# Managed Schemas in Snowflake

## 📋 Overview

**Managed Schemas** (also known as **MANAGED ACCESS schemas**) are a Snowflake security feature that provides enhanced access control and ownership delegation capabilities. Unlike regular schemas where object owners have full control over access permissions, managed schemas centralize access control management through the schema owner, making it easier to implement consistent security policies and delegate administrative responsibilities.

### Key Characteristics:

* **Centralized Access Control**: Only the schema owner can grant/revoke access to objects within the schema
* **Object Owner Limitations**: Individual object owners cannot grant access to their own objects
* **Enhanced Security**: Prevents unauthorized access grants and maintains consistent security policies
* **Simplified Administration**: Reduces the complexity of managing permissions across multiple object owners

## 🎯 When to Use Managed Schemas

### **Primary Use Cases:**

#### 1. **Multi-Tenant Environments**

-- Example: SaaS application with multiple customers  
CREATE SCHEMA customer\_data WITH MANAGED ACCESS;  
-- Each customer's data is isolated, but access is centrally managed

#### 2. **Regulated Industries (Banking, Healthcare, Finance)**

-- Example: Financial institution with strict compliance requirements  
CREATE SCHEMA pii\_data WITH MANAGED ACCESS;  
-- Ensures PII data access is controlled and auditable

#### 3. **Data Sharing Scenarios**

-- Example: Sharing data with external partners  
CREATE SCHEMA shared\_analytics WITH MANAGED ACCESS;  
-- Prevents partners from granting access to unauthorized users

#### 4. **Enterprise Data Governance**

-- Example: Centralized data warehouse with multiple teams  
CREATE SCHEMA enterprise\_warehouse WITH MANAGED ACCESS;  
-- Ensures consistent access policies across all teams

### **Benefits of Managed Schemas:**

#### 🔒 **Enhanced Security**

* **Prevents Privilege Escalation**: Object owners cannot grant access beyond their intended scope
* **Consistent Access Policies**: All access grants go through the schema owner
* **Audit Trail**: Clear visibility into who granted access to what

#### 👥 **Simplified Access Management**

* **Single Point of Control**: Schema owner manages all access permissions
* **Reduced Complexity**: No need to track multiple object owners’ permissions
* **Centralized Administration**: Easier to implement and maintain security policies

#### 📊 **Better Compliance**

* **Regulatory Compliance**: Meets requirements for data access control
* **Data Governance**: Supports enterprise data governance frameworks
* **Audit Readiness**: Clear access control documentation

## 🏗️ How Managed Schemas Simplify Ownership Delegation

### **Normal Schema vs Managed Schema Comparison**

#### **Normal Schema Challenges:**

-- Normal schema example  
CREATE SCHEMA normal\_schema;  
CREATE TABLE normal\_schema.sensitive\_data (id INT, ssn VARCHAR);  
  
-- Problems with normal schemas:  
-- 1. Object owner can grant access to anyone  
GRANT SELECT ON normal\_schema.sensitive\_data TO ROLE analyst\_role;  
-- 2. No centralized control  
-- 3. Difficult to track who has access to what  
-- 4. Object owners can bypass security policies

#### **Managed Schema Solution:**

-- Managed schema example  
CREATE SCHEMA managed\_schema WITH MANAGED ACCESS;  
CREATE TABLE managed\_schema.sensitive\_data (id INT, ssn VARCHAR);  
  
-- Benefits of managed schemas:  
-- 1. Only schema owner can grant access  
-- 2. Object owners cannot grant access to their own objects  
-- 3. Centralized access control  
-- 4. Consistent security policies

### **Ownership Delegation Scenarios**

#### **Scenario 1: Data Team Handoff**

**Normal Schema Problem:**

-- Data engineer creates objects  
CREATE TABLE normal\_schema.customer\_data AS SELECT \* FROM source\_table;  
  
-- Data engineer can grant access to anyone  
GRANT SELECT ON normal\_schema.customer\_data TO ROLE external\_partner;  
-- This bypasses security review!

**Managed Schema Solution:**

-- Data engineer creates objects  
CREATE TABLE managed\_schema.customer\_data AS SELECT \* FROM source\_table;  
  
-- Data engineer CANNOT grant access (will fail)  
-- GRANT SELECT ON managed\_schema.customer\_data TO ROLE external\_partner;  
-- Error: Insufficient privileges to operate on table 'CUSTOMER\_DATA'  
  
-- Only schema owner can grant access  
GRANT SELECT ON managed\_schema.customer\_data TO ROLE approved\_analyst;

#### **Scenario 2: Multi-Team Collaboration**

**Normal Schema Problem:**

-- Team A creates table  
CREATE TABLE normal\_schema.team\_a\_data AS SELECT \* FROM source\_a;  
  
-- Team B creates table   
CREATE TABLE normal\_schema.team\_b\_data AS SELECT \* FROM source\_b;  
  
