition/Access Records:
System User
User Settings
User Shared Records
Team Shared Records

The sharing model includes two primary tables.

- The **SystemUserPrincipal** table contains a record for each of the security principals of the system (Organization, Business Units, Teams, and System Users). Where a security principal is explicitly linked to other related principals, such as a team linked to the users who are its members, a record is contained with that link.
- The **PrincipalObjectAccess** (PoA) table records the sharing rules between security principals and individual entity instances. Each record links a particular system user or team to the entity record that it's been granted sharing privileges to. The record also stores the level of privilege that has been granted, which specifies the ability to read, write, assign, share, and so on, the record. Additionally, it stores the privileges that have been inherited from a cascading share that are held separately from the explicit privileges. Finally, the PoA table also contains records granting explicit access for organization-owned entities and records that define the user (SystemUser and UserSettings). Shares to a particular record can be created:
    - Manually and explicitly by a user.
    - Programmatically through the SDK.
    - Automatically by the system (for organization-owned entity records, user records, or as people involved in an activity).

An example of how data is held in these tables and used in the access checks is shown in this diagram.



To determine the records that a user has access to through sharing, Dynamics CRM joins the:

- SystemUser table to the SystemUserPrincipals table to extrapolate all the principals by which the user could have access to data records. This results in a list of the SystemUser record and all the teams the user is a member of.
- Resulting data set to the PoA table to determine all the permissions for the records.

Note: Sharing fields follows a similar model with a PrincipalObjectAttributeAccess table defining specific attributes and the privileges a user or team has shared to the fields for that attribute. This, therefore, has similar characteristics as sharing records in terms of granularity of control but also in terms of the performance and data storage implications. It can be a very useful mechanism but should be used with care at higher volumes.

POA table records

A common challenge in high volume sharing scenarios is the volume of data created in the PrincipalObjectAccess (POA) table. Understanding the types of data created in the POA table can be useful in appreciating the impact a design can have on the scalability of a system.

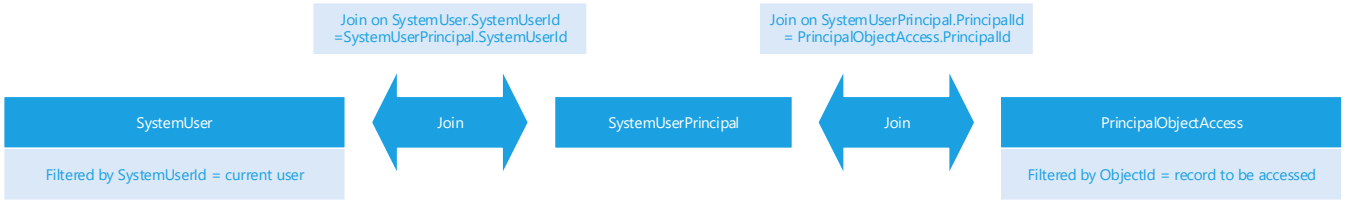
The POA table will contain the following types of records:

- Organization-owned.

    - An entry for each organization owned entity type.
- User entity records.
    - Each user will automatically have shared to their:
        - System User record.
        - User Settings record.
- User sharing
    - Each time a record is shared with a user, it's recorded with a record in the POA table linking that user to the record along with the privileges they have shared.
- Team sharing
    - Each time a record is shared with a team, this is recorded with a record in the POA table linking that team to the record along with the privileges they have shared.
- Activity participation
    - Whenever a user is included as a participant in an activity, they are automatically shared to that activity whether they would normally have access to it or not.

Single record sharing access check

When a user accesses a specific record, an access privileges retrieval and check is performed before the record is opened. This retrieves the maximum directly provided and inherited privileges that the user is directly or indirectly granted through sharing by querying and checks that allows access for the user for this record. To retrieve all the potential access privileges a user has for a particular record, the following query is run.

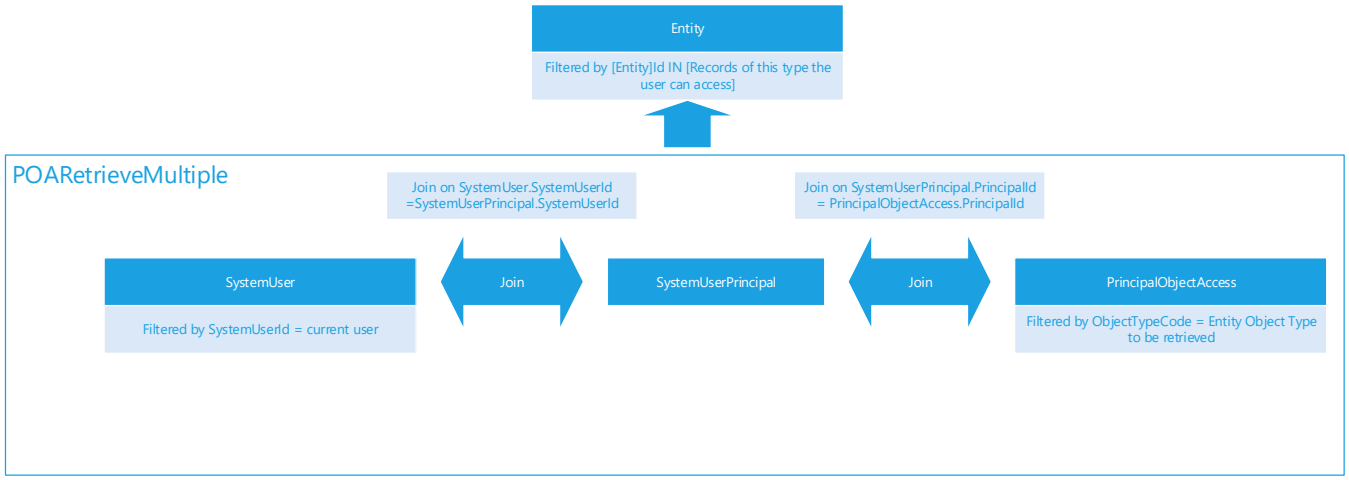


Multiple record sharing access check

When accessing a view or running a RetrieveMultiple query, rather than accessing a single record, all the records that match the filter criteria for that entity type that the user can access are returned (subject to the limit set for the maximum number of records to return).

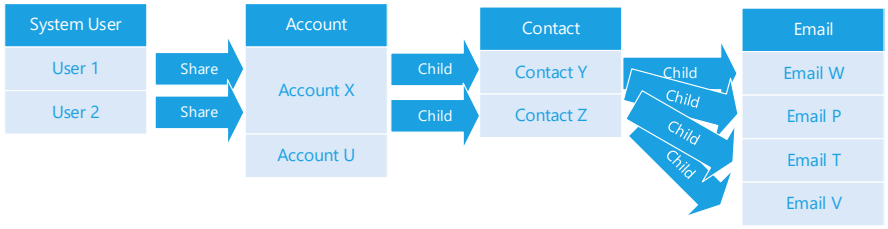
To perform this query for multiple records, the record type is queried with the filter criteria set and with a join to an in-memory table that is created with the records of that type the user can access directly through sharing or indirectly through team sharing.

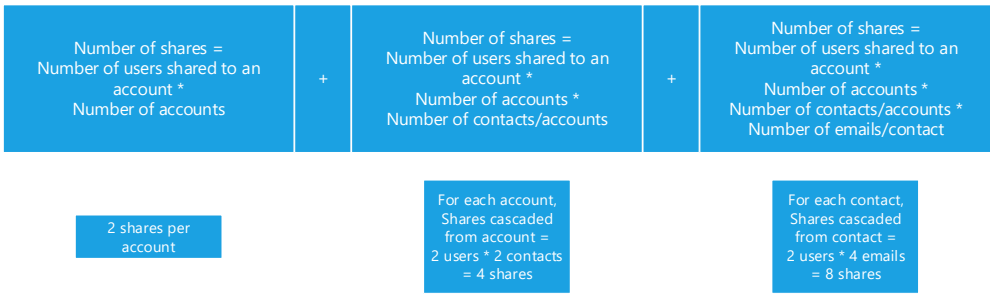
Clearly, this mechanism for calculating access for multiple records using sharing is intended for exception handling, rather than for very high volume access. Sharing at very high volumes will have an impact on performance and scalability as a result of the number of sharing record access definitions that need to be checked in complex scenarios. The processing performed is shown in this diagram.



Cascading sharing

A classic scenario where sharing challenges can be more problematic relates to the inheritance of sharing through parent-child relationships. In this scenario, whenever the parent is shared, the sharing is inherited by all the children. To achieve this, the system needs to create a record for each child record, and recursively to any children of that record, and record the inherited sharing against each.

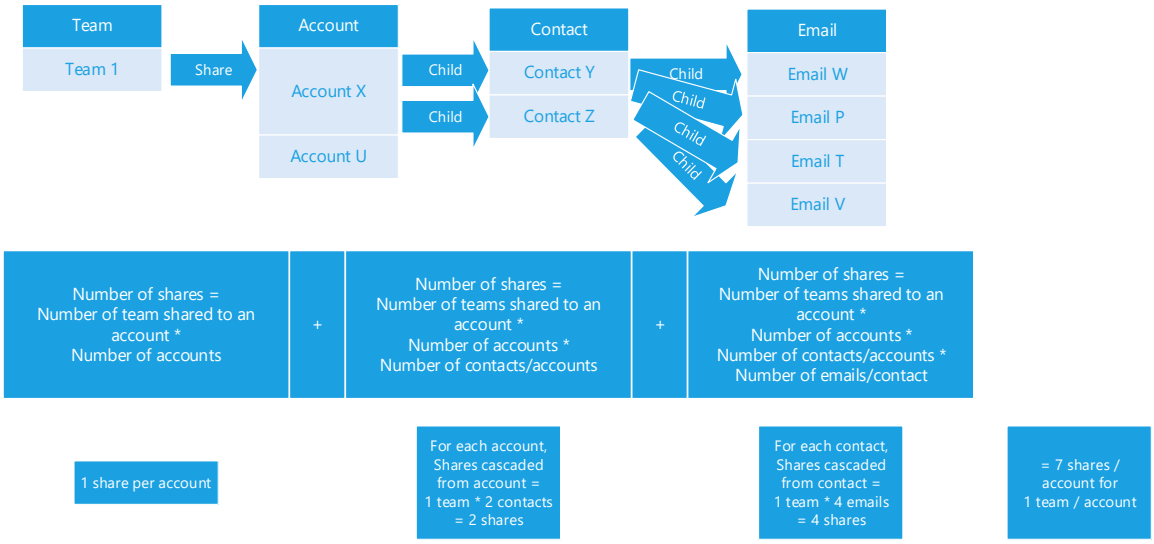




For smaller numbers, as occurs in an exception scenario or to provide less common variations from a more general access approach, this is an effective and efficient mechanism. However, for scenarios that use sharing as the primary access mechanism, carefully consider the volume of sharing records and impact on performance and scalability. Particularly because there are other mechanisms better suited to define volume access.

Team sharing

One immediate approach that can pay dividends is to share to teams rather than to individuals. As shown in the simple example, even a team with two members can reduce the overall number of shares by half. For situations where teams represent larger groups of users, the saving in terms of record growth can be significant.



However, it should be noted that while sharing by using teams reduces the number of records in the table, for a particular user it doesn't reduce the final number of sharing access records that are instantiated as part of the join of tables describing the actual sharing rules to be considered during the access check.

Removing sharing

When sharing for a particular user is removed, the directly granted privileges and any inherited privileges through cascading are removed. Particularly for cascading, the level of privilege inherited for a particular record could have cascaded down and been combined from sharing through either a direct or indirect parent. When removing shares, any remaining inherited privileges need to be recalculated and recorded. If no privileges, direct or inherited, remain, the record will be updated to indicate no privileges are recorded against that record. There is a maintenance clean up job that runs regularly that looks for and removes any sharing records that have no remaining privileges.

It is possible to see records with no shared privileges in this period between the sharing being removed and the job running to remove these records.

Sharing implications

After having reviewed these different options, a summary of the implications associated with sharing include the following considerations:

- Sharing works well for the intended purpose, that is to serve as an exception mechanism or to overlay variations on a more general access model.
- When used to model large scale solutions, sharing can result in:
  - Large volumes of data that need to be recorded to account for various sharing rules in the system.
  - Higher processing demands associated with checking each of these rules on each user access. This can result in potential performance and scalability implications.
- The administration of complex sharing rules should be considered carefully. Each time that a record is created or amended or when the related user's organization changes, the related sharing records may also need to be altered.
- For scenarios that require broader access, Dynamics CRM provides more effective mechanisms.

Access team lifecycles

Access teams are created as any other team, just without the ability to have security roles or own records.

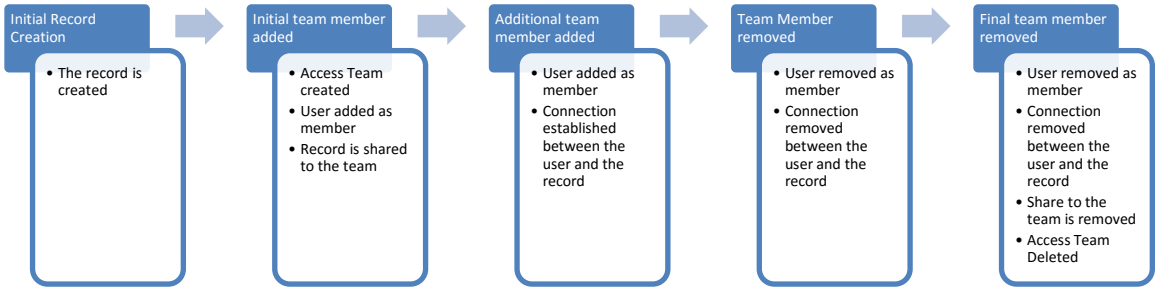
Access teams can then have users added to them as members and can have records shared with them.

The advantage of splitting out these types of teams from owner-based teams is that the overhead of adding security roles to a team can be avoided when it is not required. When security roles are added, these can impact performance and scalability in key ways:

- Team security roles act in a cumulative way, and as a result for each user who is a team member. The privileges for that user must take into account all the roles of the teams to which they are a member.
- The more teams a user is a member of with security roles, the more complex the calculation to determine access.
- To reduce the impact of doing access calculation on each request, the system caches the cumulative permissions for the user as they first connect to Dynamics CRM.
    - When a user has a large number of teams with security roles, this can cause a delay on initial connection due to either one of the following.
        - After a computer restart.
        - When the user's record has been flushed from the cache due to twenty minutes of inactivity.
- Whenever the user is added or removed from a team, or the team has its security roles changed, the cache for each user affected needs to be flushed and recalculated on the next connection.
- For rapidly changing teams or team memberships, access calculation can introduce a significant performance and scalability impact to the system.
- In these cases, access teams can avoid this impact for team memberships where this is not necessary. For example, when a combination of ownership and sharing is used, access teams can be used for the sharing cases. This avoids the cache and access calculation impact when they change.

Service scheduling uses caching extensively to optimize access calculations. When teams are used as resources in service scheduling, Dynamics CRM loads the teams into the cache. Access teams, therefore, can't be used as resources in service scheduling to avoid the need to cache and to avoid the impact of recalculating the resource groups when a user's team memberships change.

When access teams are enabled for an entity, the life cycle of the team is managed automatically.

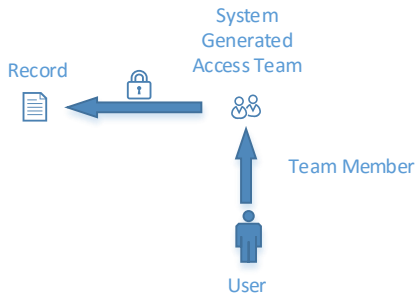


As shown in the preceding diagram, the access team is generated automatically on demand as the first team member is added.

When the final team member is removed, the team is deleted.

This brings advantages where you have rapid changing of large numbers of teams, removing any old and redundant access that may no longer be needed.

When a user is added to the team connected to that record, a membership is set up for that team and the user is linked to the team in the SystemUserPrincipals table.



Access to the record is established through automatically sharing the record with the team, with the privileges defined in the access team template for that entity type.

If the record is deactivated, this won't affect the related access team. If all the team members are removed, the automatically generated access team will be deleted.

Design considerations of using access teams

Access teams can't own a record; they don't have any security roles so they can't be granted the privileges to own a record. Nor can they access records through security role privileges of Owner, Business Unit or Organization level scopes.

For very large volumes the implications of the sharing still needs to be considered carefully. In particular, managing the life cycle of an access team's existence for records that no longer need to be directly viewed should be considered.

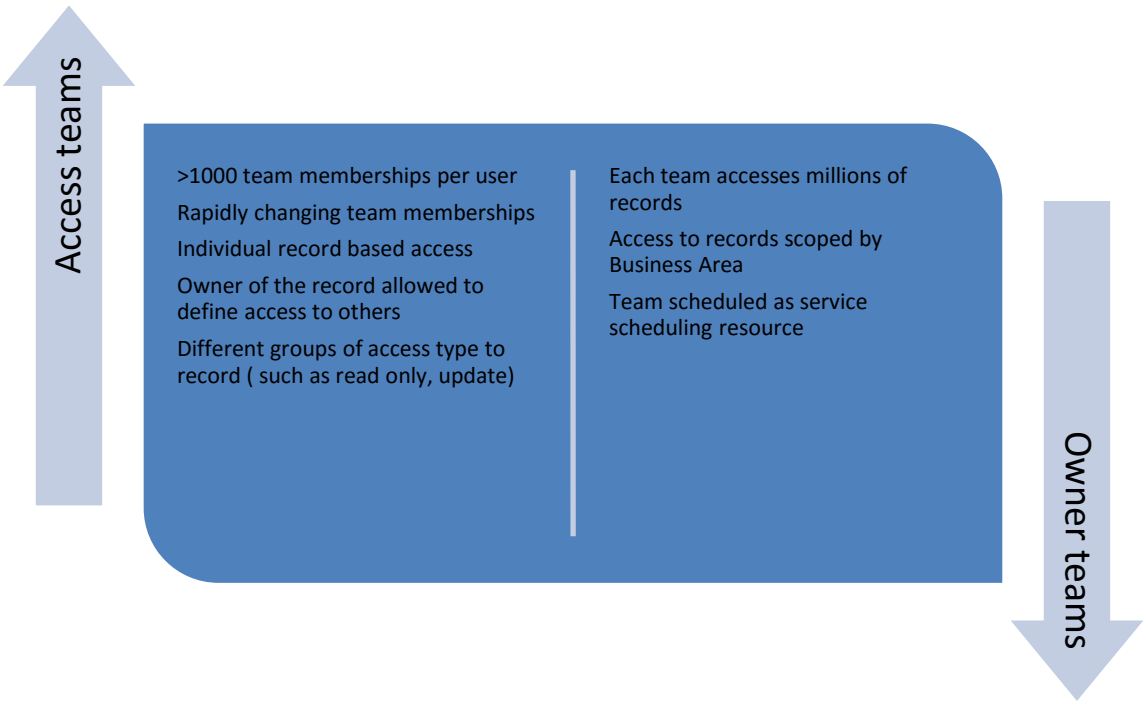
When a record is deactivated, this doesn't change the related access team membership as this may be needed if the record is reactivated or for compliance purposes. To remove a related access team for a record, remove all the team members, either through the user interface or programmatically.

