## Release Date

July, 2016

## Personnel

|  |  |
| --- | --- |
| Mark Dean | CDW Enterprise Architect |
| Richard Pham | CDW Enterprise Architect |
| Thomas Huneke | CDW ETL IT Specialist |
| Dennis Swenson | Microsoft (CIS) |

## Release Overview

The Active Directory 1.0 CDW data domain does not contain references to other CDW domains, Staff, and VistA-sourced data. The Active Directory 1.0 domain is the first iteration of creating a domain SQL representation of the data found in VA’s Active Directory.

## Special Notes for this Release

### SPV and LSV View Versions

SPV view versions have not been created at the time of this release because the columns associated with SPV objects (e.g. *ETLCreateDateTime*, *ETLEditDateTime*, and *ETLBatchID*) are not being populated by the current ETL process.

LSV view versions are not being populated as Sta3n is not a part of this domain.

### AccountSID

AccountSID on NDim.ActiveDirectoryAccount is not a foreign key to another domain object. For more information about this column’s values, see the DW View Field Description in the metadata.

## Schema & Security

List all the database schemas included in the data domain.

* **NDim**: Dimensions without a primary VistA data source.

## Details

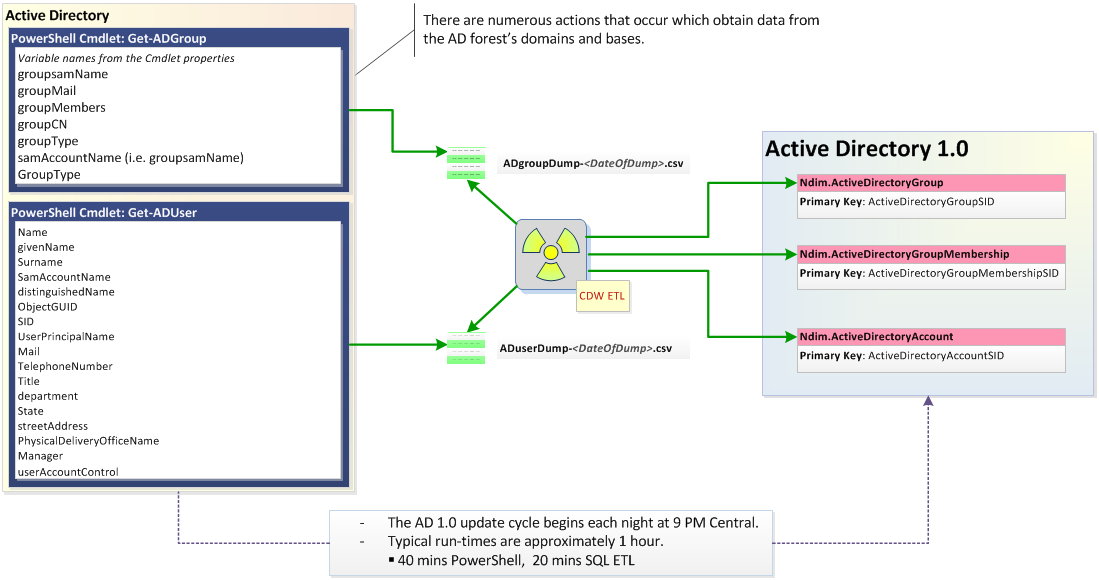
### View Versions

ActiveDirectoryGroup: Contains a set of group object properties from Active Directory.

ActiveDirectoryAccount: Contains Active Directory attributes for group or user objects (i.e. Active Directory Accounts).

ActiveDirectoryGroupMembership: Contains a set of group and group member object properties from Active Directory including the recursive relationship a group has when it is the first ancestor.

### Data Flow Diagram



## Data Quality

### View Column Names Issues

*NDim.ActiveDirectoryGroup* contains columns that have been misidentified as to their source values. Use column aliases until a new view version can be released.

**CommonName**: incorrect label. The identifier “groupCN” used during the AD query has been found for calling $\_.canonicalname. The canonical name is not the same as the common name (i.e. CN) which is an element of the Distinguished Name (i.e. a value that defines the path to a specific object). The difference is a path versus a single value.

**GroupName**: incorrect definition. This column’s values may stem from $\_.samaccountname. Any group’s name and SamAccountName values may differ.

