1 The Metadata Engine - Concepts

1.1 Introduction

The previous two chapters have discussed in detail some of the concepts, projects and policies that are related to the concepts investigated in this thesis. The purpose of this chapter in is to outline all the concepts that can be used in the development of an operational Metadata Engine. The chapter will concentrate on the concepts that have to be mastered in order too develop a metadata engine, what the different options are to develop a engine, along with the advantages and disadvantages of these options. The chapter does not describe the specific approach that was undertaken in the prototype that was developed in this research. Chapter 5 "The Metadata Engine - Developed" describes in detail which of the approaches outlined in this chapter were adopted in the prototype.

1.2 The Metadata Engine Concept

The importance of metadata engines is probably best emphasised when one looks at how decision makers use spatial data. Currently decisions requiring spatial data are generally made with the assistance of a stand-alone GIS. This GIS does of course have a limited amount of data, on a limited number of topics, stored within it. Thus the decisions that one makes while using this GIS are obviously biased to that data. The best decisions are made when as much information as possible is taken into consideration. To get as much information as possible into the decision making process the decision makers could keep adding more and more data to their closed GIS, or they could undertake a distributed approach where they essentially have an open GIS. This approach takes advantage of other relevant datasets, that are available over the network, and consults them to gain the results that the user wants. This approach is better as:

1) The physical storage in the users own GIS is minimised;

2) The data that they are using in the remote databases is more likely to be up to date as it is likely to be being drawn from the data custodians own database, or at least a mirror database; and

3) It is likely to be much more economical to access small pieces of the required datasets remotely when it is necessary, rather than purchasing the whole dataset and the subsequent updates.

When implemented distributed processing allows the creation of what could be called a "virtual database". The users of a "virtual database" would be able to use it as if they were accessing a single database that is located on their own machine. They do not need to know that the "virtual database" that they are using is in fact made up of any number of distinct databases located at any number of different locations. The databases could also be under the control of differing custodians from both the private and public sectors as well as being stored in differing proprietary database systems. All this is hidden from the user.

A good example of where a "virtual database" could be implemented is in any state or national government throughout the world. Each of the departments that exist within the government are likely to be custodians for one or more spatial datasets and are hence responsible for collecting, maintaining and distributing the data from those datasets. This is part of their core business. It is also part of their core business to use that data to make decisions. It is likely that other datasets maintained by other government departments are used in conjunction with their own datasets to make these decisions. It is not part of their core business to duplicate the collection of these datasets and whole dataset transfers between the departments is costly.

If a "virtual database" were established between the government departments it would be possible for a user to locate a parcel boundary on their own GIS application and then, via a network, reference required themes, such as base mapping, surface geology, environmental constraints, flood hazards, land ownership, planning zones

and transportation routes. These themes may be owned and maintained by any number of other government departments and agencies, however the user is unaware of this and has the capability to analyse them spatially without the need for file transfer and data duplication (Glover 1997). With the advances in communication technologies meaning faster networks this style of application becomes a reality.

One of the key processes that has to occur for distributed processing to become a reality is for metadata, for spatial data, to move away from being an end product in its own right, to being a background tool. As a background tool it would be used in a very similar way as in a standalone GIS/database which uses it for describing the internal make up of the GIS/databases. In this case the metadata would describe the make up of the "virtual database". The metadata would outline where each database was located, its format, access constraints and so on, all of which would be needed to access the remote databases over a network. A metadata engine could use these metadata to parse user queries so that they can be sent to the individual databases. The individual databases would resolve the queries and then return the results to the metadata engine for recomposition for the user to view. Figure 1-1shows the architecture of this style of system.

