Data modeling hierarchy with multiple node types – SAP NetWeaver MDM

Applies to:

SAP NetWeaver MDM 5.5 Data Modeling

Summary

Using SAP NetWeaver MDM data modeling we can design complex modeling requirements. In this document, I present a business requirement to create a complex hierarchy with multiple node types and show how we can answer this business requirement while demonstrating three possible solutions.

Author(s):

Company: SAP

Created on: 20 November 2006

Author Bio

Noam Berda, Solution Architect has joined SAP in 2001 to the Portal Development team. Noam's current role is solution architect in the SSM group and he is leading the solution office MDM and Enterprise SOA activities in JAPAC. As part of his role, he is working with light house customers to adopt MDM and Enterprise SOA solutions.

Table of Contents

Table of Figures

Business Scenario

A customer (a global retailer) wishes to create a master data repository to store information about the locations it is active at. The data includes information about states, regions, counties, and cities (for example, the state capital, county population) and specific outlets. Each outlet is associated with a geographical hierarchy (state, region, etc.) It's expected that it will be easy to search and maintain information using the MDM Data Manager, and that the repository will be designed in a way that will ensure data integrity.

The information to store includes the following objects and hierarchy:

1. States – each state holds a name, postal, traditional name and capital.
2. Regions – a region is associated to one state. It holds the name of the region.
3. Counties – a county is associated to one region. It holds the county name, population, FIPS code, and the county code.
4. Cities – a city is associated to one county. It holds the city name.
5. Outlet – an outlet is associated to one city. It holds a unique identifier, a name, outlet manager name, and contact information (address and phone number.)

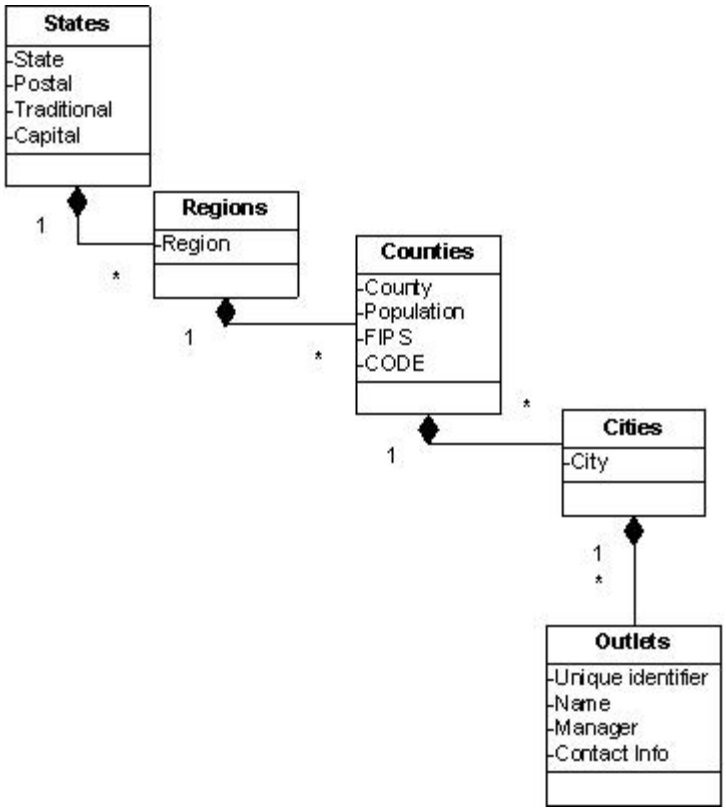

Figure 1 - Data hierarchy

Finding the right model

Identifying the main entity

The main entity in a repository is the central object users will be interested in. The main table, which holds the main entity, is designed to store a large volume of records, which means the main entity is usually also the entity which has the most occurrences. In this case, the main entity would be the outlet records – all the other objects (state, region, county and city) are used to give a geographical and organizational context to outlet objects, which are the main focus of attention. Another sign that Outlet is the main entity is that we see it has many-to-one relationships to the other objects (either directly in the case outlets → cities or indirectly through cities to the other objects) – each city object is likely to be referenced by multiple instances of Outlet. The main entity of a repository should have many-to-one relationships with the secondary objects. In "MDM-speak" it means that records in the main table should have many-to-one relationships to records in lookup tables and not vice versa.

