Scalable Security Modeling with Microsoft Dynamics CRM 2013

VERSION: 1.0

AUTHOR: Roger Gilchrist
COMPANY: Microsoft Corporation

RELEASED: October 2013

Copyright

Feedback

To send comments or suggestions about this document, please click the following link and type your feedback in the message body: http://go.microsoft.com/fwlink/?LinkID=328762

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

Microsoft Dynamics CRM 2013 offers 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 a user access 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 within the system they are permitted to see or do

Authentication in Microsoft Dynamics CRM 2013 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 user identity 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 is therefore out of the scope of this document.

After a user has been identified, information recorded about the user within the Microsoft Dynamics CRM 2013 system itself, such as the associated security roles and team memberships, is used to determine whether that user is allowed to use the system and what that user is allowed to see and act on within the system, or what that user is authorized to do.

This paper describes how these security modeling features in Microsoft Dynamics CRM 2013 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 Microsoft Dynamics CRM 2013 security at scale, incorporating teams as appropriate.

Important: For additional information about scalable security modeling in Microsoft Dynamics CRM 2013, see the topic *The security model of Microsoft Dynamics CRM* in the Microsoft Dynamics CRM 2013 SDK.

Common business scenarios

In most CRM implementations, access to information is either provided openly within the organization or it is 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 would not normally be exposed to them.

While there is no one-size-fits-all model and different business/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 e.g. providing cover for absence of actively involved staff ▪ Support others who have a personal relationship with customer e.g. providing advice/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 centre
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

Within 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 with no active business is taking place at that moment in time.
- 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.

Interaction value	Characteristics	
Low	▪ Minimal investment ▪ Transactional relationship	▪ No personal relationship ▪ Wide access
Medium	▪ Proactive management ▪ Perception of personalized approach	▪ Group relationship/access
High	▪ Large time investment ▪ 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 is not a specific pattern to 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 is important to be able to grant permissions to individual records or sets of records (such as a case and all the supporting activities related to the case) only 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 examples above 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 do not require the extra controls, particularly 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, 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.

For situations in which it is important to control of access to the individuals directly involved in a deal or case, team ownership can become an effective way to model access to information. It is also worth noting that 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, and their access needs to be revoked. As a result, be sure to carefully consider the lifetime of information access permissions.

Microsoft Dynamics CRM access control options

Dynamics CRM offers a variety of ways to model user access to information, and understanding the options available is vital to determining to the most appropriate solution for a particular scenario. The following sections provide additional detail about:

- The objects to which security access can be granted (security principals)
- The components that are used to grant security access
- A range of approaches for providing users with access to information within Microsoft Dynamics CRM.

Security principals

Access to data and actions within Microsoft Dynamics CRM is provided by granting privileges to specific security principals. In Microsoft Dynamics CRM 2013, the term *security principals* refers to system users and teams.

System users

A system user can either be a physical person or a logical identity (for example, providing integration with other applications) that has been defined as a member of a Business Unit. Each system user has a distinct identity for accessing Microsoft Dynamics CRM.

Teams

A team is a logical entity in Microsoft Dynamics CRM that is used to model common groupings for security purposes. As with system users, teams are defined as members of a Business Unit, but teams cannot define an identity with which to access Dynamics CRM. Each team can contain zero or multiple users as members. To provide for modeling a wide range of optimized scenarios with teams, Microsoft Dynamics CRM 2013 introduces two types of teams:

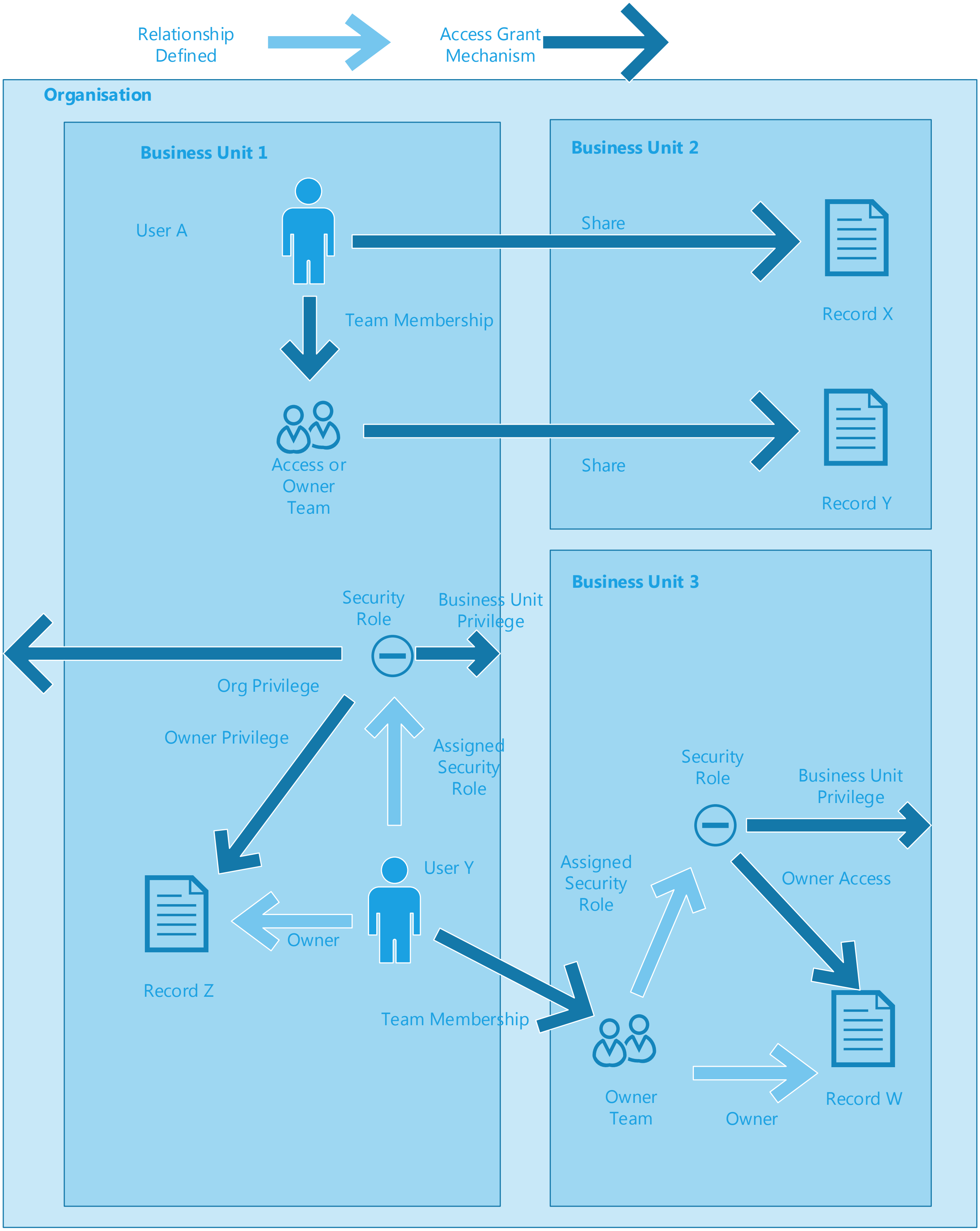
- **Owner teams**. Owner teams are similar to Microsoft Dynamics CRM 2011 teams that have been assigned one or more security roles. As their name implies, owner teams can own records, and they can act as resources in Service Scheduling. The owner teams of which a user is a member are cached when that user access the application. Owner teams must be create and managed manually or programmatically; they cannot be system managed.
- **Access teams**. Unlike owner teams, access teams cannot be granted security roles, own records, or be scheduled as resources in Service Scheduling. Access teams are not cached (because access is not derived using a privilege or ownership check) and are not displayed in most team views. Access teams can be system managed directly from the Form associated with the record to which it relates. They can also be managed manually by the user or programmatically through the SDK.

Access control components

The access control components for modeling security in Microsoft Dynamics CRM 2013 are described in the following table.

Component	Description
Sharing rule	A mechanism to define granular and explicit privileges to particular records
Organization	A particular Dynamics CRM instance, typically modeling the data for a business organization; organizations are used in scoping terms to indicate that a security principal can see any record in the system of a particular type of data.
Business Unit	A scoping mechanism that defines a grouping for security modeling purposes; business units are hierarchical in nature.
Security Role	A collection of privileges that are applied to groups of records scoped by ownership, business unit, or organization; security roles are assigned to users or owner teams
Privilege	The definition of a specific type of data access or action that can be granted as a right to a security principal; privileges are granted through a security role and are cumulative.

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

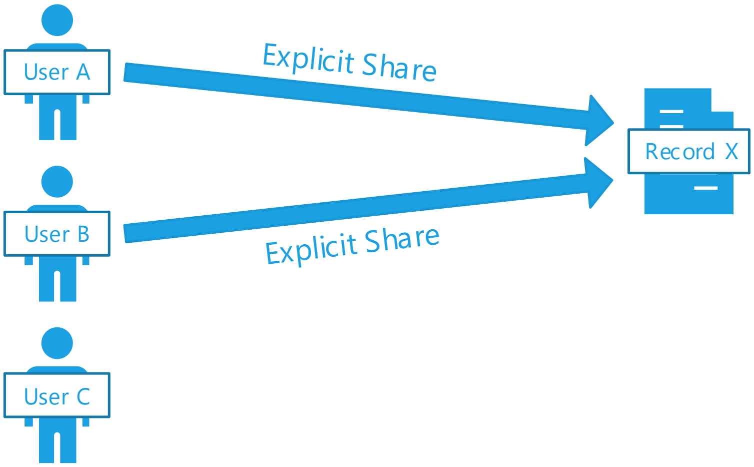


Sharing

Sharing is the act of directly granting privileges for a particular record to the users or teams who should have access to it. The grant can specify the type of access and actions, such reading or updating, that are allowed for that record.

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 should have access. This access model requires explicit sharing of each record with each user that requires access to it.

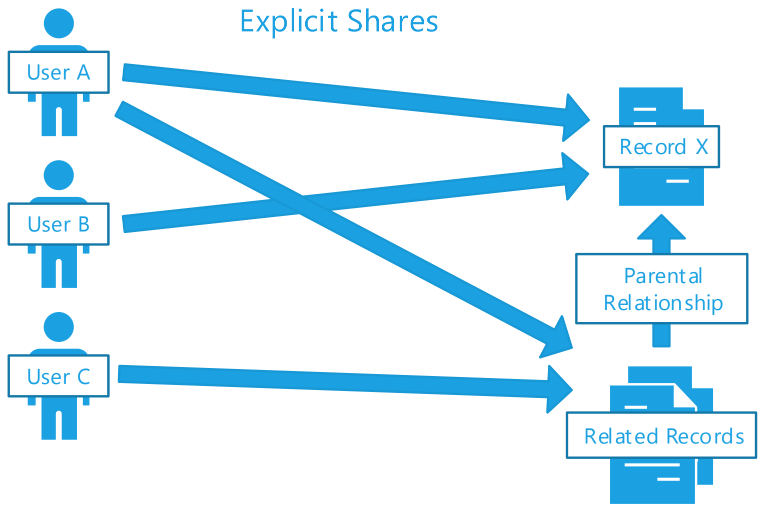


This approach works well for smaller volumes such as in smaller implementation or situations in which sharing is used to define exceptional circumstances. In larger implementations, it may be that most scenarios are handled through a more general policy, but occasionally there are specific cases that do not follow the rules. This may occur 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 need to 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 as the sharing records for each user are also duplicated for each child record as the sharing request is cascaded from parent to child records.



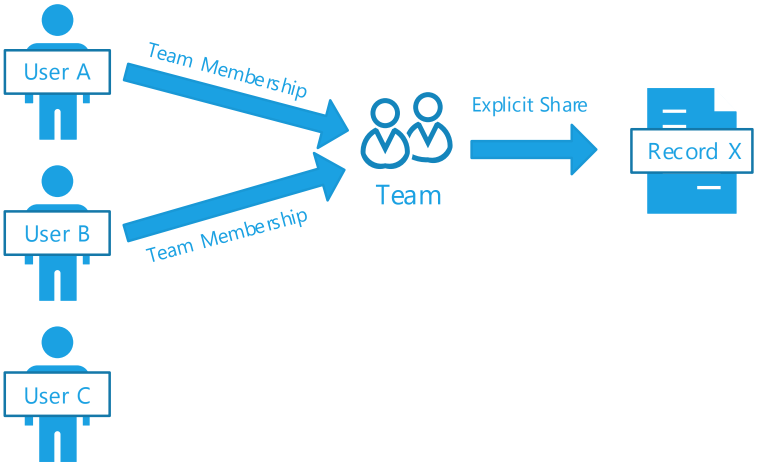
A similar scenario triggering automatic sharing can occur when records are assigned. An option available within the system is to specify that when a record is assigned to a new owner, the previous owner should have the record shared with them. This is useful for scenarios in which access permissions are granted through individual ownership, but it would still be useful for previous owners to maintain an ongoing view of the data. This is configured in the System Settings dialog box and disabled by default as the scenarios to which it applies are limited. However, this can result in sharing rules being automatically added to individual security principals.