Access teams can't be used as resources in service scheduling.

Multiple access teams can be linked to a single record, allowing different access types to be defined for the same record, such as defining a read-only access team and an update team.

Access team types are defined at the form level so they apply to all instances of a particular entity type. However, role-based forms can be used to present a different view of these to different users controlled by security roles.

With the separation of owner teams and access teams, there will be scenarios where each is the more appropriate choice. This diagram highlights some key factors to consider when deciding which to use.



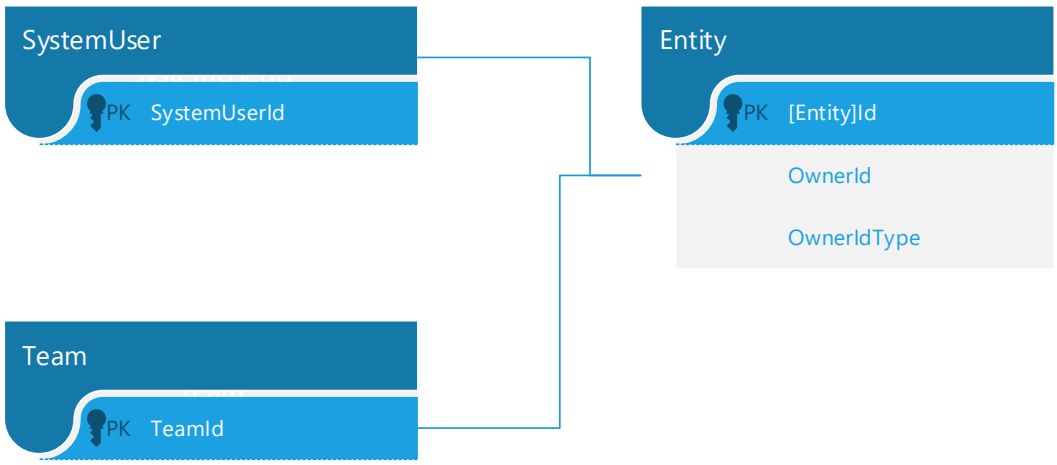
Ownership access checks

Another approach for defining access to resources is by taking advantage of ownership. This model enables granting a user privileges only to records that the user owns. From a security perspective, limitations of the user ownership approach are that:

- Only a single user can be granted permissions to the record via user ownership.
- The record also derives its business unit from its owner, such that the record's existence in the business unit hierarchy is intrinsically linked to its owning user.

By using team ownership for a record rather than user ownership, you can grant multiple users access via the ownership mechanism. Using team ownership also allows for access by users from a range of business units.

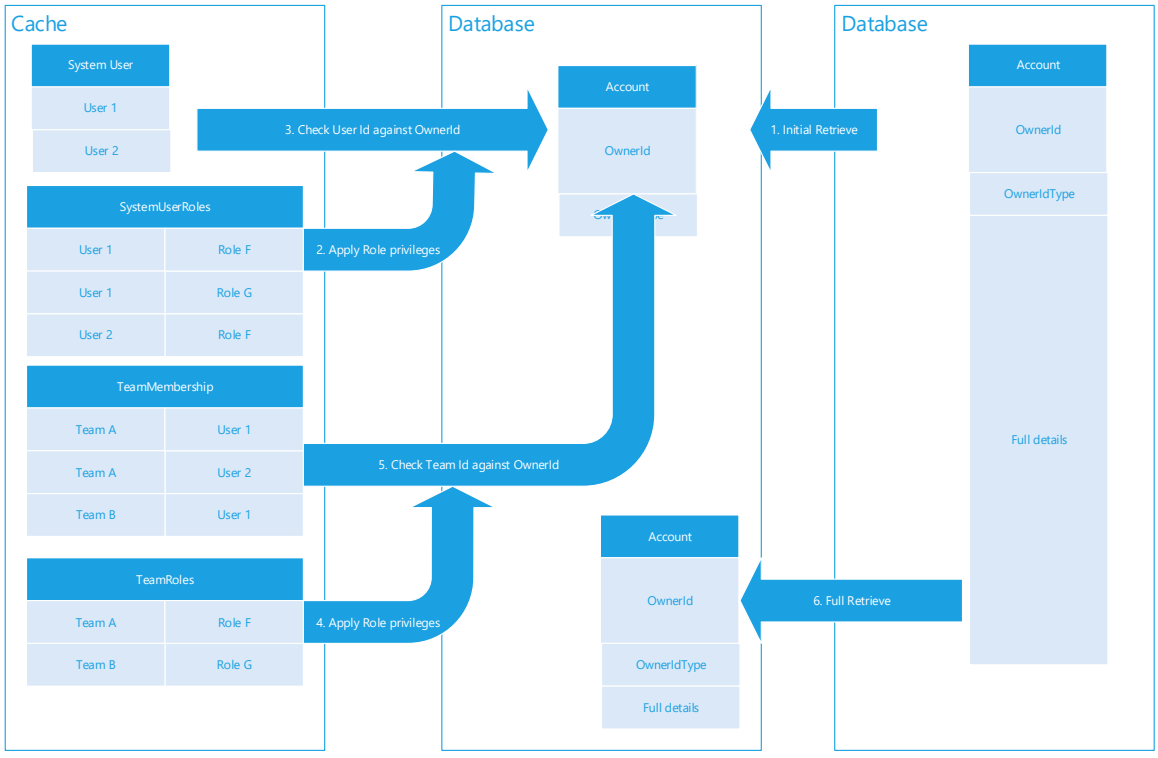
Data associated with ownership of records doesn't require additional storage. Because the information used to determine ownership is held directly on the entity records themselves, it reduces the amount of data stored as well as the complexity of querying that is required in the system to determine access, both of which qualify this as a more efficient mechanism than sharing.



When retrieving information, if the user doesn't have organization-wide access and the entity isn't organization-owned, but the user does have owner access, the system can check the ownership of the record for access. How this is performed depends on whether an individual record or multiple records are being retrieved.

Accessing an individual record

When accessing an individual record the system checks both the individual user and any owner teams they are a member of for ownership. This occurs in a two-stage process to minimize the initial request cost for information the user may not be authorized to view.



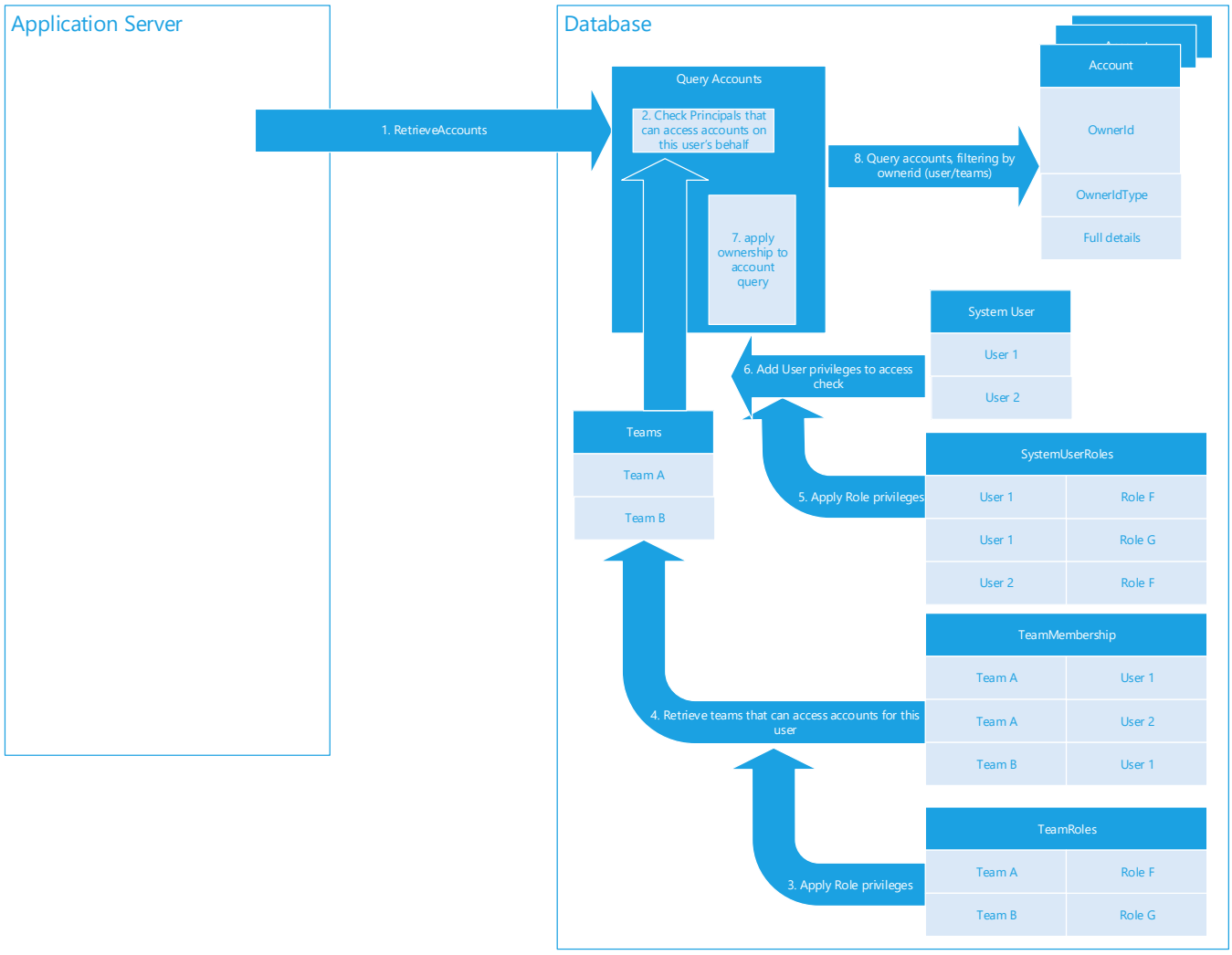
- Because a user's security roles, owner team memberships and the owner team's security roles are cached and can therefore be accessed in memory within the Application Server, only an initial set of data need be retrieved about the record, including the owner and type of owner of the record, and quickly compared against the cached information.

- This enables the application server to compare the ownerid to the ID of the user and the teamid of any owner teams that the user is a member of and whether the privileges grant ownership access to a record from an initial economic request to the database server.
- If this is the case, access rights can be confirmed within the application server and a full retrieve of the record performed.

Accessing a view or performing a RetrieveMultiple

When retrieving multiple records, as with individual ownership access checks, both the user and any teams that they are a member of should be considered if they have been granted privileges through a security role to records they own. The distinction is that instead of this being applied to a single record, this set of user and team IDs must be compared to all the records being queried that satisfy other filter criteria applied to the query.

For efficiency, this comparison is performed in the query to the database server. The list of owner teams a user is a member of that can have ownership read access to the type of record being queried is retrieved in memory in the query as a Common Table Expression. An in-memory join to the table of records being queried is performed doing a join on the ownerid column, filtering out any records owned by other users or by teams the current user isn't a member of. This allows SQL Server to apply rich query optimization and indexing to perform the query as efficiently as possible and reduce the data returned to the application server to only the results set.



Ownership implications

The results of the testing and corresponding analysis indicate that team ownership offers a good model for granular access to records. A summary of the implications of ownership is presented in this table.

Aspect	Implications
Independence from data	▪ The performance of the security model is independent of the amount of end data that the teams are owners of. ▪ This is significant difference from the sharing model.
Independence from team volumes	▪ Number of teams in the system doesn't directly impact performance. ▪ 100K teams had no significant impact.
Grows linearly with team membership	▪ Where performance is impacted is with owner team memberships per user as they each need to be checked upon access. ▪ Increase in owner team membership is linear in impact in response times while within the capacity of the system.
Bounded by CPU usage	▪ The key constraint occurring is the capacity of the application server CPUs when performing iteration access checks. ▪ Performance is good until the CPUs reach capacity.
Initial cache load hit	▪ Large owner team memberships increase the cache load impact for owner teams. ▪ With lots of users and owner team memberships, this can be significant but can be "warmed up" to mitigate. ▪ Large access team memberships doesn't have an impact on caching.
Target Team memberships	▪ Number of memberships that can be achieved will be affected by multiple factors. ▪ 1000owner team memberships per user are shown to be feasible. Significantly higher than that would need careful consideration of usage and potentially combination with other security access features for usage patterns. ▪ Users can be members of a greater number of access teams than owner teams without significantly impacting the performance and scalability of the system. The volumes of records shared with access teams still must be carefully considered.

The response time of using owner team for ownership grows independently of the amount of data in the system or that the user is given access to, unlike the sharing model that both grows in sharing record data and response time as more data is shared with the user.

Response times grow when the user is made a member of an increasing number of owner teams. This still shows good linear growth, although with an increasing impact on CPU usage until the application server CPUs reach capacity. At this point, overall system performance deteriorates quickly and impacts all usage through the application servers.

As a running system, it shows the need to monitor the CPU levels of the system as the response times will likely remain good until the point the CPUs reach maximum. But, the impact can be significant and quickly increase response times as the long running requests block new requests to IIS, resulting in queued up requests and rapidly increasing overall response times.

The initial cache load for a user is a heavy operation, where multiple users make initial requests at the system, such as at start of a shift pattern. This can introduce a spike in performance as the system loads the cache for each user. Mitigate cache load operations through pre-loading user caches after a server restart, or before a shift start.

Tests show that having a large number of owner team memberships per user is achievable with acceptable response times. However, tests also show that response time is heavily CPU-related and dependent on other factors such as:

- Number of users.
- Number of concurrent requests.
- Number of application servers.
- CPU usage by other requests or processes on the application servers.
- Specification of the application server CPUs.

While the exact volume of owner team memberships per user that can be supported will be implementation specific, testing has shown that up to 1,000 owner team memberships per user can be achieved. It's also worth noting that this is a good benchmark beyond which to consider alternative mechanisms offered by Dynamics CRM for modeling access to information.

Team ownership works well for providing granular access to records in which users either:

- Are part of groups that are provided common access to a wide range of records but there is no way to define the grouping of data directly, or
- Need independent access to a smaller subset of records and individual access to each record must be defined

Access teams can be very useful where granular access to records is needed through a sharing model, but business unit security role based access isn't also required.

Where large common access is defined, using business units may be more appropriate. This is especially the case where broader responsibility across wider sets of records is needed, such as a manager or compliance officer. But in these cases it's common that the defined boundaries of that person within the business more naturally fit with a business unit. Common access can often be the case where the manager is responsible for a specific area, such as the UK, even if their staff do not have as clearly defined boundaries of responsibilities. Providing business unit access for the manager but team ownership access for the front line staff they manage can be a good balance.

Use of team ownership can simplify the process of providing access to multiple people for a broader data set such as that for a particular deal or case. Defining the team for the broader case or deal, and then assigning any new related record to that team would immediately make the information available to all members of the team, simplifying the administration overhead. In a similar way, removing a user's team membership can quickly remove their access from a broad range of records when their responsibilities change.

Business unit access checks

When business unit access is performed, this acts in a similar way to ownership, however instead of comparing the ownerid, it instead compares the owning business unit of records against business units the user has access to. Much of the processing therefore occurs in a similar way, but with two important differences:

- As a record is created, it automatically gains a business unit based on the user or team that creates it or to which it is assigned after creation. This minimizes the overhead of managing team ownership.
- The hierarchy of access of business units provides the ability to enforce structural boundaries in a business at a broader but therefore more efficient level than individual access.

With that approach in mind, a user will have access to records in business units through a combination of:

- User privileges – A user who has been granted:

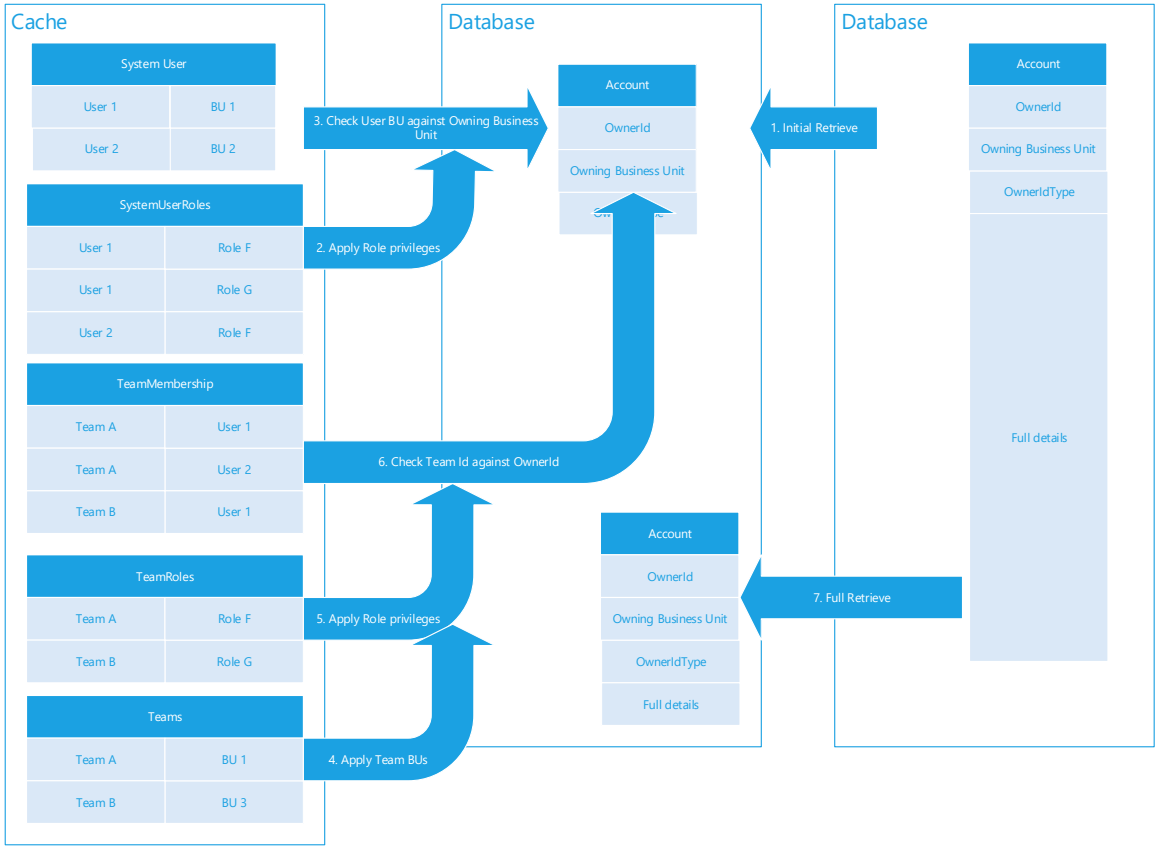
- Business unit access to an entity type will have access to any records owned by a user or team in the same business unit as the user.
- Parent:Child business unit access to an entity type will have access to any records owned by a user or team in the same business unit as the user or any children of that business unit.