Modeling the geographical hierarchy

SAP NetWeaver MDM provides some tools to create hierarchies. The first step should be to evaluate whether we can use one of these tools to meet our requirements.

The hierarchy table type

Each Outlet record should be associated to a hierarchy of geographical information – a city, county region and finally a state. The SAP NetWeaver MDM hierarchy table type was designed to describe a hierarchy in which all objects are from the same type and share the same set of fields (for example, a hierarchy between main table records.) The case at hand requires a hierarchy between different types of objects (outlet, city, county etc), which means the hierarchy table type can not be used.

The taxonomy table type

SAP NetWeaver MDM taxonomy tables give us all the benefits of a standard hierarchy with the advantage of attributes. Attributes attached to a node in the hierarchy tree are used to identify all associated master records under this node. For example in the case of an electronics shop, the hierarchy branch TV can have the attribute size and therefore all products that are attached to the TV branch will need to specify their size.

For our case it is not suitable enough. As mentioned above, the attributes are used to identify the master record attached to the hierarchy node rather than the hierarchy node itself. For example "Population" in counties is used to identify the population of a certain county other then the outlet that is associated to this county and therefore taxonomy is not suitable for our case.

A custom hierarchy

Since neither the hierarchy nor the taxonomy table types meet our requirements, a custom solution is required to model the hierarchy. Lookup tables are the simplest way to relate objects in the main table to other objects in an MDM repository – an outlet record can be related to a city record using a lookup to a cities lookup table. There are several ways of utilizing lookup tables to build the hierarchy we need, each with its advantages and disadvantages – some solutions enhance the flexibility of the model but limit the ways information can be searched, while others maximize the search capabilities but make it possible to have data integrity issues.

The selected data model should:

1. Have the ability to meet the requirements – store all the information in the best possible way.
2. Ensure information will not be duplicated in the repository.

Solution 1 – Maximum search capabilities using multiple lookup tables

The first approach maximizes the MDM Data Manager search capabilities for the repository. Each outlet master table record is connected to 4 flat lookup tables which represent the different geographical objects.

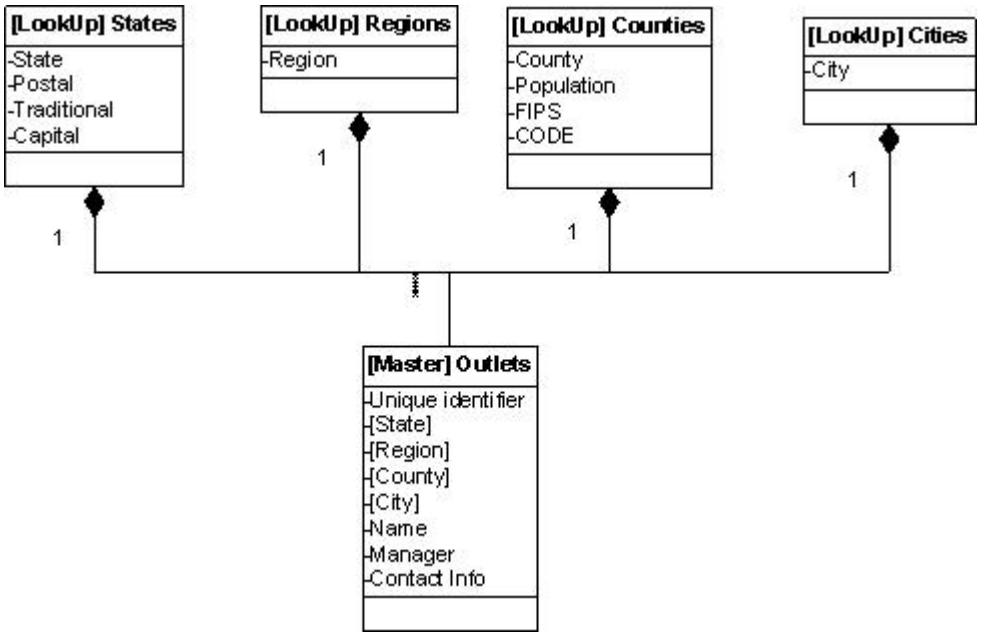

Figure 2 - Main table connected to 4 lookup tables, one for each hierarchy object

Each outlet record has lookup fields to the states, regions, counties and cities tables. For each record the user will have to fill all 4 values independently of one another. This solution does not really create a hierarchy between the geographical objects and the outlet records.