This approach to access control is enabled by using explicit shares to individuals, which 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 that is shared generates a sharing record linking that record and the individual that it is shared with to define that user's access to the information; typically, this also applies to child records ▪ An increase in the number of individual users and/or the records that need to be shared with those users can rapidly 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.

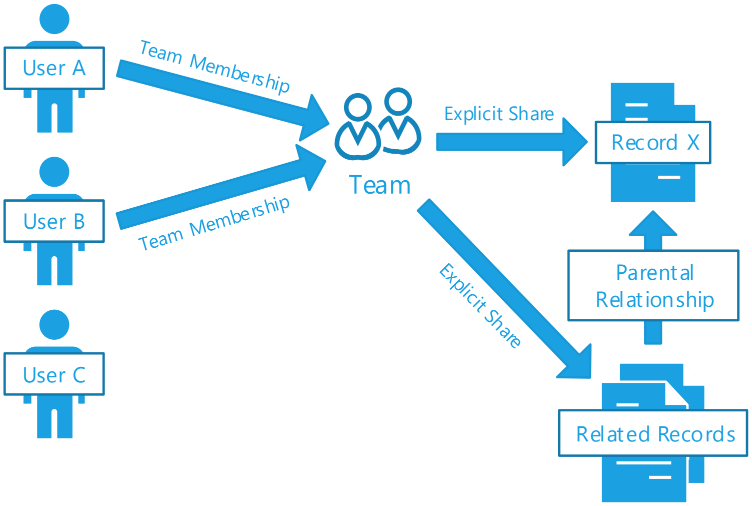
Sharing with teams

It is more efficient to use explicit sharing to manage user access to data by taking advantage of teams in Dynamics CRM 2011. 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.



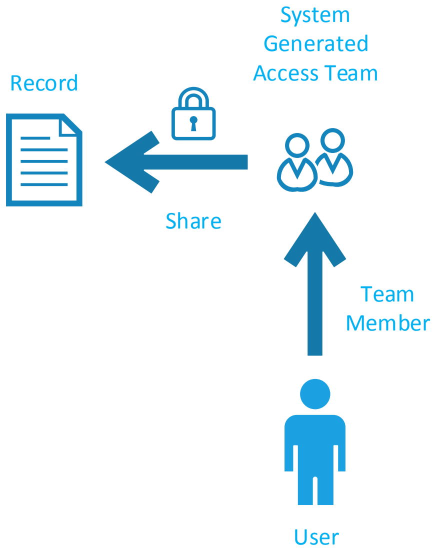
While using this approach will reduce the overhead associated with extensive use of explicit sharing, it still requires that a sharing record be created for each record and that each record have an owner, which defines the business unit to which the record belongs. In addition, solutions that employ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

Access teams

Microsoft Dynamics CRM 2013 include access teams, which are lightweight teams that are specifically designed to accommodate high volume sharing scenarios. Creating and managing access teams and their membership is automated, which simplifies administration for scenarios in which individual teams are defined, for example, to manage a particular customer.

Note: An administrator can also create an access team manually via the Teams view.

Access teams can be enabled for particular entity types through configuration, and a sub-grid is used to manage the membership of the team directly from within the record form. As users are added to the team through the sub grid, 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.



For scenarios in which sharing is the only mechanism used to provide users access to records, using access teams is an effective way to minimize the overall impact on the system, especially at high volumes.

Record ownership

In Microsoft Dynamics CRM 2013, there are many types of record ownership, as described in the following table:

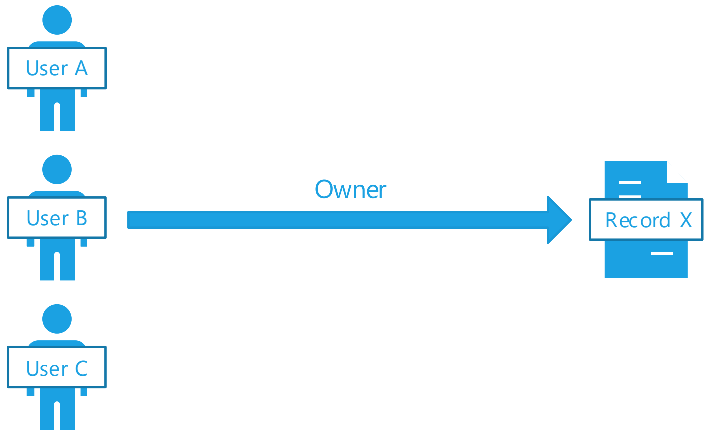
Ownership Type	Description
Organization Owned	Contains data involving something that belongs to or that can be viewed by the whole organization. Organization-owned entities cannot be assigned or shared. For example, products are owned by the organization. These records have a field named **organizationid**.
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 to a 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 as access teams cannot 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 cannot 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.

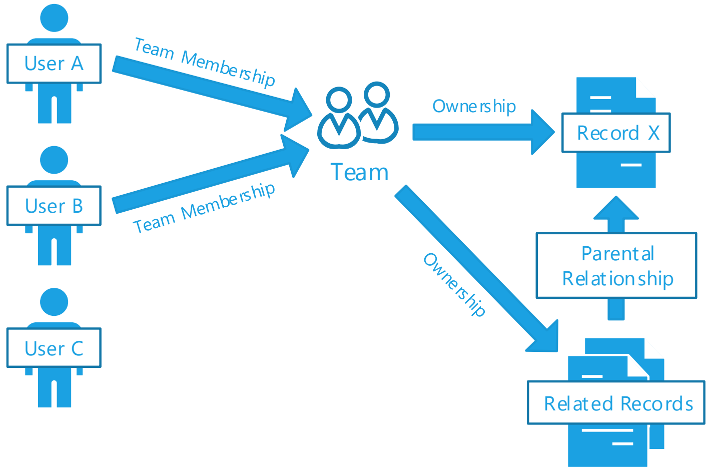


The user ownership model provides the granularity necessary to grant access to specific records by using security roles to define privileges that ensure that a user sees only the records that are owned by that user. For businesses in which individuals work independently on specific deals or on customer collaborations, this is a very effective and efficient model.

For scenarios in which 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. An additional challenge therefore comes when the user owning records moves role within an organization, potentially forcing the ownership of that record to change to maintain the access that any other users may gain via the business unit of the record.

Team ownership

Using team ownership introduces a solution pattern that can simplify many of the challenges associated with situations in which users own records directly. 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 cannot 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 cannot have security roles. To have the rights to own a record, a team must have the appropriate privileges, which an access team (without a security role) does not have.

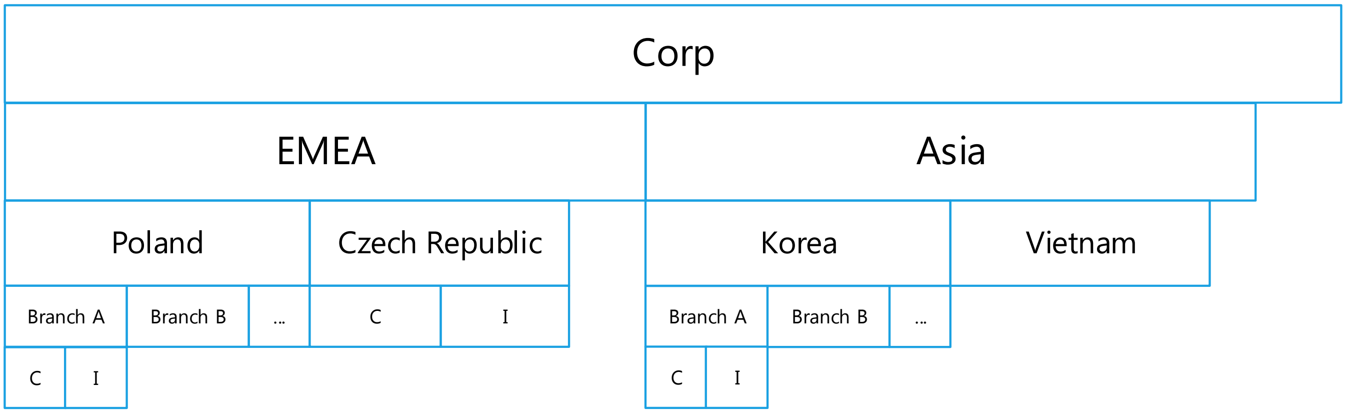
In the following sections, anywhere a team is specified that has ownership or derives access to records through security roles rather than sharing, this is referring to an owner team.

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 of the access rules to each individual record.
- Scalability, in terms of adding significant processing overhead to each request for situations in which individuals access a large number of records but individual access is defined.

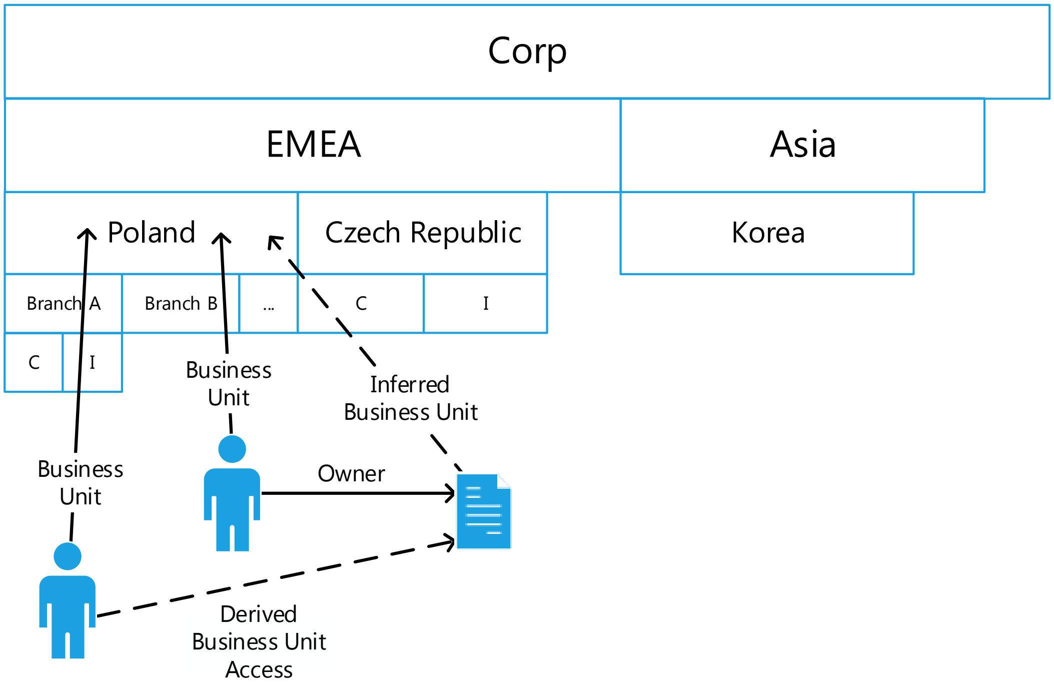
To address these challenges at scale, Dynamics CRM offers the ability to implement business units, a hierarchical structure of organizational areas, to manage access to large groups of records for situations in which a set of records are defined as a group so that collective access to be granted to multiple users.