- Owner team privileges – A user who is a member of an owner team granted:
  - Business unit access to an entity type will have access to any records owned by a user or owner team in the same business unit as the team.
  - Parent:Child business unit access to an entity type will have access to any records owned by a user or owner team in the same business unit as the owner team or any children of that business unit.

The process of performing this access check will vary based on whether an individual record is being accessed or is a view of multiple records.

Accessing an individual record

When accessing an individual record, the system checks both the individual user and any owner teams the user is a member of for business unit access in a two-stage process, which minimizes the initial request cost for information the user may not be authorized to view.



- Because a user's security roles, owner team memberships, and the team's security roles are cached, an initial set of data can be economically retrieved from the record including the owning business unit of the record and quickly compared against the cached information.
- This enables the application server to compare the owning business unit to the business unit of the user and the business unit of any owner teams that the user is a member of and whether the privileges grant ownership access to a record.

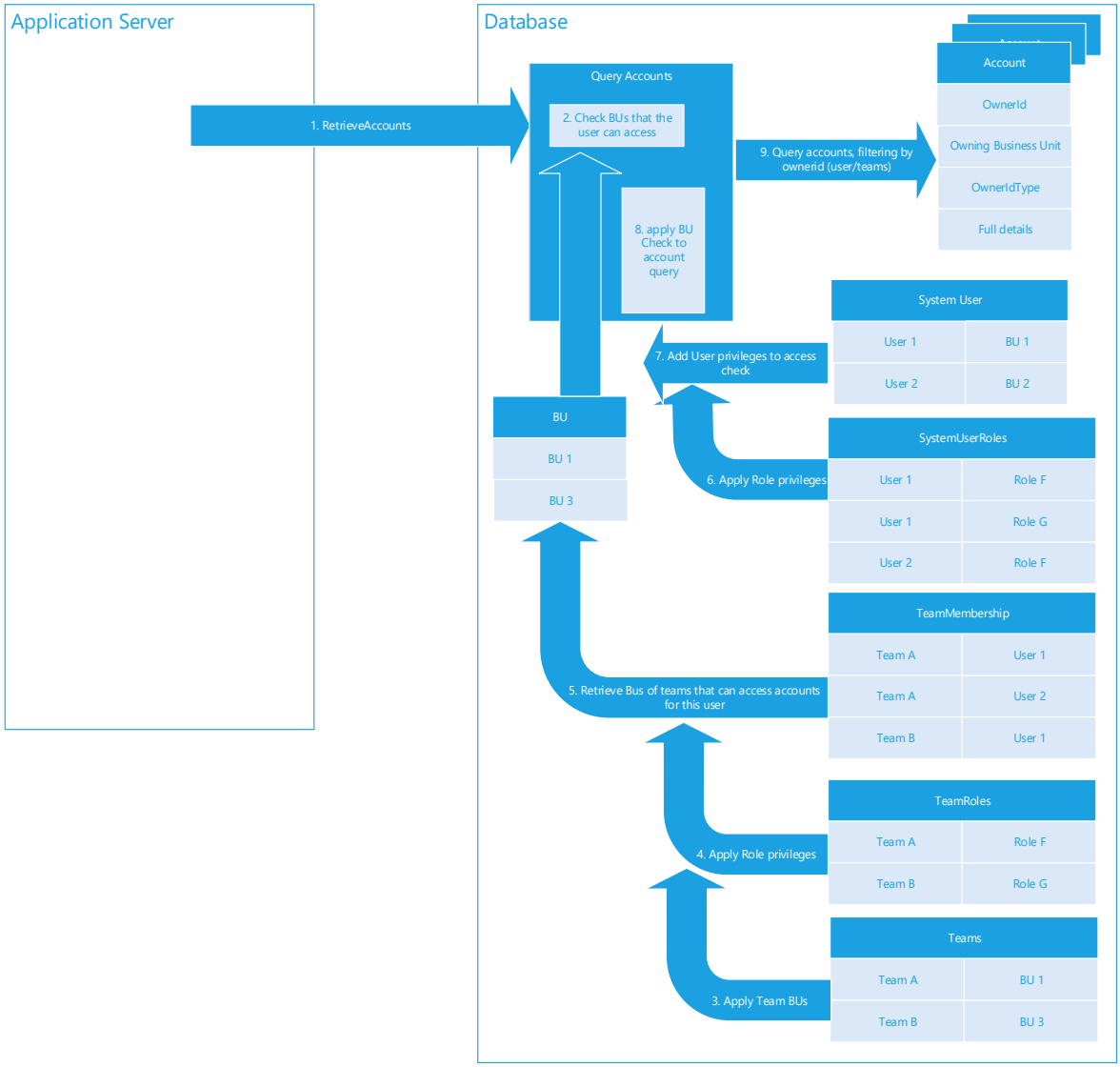
- In the Parent:Child business unit access scenario, the list of children business units the user can access for this entity type are also enumerated and checked against the owning business unit.
- If this is the case, access rights can be confirmed within the application server and a full retrieve of the record performed.

Accessing a view or performing a RetrieveMultiple

When retrieving multiple records, as with business unit access checks, both the user and any owner teams to which they belong should be considered if the user has been granted privileges through a security role to records in the user's business unit. The distinction is that instead of being applied to a single record, this set of user and team ids must be compared to all the records being queried that satisfy other filter criteria applied to the query.

To enable this check efficiently, the business unit a user is in and the business units of the owner teams the user is a member of that have either Business Unit or Parent:Child business unit access to the entity type being queried are calculated before querying the actual data. This set of business units is then compared in an in-memory join to the data to be queried. This comparison determines access by joining on the owning business unit of each record.

The set of business units the user can access is derived from the user and owner team business units and their security roles, although there is a level of optimization within the database by storing a derived set of entity type and business unit access levels for users with can be quickly and efficiently retrieved.



Business unit privilege implications

Business unit privileges are designed to provide optimal access to slowly changing structures as well as access for larger groups, for example division level access by call center or branch employee, management or compliance structures, for reporting or governance.

High volumes of business units (> 1,000) or business units that are rapidly changing can have an impact on system performance, for the following reasons:

- Access is pre-calculated, so high volumes of changes of business unit structure can have an impact on the system.
- Business units are aimed at controlling wider scopes of data to enable access to be granted to users or teams of users to the entire scope of data at the same time. It is therefore optimized for this purpose rather than becoming a mechanism for granular access to smaller sets of records.

Records created and assigned to a particular user or team as owner are immediately available to all users with access to data in that business unit with no additional processing needed.

The cost of processing security checks using business unit privileges as data volumes in that business unit grow is linear in nature with the growth in data, making it a good choice to keep the number of business units small where possible and benefit from the breadth of control it gives.

Organization-wide privileges

For scenarios in which a type of data is made available to all users, either it can be made organization owned or a user or team can be granted organization-wide privileges to that entity. When a user tries to access data of this type, the system can determine either from the entity type or the user's security privileges that the organization-wide privileges apply, allowing it to simply retrieve the information which provides the most efficient access model. Of course, this approach applies only to certain types of data and users. But it does allow for the reduction of overhead for scenarios in which it is applied.

One limitation is that an organization-owned entity can't be changed to a user-owned entity at a later date. But giving users organization-wide privileges can also provide efficient mechanisms to access records while still allowing for other users to have more granular access. Where it is known that data will never be sensitive using organization-owned entities can allow the platform to quickly determine that no additional checks are needed for any requests of that type.

Organization-wide access implications

Organization-wide access is simple in nature. When everyone should see the data, organization-wide access is a good choice to make as it allows other more costly security checks for that entity type to be bypassed giving optimal performance where the access control isn't needed.

Hierarchy access

From a scalability perspective, the primary thing to consider with the hierarchy access feature is that it isn't intended to drive more performant or scalable access checks than the other features described previously, but rather to make the process of managing these other modelling features simpler in typical hierarchical organizations.

What often happened previously, without hierarchies, was that when security access was granted to a user for particular records, customizations automated the process of also granting the same privilege to the user's direct and indirect managers.

Hierarchies remove the effort of customization and processing time rippling out of privileges to all the managers of users, and instead perform those same checks automatically.

The same end checks are required when a hierarchy is used and a check is performed to see whether a manager has access to a record. The process essentially is performed to check the combined set of permissions for all the users that report up to the manager.

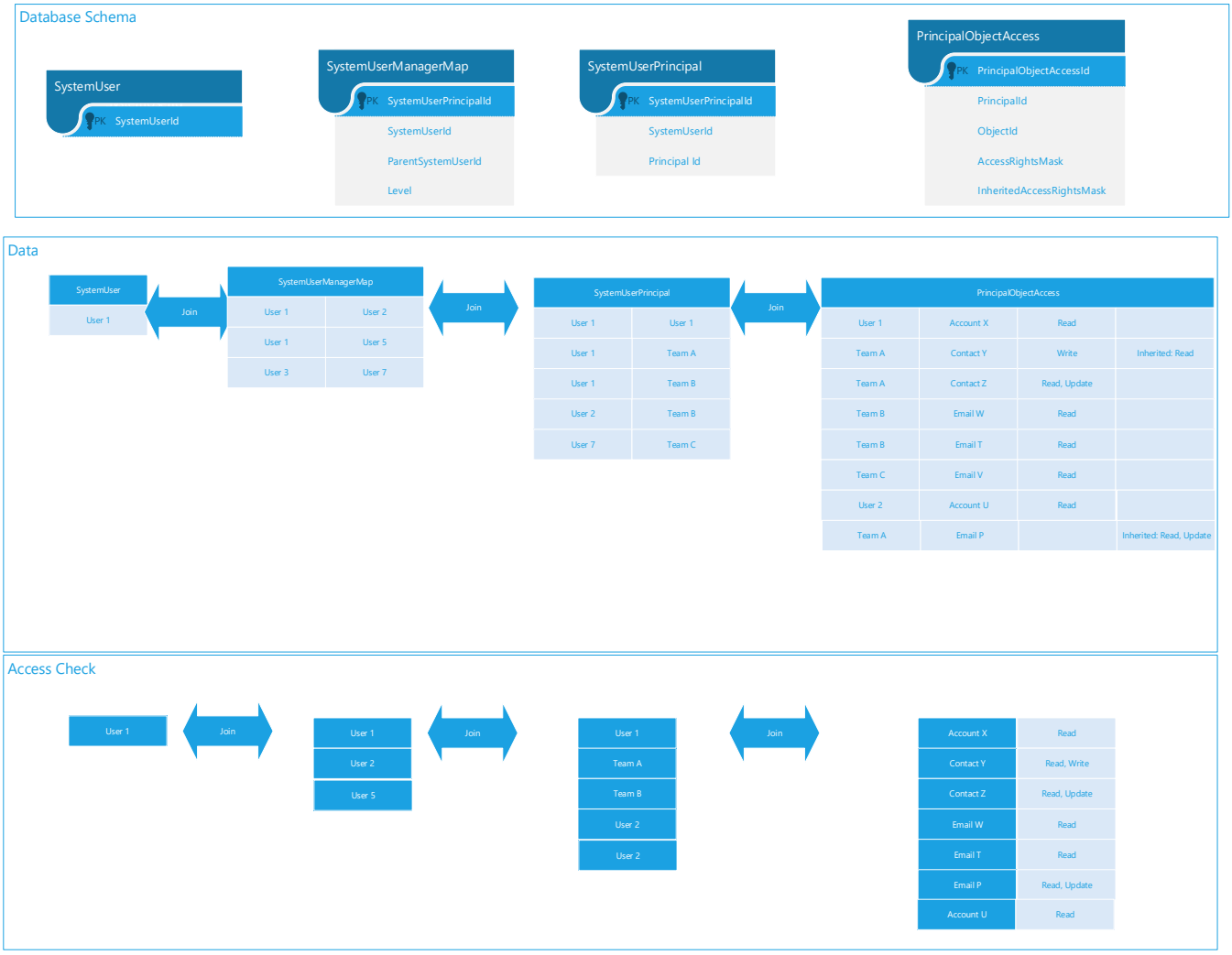
The list of users that report to a particular manager is pre-calculated and de-normalized in a table called SystemUserManagerMap, which reduces the overhead of recalculating this each time the user accesses the system.

After the SystemUserManagerMap table process, these tasks occur:

- Perform business unit and organization level access checks for the current user.
- Retrieve the list of users who report to the current user.

- Perform ownership lookup/sharing for each of that set of users.

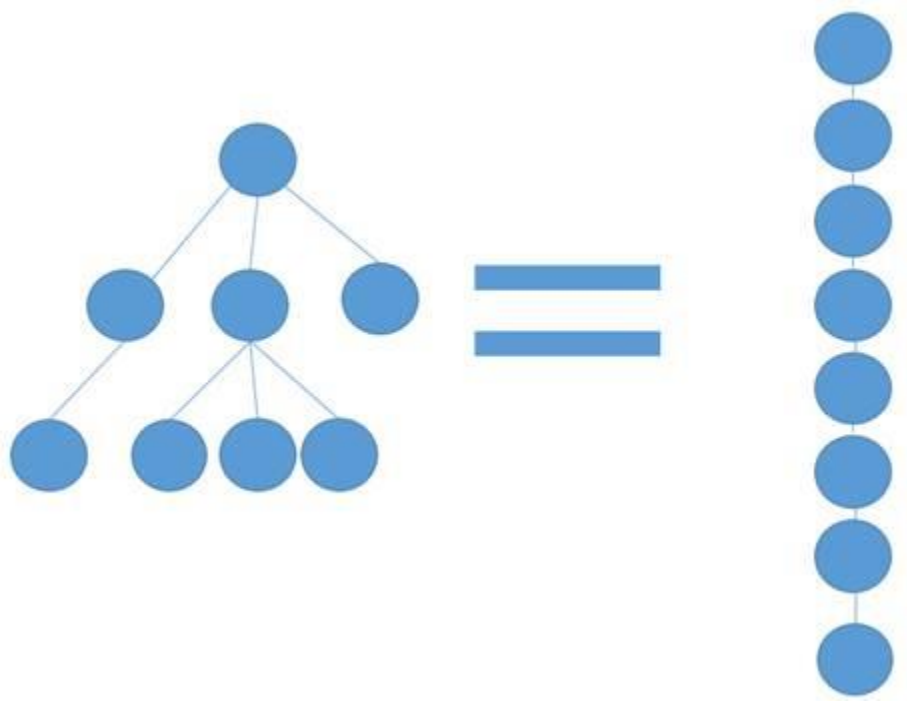
So if a user A has 100 reports, access checks for 100 users are performed. Therefore, the same number of checks is performed as if explicitly granted, but this avoids the maintenance and storage of all the records for managers.



Hierarchy levels

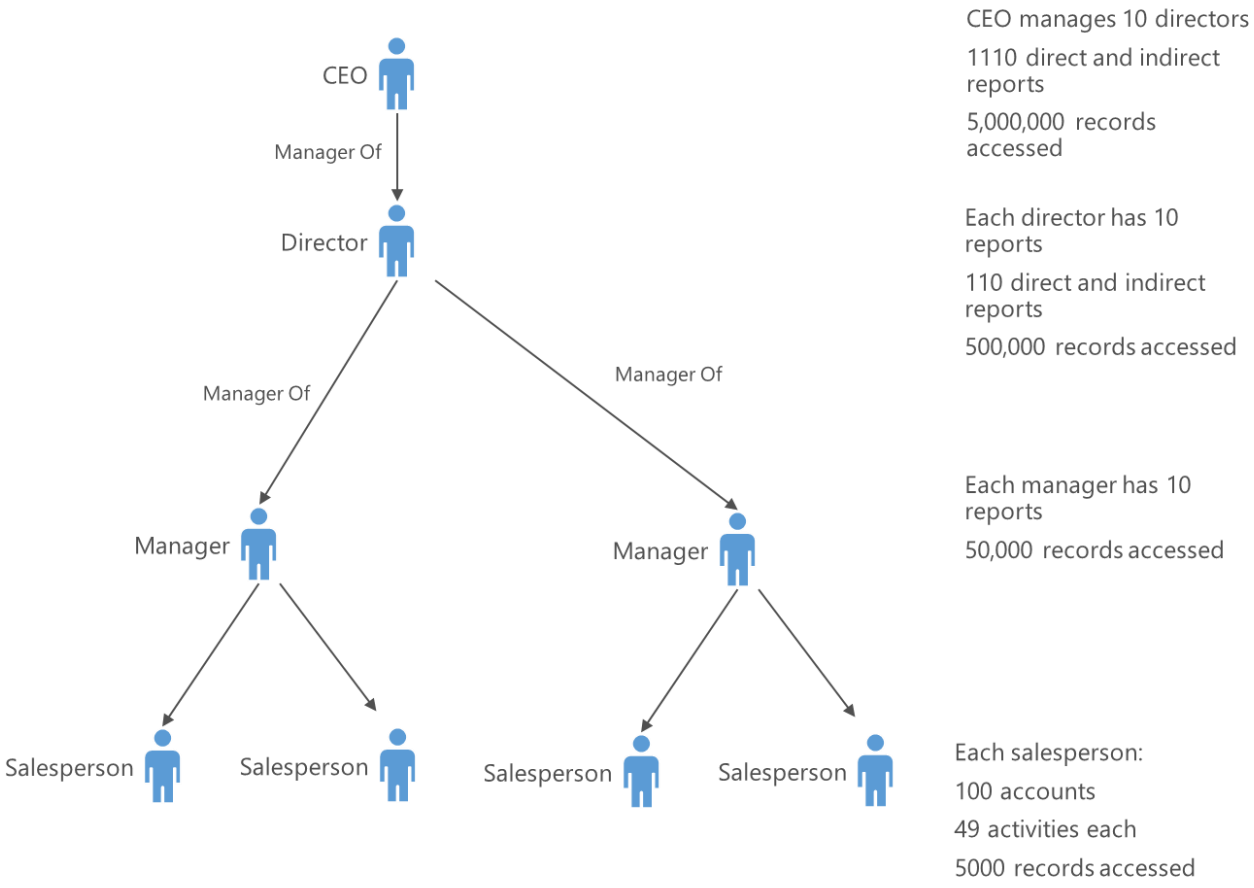
Although the exact implications of deep hierarchies depends on the specific implementation, a guideline is that the target is about four levels of management hierarchy. The maximum is 100 levels. But, this is an extreme scenario and in most cases would have scalability implications.