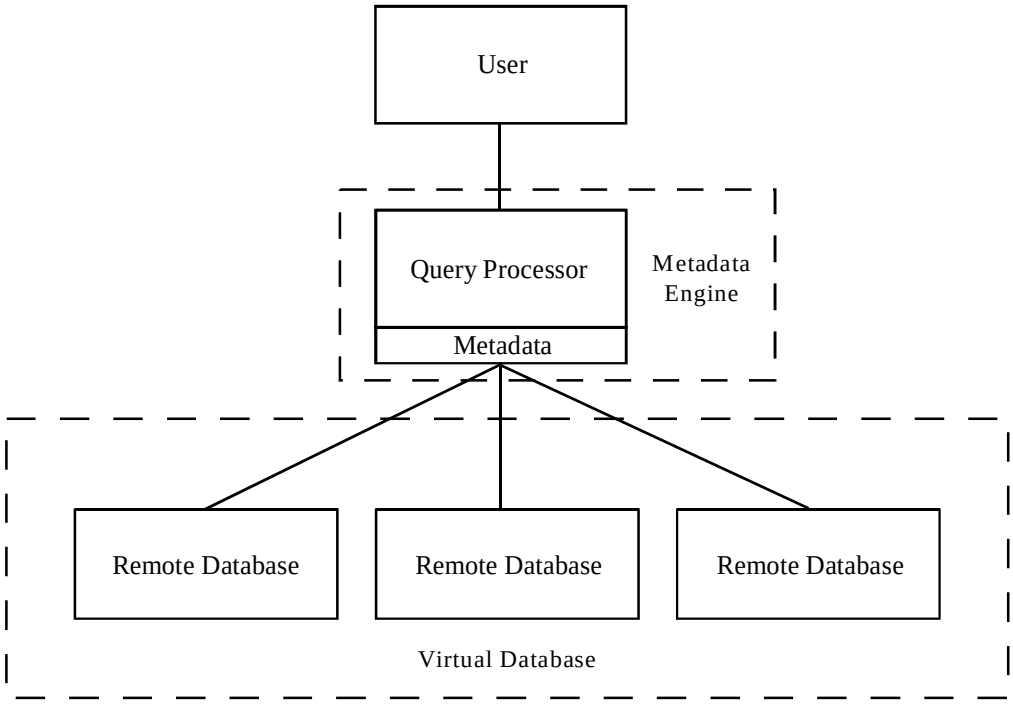

Figure 1-1: Virtual database architecture (Phillips, *et al.* 1998b).

Another example of where distributed processing is likely to be used in the future was recently outlined by Al Gore (1998) when discussing his vision for a "Digital Earth". He said:

> *Imagine, for example, a young child going to a Digital Earth exhibit at a local museum. After donning a head-mounted display, she sees the Earth as it appears from space. Using a data glove, she zooms in, using higher and higher resolutions, to see continents, the regions, countries, cities, and finally individual houses, trees, and other natural and man made objects. Having found an area of the planet she is interested in exploring, she takes the equivalent of a "magic carpet ride" through a 3-D visualisation of the terrain. Of course, terrain is only one of the many kinds of data with which she can interact. Using the systems' voice recognition capabilities, she is able to request information on land cover, distribution of plant and animal species, real-time weather, roads, political boundaries, and population. She can also visualise the*

> *environmental information that she and other students all over the world have collected as part of the GLOBE project. This information can be seamlessly fused with the digital map or terrain data. She can get more information on many of the objects she sees by using her data glove to click on a hyperlink. To prepare for her family's vacation to Yellowstone National Park, for example, she plans the perfect hike to see the geysers, bison, and bighorn sheep that she has just read about. In fact, she can follow the trail visually from start to finish before she ever leaves the museum in her home town.*

While this scenario may seemed far fetched to some, most of the technologies and capabilities that would be required to build a Digital Earth are either being developed, or indeed are already here. Essentially the ideas outlined by Gore are just an extension of the principles of distributed processing of spatial data. Once distributed processing systems for spatial data are in circulation, which is not that far away, extensions to the concept, like virtual reality interfaces, will make the Digital Earth concept of Gore possible. This of course assumes that the network and computer speeds to complete the required data processing are sufficient. The rapidly increasing speed of development of both of these technologies suggests this should not be a problem.

One might say "Why does the Digital Earth have to be developed using a distributed processing approach? Why can't all the data be integrated into one enormous data warehouse?" True all the data could be integrated into one data warehouse, computer speeds and storage are progressing rapidly enough to make this possible. The real problem with the data warehouse approach is that of data custodianships. The datasets that will be involved in the Digital Earth project will be owned by many different organisations in government and the private sector from all around the world. These organisations will wish to remain in control of their datasets. If all the datasets are merged into one data warehouse the individual organisations lose control of their datasets. It is fair to assume that most countries involved in such a project would be very keen to maintain control over the datasets that are able to be accessed. By

undertaking a distributed approach, each of the custodians retain control of their datasets as they are accessed through their site via a network.

1.3 The Five Data Types

In proposing a query management system Ezigbalike (1988), determined that there are three types of data from the users point of view. The types refer to where the actual spatial data is located and whether it is able to be accessed by outside users. These three types of data are as follows:

I. Type I: Imported/remote data.

This is the type of data that the user does not have physically located at their site. In other words somebody else has collected the data and is allowing you to gain access to that data through a network. The user does not have the right to modify this data in any way. At the server that stores this data it is of type II, Local Public Access data.

II. Type II: Local Public Access data

This is the type of data that the users organisation has collected themselves, and as such, it is stored at their site. This data also has the distinction of being used by other organisations outside the one that “owns” the data. In other words it is data that your store on your server and allow other sites to look at it. Only users at the location that the data is stored at have the ability to modify this data.

III. Type III: Local private data

This is data that the organisation collects and maintains at their site for their private use only. Reasons for making the data invisible to others are that the data is either of no use to anyone else, their site is incapable of acting as a server, or the data is of a confidential nature.

As an extension to these three main types of data two more should be added to cover special cases where an individual may wish to make some spatial data available however they do not have server that is capable of serving the data to the network. The following two types of data allow for data to be stored on another organisation's server and to allow the actual owner of the data to modify that data. To be more precise the two types of data are:

IV. Type IV: Remote Modifiable data.

This type of data is data that is stored at a remote location and the remote location allows users at other locations to firstly access the data, and secondly modify, add or remove data as they see fit. This type of data would have to have a rigorous security system to prevent modification of the database by unauthorised users.

V. Type V: Local Remotely Modifiable data.

This type of data is data that is stored on your server and you allow outsiders to access and modify the data as they see fit. This data is seen as type IV: Remote Modifiable data by users at remote systems that have access to it.

In a virtual database spatial data needs to be classified for the reason that different users will have different access privileges to different datasets. Certain users will be able to access some datasets that other users may not be able to. In order for this to be possible each of the datasets will have to be classified along with each of the individual users. Certain datasets will then only allow users of a certain classification to view/update the dataset.