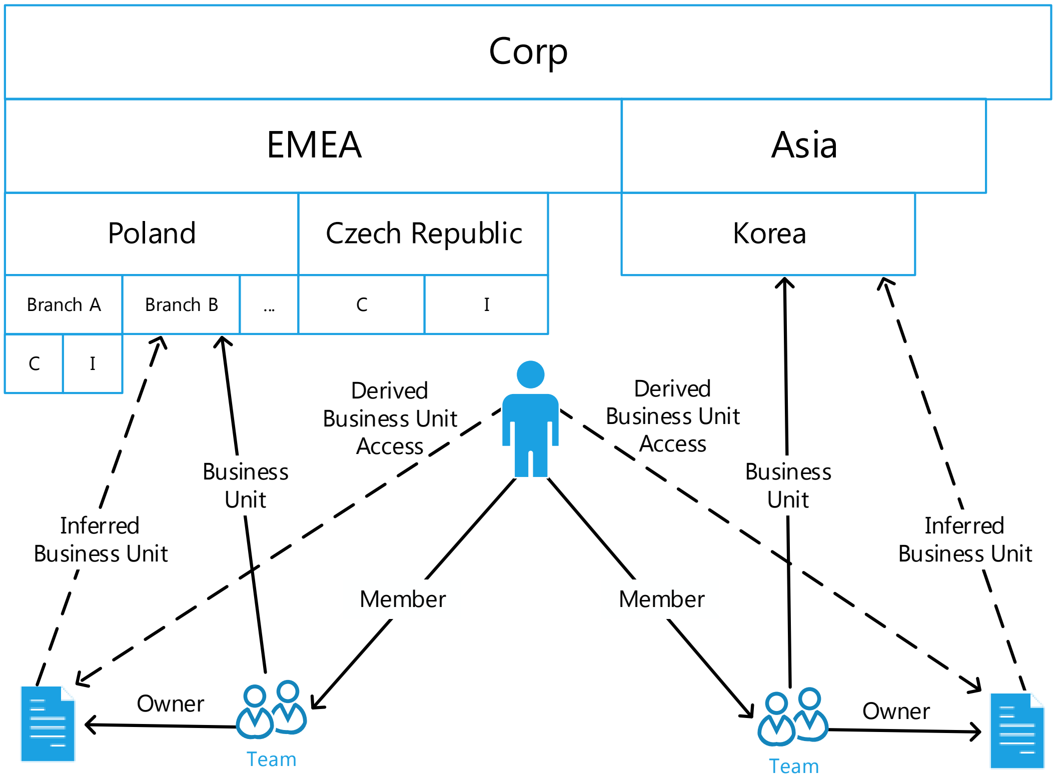
When a record is assigned an owner, either a user or a team, the record inherits the owner’s business unit.

Business unit local privileges

Users or teams are assigned security roles that grant privileges at different levels, one of which is '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. With this approach, for example, a Polish user who has been given read access to any contact records in the same business unit would be able to 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 would 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 were only able to see the Polish Branch C and items managed in the Korean head office, it would be possible to add the user to the teams created within the business units of Poland Branch B and Korea. If those business unit Teams are granted BU only level privileges, then the user would be able to see records in either of those business units without the need for the two business units to be related in any way.



Note: Though default teams are created within each business unit, they are designed to provide ownership of records by a business unit and their membership cannot 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".

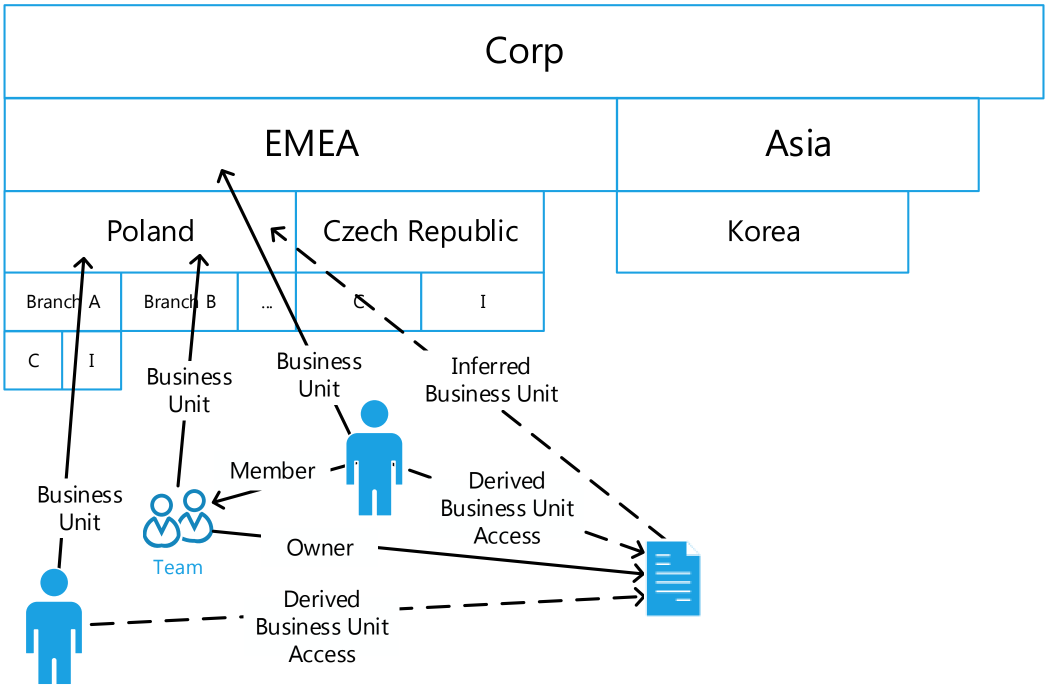
Business unit deep privileges

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 any children business units. This can be useful in hierarchical scenarios, in which one user works in a particular country 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.

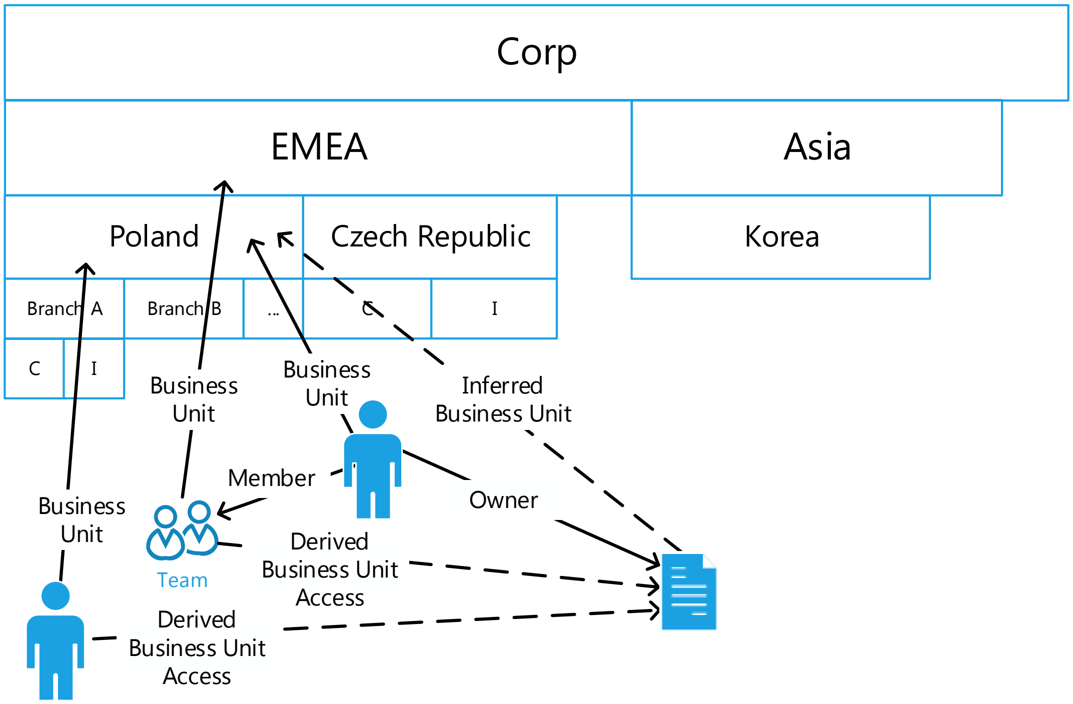
A common challenge encountered with this model is the need to accommodate users at the regional level who interact directly with accounts while other users need access bounded at a local level. To provide users regional access by taking advantage of user business unit modeling, the users would need to sit within the regional, e.g. EMEA, business unit and be given deep privileges. However, that would naturally locate any records those users own also with the regional business unit, making them outside of the scope of access of any Polish users.

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

- Locating the user at the regional level but assigning any records to the team representing access to the Polish business unit rather than directly to the user.



- Locating the user in their local country business unit but adding them to a team located at the regional level, which would grants them privileges to any records owned within the regional or through deep privileges to any of the children business units.



Organization privileges

A further option is available for records that do not need granularity of scope of access. For a scenario in which 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, as individual access checks are not required other than to verify that the user or team is allowed access to this type of record.

Access control to fields

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

- Breaking out the secure information onto separate entity types e.g. an ‘Annual Account’ record or an annual ‘Account Plan’ may be a valid approach. This would provide 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 e.g. all users of a typical role can view this type of data for all accounts
- Implementing Field Level Security (FLS), which is a finer-tuned approach to providing data access than is entity level control. If an organization determines to implement FLS, it can be set up using one of two alternative models.
    - Assigning field security profiles to users or teams, and then defining the fields across all instances of a particular entity type that a user or team can access. This can therefore be a good mechanism for scenarios in which the type of access control needed is universal.
    - Sharing fields, which combines sharing and field level security, can be implemented to allow for sharing individual fields from individual records, with specific permissions in terms of read or update access for example, to particular users or teams. Implementing this granularity of access, however, incurs similar performance impacts with the sharing of the records themselves. Using this approach can provide benefits however for scenarios in which 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.

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. This information is subject to change as future releases add optimizations and features to the product.

The primary scenarios to consider as part of general access modeling include accessing, the system initially; accessing a single record, and 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 described below only applies to read privileges.

SQL view access

The subsequent sections focus on the way that Application Servers apply security principles. 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 with Offline Access (the “Outlook offline client”.) 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 does not cover the topic separately.

Initial access: Caching

To optimize many processes within Microsoft 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, which 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, and 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 logging on at the same time can impact overall scalability as well as initial access times for individual users
- For scenarios in which owner team memberships and/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, potentially impacting both the request and overall server performance
  - Making regular changes to user records, for example to record a metric tracking the last time the user accessed the system, will not only incur the write impact to the record, but will trigger the Application Server caches to flush that user's information on each update, forcing it 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 reloading the cache.
  - This is another scenario where access teams can benefit, regularly changing access team memberships have much lower system impacts as they are not cached. They are primarily used directly within the database to calculate sharing and therefore do not 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 to which the user belongs:
  - 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 to which a user belongs:
    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 the following event occurs:

- When the system user record is updated or changes state
- When the security roles for a user are changed
- When a user is added or removed from an owner team
- When the security roles are changed for an owner team a user is a member of
- When the user has been inactive, i.e. has not made a request of the system, for 20 minutes

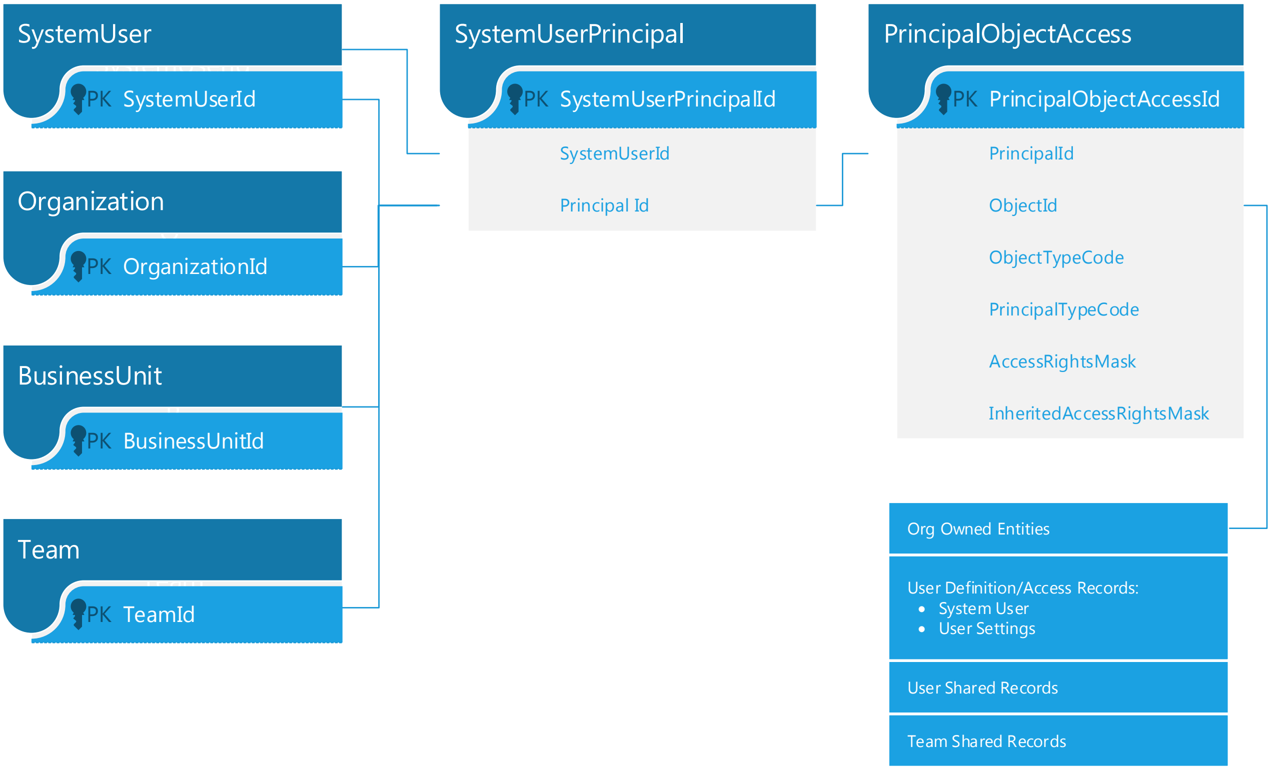
Designs that regularly update a System User record, for example to record last request, will have a significant impact on system performance and scalability as the cache will constantly be flushed and reloaded.