The limiting factor is the number of indirect reports that a manager can have, rather than how deep the hierarchy tree can get. Having 7 indirect reports would have the same implication whether they are spread across 7 levels or only two.



As you aggregate levels of access, volumes can grow rapidly so carefully consider the following issues:

- Even with only three levels of direct reports, as the following simple example shows, a manager can end up with over 1000 reports and over 5 million records.
- Now imagine 10 levels of hierarchy and the implications on scalability with the same growth patterns.



Combinations of access types

Although considering these models in isolation is useful, it's also important to understand how combinations of security models can impact performance and scalability.

When a user is granted organization-wide access to information, even if access is also provided through other mechanisms, these other mechanisms can be bypassed as the system can determine that the organization-wide privilege will always apply for this type of record, optimizing the access.

In a similar way, when accessing individual records, if business unit or ownership access can be used to determine access to a record, complex processing of sharing rules won't need to be performed. The business unit and ownership access can be performed on information cached about the user and on the minimal information retrieved about the record. For example, the owner and owning business unit, which enables an optimized access check to be performed. However, if for a particular record the business unit access or ownership rules don't allow access, sharing needs to be checked.

Most access can be modeled using one of these more optimized access approaches, leaving sharing for only exceptional circumstances. The overhead of the additional sharing checks (either when the user does not have access, or when they access through sharing only) is reduced in volume of rule and in terms of the frequency of occurrence, thus improving overall system scalability.

In particular, the isolation of different user types and usage patterns is critical to effective security modeling design. Analysis of this often yields the ability to utilize the less granular and therefore more efficient access

mechanisms for users with broader access needs and leaves greater system capacity to be focused on providing more granular access to users with more specialist needs.

Trade off with granular access

Although the granular access model offered by team ownership or sharing sounds ideal in many scenarios, the implications of granular access is that for every query of the system, as the volume increases, the system has to perform a significant amount of work. The larger the volume, the greater the impact on performance will be.

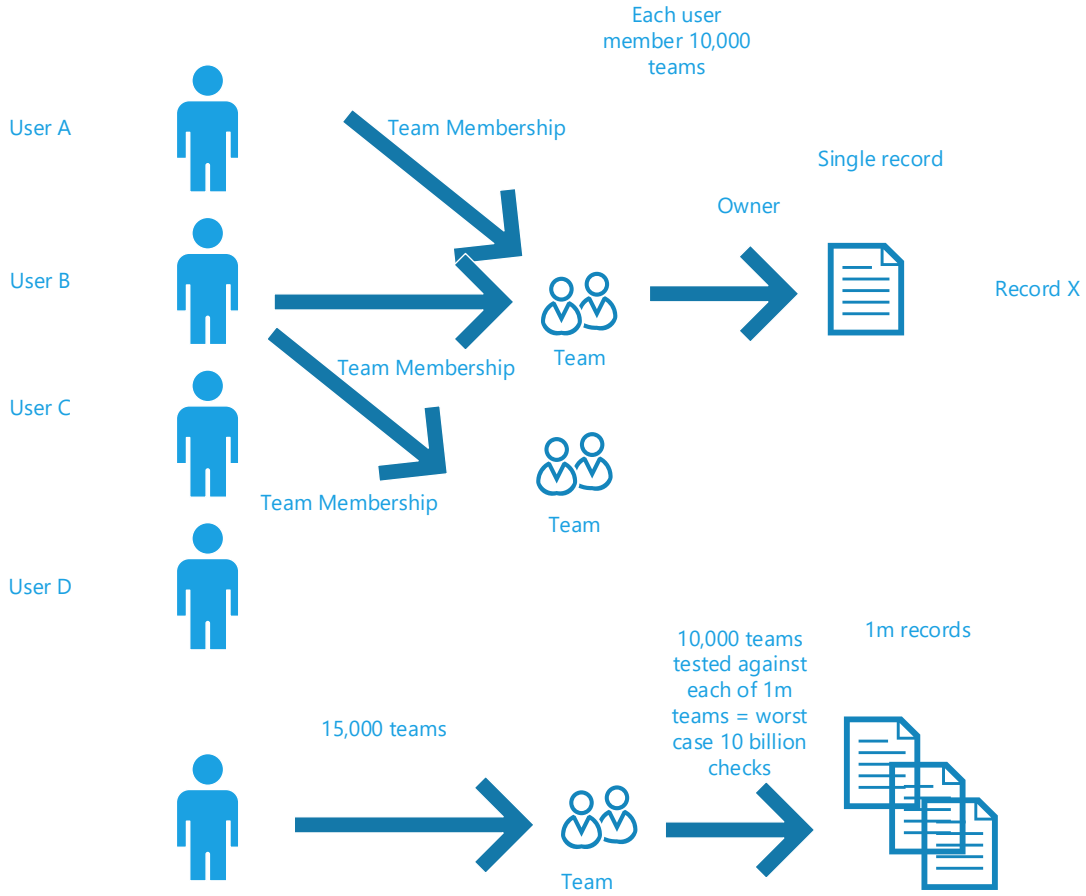
For context, let's look at the volume of access checks required in different scenarios. Our base for this calculation is one million records and a user that is a member of 1,500 teams.

Individual record access - When accessing a particular record, the system needs to:

- Determine the list of teams the user is a member, or
- Compare the team that owns the record to each of those teams, performing a total of 1,500 checks.
- The number of checks grows linearly with the number of teams the user is a member of.

Views of records - When accessing a view of records, the system needs to:

- Determine the list of teams the user is a member of.
- Using a join, compare the user and each team the user is a member of to the owner of each record.
- If there are one million records in the system and 1,500 records, the system could, in a worst case scenario, perform 1,500 checks against each record, or 1.5 billion access checks.
- The practical reality is that Dynamics CRM deliberately limits the initial results of a view, with the user defining how many results should be shown. This is typically the first 100 or 250 results. The other factor is, once a successful match on an owning team has been found giving a user access, no further checks are performed on that record.
- It's conceivable that, in this scenario, as few as 100 checks could be needed with successful access being granted for the first 100 records by the first team that the user is a member of.
- But where many records are not accessible by a user, it is important to realize that a significant amount of processing could be needed. Where few records are accessible by a user, it could be possible to need to perform the full 1.5 billion checks in the worst case scenario, a significant amount of processing.



A key aspect to evaluating the potential scalability of a given business solution lies in having a firm understanding of the contextual processing challenge being demanded of a system when driving granular, individual access. As shown above, when dealing in high volumes, requiring that individual, granular rules are checked by definition introduces a complexity to the security access checking. As a result, using granular access controls in a scenario that doesn't require them incurs a potential performance or scalability implication that may be unnecessary.

While Dynamics CRM supports efficient granular access models, it also offers other capabilities to reduce the processing and maintenance complexity for scenarios that require larger volumes.

Comparison

Dynamics CRM features offer a great deal of flexibility of modeling, either granular in access or broad in scope. That flexibility is accompanied by a processing cost, as the feature offers more individual access, the impact of that extra granularity is paid in the number of checks that need to be performed for each request.

Different approaches can be used together, particularly when considering different user and usage types. For scenarios that require providing individual access, team ownership and sharing can be used. However, in scenarios involving users acting in a managerial or reporting role, it helps to leverage features that offer broader access to take advantage of the optimization provided by broader access models.

A comparison of features and functionality for controlling access to system data in Dynamics CRM is presented in this table.

Feature/Functionality	Access characteristics
Organization	▪ For open access information to all. ▪ Bypasses need for security checking, so optimal performance. ▪ Organization owned entities offer simple and efficient mechanism but less control. ▪ Organization privileges against user owned records offers efficient access for some without limiting others.
Business unit	▪ Great for large volumes of data accessed by large groups of users. ▪ Allows for optimal access checking. ▪ Reduces maintenance overhead. ▪ Can be used for management and oversight access with more granular access.
User ownership	▪ Ideal for large volumes of overall data but individual granular access to smaller subset. ▪ Optimized access checks as it combines memory cached and record data directly. ▪ Limited to one type of access through ownership, such as owned by a single user. ▪ Drives business unit position so enables combination with business unit privilege access.
Team ownership	▪ Ideal for large volumes of overall data but group granular access to smaller subset. ▪ Optimized access checks as it combines memory cached and record data directly. ▪ Reduces cascading impact as user involvement changes. ▪ Isolate users from record business unit structure, enables user movement without data impact.
Sharing	▪ Ideal for modelling exceptions over standard model using other features. ▪ Provides very specific unique permissions per record. ▪ But comes with cost, checking lots of rules is expensive in terms of performance, storage, and maintenance overhead. ▪ Sharing with teams, and in particular access teams, mitigates this.

Alignment with real world usage

How do these findings map to real world usage? Considering some common usage patterns helps to highlight where granular access controls like team ownership can play a significant part in meeting business needs in a scalable way and where other more efficient volume access controls may be more appropriate.

Usage patterns

By breaking out some of the key usage patterns that often require access to data in Dynamics CRM, it's possible to analyze the typical patterns of access for different user types and determine how the characteristics of these security modeling capabilities map to the needs of each user type.

Usage Pattern	Characteristics
Active involvement	▪ Common user types include sales or case management. ▪ Typically need full time access to the records they work with.
Managerial reporting	▪ Manager and support teams. ▪ Predominantly review aggregated data not individual deal data.

Usage Pattern	Characteristics
	▪ Occasional access to individual records within their scope of responsibility. ▪ Dive into individual deal and matter data on demand due to exception raised or noted.

Active involvement

Looking at the active involvement roles, the data coverage and usage patterns are often dictated by the value to the business of the relationship.

High value

Considering high value accounts in which a personalized, managed relationship exists between specific employees and customer staff members, it is possible to estimate a realistic volume of customers or cases to which a particular employee would need access. Characteristics of high value interactions include:

- Heavy involvement - Where a deal is high value and, therefore, a personal relationship is developed.
- Minimum of one day's effort per deal – Achieving that level of personal relationship requires a minimum of one day of involvement directly in the deal, either in a concentrated period or over a period of time.
- 200 working days per year - Allowing for weekends, public and personal vacation time, as well as training and other activities, 200 working days a year is a common level of workload to assume.
- Maximum 200 deals per year per person - With a working capacity of 200 working days a year, with a minimum of one day per deal, gives a maximum possible capacity to interact effectively on 200 deals per year.
- In reality, many high value deals will require more time than that, so in practice a user will interact with fewer deals per year than this.
- As an estimate, basing an average on a typical working pattern of half that gives an estimated 100 deals per year, affected by bigger deals taking a greater proportion of time than the minimum.
- With typical data history requirements requiring access to between 5-7 years, this could mean that users could require access to an estimated 100 deals per year, with up to 700 deals including historical access.
- In this scenario, where security rules dictate that only people directly involved with a particular customer or deal can be given access, it is shown that the volumes of deals the user would need to access are within the scope of team ownership to achieve. Team ownership, therefore, can be a good solution to this model of relationship.

Medium value

Characteristics of medium value interactions include:

- Where deals are of medium value to the business, there may still be a level of personal relationship. Where a personal relationship is appropriate there is a minimum level of time that needs to be allocated to build the relationship.
- The minimum time needed to build an effective relationship is estimated at 0.5 day effort per deal.
- Using the same estimate of 200 working days per year.
- Gives a maximum 400 deals per year per person.
- Average typically maximum a third of that, such as 134 deals per year, with the average affected by bigger, more time consuming deals where at such low involvement levels even a single days meeting has a distortion effect on the overall results.
- With typical data history requirements requiring access to between 5-7 years, this could mean that users could require access to an estimated 134 deals per year, with up to 940 deals including historical access.
- In this scenario, in which security rules dictate that only people directly involved with a particular customer or deal can be given access, it is shown that the volumes of deals the user would need to access are within the scope of team ownership to achieve. Individual team ownership, therefore, can be a solution to this model of relationship. Although at this point, the level of complexity of maintenance and need for groups of people to provide cover may mean that more efficient access methods are permissible and which would give easier administration and better performance.
- Having common teams or business units managing multiple deals or customers, rather than having individual teams for each deal or customer, may be a better balance both from a business and technical perspective.

Low value

Characteristics of low value interactions include:

- For lower value relationships, the practical reality is that because of the limited time that can be devoted to individual involvement, these typically do not depend on individual relationships.
- In this scenario instead of designated individuals interacting with particular customers, it is more common for a group or team of employees to manage a segment or type of customer.
- As particular interactions are required, the most available or applicable person picks up the activity as they occur.
- In these scenarios, individuals could potentially need to access a wide range of possible customers. But, in the same way, a wider range of users would also have the potential to interact with any one customer.
- As a result, when managing lower value relationships, using common teams or business units to manage groups of deals or customers is more realistic and effective.

Management involvement

When considering requirements for management access to information, it can be useful to understand that managers' work patterns are typically organized around:

- *Lines of responsibility*, with clear delineation of areas of ownership.
- *Speed of allocation*, with clear allocation rules to enable rapid work allocation.
- *Minimal overhead*, with limited management intervention except in special cases.

Given these factors, managers are typically given responsibility allocated or aligned on hierarchies based on:

- Staff reporting line
- Industry
- Sector
- Region/area boundary
- Client type/ value
- Work type

Managers working in these scenarios typically need to provide a rollup of activity within a clear area of responsibility, such as the financial performance of a division, with access to the detailed data provided only in cases where there is a need to understand or address concerns raised about a particular area.

In these cases, in which a manager is typically responsible for an area of the business that includes a large range of customers or deals, the level and scope of access required can often be dictated by that manager's level within the business. This works well with an organizational structure mapped to business units in Dynamics CRM with managers being given either Local or Parent:Child business unit privileges. In complex scenarios, use of team privileges against non-hierarchical business units gives additional flexibility. The use of business unit privileges provides an efficient way to access the large volumes of data scoped around an area of responsibility that typically reflects a manager's working patterns.

The higher in the management hierarchy and, therefore, the broader the scope of a manager's responsibility, the more likely that the manager needs aggregated information. For scenarios in which there is such a need, particularly at larger volumes, using an integrated BI solution, such as an OLAP cube in SQL Server Analysis Services, may mean that the majority of a manager's access is through reporting directly against OLAP cubes rather than directly onto Dynamics CRM data. For situations in which users do need access to Dynamics CRM data, it's possible that at a higher level of management it is much better aligned with business divisions or area responsibilities that can be efficiently managed by using business units.

Design considerations

When modeling security in Dynamics CRM, there are a range of capabilities that can be used to provide both granularity of access and scalability at volume.

Understanding business needs and scenarios

The first step in modeling security is to fully understand the business needs and scenarios that the solution must support, because environments with varying user types and usage patterns have different needs. Attempting to apply one approach or capability to modeling security uniformly across all the different usage patterns often leads to a belief that the only solution that is capable of meeting all the needs is granular, individual access. This is quickly followed by the realization that trying to apply that granular access at higher volumes, for example to managers, becomes a scalability challenge.

In reality, the more common scenario is that a combination of requirements and approaches can be effectively used to meet the needs of both of those groups of usage types, but in a complimentary and overlapping way rather than attempting to use a single uniform model for all.

User types and usage patterns

When gaining a better understanding of the scenarios that need to be addressed, be sure to consider the breadth of user types and usage scenarios, and identify key usage roles and patterns. For each user type, define the scope of data that must be accessed and determine whether there are ways to align that scope of access around existing business structures or to model by, for example, business division or area of responsibility.

A commonly overlooked aspect is recognizing the difference between two separate factors: the data that a user does not commonly need to access and the data they should not be able to see. Often it may be perfectly valid to have access to data that a user does not commonly need to view.

Similarly, determining what data may occasionally need to be accessed may highlight scenarios that simplify access to data, for a branch at which staff regularly support specific local customers, but on occasion may need the ability to access the data from any other branch for a customer who is visiting the area.

In these cases, it is often possible to simplify access because staff may need access to a broader scope of data than is immediately obvious, but instead using filtering in the user interface to offer an optimized daily experience to regularly used data while still allowing access to broader sets of data as needed. In this case, the security access process is sometimes greatly simplified, which offers more efficient access and therefore a higher level of performance.

Granularity of access

Where a granular access approach is required, be sure to identify the factors that affect scalability. One key consideration is the amount of work that needs to be processed, which is a factor of:

- The number of teams to be checked per request
- The number of requests made, such as every few minutes or once per day
- When requests are made, such as at the start and end of the day, off hours, or throughout the working day

For example, assume that a business has sales staff and managers to support the people working in those roles.

- Members of the sales staff work on deals every day, so they interact with a smaller range of clients but with higher level of activity, for example one interaction every 8 minutes.

- Managers report across a broader range of clients, but they pull reports and access client information less often, only a few times per day.

Although it may be hard to reconcile the workloads of these very different working patterns because sales staff access small amounts of data regularly and managers access large amounts of data occasionally, the actual number of individual records each group accesses may be similar. These two patterns may ultimately place an equivalent overall load on the system, but the way that load is spread over time could prove a difference. If such load and usage were to be averaged out across large user populations, even they might prove nearly equivalent.

Having an accurate understanding of the scope of data access by role can significantly enhance the ability to estimate volume and load, each a key factor on the overall scalability of a solution. Key areas of consideration and specific aspects are listed in this table.