A user on the network will have or use at least one of the first three types of data, however they may or may not have or use all of them. For example a surveyor would have data of type III (the subdivision that they are developing) and would use data of type I (the DCDB served by Land Victoria). It is unlikely that they would have data of type II as their computer is unlikely to be a server. The DOI on the other hand, would

have all three types as they have their planing scheme which they allow other sites to look at, type II data, they have their own private data like payroll information, type III data, and they also use outside information like the DCDB from Land Victoria, type I data. Finally a person just surfing the web for information would only use data of type I as they have no local data. All these users use are data stored by others.

1.4 The Data Models

During the course of this research five main options have been devised as possible data models for the metadata engine. Each of the options differ in the location of the metadata engine and where the metadata is located. This section of the thesis will detail what each of these options are and what their relative strengths and weaknesses are.

1.4.1 Option 1: Metadata and Metadata Engine on local machine

In terms of coding option 1 is the simplest method to implement a metadata engine with. This model requires that the metadata for the entire network is stored on the user's own computer. Users are able to submit queries to the metadata engine, also located on the users own machine. The metadata engine searches through the metadata and returns a result to the user. The metadata located on the user's computer will contain the location of the actual data, a description of the dataset, and any access restrictions. If the dataset is to be viewed the relevant server will have to contacted using the information in the metadata and the data transferred. Figure 1-2 illustrates this data model.

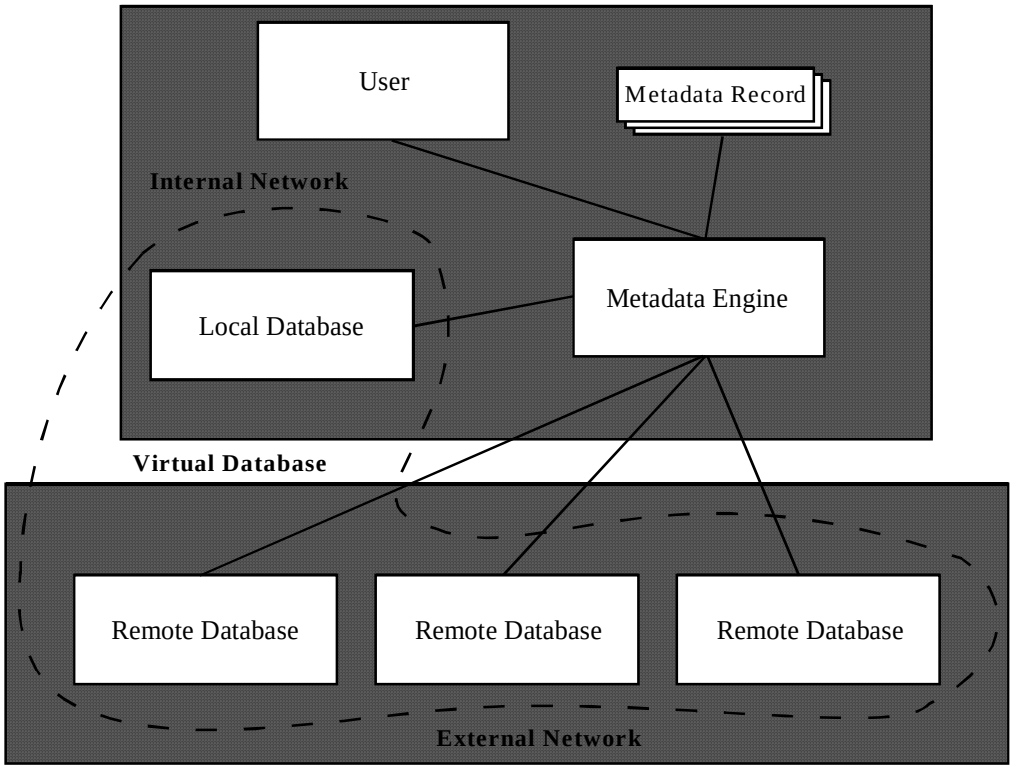

Figure 1-2: Example of Metadata and Metadata Engine located on the users own machine.

The main advantage of this approach is that the actual metadata engine would be relatively simple to design as all the metadata is found locally. The queries to find datasets should be quick as they are done locally, and don't have to be transferred to a remote site.

The main disadvantage of this approach is that there will be multiple copies of the same metadata in existence. Each of the sites that use the system will require a copy of all metadata that they may at some stage use. Apart from leading to a lot of wasted storage space, the actual task of maintaining metadata integrity would be a major hassle. It is likely a central server would have to be established that would contain a copy of all the latest versions of the metadata. When new metadata was produced the custodian would have to submit the metadata to this location. All other users of the system could obtain a copy from this location.

1.4.2 Option 2: Metadata Engine on local machine, Metadata stored on custodians server

In this option the metadata engine is located on the users own computer and contains the network addresses of the remote servers that are known to contain metadata records and their corresponding spatial datasets. When a query is made the metadata engine sends the query to each of the remote servers. The remote servers individually query their metadata to see if any datasets satisfy the users query. If a datasets metadata satisfies the users query the metadata is returned to the metadata engine. The user can then view the metadata to see if the dataset satisfies their needs before they view the dataset itself. Depending on the users query, metadata for many datasets may be returned. Figure 1-3 illustrates this data model.