Sharing access checks

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

Sharing records

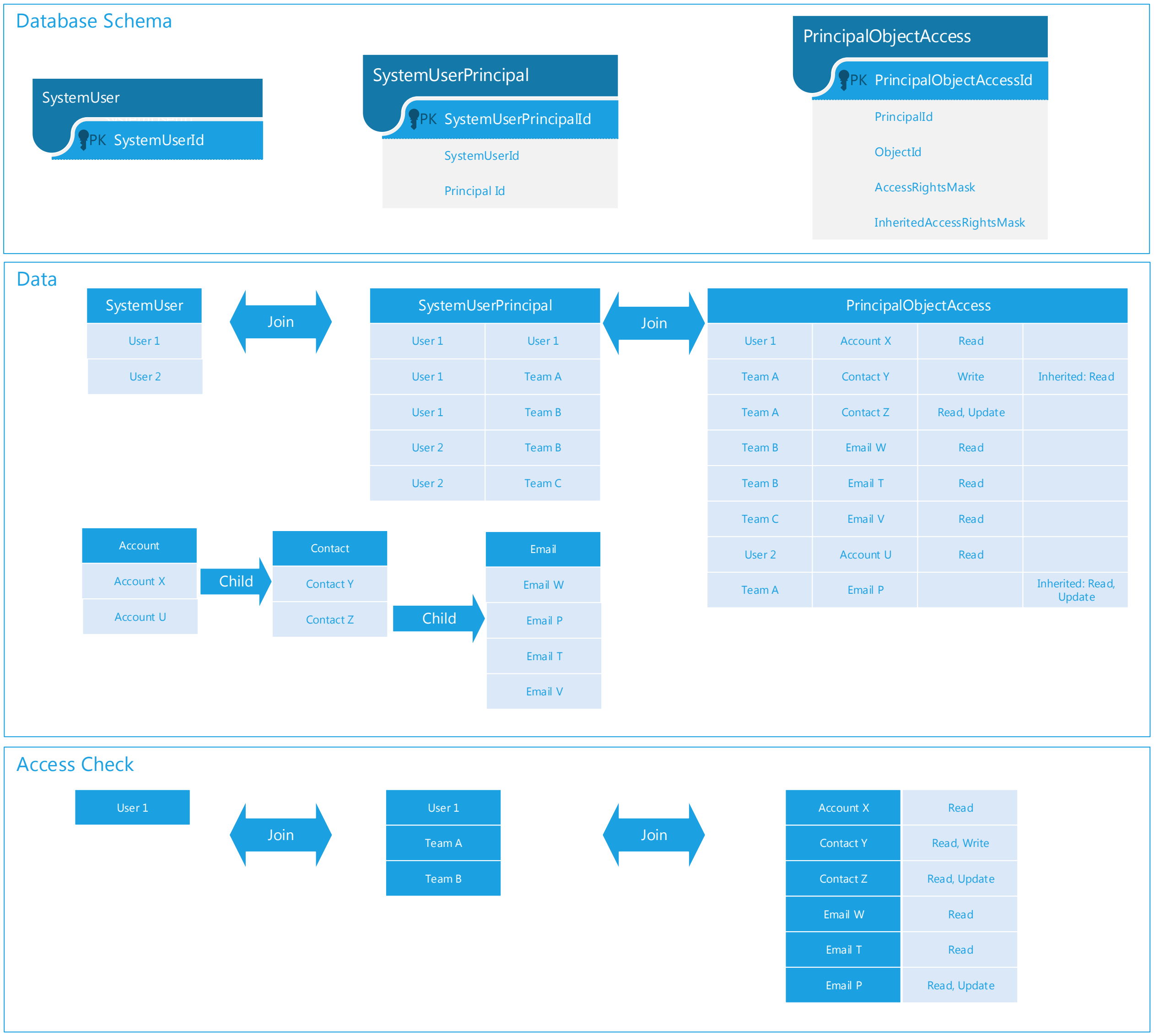
Of initial importance is having an understanding how the sharing rules are recorded in the system. The sharing model is shown in the following graphic:



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 to which it has been granted sharing privileges. The PrincipalObjectAccess 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, as well as 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 either: manually and explicitly by a user, programmatically through the UK or 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 the following graphic:



SystemUser
PK SystemUserId

SystemUserPrincipal
PK SystemUserPrincipalId
SystemUserId
Principal Id

PrincipalObjectAccess
PK PrincipalObjectAccessId
PrincipalId
ObjectId
AccessRightsMask
InheritedAccessRightsMask

SystemUser
User 1
User 2

SystemUserPrincipal	
User 1	User 1
User 1	Team A
User 1	Team B
User 2	Team B
User 2	Team C

PrincipalObjectAccess			
User 1	Account X	Read	
Team A	Contact Y	Write	Inherited: Read
Team A	Contact Z	Read, Update	
Team B	Email W	Read	
Team B	Email T	Read	
Team C	Email V	Read	
User 2	Account U	Read	
Team A	Email P		Inherited: Read, Update

Account
Account X
Account U

Contact
Contact Y
Contact Z

Email
Email W
Email P
Email T
Email V

User 1

User 1
Team A
Team B

Account X	Read
Contact Y	Read, Write
Contact Z	Read, Update
Email W	Read
Email T	Read
Email P	Read, Update

To determine the records to which a user has access 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 will result 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

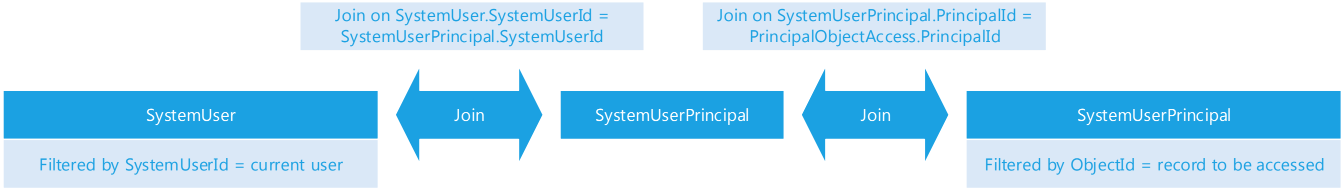
A common challenge in high volume sharing scenarios is related to the volume of data created in the Principal Object Access (POA) table. Understanding the types of data created in the POA table can be useful in appreciating the impact a design can have on the scalability of a system.

The POA table will contain the types of records shown in the following table.

Type of record	Detail
Organization Owned	An entry for each Organization Owned entity type
User Entity records	Each user will have the associated System User record and User Settings record shared to him or her automatically
User Sharing	Each time a record is shared with a user, this is recorded with a record in the POA table linking that user to the record along with the privileges they have been 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 been shared
Activity Participation	Whenever a user is included as a participant in an activity, they will be 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 needs to be run:

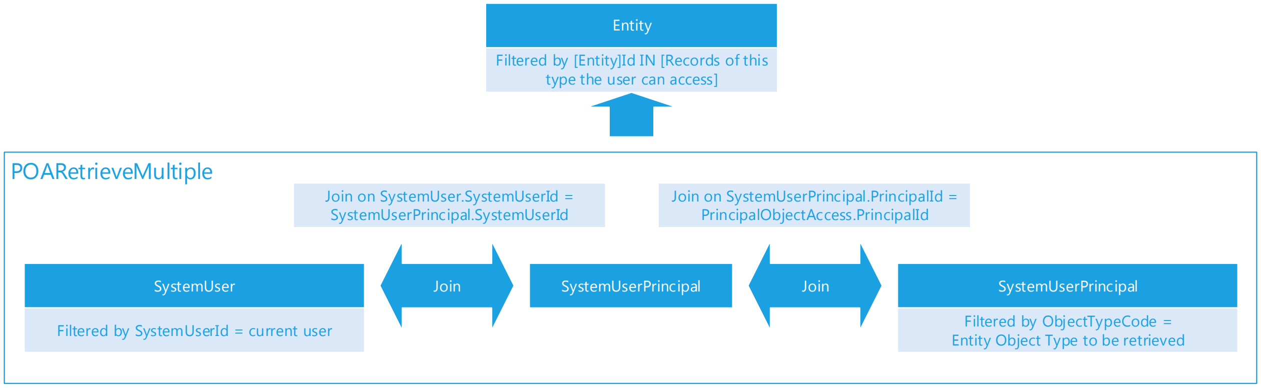

Join on SystemUser.SystemUserId =
SystemUserPrincipal.SystemUserId
Join on SystemUserPrincipal.PrincipalId =
PrincipalObjectAccess.PrincipalId
SystemUser
Filtered by SystemUserId = current user
Join
SystemUserPrincipal
Join
SystemUserPrincipal
Filtered by ObjectId = record to be accessed

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).

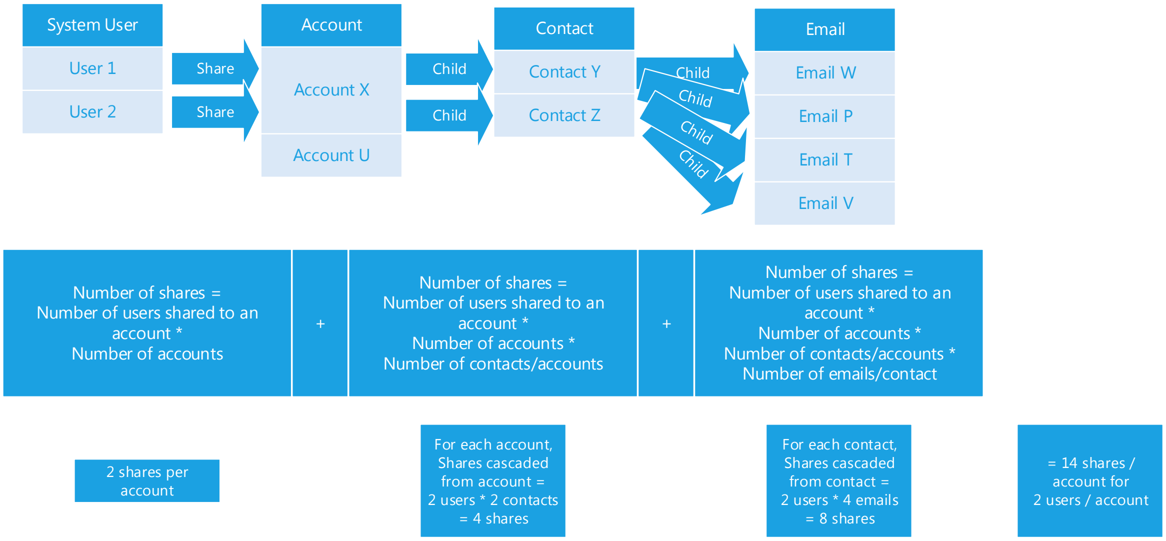
To perform this query for multiple records, the record type is queried with the filter criteria set and with a join to an in memory table that is created with the records of that type the user can access directly through sharing or indirectly through team sharing.

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 the following graphic.