Area	Considerations
Volumes	▪ Teams per user ▪ Currently active records accessed ▪ Inactive records
User roles	▪ Teams per user type ▪ Need for individual access ▪ Need for aggregated reporting
Usage patterns	▪ Frequency of data access ▪ Access times, such as end of week reporting

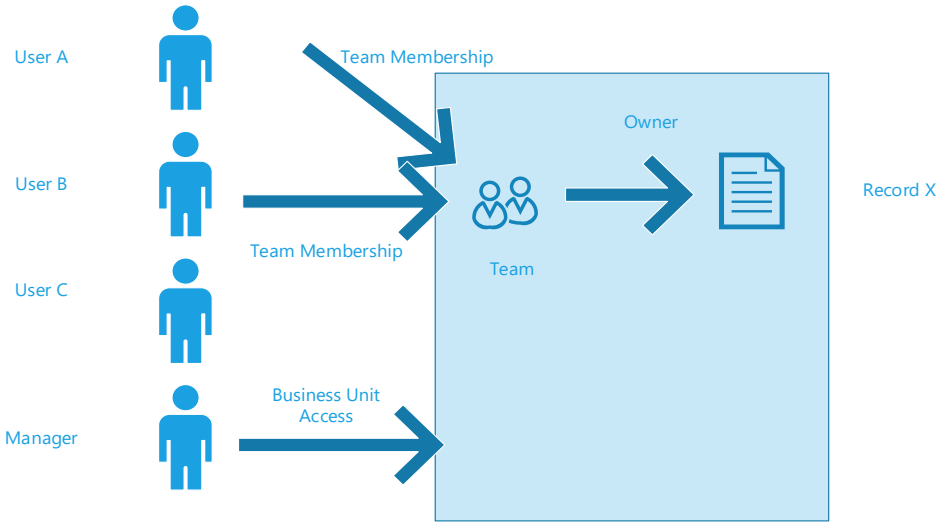
Design patterns

There are several design patterns to consider when modeling high volume security in Microsoft Dynamics CRM.

Separating and optimizing different usage patterns

There is rarely a security model that fits all usages with a single approach in the most efficient way possible. Dynamics CRM offers a range of capabilities that can be combined to offer rich security models for different user types and usage patterns while allowing for efficient and scalable access.

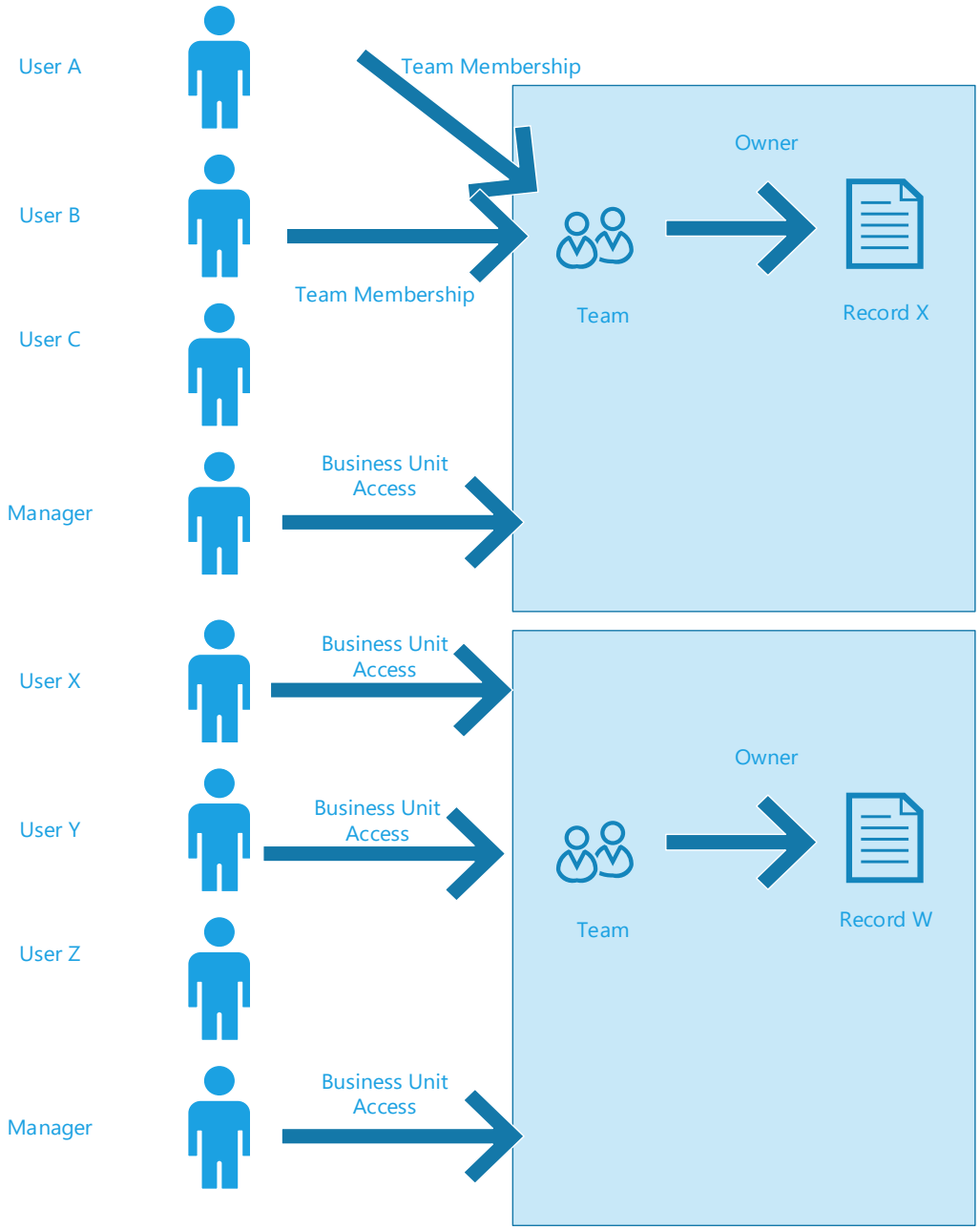
Identifying different user types and then modeling the associated access appropriately can be an extremely effective way to model for high scalability. For example, using a team ownership model for individual sales staff together with a business unit model for management staff can often align well with the way the business works. Using this approach allows for a granularity of access to sales end users together with a rich and scalable model for the managers who are responsible for business areas, providing them with business unit access at scale.



Customizing the security model for different business areas

A commonly encountered challenge is that despite having common job titles, users in different business areas actually work in very different ways. Trying to identify a single, common denominator approach may seem to ensure a simple design with a single mechanism, but in reality it can often complicate requirements significantly because the selected mechanism must become overly complex and potentially granular in nature.

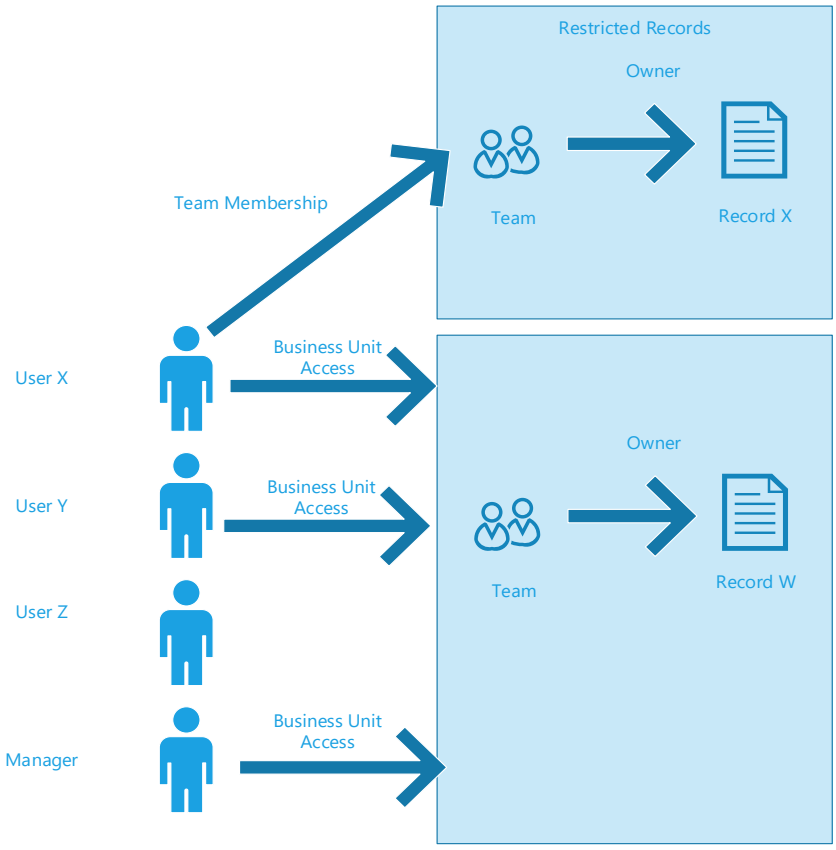
Should this occur, being able to recognize the variations in the ways that different areas of the business work and to model each area independently can achieve simplifications via offering separate security models. For example, this might be this case if one country/region requires individual access while another country/region, either because of different legislation or a different type of client base with fewer stringent privacy requirements, needs an alternative solution. In this scenario, modeling team ownership for one country/region and business unit access for another can significantly reduce the scalability challenge on the solution.



Customizing the security model to account for exceptions

Another common challenge is the need to accommodate specific rare, edge cases. A classic problem encountered is to try and build the exception case into the general model, often leaving an extremely complex model for all. In reality, recognizing and identifying the edge cases and modeling a specific example for these cases can be an effective way to maintain the simplicity and scalability of the solution. For example, consider a scenario in which VIP or sensitive customer records require special handling. Rather than setting up individual access for all records, one might use an approach by which specific records are identified as sensitive, which triggers a subsequent process that moves the subject record outside of the general BU hierarchy and into a specially designated business unit that is accessible only to users with specific team ownership permissions or to who the records have been explicitly shared. In this example, a majority of the percentage of usage follows efficient and general access approaches, while only exception cases receive special access treatment, minimizing the overall impact of their use.

The following diagram shows most users having business unit access to general records, while an individual who is authorized to access a particularly sensitive client is granted that access within a different business unit through membership of a team that owns that client record.



Separating historical data and active data

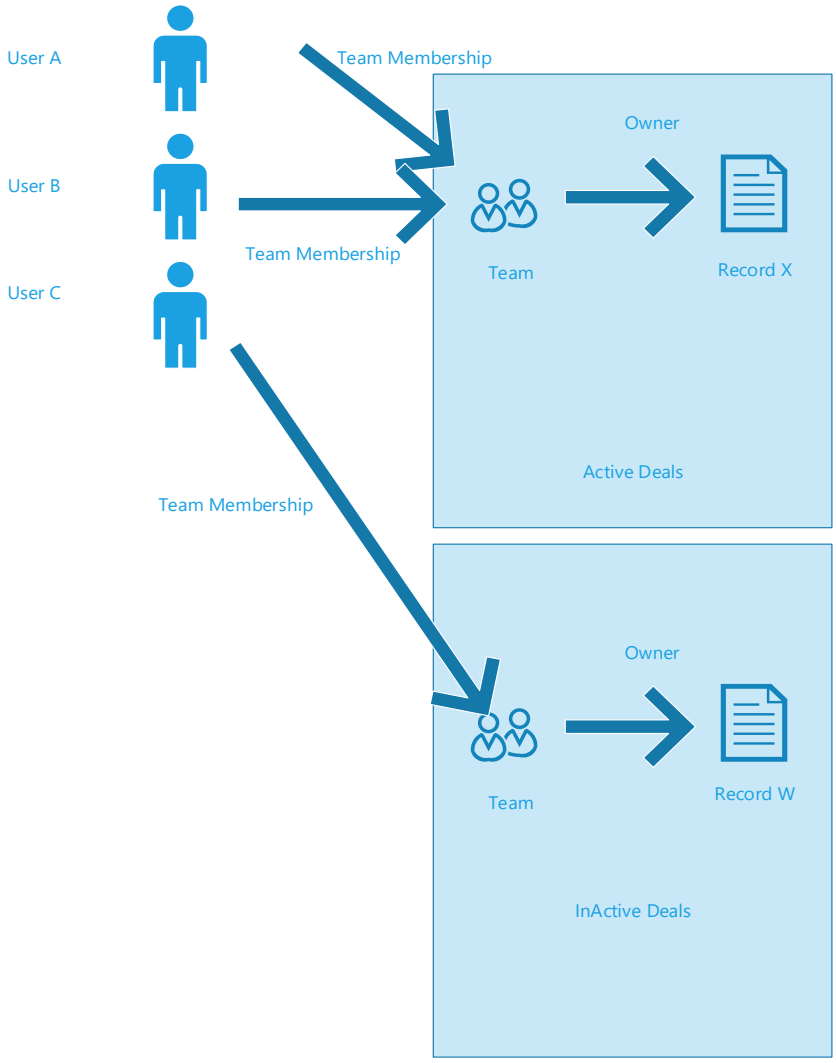
Accommodating historical data presents another frequent challenge. There are many scenarios in which providing access to historical data is important, but including that historical data in the operational system can negatively impact performance, particularly for a solution that uses team ownership or sharing security model. Rarely, however, are access needs for historical data the same as the access needs for active data. Often historical data cannot, and should not, be written to or updated; rather, it is provided in a read-only form merely for reference purposes.

Partitioning data to provide direct access to active data while using a secondary mechanism to address the need for occasional access to historical data can help to optimize for performance. The experience can be presented seamlessly to users by using customizations, but it also allows access to historical data to be provided through a variety of technical means.

Possible approaches to accommodating this type of scenario include providing a:

- Read-only summary pulled from what is stored in a data warehouse or data-mart.
- Secondary instance of Dynamics CRM to which historical data is copied for a suitable period after it is no longer active or at year end.

In the following example, this is shown through access to active current deals in one tenant of Dynamics CRM, but access to historical data is provided through a second tenant of Dynamics CRM to which users can be granted access as appropriate.


User A
User B
User C
Team Membership
Team Membership
Team Membership
Owner
Team
Record X
Active Deals
Owner
Team
Record W
InActive Deals

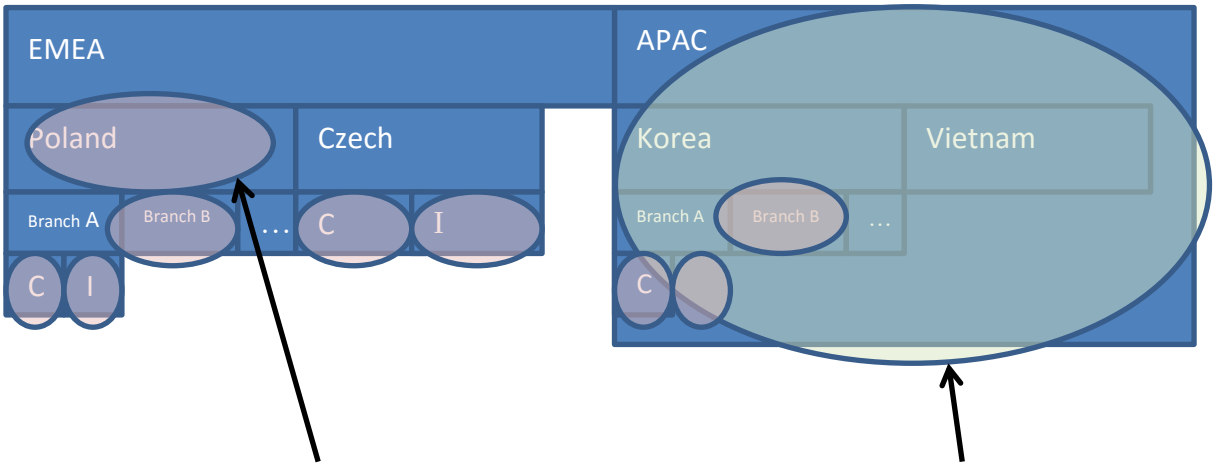
Another option to consider is just in time access to data. Particularly where data has become dormant, while the records themselves may now be held as inactive, any teams or sharing would still be active in the system and could have an impact if they exist in large volumes.
By removing team or sharing access to inactive data that a user doesn't typically need to access can have a beneficial effect on operational access to active data. When a user needs to gain access to historical data, this access can be re-established on demand. This can either be a manual process or, where there are rules about which user can see which data, it could be automated.

Modelling security walls rather than the organizational hierarchy

The instinctive design for business unit hierarchies often is to model them directly on the existing organizational structure of the business. In reality, however, business units usually are used to model the security structures of the system rather than the organizational structure.

There are many cases in which modeling security based on the occurrence of differences in access requirements, not on organizational boundaries, can significantly simplify the overall solution and allow for a more natural and efficient security access model.

In the following diagram, the APAC region has one, all-encompassing business unit, which reflects an organizational structure with one team that can access data from the entire region. On the other hand, in the EMEA region, team control is managed more locally in a classic hierarchical model. The ability to model security boundaries independently of organizational boundaries allows for the representation of actual working practices, which may not be directly reflected in the organizational structure.



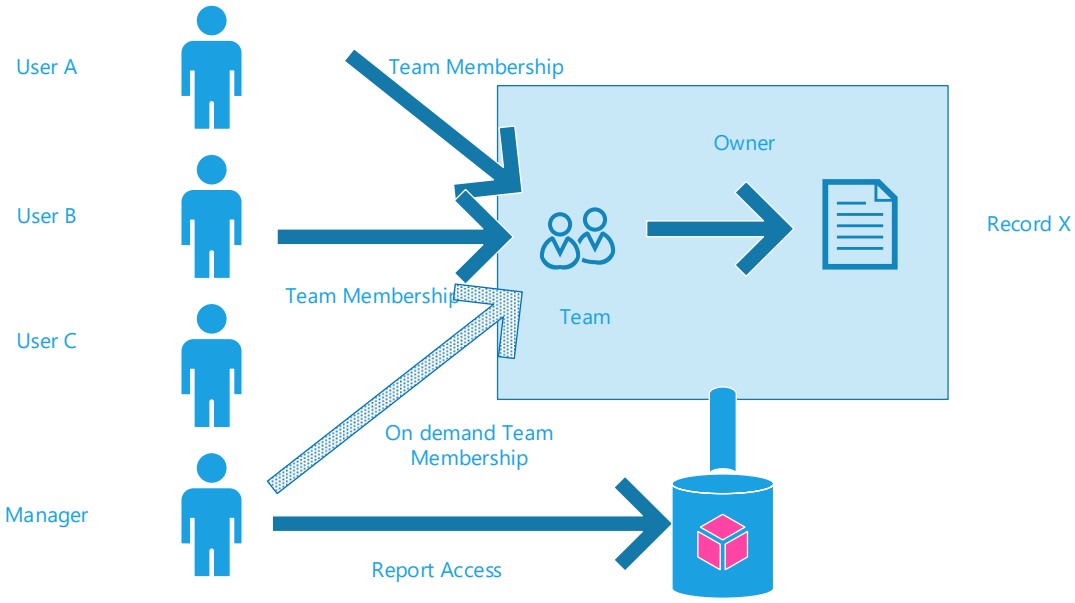
There are other cases, particularly in the banking industry, in which customers have a global presence and may need to interact with exchanges around the globe on a 24-hour basis. In such cases, it may be useful to manage these customers separately from the more typically geographical structure used to manage smaller, more local customers. This provides a centralized business unit to cover the global customers while maintaining a country/region business unit structure for smaller customers may be an appropriate approach here.