Advantages

- Maximal search capabilities: it's possible to search using each of the 4 lookup tables, giving the MDM Data Manager GUI maximum strength

Challenges

- Hard to maintain: Every change in the geographical tree structure will require changes in multiple records. For example, if a county will move from one region to another, the region field will have to be changed in all the outlet records which belong to that county.
- No visible hierarchy structure: When looking at an outlet record it is hard to understand the geographical hierarchy the record is associated to.
- No fixed hierarchy tree: Since the different geographical objects are unrelated to each other (city has no relation to county, county has no relation to region, etc) the hierarchy structure cannot be pre-defined. The structure is built for each outlet record when filling values in its fields.
- No constraints: Because there is no real hierarchy structure there are no constraints limiting the geographical combinations – an outlet record could theoretically be connected to a state and a region which does not exist in that state.

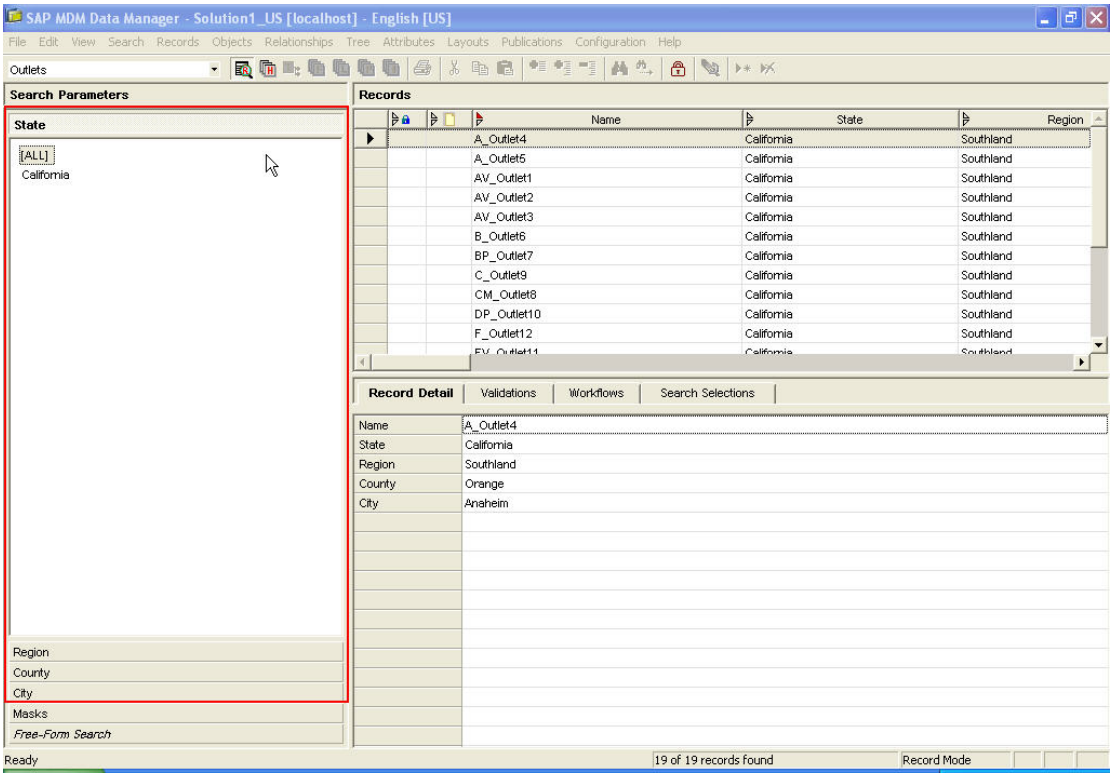

Figure 3 - 4 Lookup tables solution gives a lot of search capabilities

Solution 2 – A “clean” data model using nested lookup tables

The 2nd approach tries to create the “cleanest” data model possible – to replicate the exact requirement in the SAP NetWeaver MDM repository structure. In this model we again use 4 lookup tables – one for each geographical object, but