Entity
Filtered by [Entity]Id IN [Records of this type the user can access]
POARetrieveMultiple
Join on SystemUser.SystemUserId =
SystemUserPrincipal.SystemUserId
Join on SystemUserPrincipal.PrincipalId =
PrincipalObjectAccess.PrincipalId
SystemUser
Filtered by SystemUserId = current user
Join
SystemUserPrincipal
Join
SystemUserPrincipal
Filtered by ObjectTypeCode =
Entity Object Type to be retrieved

Cascading sharing

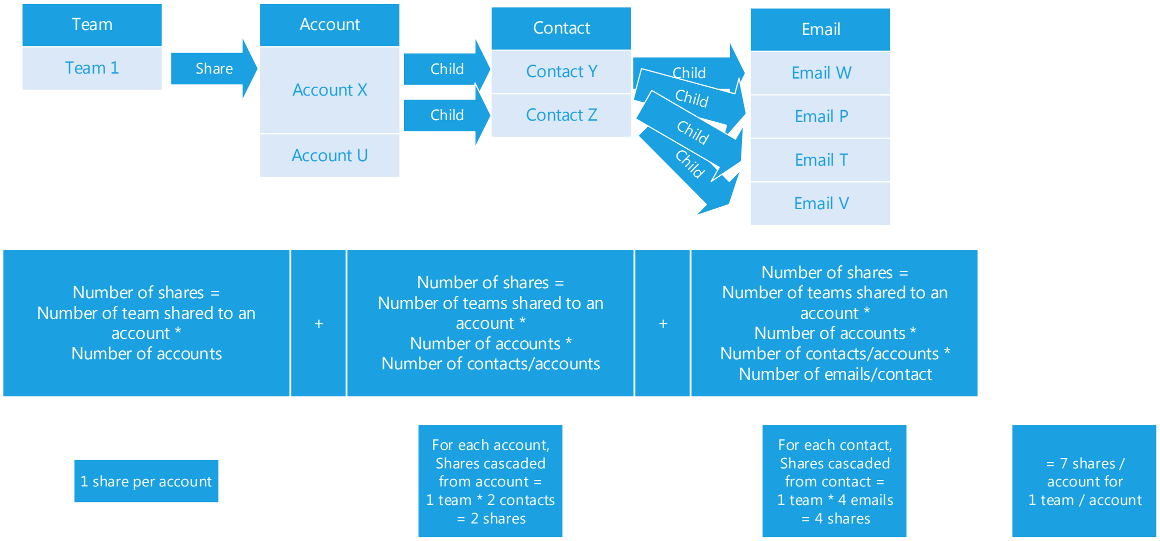
A classic scenario in which sharing challenges can be more problematic relates to the inheritance of sharing through parent-child relationships. In this scenario, whenever the parent is shared, the sharing is inherited by all the children. To achieve this, the system needs to create a record for each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


System User
User 1
User 2
Share
Share
Account
Account X
Account U
Child
Child
Contact
Contact Y
Contact Z
Child
Child
Child
Child
Email
Email W
Email P
Email T
Email V
Number of shares = Number of users shared to an account * Number of accounts
+
Number of shares = Number of users shared to an account * Number of accounts * Number of contacts/accounts
+
Number of shares = Number of users shared to an account * Number of accounts * Number of contacts/accounts * Number of emails/contact
2 shares per account
For each account, Shares cascaded from account = 2 users * 2 contacts = 4 shares
For each contact, Shares cascaded from contact = 2 users * 4 emails = 8 shares
= 14 shares / account for 2 users / account

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 defining 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 2 members can reduce the overall number of shares by half. For situations in which teams represent larger groups of users, the saving in terms of record growth can be significant.


Team
Team 1
Share
Account
Account X
Account U
Child
Child
Contact
Contact Y
Contact Z
Child
Child
Child
Child
Email
Email W
Email P
Email T
Email V
Number of shares = Number of team shared to an account * Number of accounts
+
Number of shares = Number of teams shared to an account * Number of accounts * Number of contacts/accounts
+
Number of shares = Number of teams shared to an account * Number of accounts * Number of contacts/accounts * Number of emails/contact
1 share per account
For each account, Shares cascaded from account = 1 team * 2 contacts = 2 shares
For each contact, Shares cascaded from contact = 1 team * 4 emails = 4 shares
= 7 shares / account for 1 team / account

However, it should be noted that while sharing by using teams reduces the number of records in the table, for a particular user it does not 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.

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 needing 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, which can in turn lead to potential implications performance and scalability implications
- The administration of complex sharing rules should be considered carefully, because each time that a record is created or amended or that the related user organizations change, the related sharing records may also need to be altered
- For scenarios require broader access, Dynamics CRM provides more effective mechanisms

Access team life cycles

Access teams are created the same way that other teams are created. However, access teams cannot be assigned security roles or own records. After access teams are created, records can be shared with them and users can be added as members.

Note: Access teams behave similarly to teams in Microsoft Dynamics CRM 4.0.

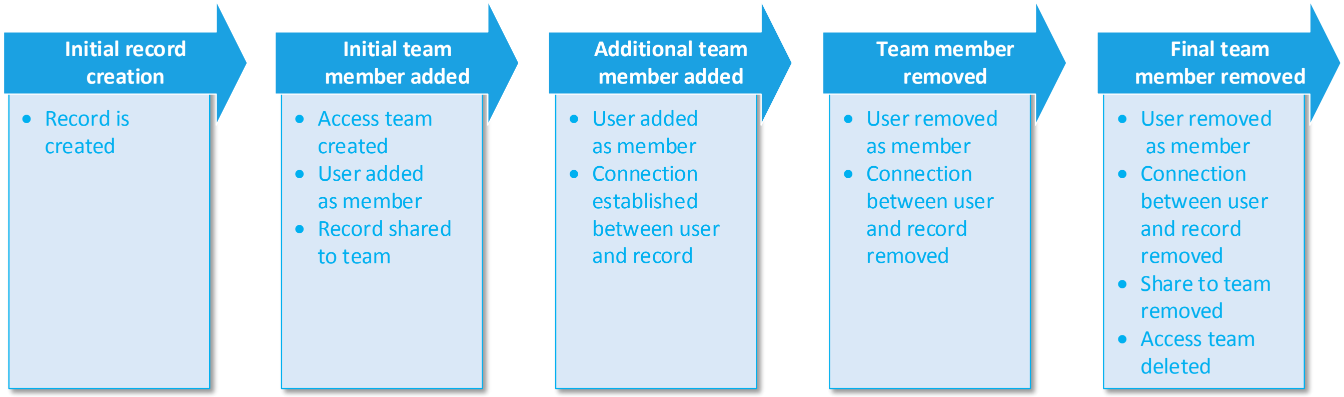
Because access teams do not require the assignment of security roles, using access teams rather than owner teams reduces overhead for scenarios in which security roles would not otherwise be required.

Adding security roles can impact performance and scalability in some 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 need to 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 that needs to be performed
- To reduce the impact of doing this 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 after a restart or when the user's record has been flushed from the cache after 20 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 re-calculated on next connection
- For rapidly changing teams or team memberships, this 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 e.g. where a combination of ownership and sharing is used, access teams can be used for the sharing cases, and avoiding the cache and calculation impact when they change

Service Scheduling uses caching extensively to optimize its calculations therefore where teams are used as resources in service scheduling we need to load the teams into the cache. Access teams therefore cannot be used as resources in service scheduling to avoid the need to cache them 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 each team linked to a particular record of that entity type is managed automatically.



Initial record creation

- Record is created

Initial team member added

- Access team created
- User added as member
- Record shared to team

Additional team member added

- User added as member
- Connection established between user and record

Team member removed

- User removed as member
- Connection between user and record removed

Final team member removed

- User removed as member
- Connection between user and record removed
- Share to team removed
- Access team deleted

As shown above in the 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 afterwards.

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 will not affect the related access team, only removing all the team members will cause deletion of the automatically generated access team

Design considerations of using access teams

Access teams cannot own a record, they do not have any security roles therefore cannot 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 however. In particular managing the lifecycle of access team's existence for records that no longer need to be directly viewed should be considered.

When a record is deactivated, this does not change the related access team membership as this may be needed if the record is reactivated or for compliance purposes. Removing all the team members, either through the user interface or programmatically, for a record can be used to remove the related access team.

Access teams cannot 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 apply to all instances of a particular entity type although role based forms can be used to present 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. Some key factors to consider when deciding which approach to use are listed in the following table.

Type of team	Key considerations
Access team	▪ Allows for >1,000 team memberships per user ▪ Rapidly changing team memberships ▪ Individual record based access ▪ Owner of the record allowed to define access to other users ▪ Different groups of access type to record (e.g. read only, update)
Owner team	▪ Owner teams best suited for scenarios in which each team accesses high volumes of data, for example millions of records, by using mechanisms such as ownership or business unit access ▪ Owner teams are required to access records scoped by business area, as security roles are required to define this and only owner teams can be granted security roles ▪ Teams used as service scheduling resource must be owner teams; access teams cannot be used for this purpose

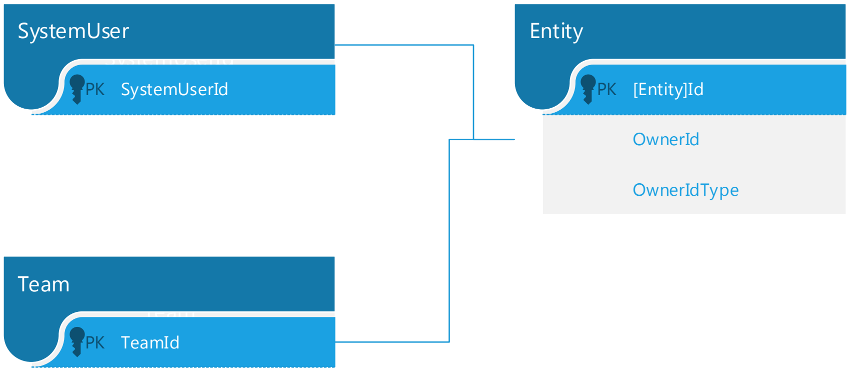
Ownership access checks

Another approach for defining access to resources is by taking advantage of ownership. This model enables granting a user privileges only to records that he or she 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.

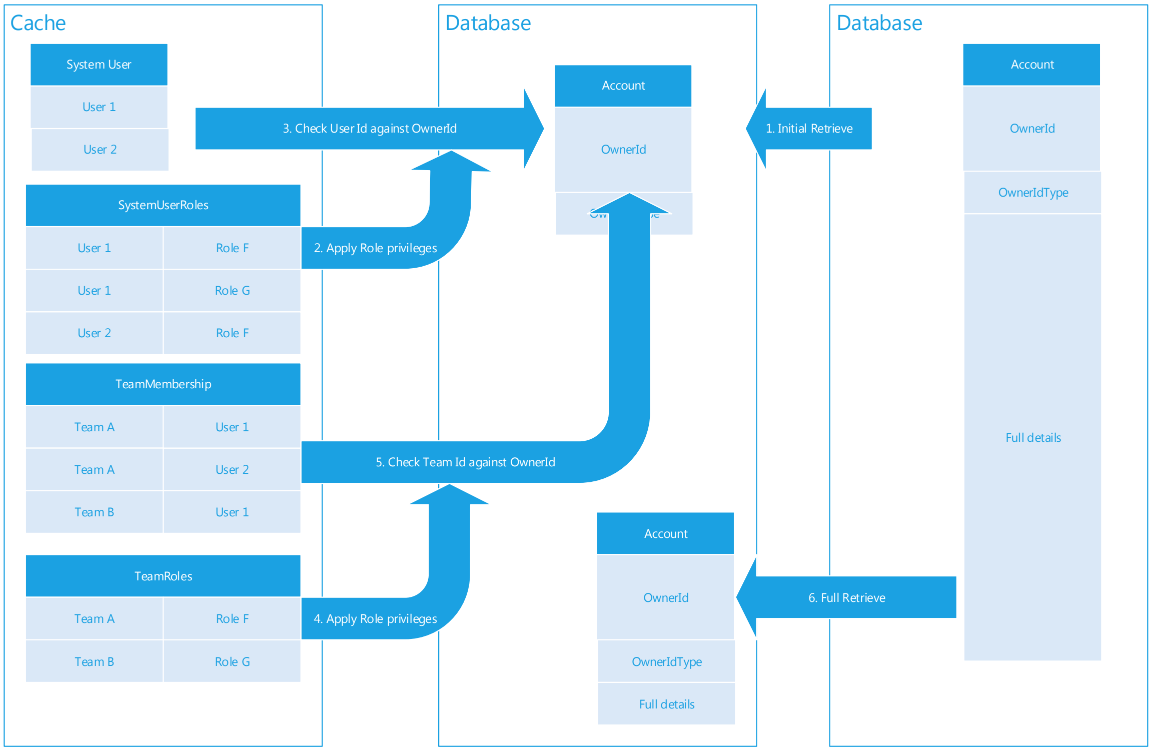
Data associated with ownership of records does not require additional storage because the information used to determine ownership is held directly on the entity records themselves, reducing the amount of data stored as well as plus the complexity of querying that is required in the system to determine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



When retrieving information, if the user does not have organization wide access and the entity is not 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 in a two-stage process to minimize the initial request cost for information the user may not be authorized to view.