Providing separate reporting

Many of the more complex security requirements arise when trying to model different matrix structures of overlapping responsibility in a business. Often, the level of access required to support the management layers of these matrix structures is for reporting rather than operational access to data.

As a result, for scenarios in which managerial access is required and different perspectives on the data are needed (which complicates the security model), it is often worthwhile to consider separating reporting and operational needs to provide the best solution. Using a tightly integrated BI solution for differing reporting needs, for example different OLAP cubes for different reporting hierarchies, can provide a satisfactory level of separation of access needs while at the same time segregating the reporting workload from the operational system. In some cases, this can allow for implementing the security model in a different way or might eliminate the need for it completely for scenarios in which the aggregation of OLAP provides data organized to preserve anonymity. So, there is no need for further security controls.

Often providing pre-defined reports that query data as a super user but that include preset filters to align results with users' permissions or responsibilities can simplify the complexity of processing without the need for the security model to provide it generically. In the following example, all users accessing individual records access the Dynamics CRM instance directly, but managers viewing reports would instead have that data provided by an OLAP cube built from the operational CRM data store rather than reporting on Dynamics CRM directly.



Each of these cases, though using different approaches, allow for potential simplification of the operational security model by reducing the complexity of the requirements by separating the reporting need to an alternative mechanism.

Controlling versus filtering

When considering individual access, it is easy to conclude that limiting a user's access only to data required for his or her primary activity is essential to locking down that user's access only to collaborations in which he or she is directly involved. However, consideration of secondary roles or activities, such as covering other people's activities or taking a call from a customer unsolicited, may indicate that users actually need to access broader sets of data than are initially indicated. For situations in which there are no policy-related or legislative reasons to enforce access controls on the data, it can often become more a case of filtering the user's primary access to data based on the perspective of user experience while still allowing the user to access broader data to accommodate exceptional circumstances.

In these cases, improved performance and simplification of solution benefits may be achieved by providing this user experience-based approach by automatically filtering data in the standard views that the user accesses rather than by implementing a complex security model.

For situations in which there are concerns about the actions that a user may take, often the concerns can be addressed by using audit records rather than prevention. This deters fraudulent or irregular activity with the promise of being caught rather than preventing the activity in the first place, particularly for situations in which a user has legitimate business needs on occasion to access records outside or his or her primary area of responsibility. Whether or not this is a valid approach often depends on the consequences of the potential actions, should they be allowed. For situations in which potential gains are high or consequences are prohibitive, enforcing prevention may be necessary. But if the consequences or potential gains are lower, recording the behavior for later compliance checking may be sufficient and appropriate.

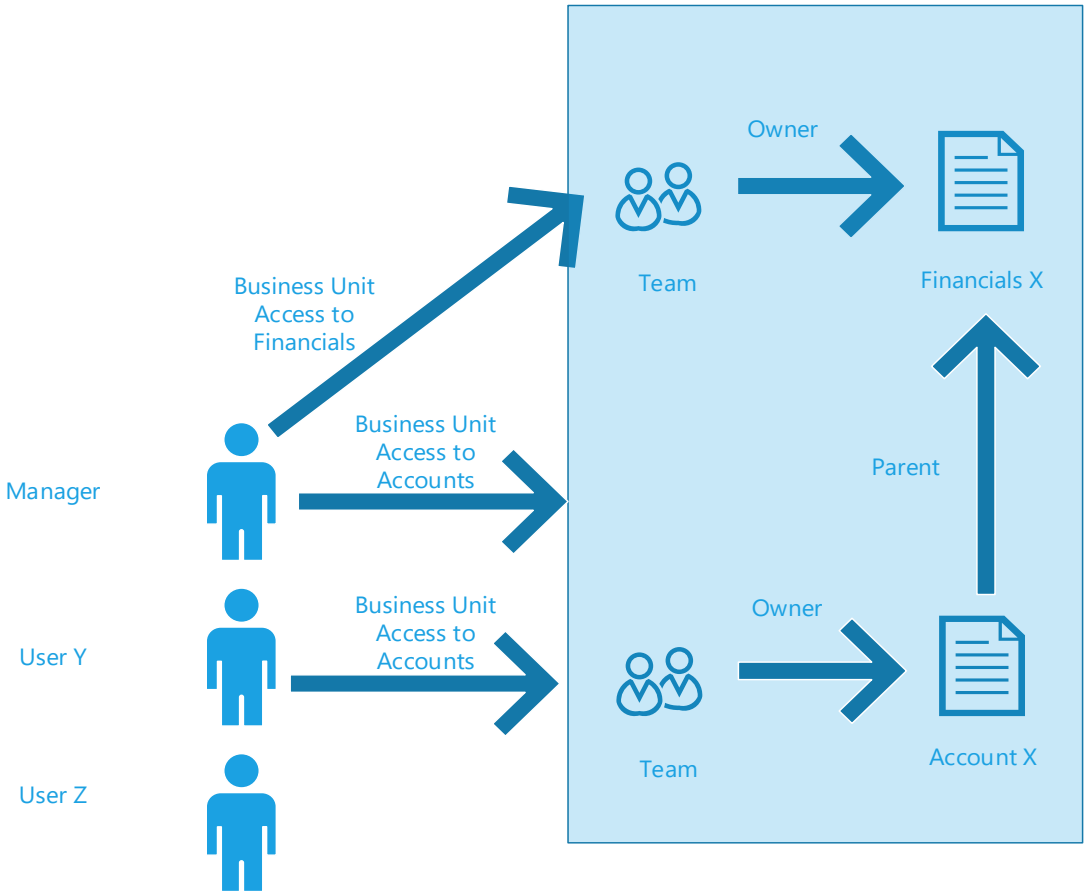
Modelling data along security lines

The first inclination when implementing a solution may be to model data the way it is in an existing system or around tangible objects, such as Account or Company, from the real world. However, taking this approach often can hide logical distinctions in data that actually reflect the way that users need to control access to data. A good example of this often relates to the financial information, such as an account's annual business with the company. While it makes sense to display as part of a customer record, often it is information that is calculated and recorded on an annual basis and that is more sensitive or competitive in nature. As a result, the information is instead made available to a more limited group of users, such as sales, but restricted from access by servicing.

The implication when not modeling this data separately is often that more onerous security controls are imposed than are needed. For example, it may be in this scenario that the Account entity requires granular team ownership access while in reality the only rationale is to restrict access to the financial data. Modeling this financial data as a separate entity could provide one of two potential beneficial outcomes:

- Recognizing that the only users accessing the financial data can be grouped directly into a team or role that is then separately granted access to the Financials entity.
- Restricting the granular lookup access checks only to the less commonly accessed Financial entity, which while not reducing the complexity of the access, does reduce the frequency with which the data can be viewed than would be the case for a commonly referred to record, such as Account.

Modeling data in this way to reflect the impact of security access provides the opportunity to consider different perspectives on the data, which could introduce new data boundaries that could simplify the security modeling approach that is used.



Security role versus privilege

When customizing solutions, one need that surfaces is the ability to control the permissions of a user to certain custom actions. A debate that often surfaces is whether to control the action through a security role or a privilege.

Within Dynamics CRM, the model used is that privileges define the specific actions or data access a user can perform. Security roles are used to group these privileges for a particular business role of a user.

The recommendation is to key any custom processing from a privilege that a user holds rather than a security role. There are a number of reasons why this is beneficial:

- Privileges are cumulative.
    - If a user is granted multiple roles, the combination of their rights to particular actions is determined by a cumulative calculation of their privileges from all roles.
- The cumulative privileges assigned to a user are cached and optimized for querying, security roles are not.
    - Where testing the rights of a user to perform an action will occur a lot, as it often would if used to determine if a user interface element is enabled or not, this needs to be as performant and scalable as possible.
    - Making a request of Dynamics CRM to ask for a privilege will be quicker to return and will have less impact on the scalability of the system because once the cache is loaded for a user the request can be served from the cache. Requesting a security role will require a database request for each query.
- Allows more business flexibility.
    - As businesses change, the actions individual user roles need to perform may change. Managing the right to perform an action through a privilege enables changes in role responsibilities to be performed by changing a security role centrally.

- If more granular security roles are used and the role itself is used to define rights for a custom action, any change in responsibility will require the roles granted to individual users to be changed. This is often a significant amount of change to propagate out to the user base.

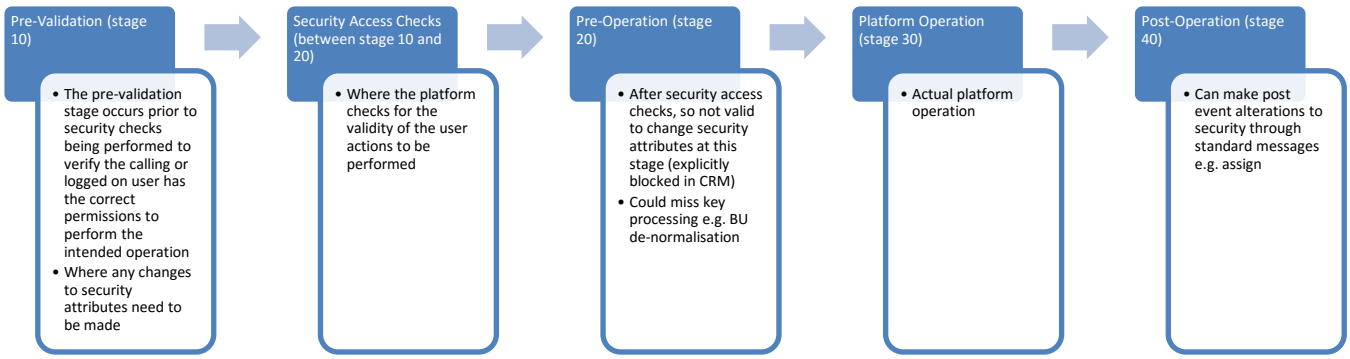
It is therefore recommended to follow the approach used by the platform itself. Use privileges to define the right to perform a particular action and security roles to collect these actions into usable groups of rights for particular common business roles. Directly querying the security roles to determine the rights to particular actions is not recommended.

Controlling security through automation

When automating the manipulation of security attributes, there are some constraints that need to be considered.

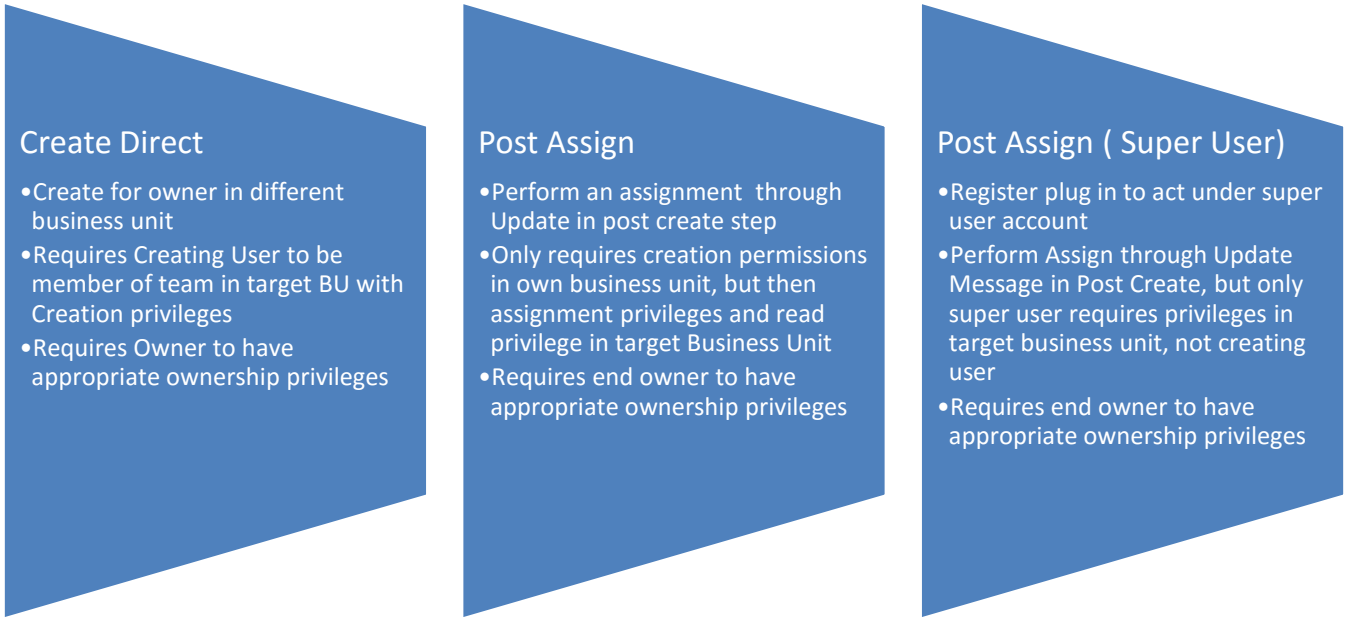
The primary attribute that is relevant here to be changed programmatically is the owner of a record. This is often updated either to specifically reassign a record to a different user or team or to assign a record to a different team to implicitly change the business unit of a record. Because the business unit of a record cannot be directly set, changing the business requires change of ownership and the business unit of the record will automatically be changed to that of the newly owning user or team.

In a create or update message, security attributes can only be edited in stage 10, before the security attributes are checked by the platform against the operation being requested, which occurs before stage 20. Modifying security attributes beyond stage 10 is not permitted. Starting with Microsoft Dynamics CRM 2015 Update 1, using the Assign message to change ownership is deprecated in favor of changing ownership using the Update message.



Reference SDK 'Event Execution Pipeline'

"The pre-validation stage occurs before security checks are performed to verify that the calling or signed-in user has the correct permissions to perform the intended operation" More information: Event execution pipeline



Optimization example

As an example of applying the optimization approaches suggested previously, a case study will be presented showing a common challenge and approach to resolve it.

In this scenario, granular access to particular customer records and deals is required, particularly to ensure that individual sales people can only access the deals they are involved with.

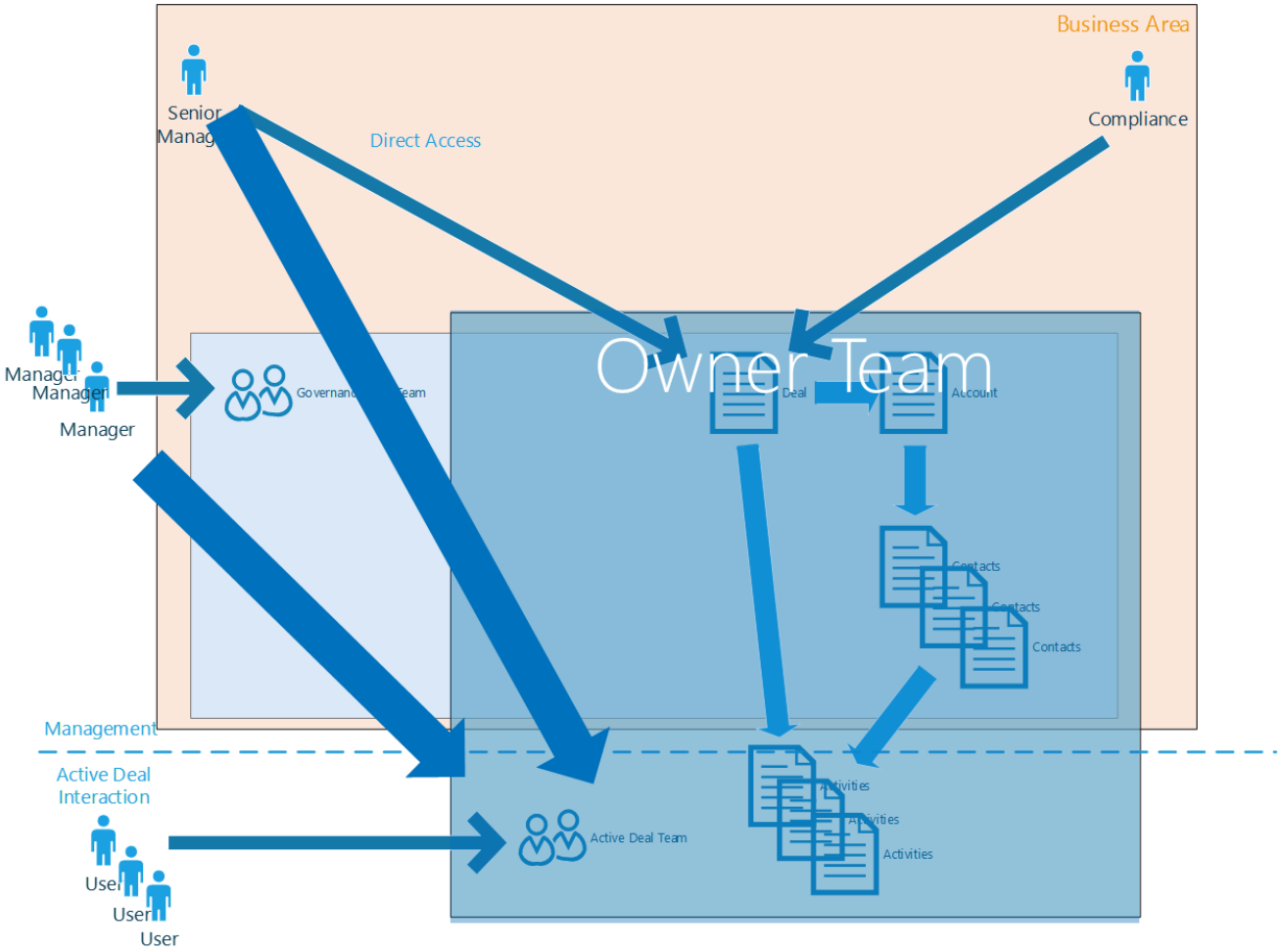
As part of the design analysis, there are some key metrics we will consider that we need to optimize and balance as much as possible.