The advantage of this approach is that only one copy of the metadata needs to exist. Apart from reducing wasted space, it also means that the metadata can reside on the server at the site where it was produced. This means that the data producer has full responsibility for their own metadata and data which can updated, added and removed from the system at anytime with ease. This is much easier than sending out copies of the metadata to each of the remote systems and assures metadata integrity.

Another advantage is that the metadata engine need only have the network addresses to the servers of metadata custodians that contain datasets that the user is likely to need. This will mean that the system will be relatively efficient, as the metadata engine will only search a minimum number of remote servers, not visiting remote servers that are not going to have relevant data.

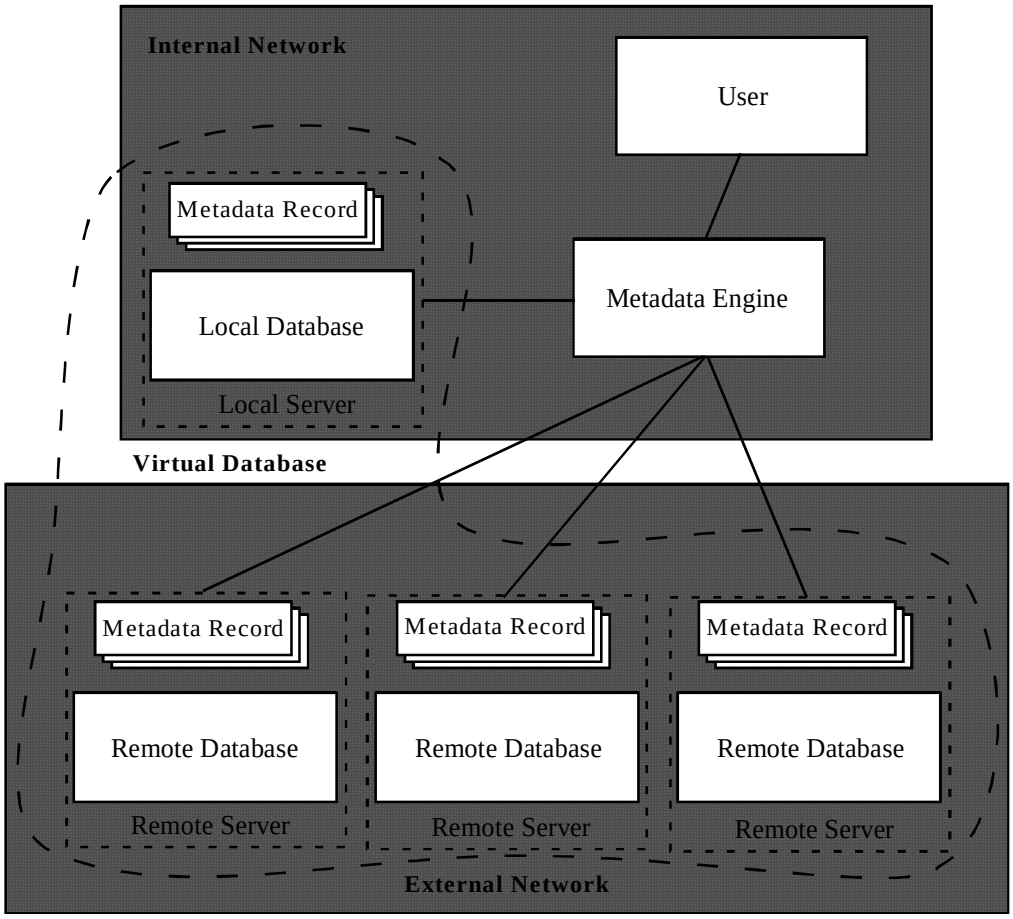

Figure 1-3: Example of metadata engine on local machine whilst the metadata is stored on custodians server.

The disadvantage of this approach is that if the metadata engine does not have the network address of a remote server that contains relevant dataset, then that dataset may as well not exist. This means that the metadata engine has to know the location of all the datasets that they require before a query is made. This is a major disadvantage if a user of the system wishes to search entirety of the ever expanding network to see if there is a certain type of datasets out there that might satisfy their needs.

A possible solution to this problem is to have a central server that has the network addresses of all the servers that are participating in the network. When a new server wishes to join the network the server's administrator will be required to register with this central server. Their network address will be placed on the central server for other

servers to update their server list from. Metadata engines can update their address lists periodically from this central server.

1.4.3 Option 3: Metadata engine and metadata located on one central server

In this option all metadata for the entire network is stored on one large central server along with the metadata engine. When a users submits a query it is passed to the central server. The metadata engine, located on the central server, receives the query and then searches its metadata. Metadata that satisfies the query is then returned to the users system where it can be examined by the user to see if it meets their needs. Figure 1-4 illustrates this data model.

The advantage of this approach is that all the metadata is stored in one location which results in user queries being resolved quickly. As with option 1 there is redundancy of metadata (the owners site has the metadata as well as the central site) however it is reduced greatly when compared to option 1 due to the centralised approach. With all the metadata in one location it should be relatively straight forward for a user of the metadata engine to find out what data is available on the network.