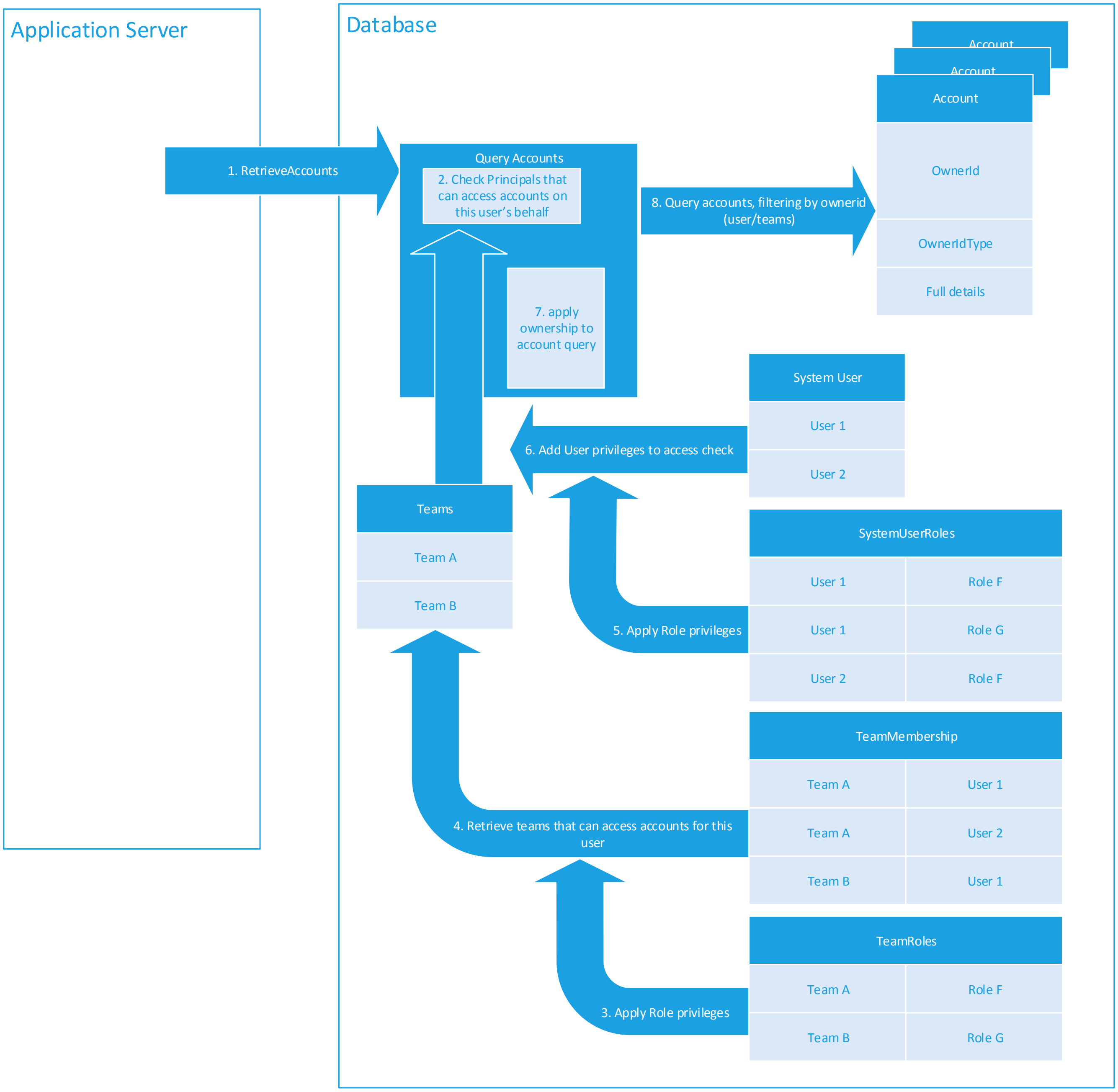


- Because a user's security roles, owner team memberships and the owner team's security roles are cached and can therefore be accessed in memory within the Application Server, therefore only an initial set of data need be retrieved about the record including the owner and type of owner of the record 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 cheap 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, here 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 is performed within 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 within 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 is not 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 the following 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 sharing model
Independence from team volumes	▪ Number of teams in the system does not directly impact performance ▪ 100k teams had no significant impact

Aspect	Implications
Grows linearly with team membership	▪ Where performance is impacted is with owner team memberships per user as they each need to be checked on 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 are maxed
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 does not impact on caching
Target Team memberships	▪ Number of memberships that can be achieved will be affected by multiple factors ▪ 1000 owner team memberships/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 needs to 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 which both grows in sharing record data and response time as more data is shared with the user.

Where response times do grow is where 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 are maxed. At this point, overall system performance deteriorates quickly and impacts on 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 after this can be significant and increasing quickly in response times as the long running requests block new requests to IIS so queuing up requests with rapidly increasing overall response times.

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

The testing has shown that having 1,000 owner team memberships per user is achievable with acceptable response times. However, it has also shown that this is heavily CPU related and very 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 very much implementation specific, the testing has shown that up to 1,000 owner team memberships per user can be achieved. It is 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 with individual access to each record 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 it not also required.

Where large common access is defined, using business units may be more appropriate. This is in particular the case where broader responsibility across wider sets of records is needed such as a manager or compliance officer, but in these cases it is common that the defined boundaries of that person within the business more naturally fit with a business unit. This can often be the case where the manager is responsible for a specific area e.g. the UK, even if their staff do not have as clearly defined boundaries of responsibilities. Providing BU 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

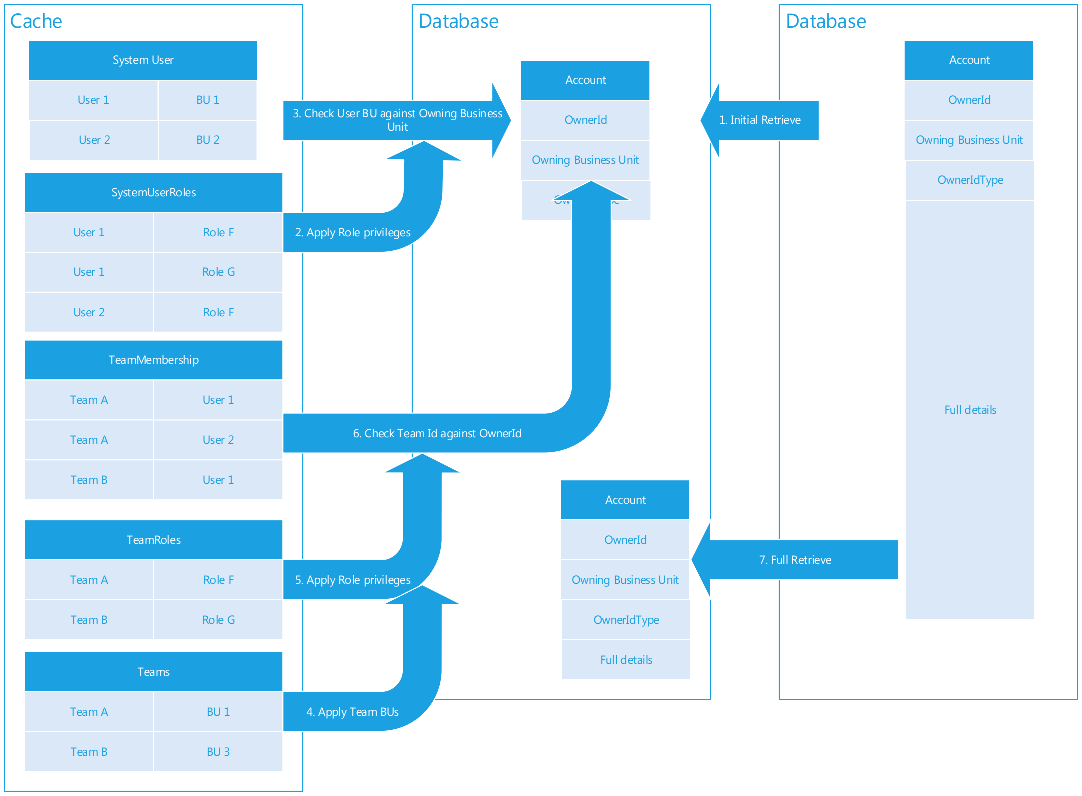
With that approach in mind, a user will have access to records in business units through a combination of:

- 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 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.



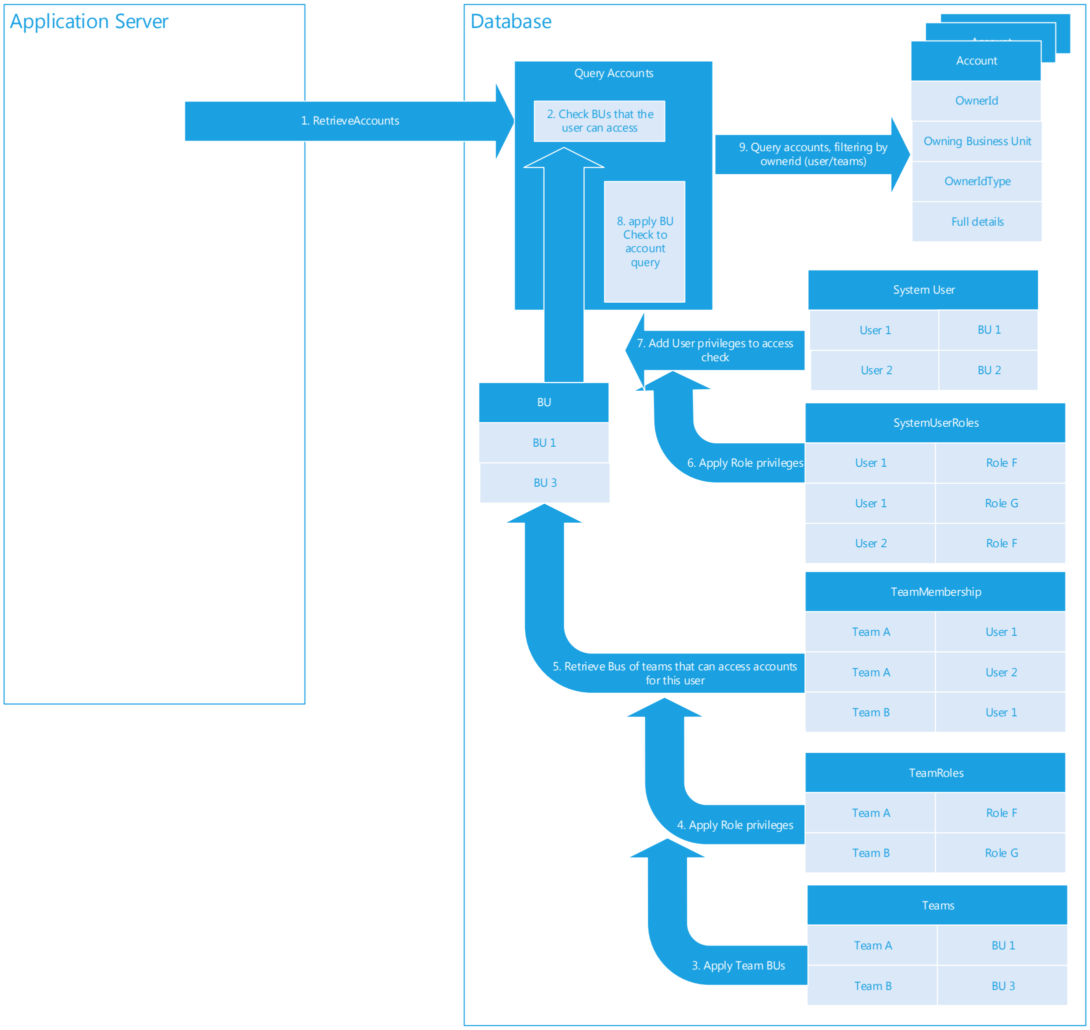
- Because a user's security roles, owner team memberships, and the team's security roles are cached, an initial set of data can be cheaply retrieved about the record including the owning business unit of the record and quickly compared against the cached information.
- This enables the application server 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 would also be 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 the user belongs 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 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, determining 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 BUs or rapidly changing BUs can have an impact on system performance, as the access mechanisms are not intended for very high volumes of BUs (> 1,000)

- Access is pre-calculated, so high volumes of changes of BU structure can impact on 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 as they are 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 within 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/ 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 cannot 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, where everyone can see the data it 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 is not needed.

Combinations of access types

Although considering these models in isolation is useful, it is also important to understand how combinations of security model can impact on 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, then complex processing of sharing rules will not 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 i.e. the owner and owning business unit, enabling an optimized access check to be performed. However, if for a particular record the business unit access or ownership rules do not allow access, then sharing will need to be checked.

Where 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, 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.