Response time: Senior managers	•User interface response times for senior managers •Time to view the data they need
Response time: Sales people	•User interface response times for sales people •Time to view the data they need
CRM Server CPU	•Overall loading of the CRM Server CPUs
DB server CPU	•Overall loading of the DB server CPUs
POA table usage	•Use of the Principal Object Access tables in CRM Used for sharing within CRM

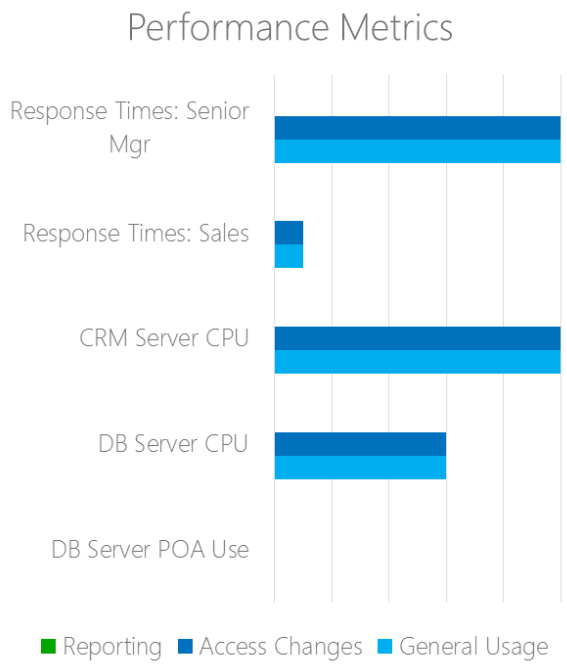
Original approach

The original approach for this problem was to tackle it from the bottom up, recognizing that individual users required access to specific deals that they were working on. And by derivation from that, each manager that sales person reports to directly or indirectly should also have access to those deals.

As a result, the following design approach was considered where an owner team was created to own the client and deals and all the users who either directly work on the deal or are in the management chain above will be allocated as members of that team.



The challenge that this highlighted, however, is that, due to the volumes and granularity of access, this led to some significant scalability challenges. In relative terms, the impact on our key indicators is illustrated here.

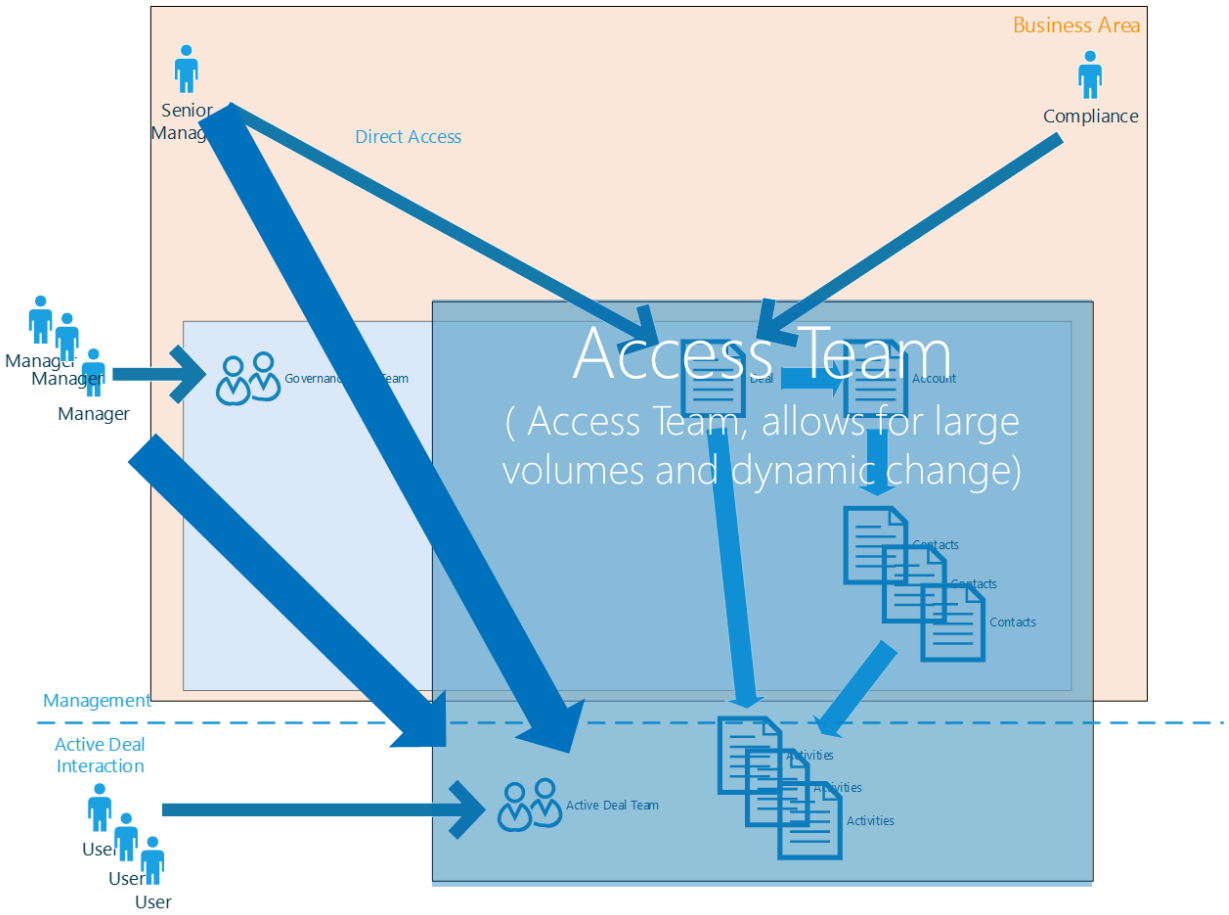


As the illustration shows, this not only had major impact on the response times for senior managers, but also on the underlying resources of the platform.

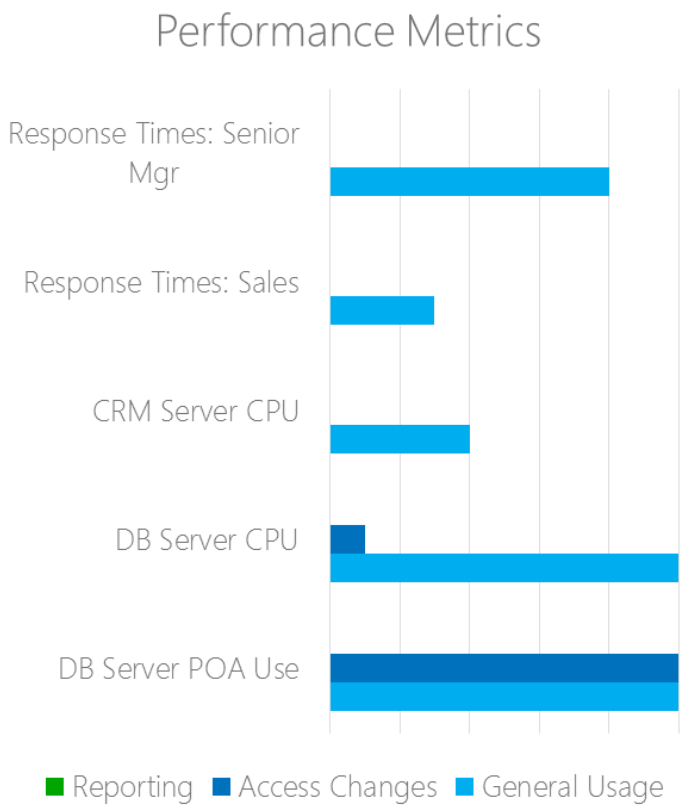
Alternative sharing approach

One common problem to consider is that the impact on senior managers is the main issue and these users will be members of many teams and their team memberships will change regularly.

With that in mind, owner teams are not the best fit for them, whereas access teams and sharing may be a better fit. So a common approach is to change to use sharing to the deal teams rather than ownership.



The concern here is that while this does avoid impact as new deals are created and the senior manager's team memberships therefore dynamically change, this moves the problem to another area.



By moving to a completely sharing based model, the POA table is now much more heavily used with a corresponding increase in database resource as well as an increase in the response time for the sales team.

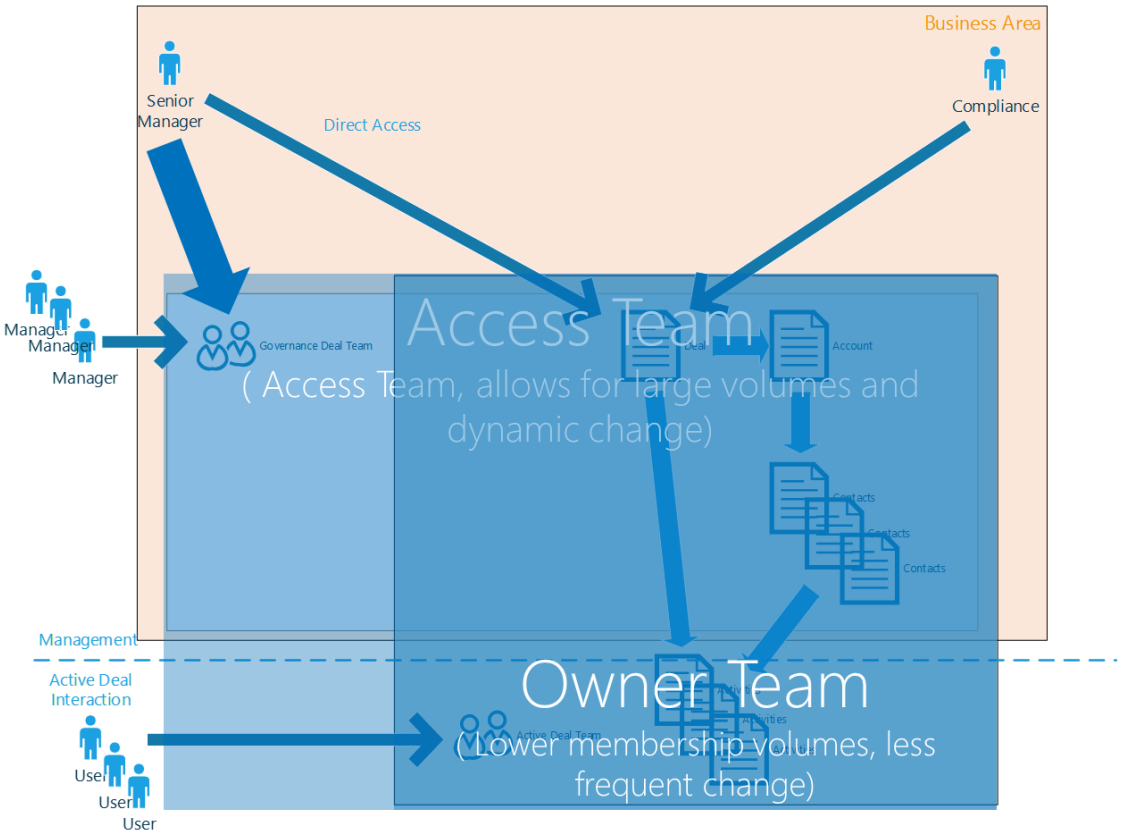
Use owner teams for sales, access teams for management

Looking at the different usage characteristics of sales and management users, instead of trying to fit both into a single model, targeting a more appropriate model to each user type may help here. In particular recognizing that:

- Sales users will
    - Access a smaller number of clients and deals and that set of clients and deals will change less frequently.
    - But they will access the system more frequently and access more details of those records.
- Management will
    - Have access to a much wider scope of clients and deals and that set of records will change much more frequently.
    - But they will access the system less frequently when they go in to review the state of the business or react to specific events.

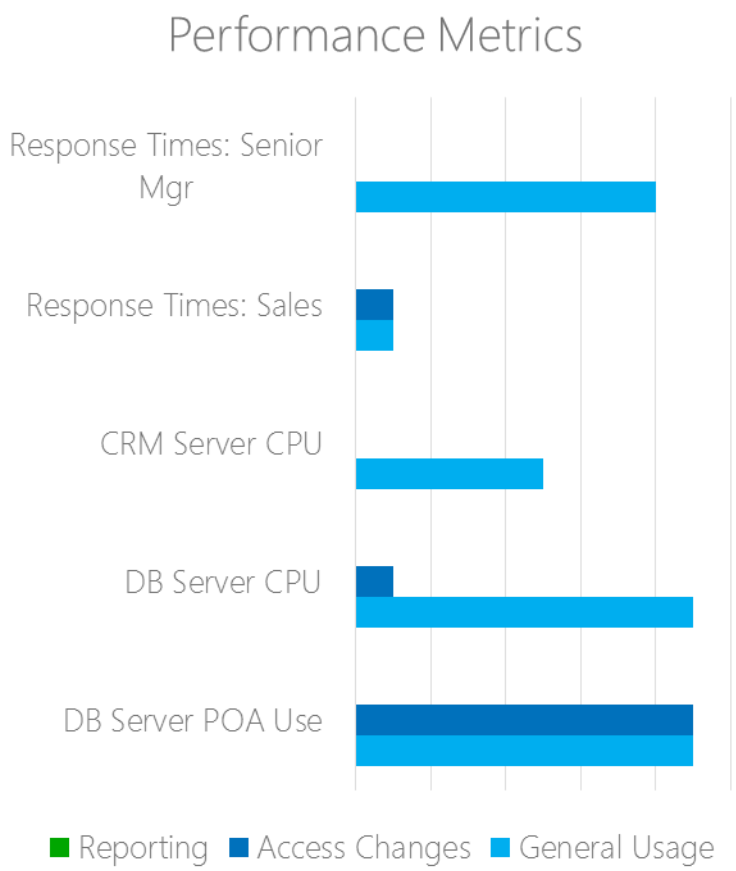
This allows us to consider a model where we use:

- Owner teams for sales people: These offer a very efficient model for smaller groups of team memberships that change less frequently.
- Access teams for managers: These offer an efficient model for larger team memberships but also where the team memberships change more often.



The other advantage of this approach is that it shares out the load on the system. Owner teams predominantly hit in terms of CPU use on the web server as we iterate through all of each user's teams as they access records to check if it grants ownership access to a record. Access teams and sharing use much more in the way of database server resources as we do a lookup of the large POA table to the appropriate record being accessed. Splitting these out can scale out the needs on the system across different components giving more scalability.

This therefore changes the system impact as illustrated here.

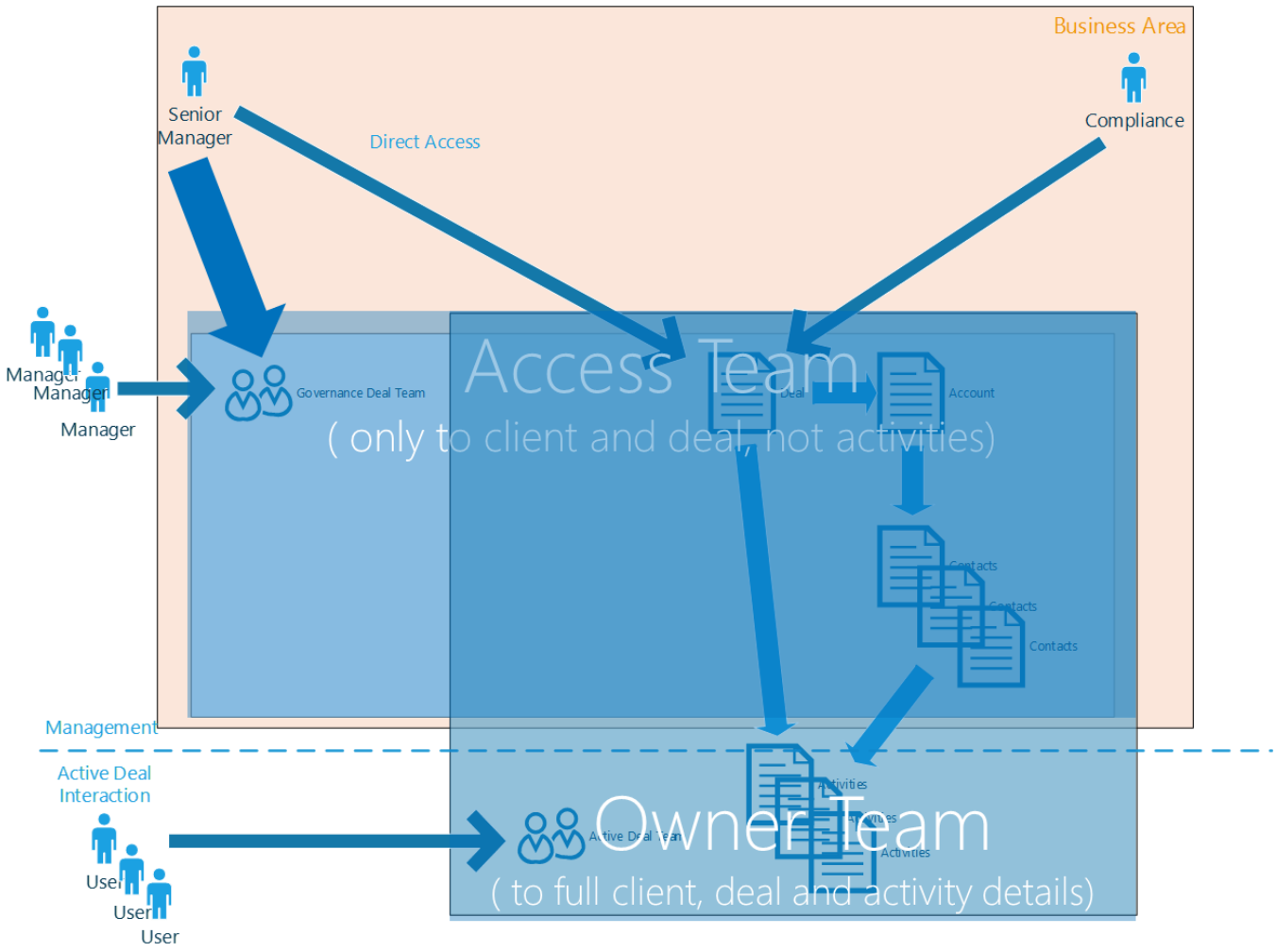


This hasn't helped with the manager response times as there is simply a lot of data and access checks to process. But what it has done is help make sure this is more consistent and does not hit major spikes whenever there is a new deal the manager is added to. Nor does their use impact on the sales users accessing the system so significantly improving their performance.

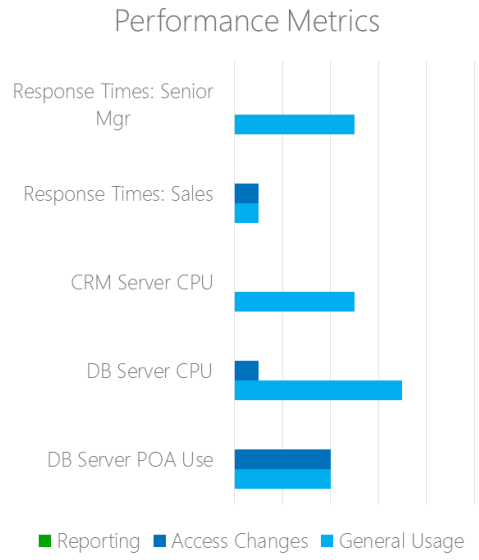
Consider different role needs

The other factor we can consider is the different needs of a sales person and a manager. As a manager is looking for higher level insight to the state of the business, the need to review individual activities of every client and deal is not necessary. In fact it may well lead to privacy or conflict of interest concerns for a manager whose role is to manage business having access to very detailed information about a particular client.