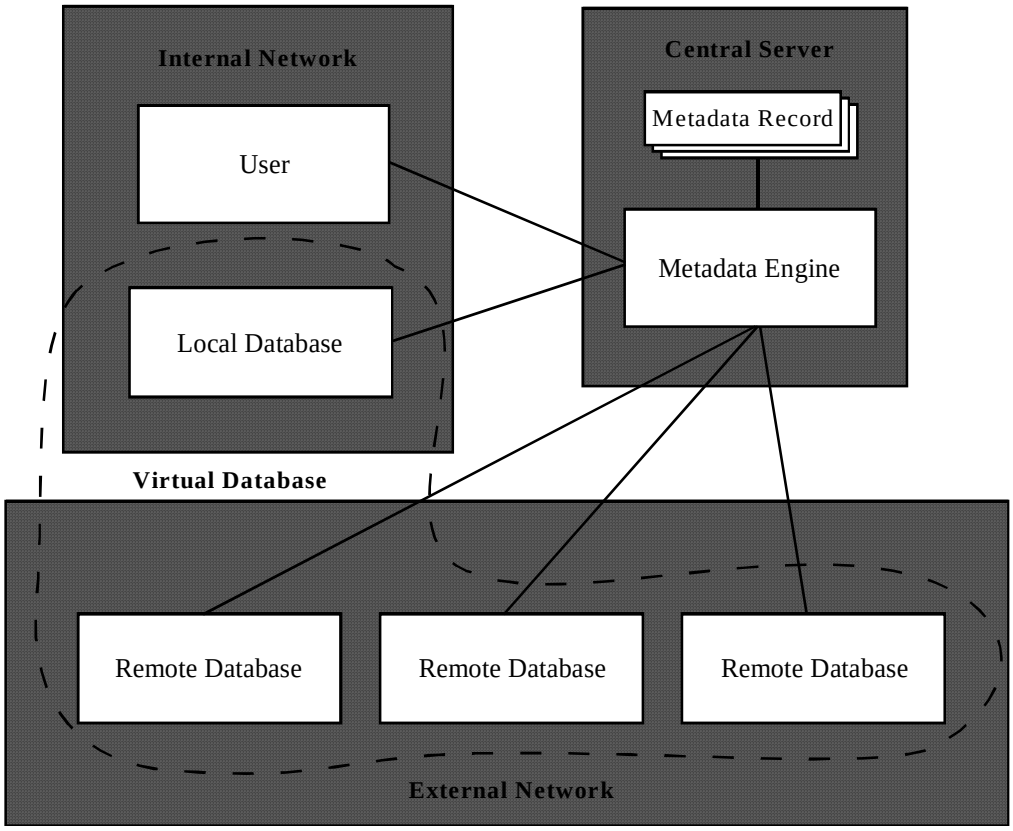

Figure 1-4: Example of metadata and metadata engine located on a single central server.

The main disadvantage of this approach is that each custodian that has data that is available over the network has to supply their metadata to the metadata engine on the central server. The difficulty here is in maintaining the metadata's integrity. To make sure that the central server is kept accurate and up to date, the addition, update and removal of metadata has to be dealt with. One way to deal with them is to have the custodians that change metadata locally to automatically pass on the changes to the metadata engine on the central server where they are subsequently added to the system. Another approach is for the metadata engine at the central server to periodically check each of the remote servers that serve data to the network. The metadata engine could compare its metadata records to the metadata records located at each of the remote servers to see if they matched. If a discrepancy exists the central servers catalogues are modified to match those of the remote servers. All new servers

would have to supply the first copies of their metadata so that the central server would know where to look to get the updated versions.

Another disadvantage of this approach is that the server would have to be large and powerful to cope with all the metadata and queries that would be potentially coming through the metadata engine on the central server. The amount of traffic that the server may have to deal with could easily exceed the server's capability and hence slow the whole network down.

1.4.4 Option 4: Metadata engine located on an independent central server, metadata located on custodians servers

In this option there is once again a central server, however in this case it only the metadata engine and the network addresses of all the custodians servers, it doesn't store the metadata. When a user's query is received by the metadata engine at the central server it is passed to each of the custodians servers at the network addresses stored on the metadata engine. Each of the custodians servers search their metadata for datasets that satisfy the user's query. Metadata records corresponding to all datasets that satisfy the query are then passed back to the central server. The metadata engine combines the results and passes them back to the user's computer where they can be examined by the user to determine their relevance. It is a similar approach to that of option 2 except there is only one metadata engine located on a central server. Figure 1-5 illustrates this data model.

The advantage of this option is that the metadata are maintained by the data custodian. This is an advantage as they can change, add to, and remove it from the system simply by modifying their own metadata. Metadata integrity is easily maintained and there is no duplication of metadata.