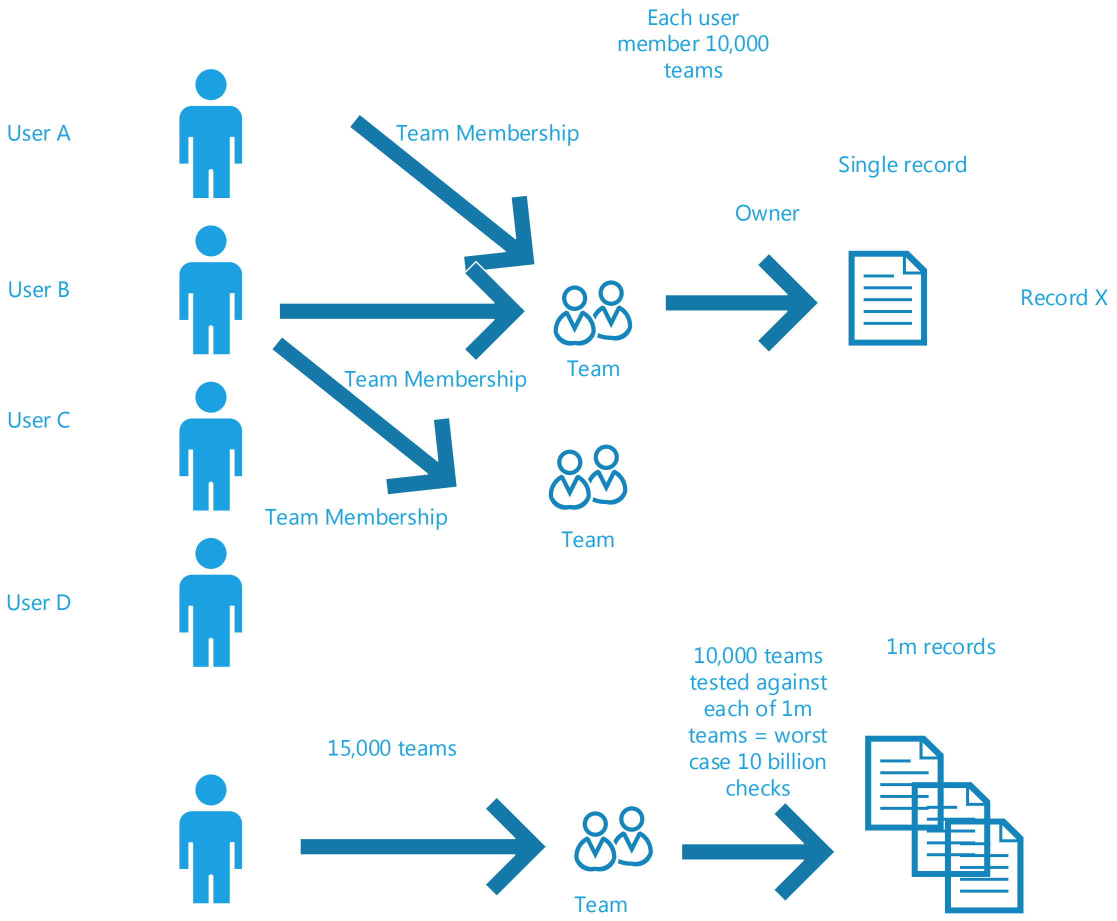
For context, let’s look at the volume of access checks required in different scenarios. Our base for this calculation will be 1 million records and a user that is a member of 1,500 teams.

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 1m records in the system and 1,500 records, then the system could in the worst case scenario need to 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, typically the first 100 or 250 results. The other factor is that once a successful match on an owning team has been found giving a user access then no further checks need be performed on that record.
- It is conceivable that in this scenario that as few as 100 checks could be needed with successful access being granted for the first 100 records by the first team 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 is 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 scenario that does not 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 utilized together, particularly when considering different user and usage types. For scenarios that require providing individual access, team ownership and sharing can be used. However, scenarios involving users acting in a managerial or reporting role, which require broader access, 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 the following 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/ oversight access with more granular access lower
User Ownership	▪ Ideal for large volumes of data but individual granular access to smaller subset ▪ Optimized access checks as combines memory cached and record data directly ▪ Limited to one type of access through ownership i.e. owned by single user ▪ Also drives BU position so enables combination with BU privilege access
Team Ownership	▪ Ideal for large volumes of overall data but group granular access to smaller subset ▪ Optimized access checks as combines memory cached and record data directly ▪ Reduces cascading impact as user involvement changes ▪ Isolates users from record BU structure, which enables user movement with no data impact
Sharing	▪ Ideal for modeling 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 the real world

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 within Dynamics CRM, it is 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/ Case management ▪ Typically need full time access to the records they work with
Managerial Reporting	▪ Manager/ Support teams ▪ Predominantly review aggregated data not individual deal data ▪ Occasional access to individual records within their scope of responsibility ▪ Dive into individual deal/matter data on demand due to exception raised or noted

Active involvement

Looking at the active involvement roles, the data coverage, and the 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
- Max 200 deals per year per person - With a working capacity of 200 working days a year, with a minimum of 1 day per deal, gives a maximum possible capacity to interact effectively on 200 deals/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 therefore, 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 which 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 i.e. 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 therefore, 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.
- Therefore 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 individuals involvement, these typical 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 would pick 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 on exception for in cases in which 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. 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, then 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 is 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, for example.

Design considerations

When modeling security within 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 not regularly but on occasion 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 will affect scalability. One key consideration is the amount of work that needs to be processed, which is a factor of:

- How many teams to be checked per request?
- How many requests made e.g. every few minutes, once per day?
- When are requests made e.g. at the start/end of the day, out of hours, 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 the following 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 e.g. end of week reporting

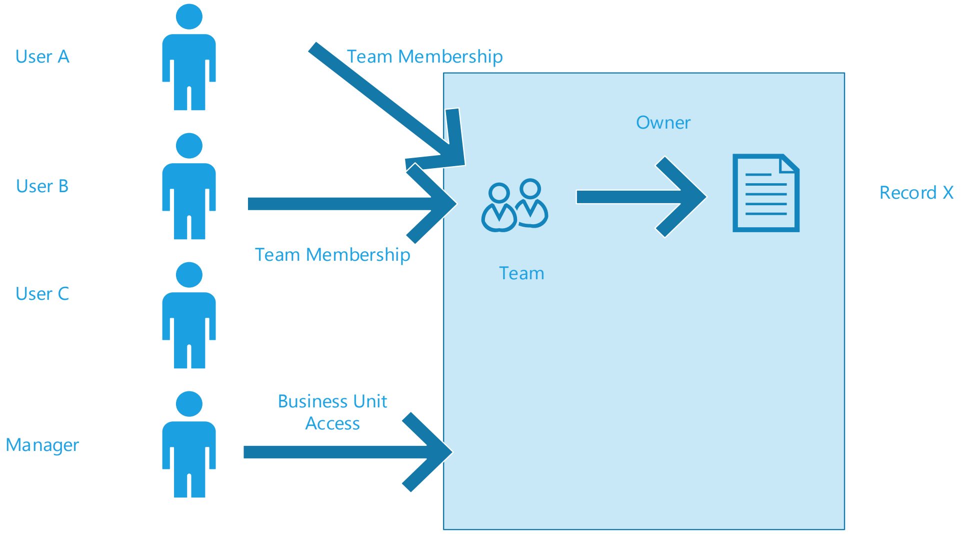
Design patterns

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.

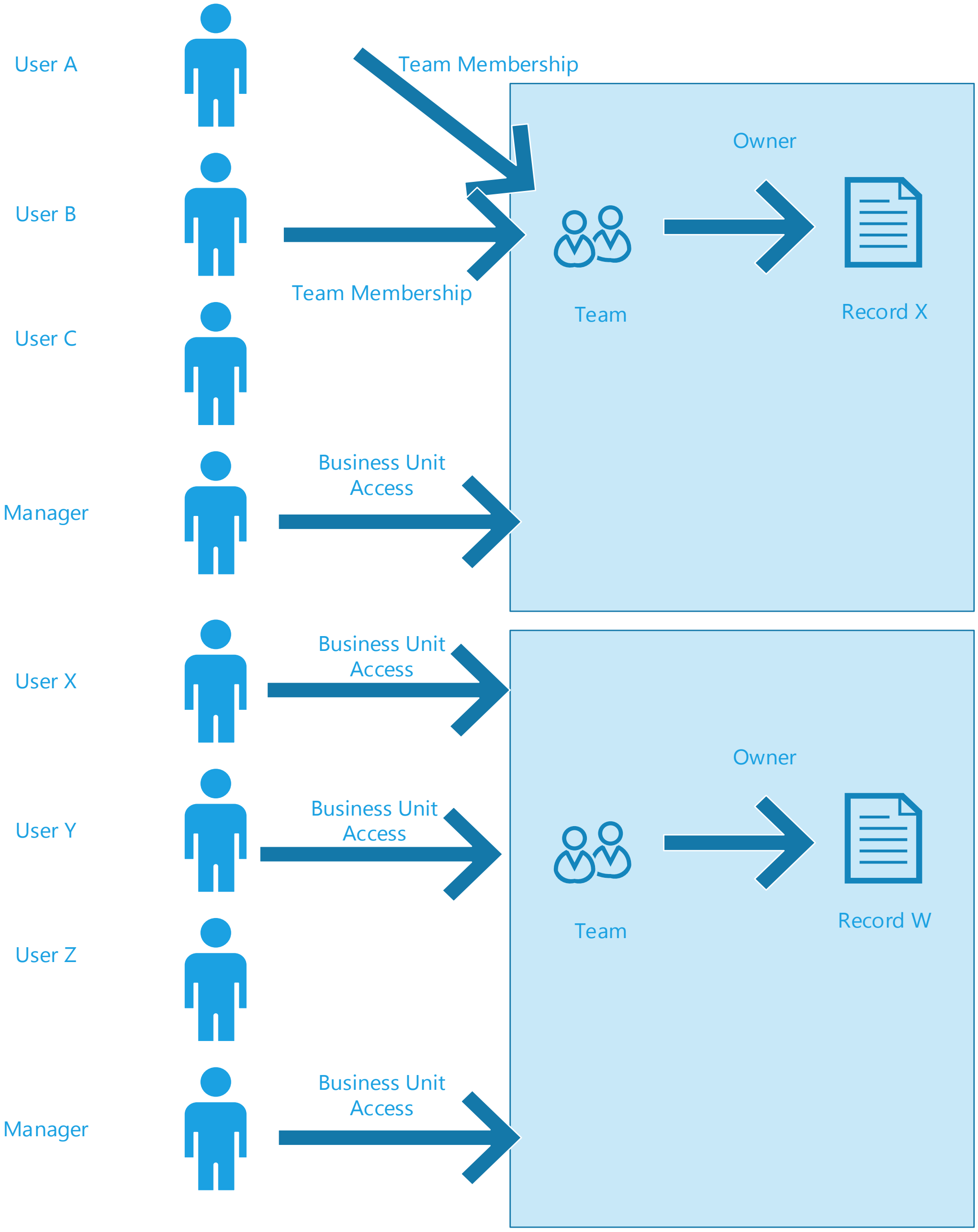
Identifying different user types and then modeling the associated access appropriately can be an extremely effective way to model for high scalability. For example, using a team ownership model for individual sales staff together with a business unit model for management staff can often align well with 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 requires individual access while another country, 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 and business unit access for another can significantly reduce the scalability challenge on the solution.