instead of associating outlet records to each of the four lookup tables directly we use a single lookup field to link Outlet to a single record in the cities tables, and link every lookup table to the next object in the hierarchy using lookup fields.

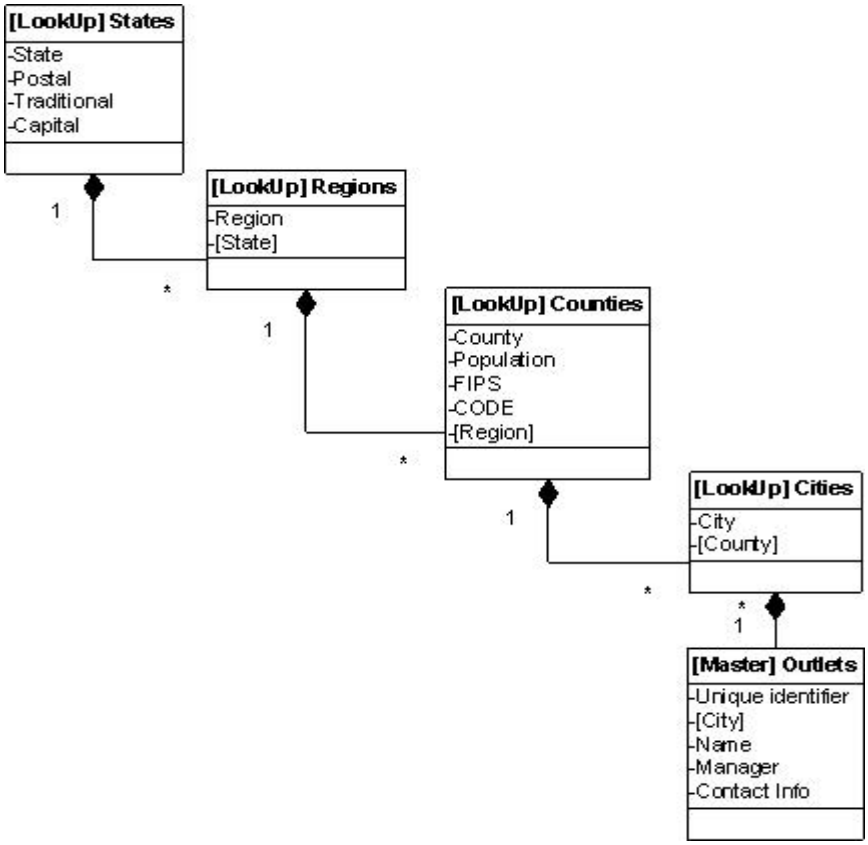

Figure 4 - Using nested lookup tables to mimic the create a "clean" model

Unlike the previous suggestion, this structure really creates a hierarchy – each node is associated to the parent node and by using different lookup table we can add different fields to each object type in the hierarchy.

Advantages

- Easy to maintain: Because of the structure created by nesting the lookup tables it's easy to change the hierarchy structure. Moving a county from one region to another requires only a change in the corresponding record in the counties table. The change is done only on one field in one location, much like in a standard hierarchy table. Adding a new level to hierarchy structure is easy as well.
- Visible hierarchy structure: The hierarchy structure is fairly easy to understand, visualize and manage. The similarity to the standard hierarchy tree is a strong advantage.
- Fixed hierarchy tree: The hierarchy structure is predefined using the nested lookup tables. Outlet records can only associate to the lowest level of the hierarchy (in this case, a city) – the rest of the tree is predefined.
- Constrained structure: The hierarchy structure is constrained. Every node can be associated only to one parent.