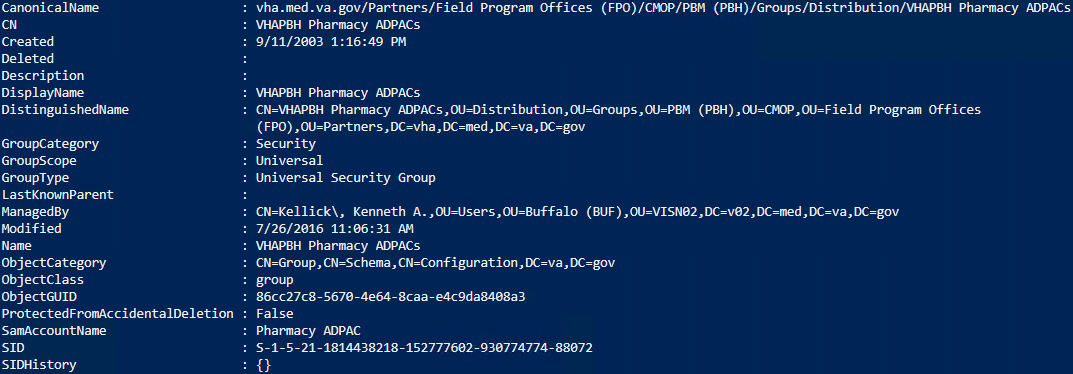


Figure 1. Ex: CN, Name, and SamAccountName not the same for a group.

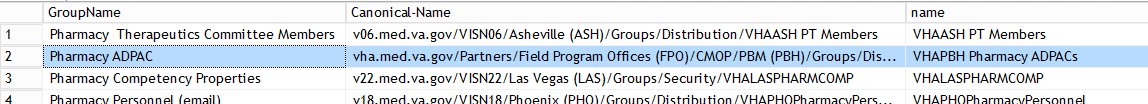


Figure 2. Ex: Entry in view

### Data Types

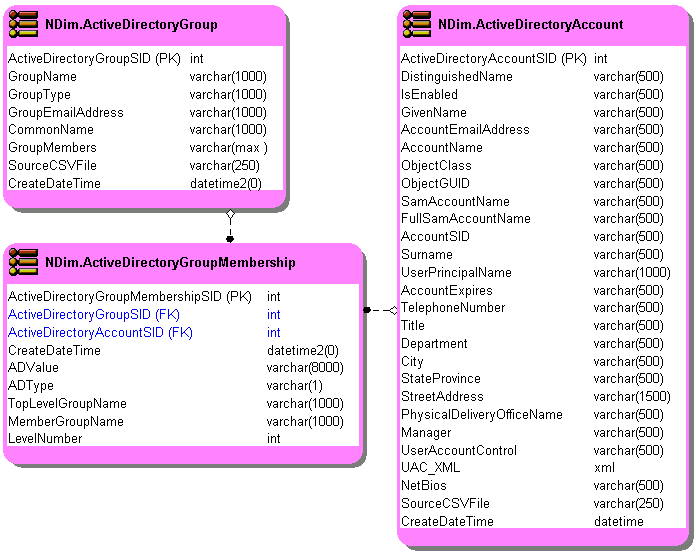
There may be performance issues in a SQL WHERE clause, and an error will be raised about the byte size limitation in the Order By statement inside an OVER clause (i.e. a sort used inside a Range window frame) when using varchar(1000) columns. In the latter case, use the surrogate SID key values in the view versions instead.

### 100% Null Fields

There are 3 columns in NDim.ActiveDirectoryAccount where there are no known values in this domain release: **IsEnabled**, **ObjectClass**, and **AccountExpires**. These were retained in the views with the expectation that the values will become available as the domain matures without the need for a subsequent domain release.

Please notify the [CDW Architect Team](mailto:CDWArchitectTeam@va.gov?subject=Domain:%20Active%20Directory%201.0;%20Data%20Quality) of any data quality issues found.

## ER Diagram



## Sample SQL Queries

### Create a result set of group member

Let’s say you want to create a result set with the user details for a given Active Directory group. We will return a modest set of the total available columns for the user and collect those users together based on their membership in a single group (ArchTeam will play guinea pig on this).