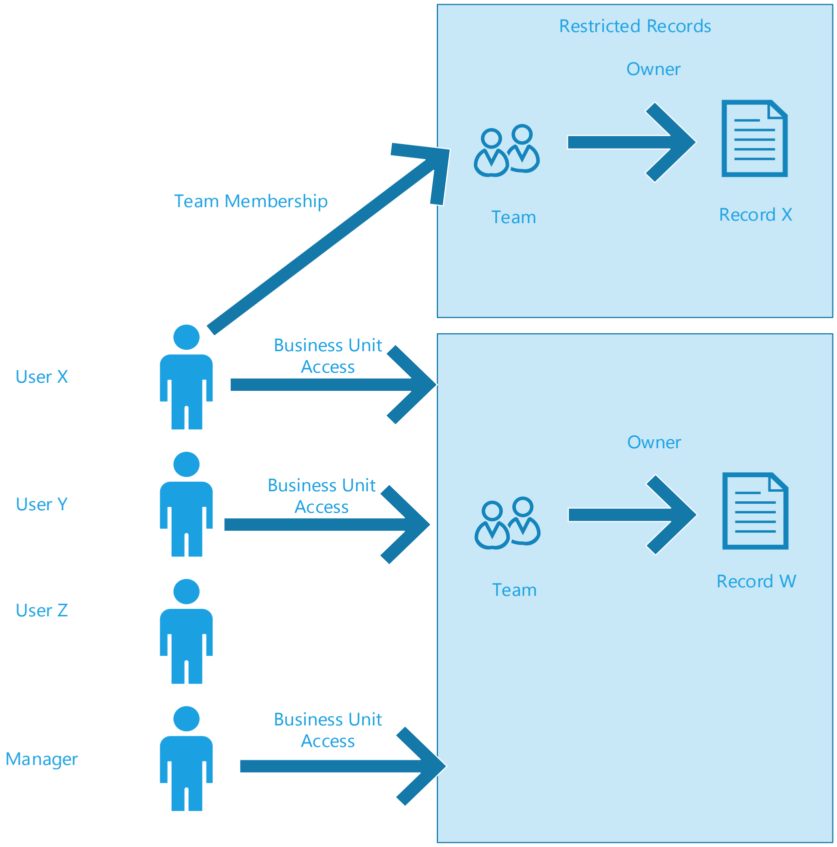


Customizing the security model to account for exceptions

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

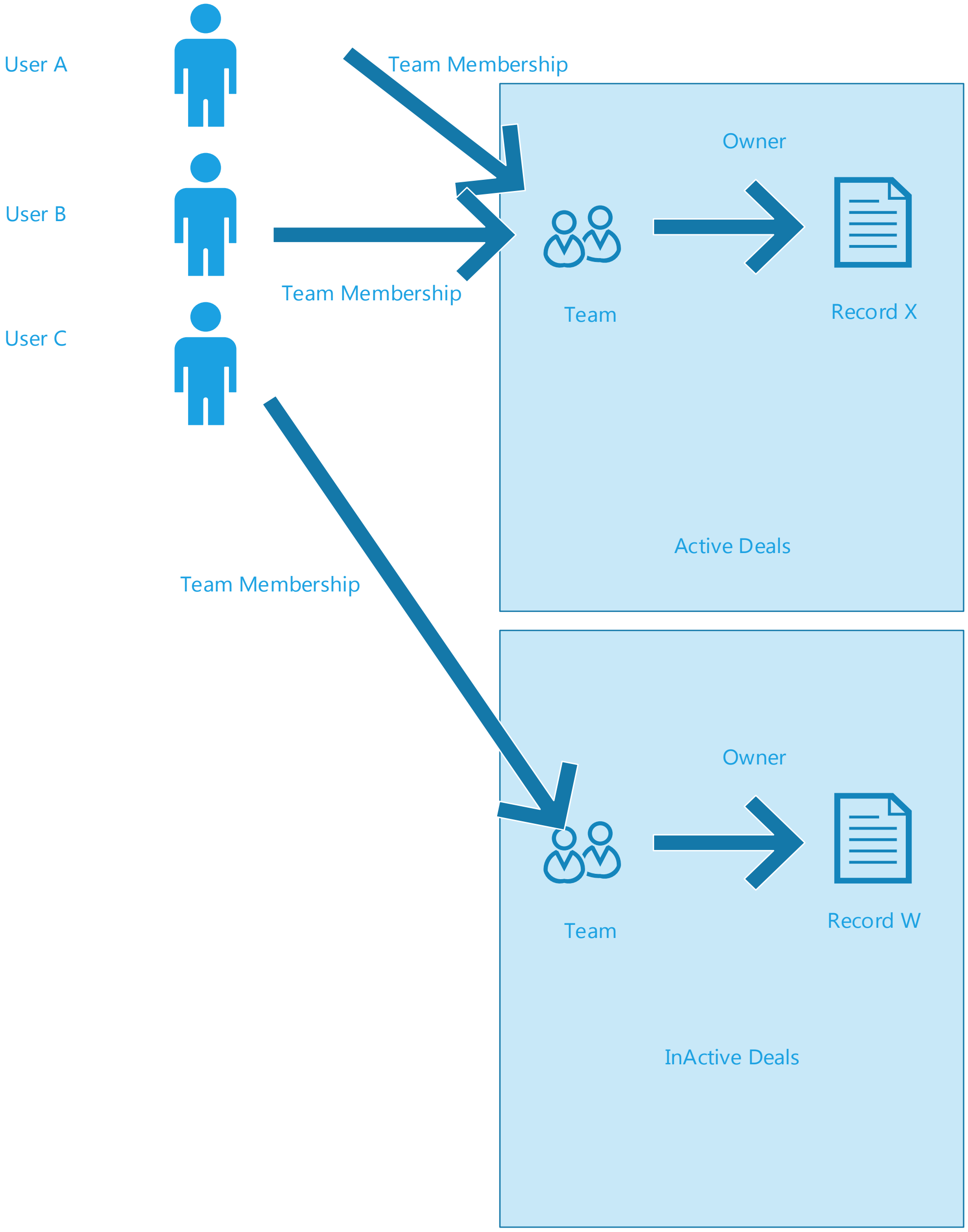
Accommodating historical data presents another frequent challenge. There are many scenarios in which providing access to historical data is important, but including that historical data in the operational system can negatively impact performance, particularly for a solution that uses team ownership or 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 the 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.

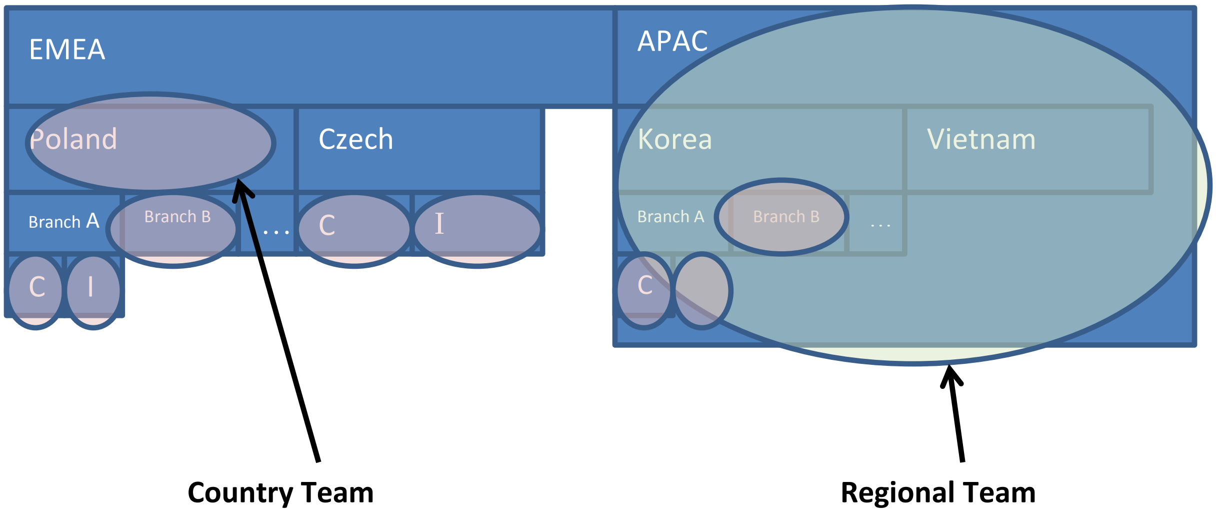


Modeling security walls rather than the organizational hierarchy

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 are typically 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.



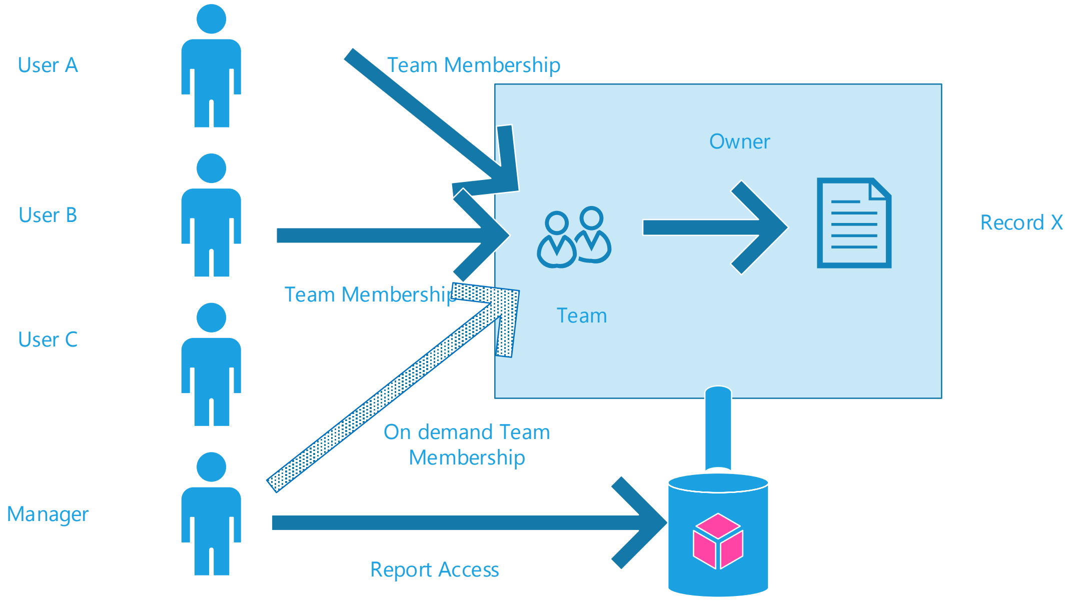
There are other cases, particularly in the banking industry, in which customers have a global presence and may need to interact with exchanges around the globe on a 24-hour basis. In such cases, it may be useful to manage these customers completely separately from the more typically geographical structure used to manage smaller, more local customers. Providing a centralized Business Unit to cover the global customers while maintaining a regional/country 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 within a business. Oftentimes, 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, then recording the behavior for later compliance checking may be sufficient and appropriate.

Modeling data along security lines

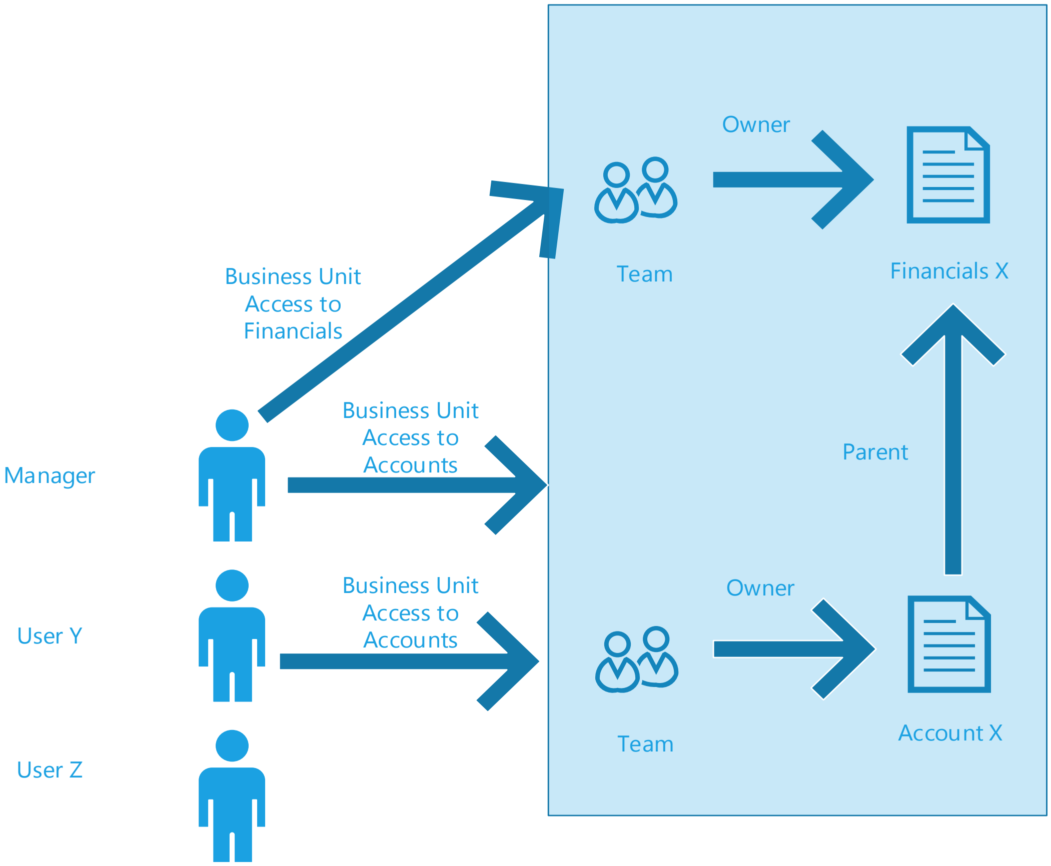
The first inclination when implementing a solution may be to model data the way it is in an existing system or around tangible objects, such as Account or Company, from the real world. However, taking this approach often can hide logical distinctions in data that actually reflect the way that users need to control access to data.

A good example of this often relates to the financial information, such as their annual business with the company, about an account. 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 need to be. 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 granted access to the Financials entity separately
- 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.


Owner
Team
Financials X
Business Unit Access to Financials
Business Unit Access to Accounts
Manager
Parent
Business Unit Access to Accounts
User Y
Owner
Team
Account X
User Z

Security role versus privilege

When customising solutions, the need arises 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 defines the specific actions or data access a user can perform and 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 of benefit:

- 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, then 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 as 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, then 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.

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; rather 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 a 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.