Challenges

- Limited search capabilities in the Data Manager GUI: By using nested lookups we limit the search tabs to the lookup table which is directly connected to the main table records (the "cities" lookup table). This means that it's impossible to search or filter outlets by state, region or county.

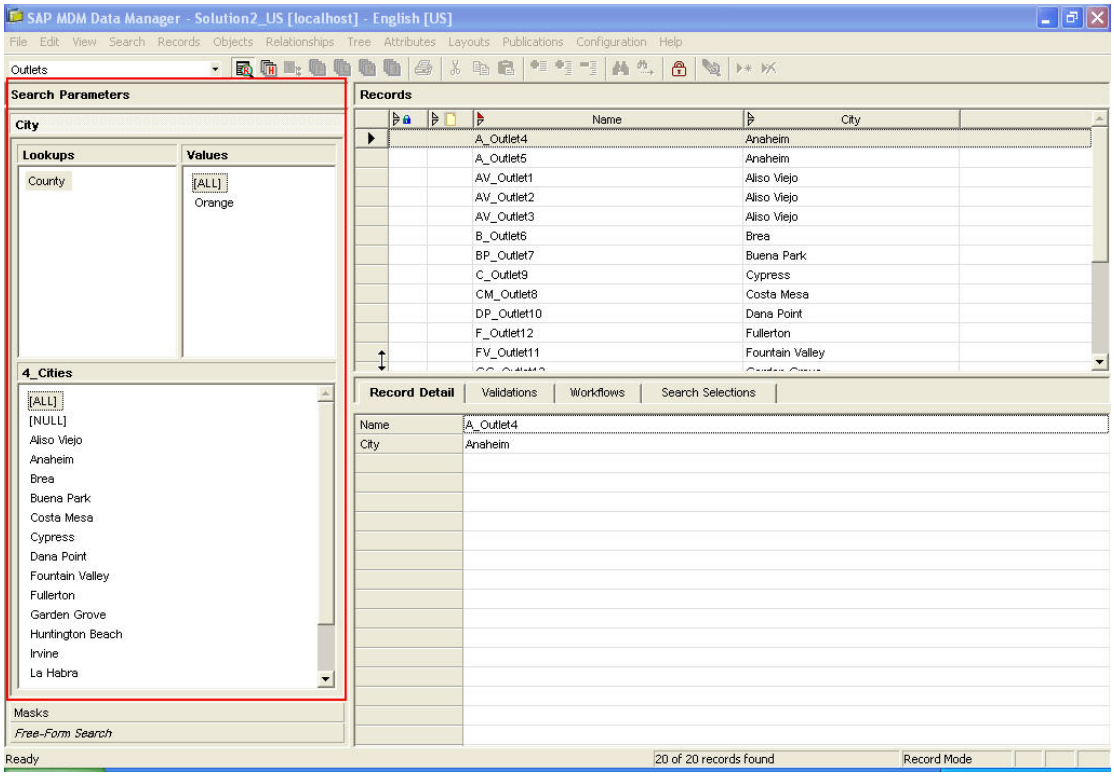

Figure 5 - Using nested lookups provides a clean model but limits the search capabilities

Solution 3 – Nested lookups with a "Helper" table

The 3rd approach is a hybrid of the first two – it combines elements of both ideas to create a third option. This approach uses 4 lookup tables, one for each geographical object, which are not related to one another (similar to the 1st approach). It also adds another lookup table, "Flat Tree", which has lookup fields for each of the other four tables (states, regions, counties and cities). By making all the lookup fields in the table "Flat Tree" display fields and ordering them according to their hierarchal levels (state is first, region second, etc) the records in that table can be used to create branches in the tree. The main table has a lookup field connected to the "Flat Tree" table which is used to set the hierarchy for a specific outlet object. For example, a record in the "Flat Tree" table can have the following values:

- State: USA
- Region: California
- County: Northern California
- City: City 1

When looking at this record from the lookup field in the main table it will show like this: "USA; California; Northern California; City1"