-- Both teams can grant access independently  
-- No coordination or consistency in access policies

**Managed Schema Solution:**

-- Both teams create objects in managed schema  
CREATE TABLE managed\_schema.team\_a\_data AS SELECT \* FROM source\_a;  
CREATE TABLE managed\_schema.team\_b\_data AS SELECT \* FROM source\_b;  
  
-- Only schema owner coordinates access  
-- Ensures consistent security policies across all teams  
GRANT SELECT ON ALL TABLES IN SCHEMA managed\_schema TO ROLE data\_analysts;

#### **Scenario 3: External Data Sharing**

**Normal Schema Problem:**

-- Internal team creates shared data  
CREATE TABLE normal\_schema.shared\_metrics AS SELECT \* FROM internal\_data;  
  
-- Internal team member grants access to external partner  
GRANT SELECT ON normal\_schema.shared\_metrics TO ROLE external\_partner;  
  
-- External partner can then grant access to others  
-- Security breach potential!

**Managed Schema Solution:**

-- Internal team creates shared data  
CREATE TABLE managed\_schema.shared\_metrics AS SELECT \* FROM internal\_data;  
  
-- Only authorized schema owner grants access  
GRANT SELECT ON managed\_schema.shared\_metrics TO ROLE external\_partner;  
  
-- External partner cannot grant access to others  
-- Security maintained!

## 🛠️ Implementation Examples

### **Creating Managed Schemas**

-- Basic managed schema creation  
CREATE SCHEMA analytics WITH MANAGED ACCESS;  
  
-- Managed schema with specific owner  
CREATE SCHEMA customer\_data WITH MANAGED ACCESS;  
GRANT OWNERSHIP ON SCHEMA customer\_data TO ROLE data\_governance\_admin;  
  
-- Converting existing schema to managed access  
ALTER SCHEMA existing\_schema SET MANAGED ACCESS;

### **Access Management in Managed Schemas**

-- Schema owner grants access to objects  
GRANT SELECT ON TABLE managed\_schema.sensitive\_table TO ROLE analyst\_role;  
GRANT SELECT ON ALL TABLES IN SCHEMA managed\_schema TO ROLE reporting\_role;  
  
-- Schema owner can revoke access  
REVOKE SELECT ON TABLE managed\_schema.sensitive\_table FROM ROLE analyst\_role;  
  
-- Bulk access management  
GRANT USAGE ON SCHEMA managed\_schema TO ROLE data\_consumers;  
GRANT SELECT ON ALL TABLES IN SCHEMA managed\_schema TO ROLE data\_consumers;

### **Object Creation in Managed Schemas**

-- Objects can be created by any user with appropriate privileges  
CREATE TABLE managed\_schema.new\_table (  
 id INT,  
 sensitive\_data VARCHAR(100)  
);  
  
-- Object creators cannot grant access (managed access prevents this)  
-- Only schema owner can manage access

## 📋 Best Practices

### **1. Schema Ownership Strategy**

-- Assign managed schemas to dedicated governance roles  
CREATE ROLE data\_governance\_admin;  
GRANT OWNERSHIP ON SCHEMA managed\_schema TO ROLE data\_governance\_admin;

### **2. Access Policy Documentation**

-- Document access policies  
-- Example: Only analysts with proper training can access PII data  
GRANT SELECT ON TABLE managed\_schema.pii\_table TO ROLE certified\_analysts;

### **3. Regular Access Reviews**

-- Query to review current access  
SELECT   
 grantee\_name,  
 privilege\_type,  
 object\_name,  
 granted\_on  
FROM snowflake.account\_usage.grants\_to\_roles  
WHERE object\_name LIKE 'MANAGED\_SCHEMA%'  
ORDER BY grantee\_name, object\_name;

### **4. Gradual Migration Strategy**

-- Phase 1: Create new managed schemas  
CREATE SCHEMA new\_managed\_schema WITH MANAGED ACCESS;  
  
-- Phase 2: Migrate existing schemas  
ALTER SCHEMA existing\_schema SET MANAGED ACCESS;  
  
-- Phase 3: Update access policies  
-- Review and update all existing grants

## ⚠️ Important Considerations

### **Limitations:**

1. **Object Owner Restrictions**: Object owners cannot grant access to their own objects
2. **Migration Complexity**: Converting existing schemas requires careful planning
3. **Access Management Overhead**: Schema owner must handle all access requests
4. **Role Dependencies**: Requires well-defined role hierarchy

### **Migration Considerations:**

1. **Existing Grants**: Review all existing grants before conversion
2. **Application Impact**: Ensure applications can handle access changes
3. **User Training**: Train users on new access request procedures
4. **Testing**: Thoroughly test access patterns in development environment

