wrong address to try to shadow the entry, and never find the new address. (Alternatively you could use a ts_bridge to make it look as if the old address still works.) You can check the master EDB has updated by making a DAP bind to the DSA managing the EDB and reading the address it has got. If it is out of date, you should wait for it to be updated; this usually takes place every six hours. To summarise, the sequence of events is

1. Change the address attribute using a DAP modify operation.

2. Wait for the DSA with the master EDB to read the entry.

3. Restart the DSA on the new address.

14.5 Connection to Other DSAs

All QUIPU DSAs should be connected. This section describes how you should configure your DSA to connect to other DSAs in order to read the data not held locally. Section 14.5.1 describes how to make sure that other DSAs can access your DSA.

A dynamic approach is used to bootstrap a new DSA. The "global master" DSA is administered at the University of London Computer Center[5] (ULCC), and other DSAs should be configured in relation to this one.

Every DSA needs to know the distinguished name and presentation address of one or more DSAs nearer to, or actually holding, the root EDB. This parameter is supplied in the *tailor* file (see Section 14.3) under the name `parent`. This information is used by the QUIPU DSA when it cannot find a DSA's entry in the local database, or can find no pointers in the local database to the whereabouts of the data. If you hold a slave copy of the root EDB, then your parent should be a DSA that is "closer" to the master copy of the root EDB, and probably the master itself.

The example databases are set up with default parents (defined in the file *quipu-db/organisation/quiputailor*). To see if your DSA can contact the "parent", connect to the local test (root holding) DSA using DISH, and type

```
list @
```

[5]The DSA has moved from University College London.

This will list the locally held root. Now try

```
list @ -dontusecopy
```

This will try to connect to the parent DSA.

Care should be taken in choosing the parent DSA. If all DSAs have the same parent DSA then that DSA may become overloaded. Typically each site will have several DSAs. One of these should be the default parent for all the other DSAs, with only one DSA having a default parent outside the site.

When "walking down" the DIT, QUIPU needs access to information to tell it on which DSA the next level of the DIT is stored. To do this each nonleaf entry *must* have a `masterDSA` or `slaveDSA` attribute. The value of the attribute is the distinguished name of the DSA holding the next level of the DIT (it may be your own DSA name, if you hold the next level as well). If there is no such attribute, QUIPU assumes the entry is a leaf. If the information required is not held in the local DSA then QUIPU chains the request to the named DSA or returns a referral — depending upon the mode of operation. (The named DSAs entry is read to establish the address. This may mean a connection to another DSA in order to read the entry.)

Once you have started your DSA, you should try to connect to other DSAs using the DISH program (this connects to the local DSA, which will chain requests to other DSAs). Try listing the children of organisations other than yourself, to find which organisations are connected, try listing your countrys' children.

14.5.1 Connection to the Global Directory

To enable the global directory to connect to you, you must contact the manager of the DSA immediately above the node you want to added under. Supply them with the entry for your DSA, and the top level entry of the subtree you want to hold. For example, to add the organisation o=foobar below c=US, contact the manager of the DSA holding c=US as a master, giving them the entry for c=US@o=foobar and c=US@cn=foobar_dsa, assuming c=US@cn=foobar_dsa is the name of your DSA (see Section 14.2.2 on naming a DSA).

By convention, to find out who administers the master EDB for a particular node, run the DISH program and retrieve the `manager` attribute from the entry for the DSA holding that node.

You can then find a mail address by looking that person up in the directory.

The following example shows you how to get the mail address of the manager of the c=GB subtree.

```
% dish
Welcome to Dish (DIrectory SHell)
Dish -> showentry @c=GB -type masterDSA
masterDSA  - cn = Giant Tortoise
Dish -> showentry "@cn=Giant Tortoise" -type manager -edb
manager= c=GB@o=X-Tel Services Ltd@cn=Camayoc
manager= o=Cosine@ou=Paradise@cn=Camayoc
Dish -> moveto "@c=GB@o=X-Tel Services Ltd@cn=Camayoc"
Dish -> showentry -edb
cn= Camayoc
roleOccupant= c=GB@o=X-Tel Services Ltd@cn=Colin Robbins
objectClass= organizationalRole & quipuObject
Dish -> moveto "@c=GB@o=X-Tel Services Ltd@cn=Colin Robbins"
Dish -> showentry -type rfc822mailBox
rfc822Mailbox         - C.Robbins@xtel.co.uk
Dish -> quit
%
```

If you want to add data to the root node of the "global tree". The new entries should be sent to *QUIPU-support* at the address given in Section 2.6.