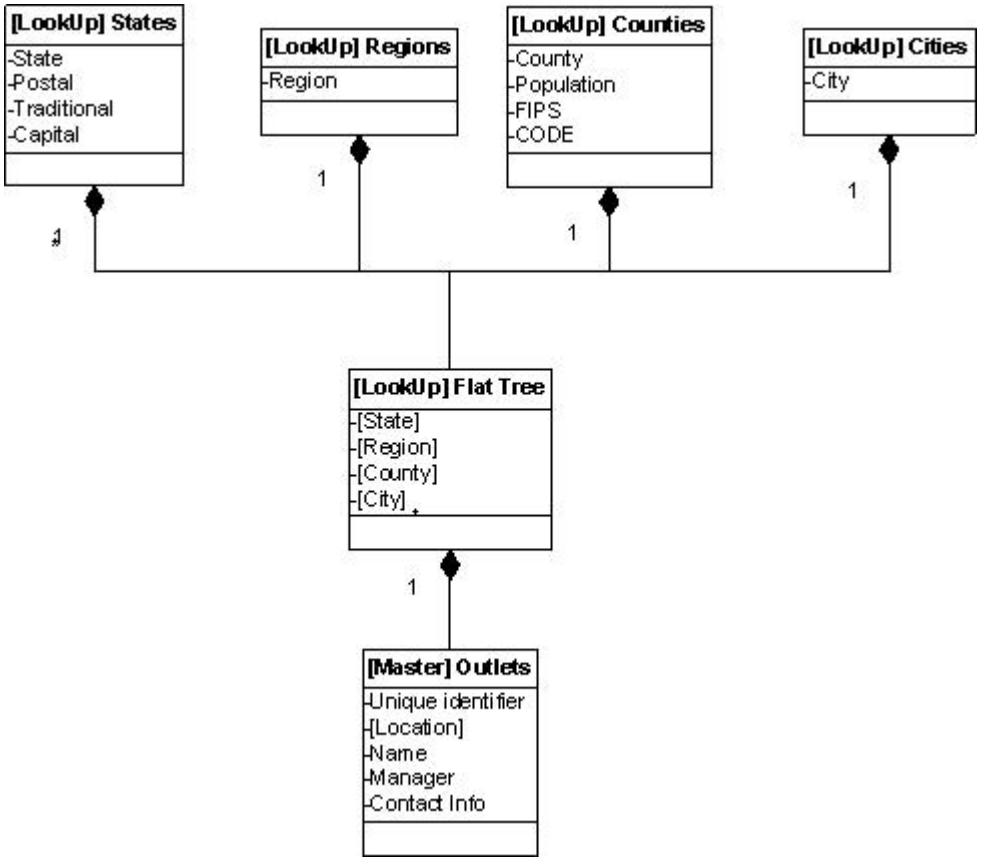

Figure 6 - Using 4 lookup tables and an extra helper lookup table give us another alternative

Advantages

- Easy to maintain: Since the hierarchy is set in a single table (the “Flat Tree” table), it’s easy to move nodes. For example moving a county from one region to another requires changing only the “Flat Tree” table records which point to that county. These records can be easily filtered using a simple search on the “Flat Tree” table.
- Fixed hierarchy tree: The hierarchy tree is predefined in the “Flat Tree” table. There’s no way for an outlet object to attach itself to a non-existing hierarchy structure which was not defined beforehand in the “Flat Tree” table.
- Good Search support in the Data Manager GUI: Since the structure is only 2 levels deep, the MDM Data Manager GUI lets us filter and search main table records based on the “Flat Tree” lookup table and its lookup fields.

Challenges

- Hard to see hierarchy structure: Because the hierarchy tree is broken down to specific branches, it’s not trivial to visualize the complete tree structure. It is however pretty straightforward to understand which County a record belongs to once it’s assigned to one. This problem could be solved by creating a tree representation of the flat table using the MDM API (the tree can even be displayed in the web pane if created in a web application)
- No constrained structure: The hierarchy structure is not constrained – it’s possible for example to create a county that has a region attached to a state it does belong to. This can be controlled by allowing only specific users with the proper knowledge to have edit permissions on the “Flat Tree” table.