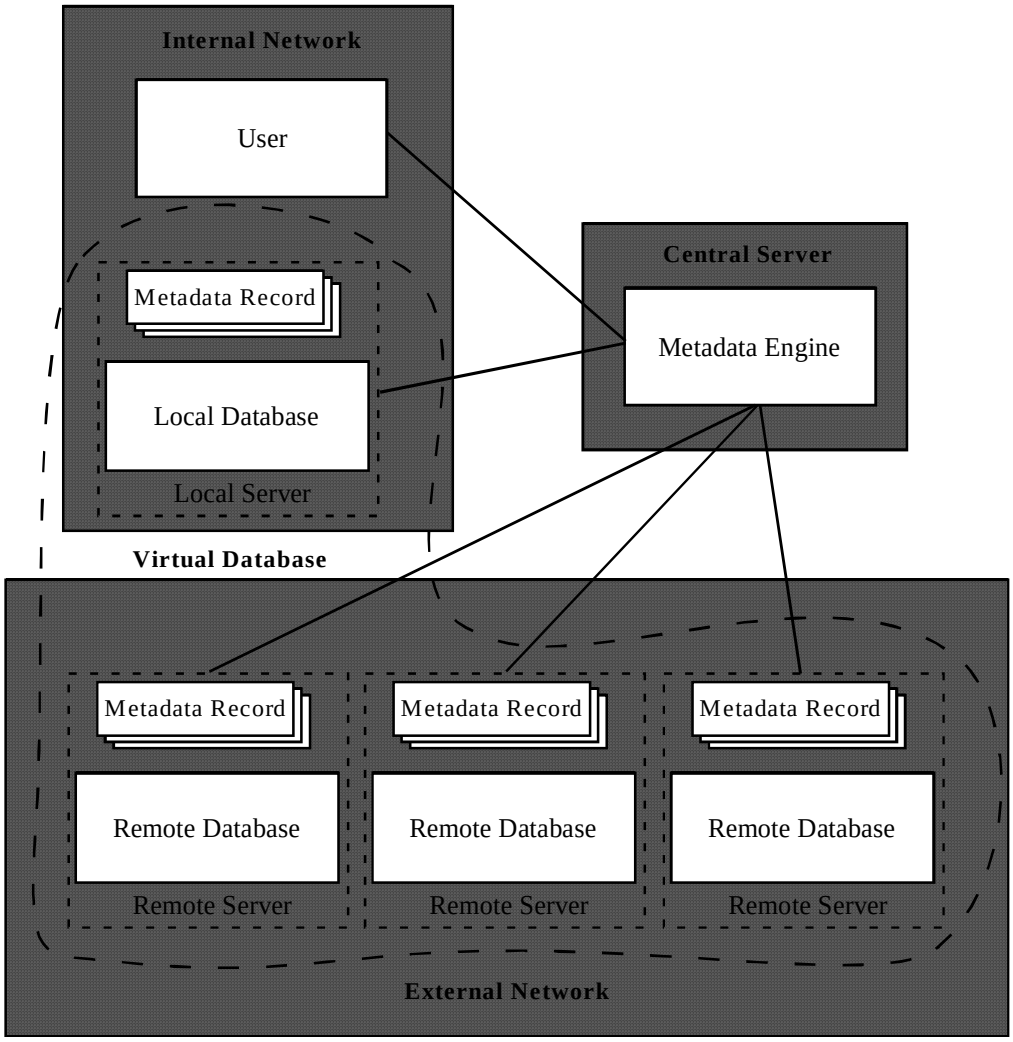

Figure 1-5: Example of a central metadata engine, and distributed metadata approach.

The main disadvantage of this option is that obtaining the results of the users query may take time as each of the servers have to be queried. The actual travel time to get from the user's computer to the metadata engine, and then from the metadata engine to the individual remote servers and back is where most of the time would be lost. One process that could be conducted in both this and the second option is to send the users query from the metadata engine to the individual remote servers in parallel. In parallel means that the query should be sent out to the first of the remote servers and then the second of the remote servers and so on without waiting for the results to come back from each of the servers before the next is sent.

1.4.5 Option 5: Combination of options 3 and 4.

In both options 2 and 4 there is an assumption made that the data custodians have, or want, the capability to that serve the metadata and data to other users on the network. In Option 5 there is once again a central server that holds the metadata engine and network addresses of all the participating custodians that have metadata and data stored on their servers, as in Option 4. Unlike option 4 this solution also has the capability to have metadata stored at the central server. This essentially means that this solution is the combination of options 3 and 4. When a custodian, that has it's metadata stored on the central server, modifies their metadata they will have to pass on the changes to the metadata engine on the central server to maintain metadata integrity. Figure 1-6 illustrates this data model.

The advantage of this solution is that it allows all data custodians that have valuable metadata to participate in the network no matter what their technical capabilities are. The disadvantage of this approach is the added complexity that the combination of the 3rd and 4th options creates. Being able to handle both situations means extra code to add this functionality.

This solution has the combined advantages of both options 3 and 4. It however also has the combined disadvantages of options 3 and 4. The extra advantage of more participation in the network is the added advantage that makes this option a valuable one. The larger the network, the more participants, and the more spatial data on the network the better. The more that spatial data is used throughout the community, the more opportunities that will be created, and more money saved by the reduction of duplicate efforts.