When you think you are connected to the "global DIT", you should test that you are. To do this use DISH to connect to a DSA higher than you in the DIT (using the `-call` flag as described in Section 13.2); then see if you can navigate to your own part of the DIT, and look at your data: if you can, then the DSA is connected.

14.6 Connecting to a Non-QUIPU DSA

QUIPU has the concept of mastering all entries in one EDB file built into its design. Other implementations may not share this model. The entry represented by the non-QUIPU DSA can be "spot shadowed". For example, if you wanted to add "o=foobar" to the DIT and this was mastered by a non-QUIPU DSA, you would add an entry of the form

```
o=foobar
```

```
objectClass= ExternalNonLeafObject & organization
subordinateReference= cn=The DSA # Internet=123.456.1.2+17003
```

where cn=The DSA is the DN of the DSA, and Internet=... is the presentation address of the DSA. Instead of a subordinate reference, a cross-reference or nonspecific subordinate reference can be given using the `crossReference` or `nonSpecificSubordinateReference` attributes respectively.

Although this is entry is in a master EDB file, QUIPU recognises that is may not actually hold the authorative master, and contacts the referenced DSA as required.

14.7 Adding More Data

Having got a DSA started, and connected to the global DIT, you should start to add lots of data. There are many sources of such data, and with just a relatively small amount of effort this data can be added to the directory.

There are two ways such data can be added. First of all you can use one of the DUA programs to bind to the DSA as the DSA manager, and send data to the directory via the *add* and *modify* operations (see Sections 4.1.5 and 4.1.8). This is probably the best way to add relatively small amounts of data or make minor changes to the data.

To add a large amount of data (i.e., during the initial creation of a large database), it is probably easiest to write a shell script using UNIX tools such as `awk` and `grep` to create the EDB files directly.

When adding data for users it is advisable to allocate a userPassword, and a suitable ACL to protect this password. Chapter 11 describes how to do this.

14.7.1 More on Object Classes

The only object class discussed so far is the quipuDSA object class. When you start to add data, you will probably want to add information about people, sub-divisions of your organisation, and other application entities. This section introduces some of the more important object classes, and the attributes they may contain. In many cases, only the attribute type is specified, for details of typical values and the value syntax you should read Chapter 10.

As already described, every entry in the DIT must belong to the object class `top`, which means every entry must have an `objectClass` attribute. Also, every entry in the QUIPU DIT should belong to either the `quipuNonLeafObject` or `quipuObject`.

The following part of this chapter describes some of the basic object classes and the attributes implied.

Person

This is a base object class used to represent a person.

There are two mandatory attributes:

1. `commonName:` which gives a (potentially ambiguous) name for the person. The value of this attribute is a string usually containing the person's first and last names; e.g.,

   ```
   Marshall Rose
   ```

   This attribute is usually multivalued, containing variations on the first, middle, and last names; e.g.,

   ```
   Colin Robbins
   Colin John Robbins
   Colin J. Robbins
   ```

   Generally this attribute will supply the distinguished attribute of the entry.

2. `surName:` which gives the person's last name.

The optional attributes are:

`userPassword`	`seeAlso`
`telephoneNumber`	`description`

OrganizationalPerson

This is a subclass of the `person` person object class and introduces the following optional attributes:

```
preferredDeliveryMethod        destinationIndicator
registeredAddress              internationaliSDNNumber
x121Address                    facsimileTelephoneNumber
teletexTerminalIdentifier      telexNumber
physicalDeliveryOfficeName     postOfficeBox
postalCode                     postalAddress
title                          organizationalUnitName
streetAddress                  stateOrProvinceName
locality
```

ThornPerson

Like `OrganizationalPerson`, this is also a subclass of `person` and introduces the following optional attributes:

```
homePostalAddress              lastModifiedBy
lastModifiedTime               secretary
homePhone                      userClass
photo                          roomNumber
favouriteDrink                 info
rfc822Mailbox                  textEncodedORaddress
userid
```

Two example `ThornPerson` entries are given in Figure 14.1 on page 132.

OrganizationalRole

Entries of this class are used to represent a position or role within an organisation.

There is one mandatory attribute:

`commonName:` gives the name of the role. The value of this attribute is a string; e.g.:

```
PostMaster
```

There are many optional attributes including:

`roleOccupant:` the distinguished name of the person who fulfills the role, e.g.:

```
c=US@o=X-Tel Services Ltd@cn=Peter Cowen
```

The other optional attributes are:

```
seeAlso                         preferredDeliveryMethod
destinationIndicator            registeredAddress
internationaliSDNNumber         x121Address
facsimileTelephoneNumber        teletexTerminalIdentifier
telexNumber                     telephoneNumber
locality                        postOfficeBox
postalCode                      postalAddress
description                     organizationalUnitName
streetAddress                   stateOrProvinceName
physicalDeliveryOfficeName
```

Alias

Objects of this class represent an alias to some other entry in the DIT. It is generally used when an entity belongs in one or more subtrees of the DIT, and is used to "point" one entry at the other.