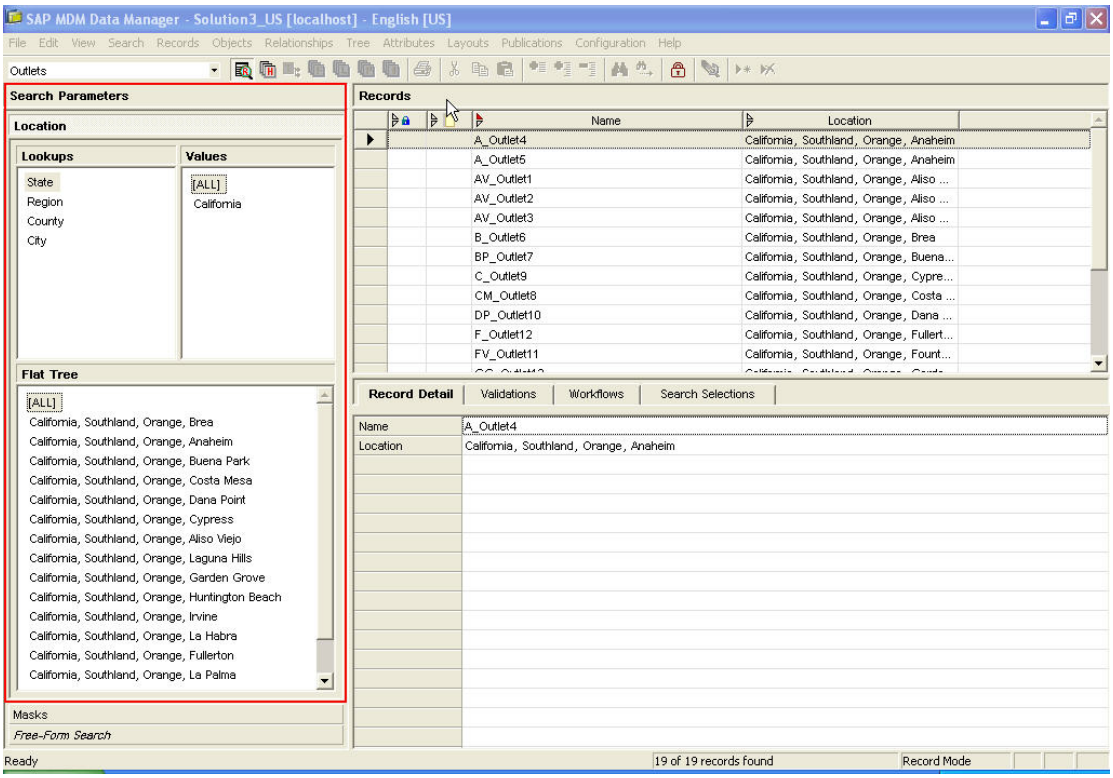

Figure 7 - The 3rd option provides good search functionality in the Data Manager

Chosen Solution

After evaluating all the possible solutions we decided that we are going to go with the 3rd option. In this particular project there was a lot of focus on the MDM Data Manager search functionality, so the 3rd approach was most appropriate as it provides a balance between the other two regarding the “cleanness” of the model and the search capabilities it gives.

Summary

In the above document we have seen a business requirement to create a complex hierarchy where each node in the hierarchy would be from different type and have different fields. The standard hierarchy and taxonomy tables are designed for different types of hierarchies, so a custom data model had to be conceived using lookup tables. This document presented three alternatives, each with its own benefits and challenges:

- Solution A – Maximum search capabilities using multiple lookup tables
- Solution B – A “clean” data model using nested lookup tables
- Solution C – Nested lookups with a “helper” table

Each project has its own requirements and on this particular project we opted for the 3rd solution for its balance between a “clean” data model and GUI support.

Appendix

The ZIP file contains everything needed to see and test the solutions proposed in this document:

- 3 repository archives, one for each proposed solution. The repositories have data in them and import maps.
- Excel files with demo data. These Excel files can be imported to the repositories using the import maps saved in the repositories.

  Zip File

Copyright