It may therefore be possible to recognize that the middle and senior managers do not need access to the detailed activity records. As they represent the majority of the records in the system and in particular the majority of the POA records as each is individually shared, removing the need to share these to the middle and senior managers can have a significant impact on the overall scalability.



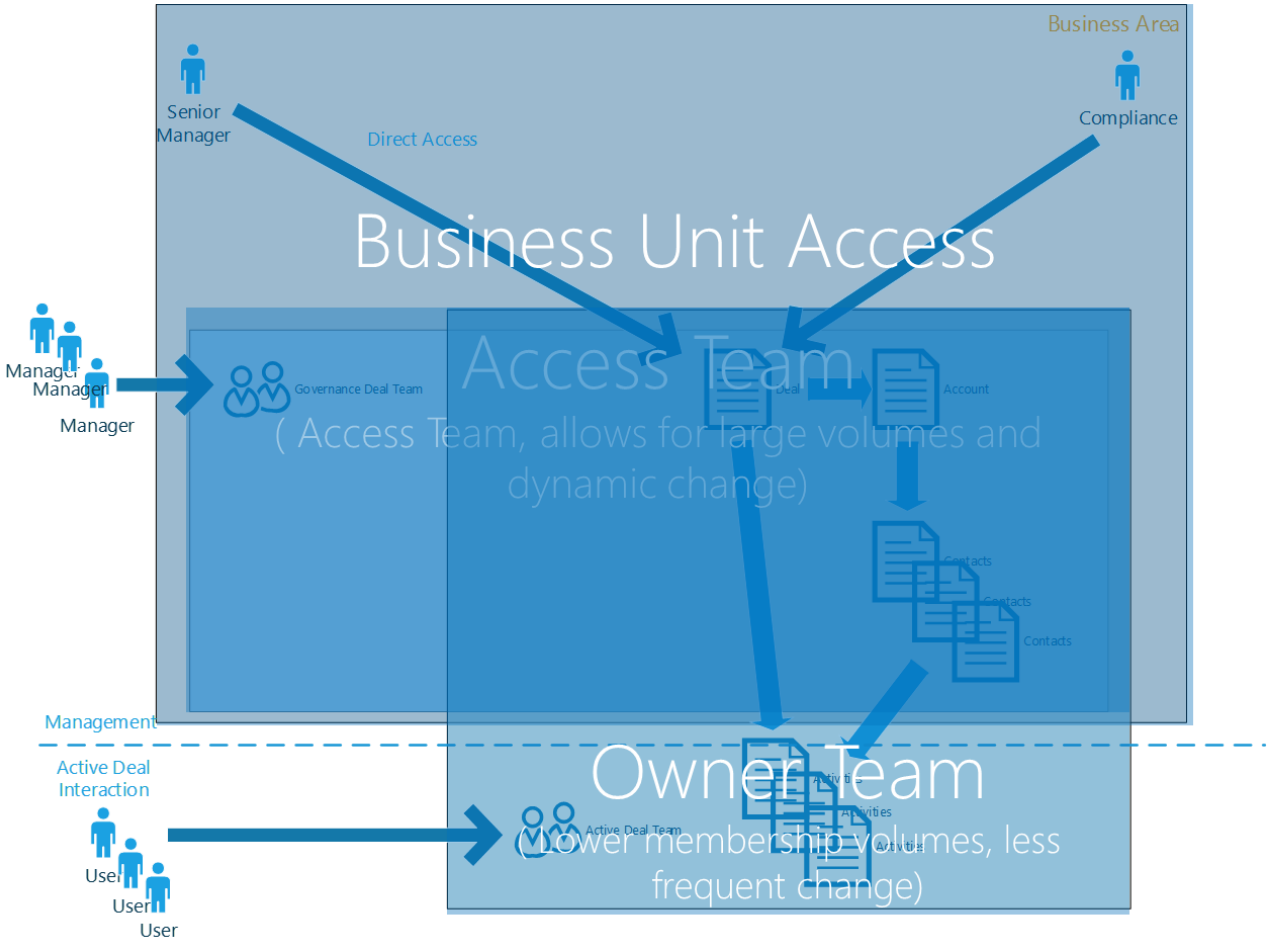
This change therefore helps reduce the overall POA use, which in term reduces the performance impact for senior managers as any access they perform now is not slowed by the sheer volume of data in the POA table.



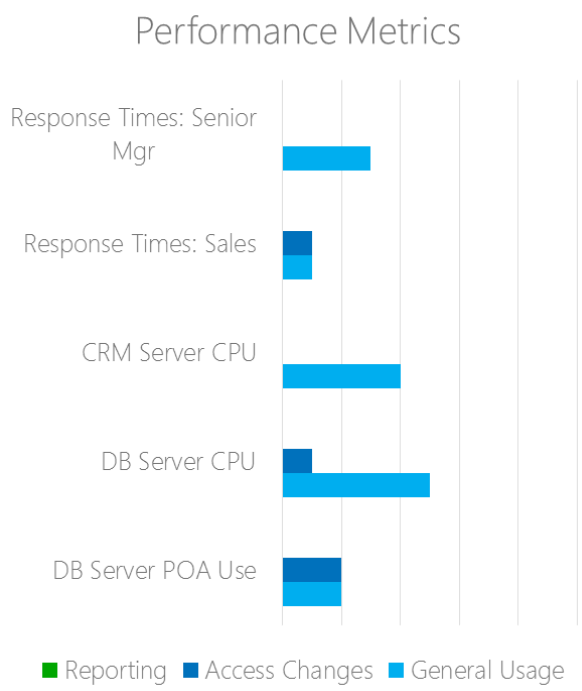
Use business units for senior managers and oversight roles

At very senior levels, many managers have access to all records or significant areas of records. These can often be managed through business units, granting access to whole areas of the business.

The benefit of this is that for larger volumes, controlling accessing this way is much more efficient than individual record access and allows even more of the sharing in the POA table to be reduced.



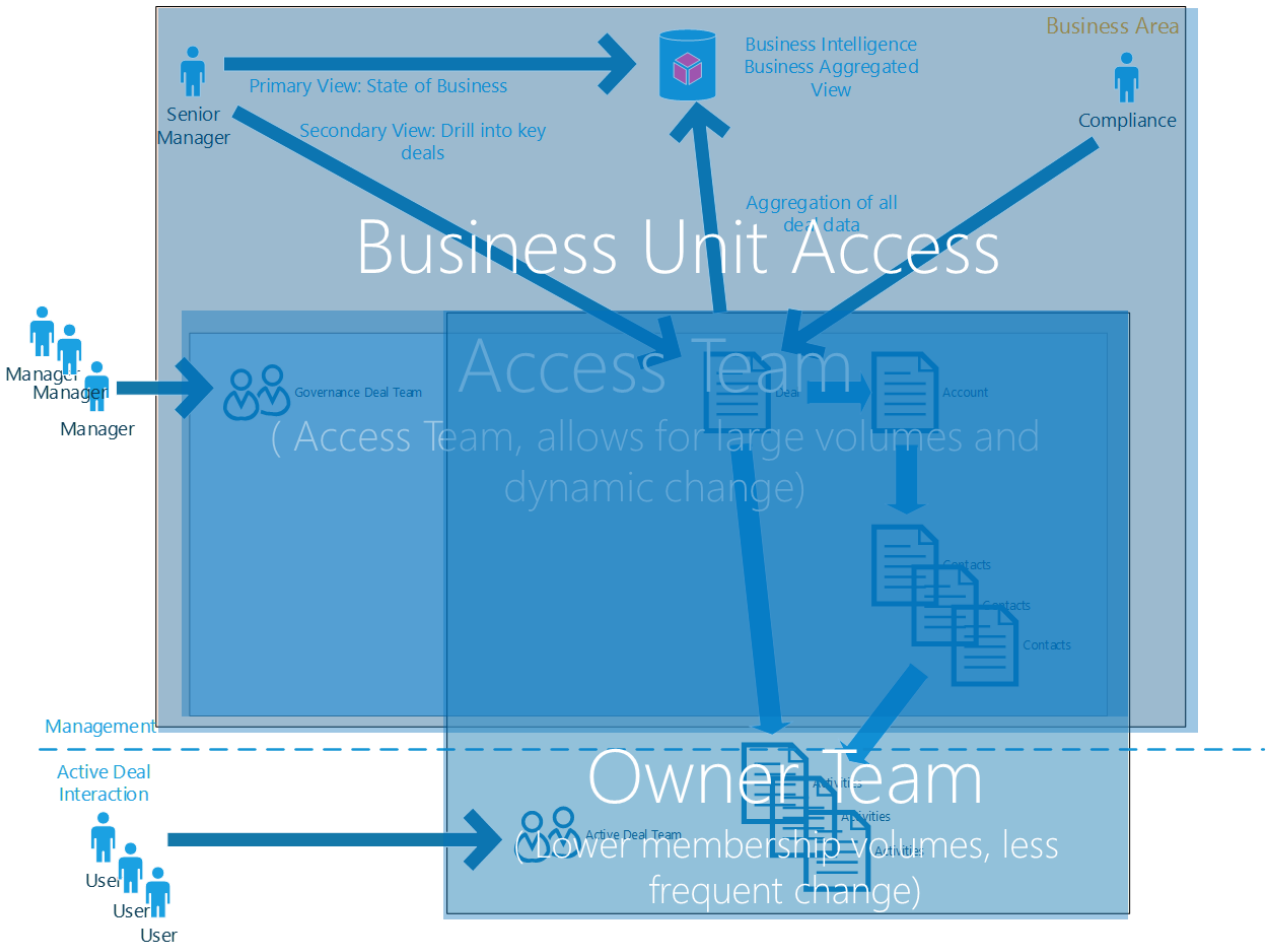
This further balances the resource use and optimizes even more the response times for the senior managers.



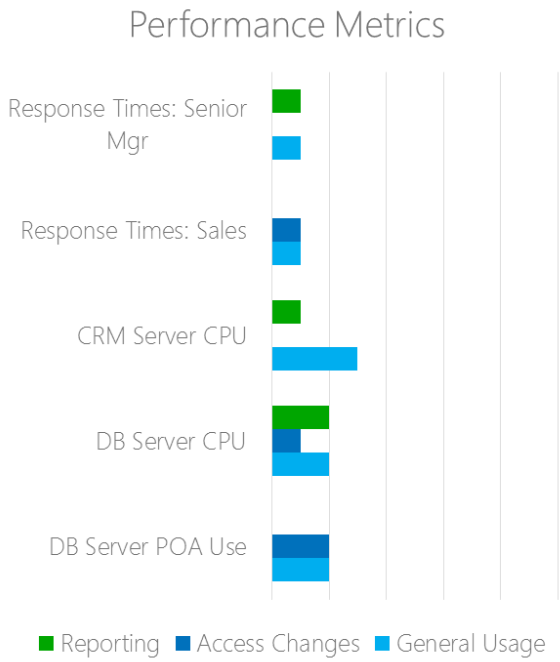
Use reporting

At management layers, even more than controlling access, the method of consuming data is typically different. Management personnel are typically interested in gaining an insight to the state of the business through aggregated summaries. They may then drill through into particular details, but the initial overall summary is much more useful.

Reporting can be a very useful tool here, to provide a more relevant insight to these users, but also to offload workload from the operational system.

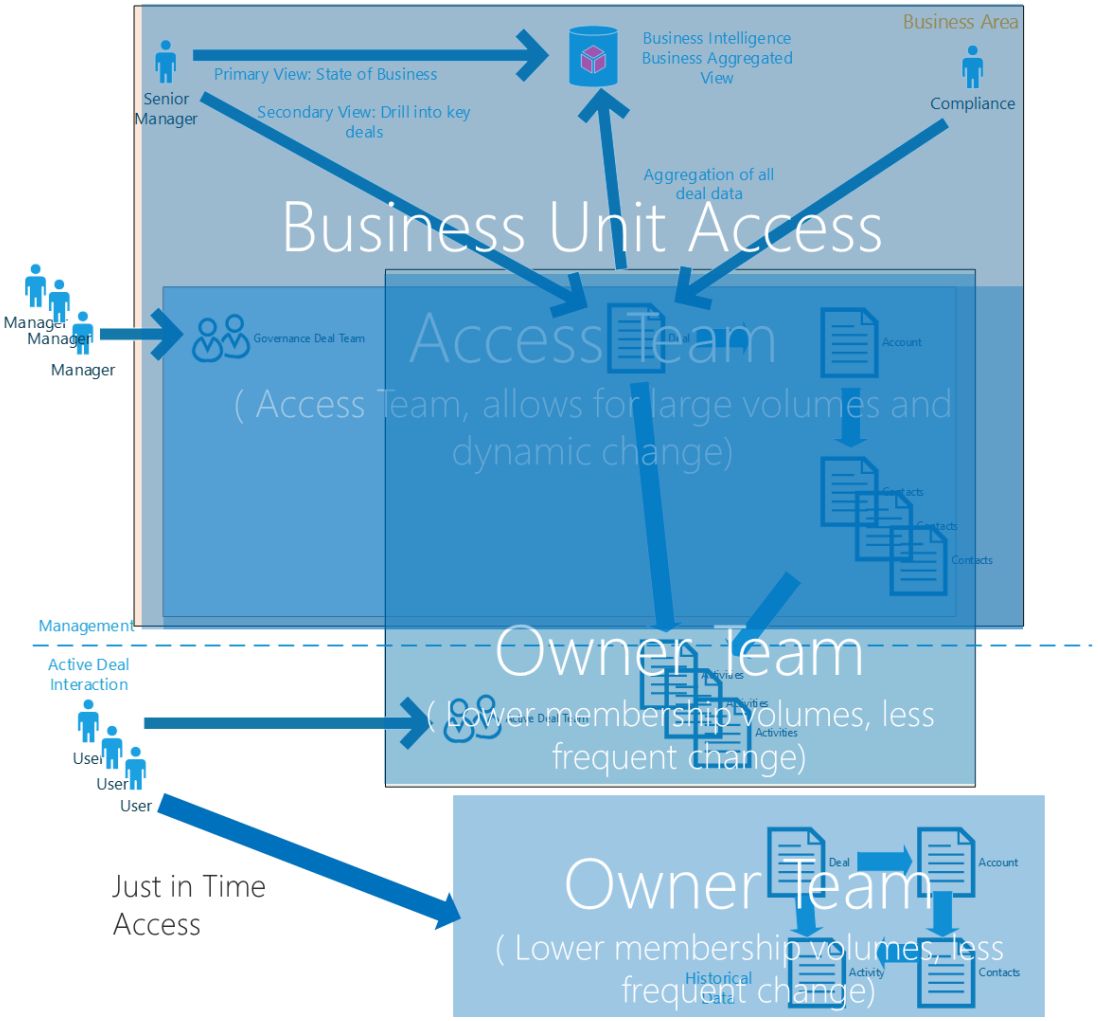


By using reports, and possibly even overnight snapshots or OLAP cubes to move much of the processing away from the daily operational system, even more benefit can be gained to the overall system scalability.

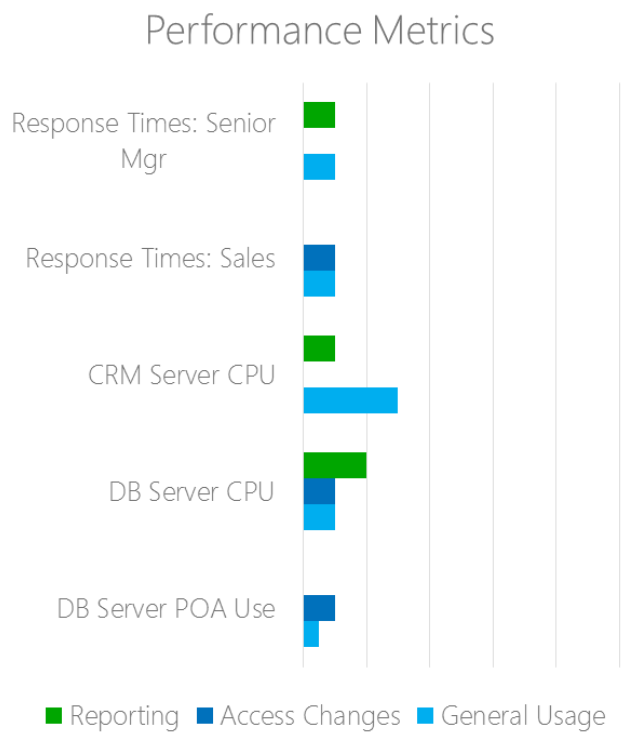


Archive historical data

Realizing that much of the scalability impact is now coming from sheer volumes of data, both record data and teams and sharing, moving historical data to an archive if it's not actively required can have another significant impact on the system.



Moving data out of the way so that the team memberships and sharing are not there for inactive data and granting access to users "just in time" as it is required to historical data can also reduce the impact on the system and therefore improved user experience.



Overall assessment

As has been shown, while there was no single immediate approach that provided the optimal design for this problem, a combination of approaches can be used to best model the solution to a particular problem.

Breaking down the need and looking at different aspects of the solution and then modelling and optimizing in a balanced way is often the best way to achieve the most scalable and performant outcome. It often also leads to a better experience for the user functionally as well as from a performance perspective.

Summary

Dynamics CRM offers a variety of security modeling capabilities. Each capability has a natural fit to different scenarios of access control, but each need not act in isolation from the others. Instead, multiple capabilities can be combined to achieve the desired result.

In more complex, larger scale implementations, it is more natural for each mechanism to play a part within the broader view of security access. A key element in defining the most appropriate capability to use depends on understanding that accommodating different user types and usage patterns are independent needs. With a clear picture of the specific needs of a given customer, you can design a security model that addresses each aspect with the right blend of security access controls, addressing multiple requirements when possible but using independent controls and approaches as appropriate. This can also lead to an overall simplification of the solution implementation as well.

While features such as sharing and team ownership can play a key part in constructing these rich security models, it is important to recognize that the granularity of control offered by sharing and team ownership has a related cost in terms of the amount of processing that is required. For large implementations, using team ownership to provide tight control, while at the same time supporting other models for broader access, can often be the best approach.