## 🔍 Monitoring and Auditing

### **Access Audit Queries**

-- Monitor managed schema access  
SELECT   
 grantee\_name,  
 privilege\_type,  
 object\_name,  
 granted\_by,  
 granted\_on  
FROM snowflake.account\_usage.grants\_to\_roles  
WHERE object\_name LIKE 'MANAGED\_SCHEMA%'  
AND granted\_on >= DATEADD(day, -30, CURRENT\_DATE());  
  
-- Track schema access patterns  
SELECT   
 schema\_name,  
 COUNT(\*) as access\_grants,  
 COUNT(DISTINCT grantee\_name) as unique\_users  
FROM snowflake.account\_usage.grants\_to\_roles  
WHERE schema\_name LIKE '%MANAGED%'  
GROUP BY schema\_name;

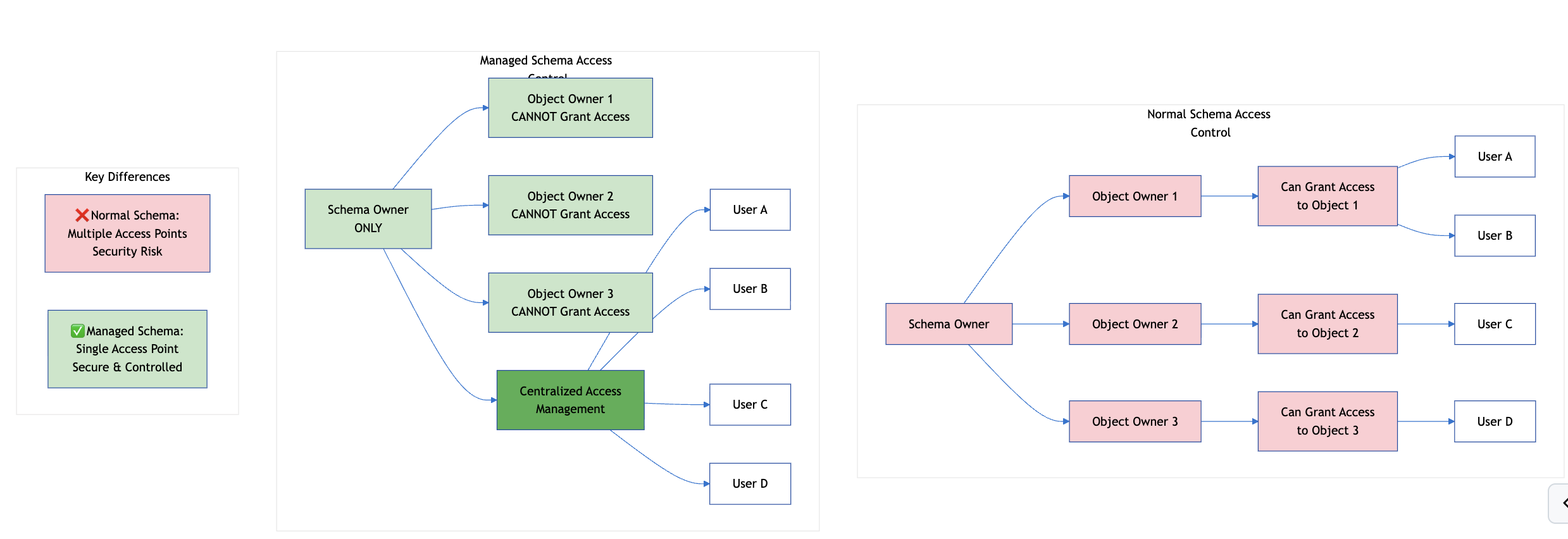
## 📚 Additional Resources

* [Snowflake Documentation: Managed Access Schemas](https://docs.snowflake.com/en/sql-reference/sql/create-schema.html#managed-access-schemas)
* [Snowflake Security Best Practices](https://docs.snowflake.com/en/user-guide/security.html)
* [Data Governance Framework](https://docs.snowflake.com/en/user-guide/data-governance.html)

**Note**: Managed schemas are particularly valuable in enterprise environments where data governance, compliance, and security are critical requirements. They provide a robust foundation for implementing consistent access control policies across your Snowflake data platform.

## 📊 Managed Schemas Access Control Flow Diagram

The following diagram illustrates how managed schemas control access and simplify ownership delegation:



## 🔄 Managed Schema Implementation Workflow

## 

## 🔐 Security Comparison Matrix

| Aspect | Normal Schema | Managed Schema |
| --- | --- | --- |
| **Access Control Points** | Multiple (Schema + Object Owners) | Single (Schema Owner Only) |
| **Security Risk** | High (Privilege Escalation Possible) | Low (Centralized Control) |
| **Audit Complexity** | Complex (Multiple Grant Sources) | Simple (Single Source) |
| **Compliance** | Difficult to Maintain | Easy to Maintain |
| **Administration** | Distributed & Complex | Centralized & Simple |
| **Object Owner Powers** | Full Access Control | No Access Control |
| **Governance** | Weak | Strong |