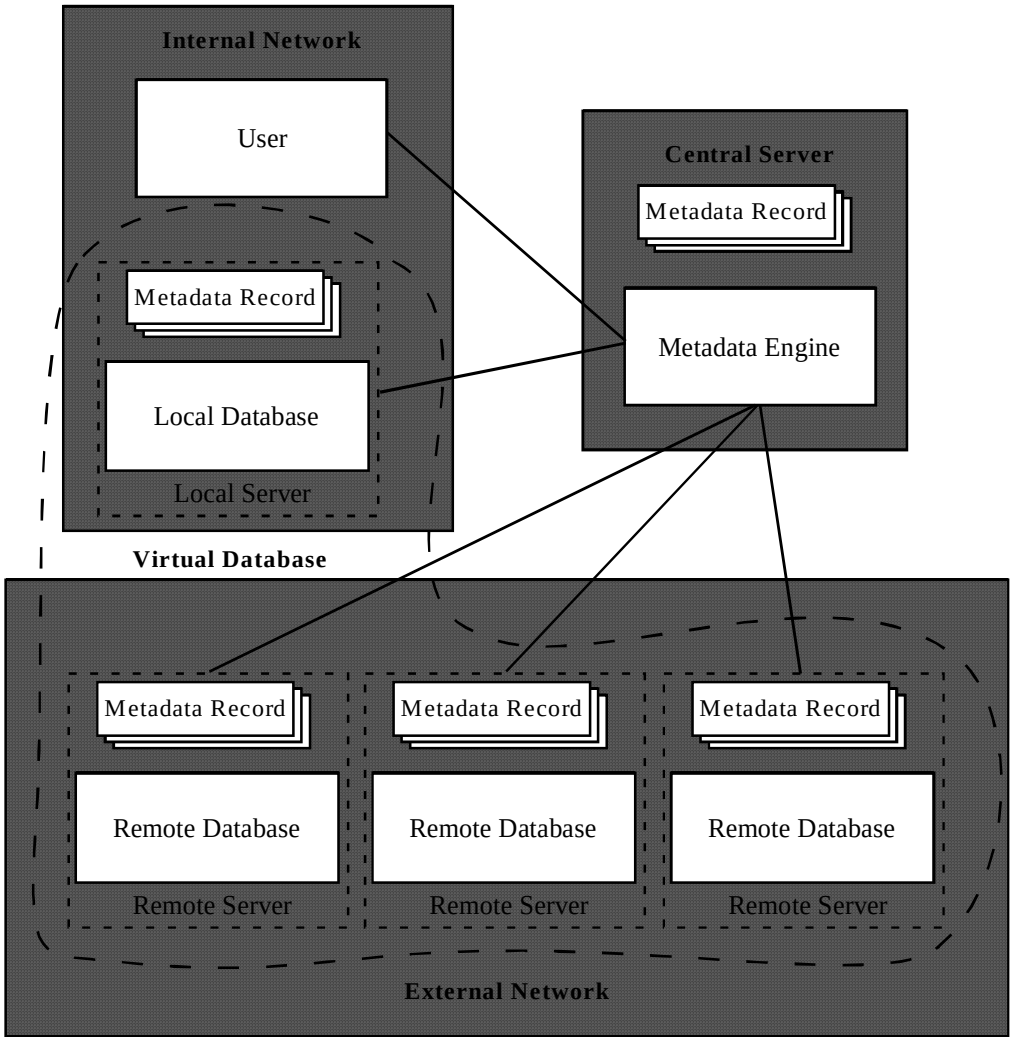

Figure 1-6: Example of a model that combines both options 3 and 4.

1.5 Live Connection to the servers

One of the key concepts that must be emphasised is the fact that when the results of the metadata query have been returned to the user, the user has the option to do one of two things:

1) **View the metadata that has been returned**. When the metadata is returned the name of the dataset is displayed on the user's screen. If more detail than this needs to be viewed to make a decision on what if any of the data found will be useful the whole of the metadata record for the dataset can be viewed.

2) **View the spatial data on the screen**. From the list of datasets that will appear on the screen as a result of the metadata search, the user can select any number of these and have them all displayed on the screen at once. For example the user may submit a query that will find all the datasets that cover the Parkville area. From the list of results from the search the user may wish to display the SDMB as the basemap and on top of that display the planning zones in effect, public transport routes, as well as where all the public telephones are. Another option might be for a contractor to view or construct a map that will show where all the underground gas pipes, power cables, and telephone lines are. This could be a viable alternative for the "Dial before you Dig" service.

   The likely method that will be used for the displaying of the data will be to convert the data from the local format into an intermediate format at the servers end which will be displayed on a web based browser. This is where the work of the OpenGIS Consortium could be extremely invaluable. The work that they have undertaken should make all the datasets of interest interoperable.

   Only the amount of data that is needed to be displayed on the screen will be sent to the browser to conserve bandwidth. A connection with the server will be maintained while the data is being viewed. This enables the user to zoom in on the data, zoom out, and move the focal point. If new data is to be displayed as a result of the user moving their focus the screen will be updated by data being resent from the server/s. It could be said that a live connection exists between the users machine and the servers where the data resides.

1.6 Gaining Access to the Remote Databases

This section discusses the use of the ANZLIC metadata guidelines to achieve the objectives of this thesis, which is to not only find the metadata record for a dataset, but also to display it. As far as this thesis is concerned a major shortcoming of the guidelines is that there is no metadata, at the page 0 level, that tells the user how to access the data across a network. Obviously this is a problem as a search for a dataset

using ANZLIC compliant metadata will not return the information required to access the dataset.

There are two possible ways to solve this problem (other than to change the page 0 metadata):

- Have all the metadata required to access the dataset over the network stored in the page 1,2,3 etc pages for each metadata record.
- At each server have a completely separate metadata file which has the extra metadata required to access each dataset over the network stored in it.