In this example, we’ll create a basic inner join between the view versions NDim.ActiveDirectoryAccount and NDim.ActiveDirectoryGroupMembership using the ActiveDirectoryAccountSID key common in the two view versions. This is typical fare for CDW using the common key as a Foreign Key reference.

|  |
| --- |
| SELECT  a.Surname,a.GivenName,a.AccountName,a.NetBios As [Domain],a.FullSamAccountName  ,a.AccountEmailAddress,a.ActiveDirectoryAccountSID  ,a.TelephoneNumber,a.Title,a.Department,a.City,a.StateProvince,a.StreetAddress  FROM NDim.ActiveDirectoryAccount As a  INNER JOIN NDim.ActiveDirectoryGroupMembership As b  ON b.ActiveDirectoryAccountSID = a.ActiveDirectoryAccountSID  WHERE TopLevelGroupName = 'CDW\_ArchitectTeam' AND ADType = 'U'  ORDER BY 1; |

Order By 1: *not exactly a best practice, but good enough for quick SQL work.*

### Construct a group hierarchy with NDim.ActiveDirectoryGroupMembership

Kindly consider the following example: you want to construct a hierarchy tree of a workgroup such that at the top of the tree is the ancestor and everything that falls under are the descendants and siblings. To avoid eye strain, the application that you’re developing only needs to display the first 4 groups at each level of the tree. Additionally, you want to include any members on the ancestor element… er, row.

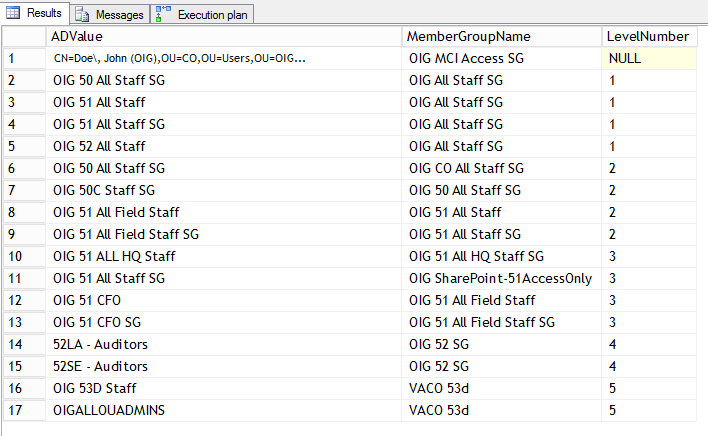
To make it interesting, we’ll use a group that has multiple levels of recursion[[1]](#endnote-1), *OIG MCI Access SG*.

#### NDim.ActiveDirectoryGroupMembership

|  |
| --- |
| ;WITH Members As (  SELECT ActiveDirectoryGroupMembershipSID,ActiveDirectoryGroupSID, MemberGroupName,ADValue,LevelNumber  ,ROW\_NUMBER() OVER(Partition By LevelNumber Order By ADValue) As RowNumber  FROM NDim.ActiveDirectoryGroupMembership  WHERE TopLevelGroupName = 'OIG MCI Access SG'  AND ADType = CASE WHEN LevelNumber > 0 THEN 'G' ELSE 'U' END  )  SELECT DISTINCT ADValue,MemberGroupName,LevelNumber  FROM Members WHERE RowNumber <= 4  ORDER BY LevelNumber; |

This query constructs a CTE to gather all the data that meets your criteria. Notice the CASE statement in the WHERE clause that uses the value in *LevelNumber* to determine the group type to return. The ORDER BY is not truly necessary at this stage and for limiting larger sets, you might do well using OFFSET and FETCH.

The windowed function, ROW\_NUMBER()[[2]](#endnote-2), creates an integer value for use in the outer query that restricts the result set to 4 or less members.



*\*ADValue in Row 1 has been changed from the actual value*

The IO and TIME statistics look good and the Execution plan is straightforward. Let’s visualize these data as an organization chart… which is a hierarchy!

## References

[CDW Metadata Page](https://vaww.cdw.va.gov/metadata/default.aspx)

[CDW Support](https://vaww.cdw.va.gov/Support/SitePages/CDWSupportHome.aspx)

1. This recursion refers to a SQL query that calls itself. Here, the recursion is done in the ETL process and the anchor and recursive members are defined by the TopLevelGroupName and the LevelNumber. Additional recursions may be done to order the table to construct a hierarchy tree. Typically, recursive SQL queries are constructed using a Common Table Expression (CTE) or with the HierarchyID datatype. [↑](#endnote-ref-1)
2. ROW\_NUMBER() is used here in place of some fancy COUNT() function footwork. A window function normally increases performance over other methods that use self-joins and cursors. They aren’t infallible and care must be taken in their usage. Yep – there will be two operators associated with window functions: Sequence Project and Segment. These usual suspects aren’t generally something to bother with as they are concerned with adding the column to the result set and the division of the input into segments. Watch out for Sort and Table Spools (i.e. a working table in Tempdb) that can creep in to the execution plan –ahem, SUM(). These are your resource killers. [↑](#endnote-ref-2)