1.6.1 Pages Approach

In the pages approach the metadata that is required by the metadata engine to access the dataset (the access metadata) across a network is stored in the subsequent pages of the metadata record. Preferably this metadata would all be located at the same page level in order to make its retrieval easier.

This approach would work by firstly searching the page 0 metadata located at each server in the system. All metadata records that satisfied the users query would be passed back to the metadata engine and subsequently the user's machine in the form of a hyperlinked address. Along with the page 0 hyperlinked address, hyperlinked addresses to the subsequent pages that contained the metadata that enables a user to access a dataset could be returned. Alternatively this access metadata could be extracted from these pages and encoded onto the return string with the page 0 hyperlinked address.

This approach has the advantage that it keeps all the metadata within the ANZLIC guidelines, and hence is relatively straight forward to understand. For a commercially developed system that has had the time and resources to invest, this would probably be the best approach to take in order to make the system as technically easy to maintain.

There is one main disadvantage with this approach which is the fact that at this stage only a format has been settled upon for page 0 metadata. There is no agreed format for metadata that will be located at levels below this. It is the responsibility of the data custodians to formulate these. It will be very difficult to develop a system to use the pages approach if there is no standard format for different page levels. Finding the required metadata within the pages will be extremely difficult if they are all stored in different places and called different names.

1.6.2 Separate File Approach

The separate file approach is a concept where each of the metadata records that exist on a server also has a corresponding entry in a separate metadata file. The entry that is in the separate metadata file is all the access metadata that is needed to access the corresponding dataset across the network.

As with the pages approach the metadata engine would work firstly by searching the ANZLIC compliant metadata records for datasets that satisfy the users requests. Hyperlinked addresses for each of the metadata records that satisfy the request would be encoded onto the return string, along with the corresponding access metadata entry from the separate metadata file. This return string would then be decoded at the user's end to gain the required metadata.

The main advantage of this approach is that it is relatively easy to code. There will be only one file containing access metadata of a standard format for accessing datasets on any one server. It could have a common name that could be hard coded into the source code. A more elegant solution, than hard coding the name and location of the file into the source code, would be to include a line in an initialization file that tells the system where the metadata file is located.

Another advantage is that the metadata engine developer establishes the standards for the separate metadata file. The file will be in the same format on every server, which means that there should not be any problems finding the appropriate access metadata

within the file. Compare this with the pages approach where the pages below page 0 have no standard format and hence are extremely difficult to navigate through to find the required access metadata.

The main disadvantage of this system is that the separate file is another file that has to be kept up to date for the system to work effectively. Whenever a dataset is added to a server two activities need to be undertaken by the system maintainer. Firstly they need to add an ANZLIC compliant metadata record for the dataset. Secondly they need to modify the separate metadata file to contain the access metadata for that new dataset. This approach tends to add complications to the system maintainer's role.

1.7 Chapter Summary

Whereas the previous two chapters outlined related concepts, projects and policies, this chapter outlined the concepts that are involved in the development of a metadata engine that is capable of allowing a virtual database to be developed. The development of metadata engines is important as they allow for decisions that are to be made using spatial data to incorporate as much spatial data as possible. It is likely to be more economically to use small pieces, of many datasets, to make a decision than it would be to have copies of the entire datasets in a stand-alone GIS.

In order for a metadata engine to allow for users with different access privileges to access datasets differently the view that the user has of each of the datasets can be classified into one of five types:

1) Imported data;
2) Local public access data;
3) Local private data;
4) Remote modifiable data; and
5) Local remotely modifiable data.

Different users will have differing views of the same datasets depending on what type of user they are. Certain datasets may be remote modifiable for some users and yet they are only imported data for others.

When implementing a metadata engine there are five basic models that could be used to develop the system. Each of the models has its own advantages and disadvantages and each of them varies in difficulty as far as implementation is concerned. The five basic models differ in where the metadata and the metadata engine are physically located. The models are:

1) Metadata and metadata engine are located on the users own machine;
2) Metadata is located on the data custodians machine and the metadata engine is located on the users own machine;
3) Metadata and the metadata engine are located on one central server;
4) Metadata is located on the data custodians machine and the metadata engine is located on an independent central server; and
5) A combination of the 3rd and 4th options.

Any of these options could be used to implement the metadata engine. It is simply a matter of which of them best suits the needs of the user group that is to use the engine.

One of the key components of the metadata engine is the fact that it has a live connection to the datasets. This means that when the results of a search on the system's metadata records is returned, the user has the ability to concurrently view any number of the datasets, corresponding to the returned metadata records. This is unlike the concept of spatial data directories and clearinghouses where only the metadata records can be viewed, or if you are lucky you may be able to download the individual datasets one at a time in their entirety.

The chapter also outlines two methods for storing access metadata that is required for the remote user to gain access to the individual datasets. This access metadata is stored at the same location as the ANZLIC compliant metadata records that they belong to. The two options have their own advantages and disadvantages and are:

1) A pages approach, where the access metadata is simply located at page levels 1,2 etc in the ANZLIC metadata record; and

2) A separate file approach, where all the access metadata for the server is located in the one file in a standard format.

The first approach is probably more logical and would be easier to maintain, however the second approach is much easier to code and implement.