There are two mandatory attributes:

`commonName:` gives the name of the alias. The value of this attribute is a string; e.g.:

```
Colin Robbins
```

`aliasedObjectName:` a pointer to another object in the directory; e.g.:

```
c=GB@o=X-Tel Services Ltd@cn=Colin Robbins
```

There are no optional attributes for this object class. An example of an Alias entry is given below:

```
cn= QUIPU-support
aliasedObjectName= c=GB@o=University College London
   @ou=Computer Science@cn=Incads
objectClass= quipuObject & alias & top
```

OrganizationalUnit

The `OrganisationalUnit` object class is used to represent a unit within organisation. There is one mandatory attribute

`organizationalUnitName:` which gives the name of the organisational unit. The value of this attribute is a string; e.g.:

```
Research and Development
```

The optional attributes are:

`userPassword`	`seeAlso`
`preferredDeliveryMethod`	`destinationIndicator`
`registeredAddress`	`internationaliSDNNumber`
`x121Address`	`facsimileTelephoneNumber`
`teletexTerminalIdentifier`	`telexNumber`
`telephoneNumber`	`physicalDeliveryOfficeName`
`postOfficeBox`	`postalCode`
`postalAddress`	`businessCategory`
`searchGuide`	`description`
`streetAddress`	`stateOrProvinceName`
`locality`	

Organisation

Objects of this class represent a top-level organisational entity, such as a corporation, university, government entity, and so on.

There is one mandatory attribute:

`organisationName:` which gives the name of the organisation. The value of this attribute is a string; e.g.:

```
Performance Systems International
```

The optional attributes are:

`userPassword`	`seeAlso`
`preferredDeliveryMethod`	`destinationIndicator`
`registeredAddress`	`internationaliSDNNumber`
`x121Address`	`facsimileTelephoneNumber`

```
teletexTerminalIdentifier      telexNumber
telephoneNumber                physicalDeliveryOfficeName
postOfficeBox                  postalCode
postalAddress                  businessCategory
searchGuide                    description
streetAddress                  stateOrProvinceName
locality
```

domainRelatedObject

If an object has some relationship to the Internet Domain Name System (DNS), then this can be represented in the DIT using this object class.

This class has one mandatory attribute:

`associatedDomain:` identifies the domain which corresponds to this object. The value is a domain string; e.g.:

```
psi.com
xtel.co.uk
```

14.7.2 Schemas

Directories should provide a very flexible tool which enable any information to be stored. There is a danger that schemas, as specified in the OSI directory, will lead to Procrustean directory implementations which impose unreasonable restrictions. The QUIPU directory does not, per se, place restrictions on what can be placed in a DSA. It does, however, give control so that managers may control what is stored in the directory.

The first aspect of structure is with respect to attributes which may be present in an entry. A QUIPU DSA will allow an entry to belong to one or more object classes which are known to the DSA (stored in a table). An entry will typically have a small number of object classes (e.g., TOP (implicit) + Person + Organisational Person + QUIPU Object). The DSA will maintain a table of mandatory and optional attributes for each object class supported. This will follow the guidelines of the standard or specification identifying the object class in question. From this information, the DSA can determine the permitted and mandatory attributes for a given entry by calculating the union of all the object classes of that entry. Free extension (i.e., the ability to

store any attribute) was rejected, as there does not appear to be a reasonable mechanism to manage this. However, it is straightforward for managers to create new object classes as desired.

```
TreeStructureSyntax ::= SET {
    mandatoryObjectClasses [1] SET OF OBJECT IDENTIFIER,
    optionalObjectClasses [2] SET OF OBJECT IDENTIFIER OPTIONAL,
    permittedRDNs [3] SET OF SET OF AttributeType }

TreeStructure ::= ATTRIBUTE
    WITH ATTRIBUTE-SYNTAX TreeStructureSyntax
    MULTI VALUE
```

Figure 14.3: Schema Definition

It is important to allow management control of what is permitted at a given level. Therefore a `treeStructure` attribute may be created. This attribute is defined in Figure 14.3, with the string syntax discussed in Section 10.2.2. This specifies for the level below, what types of object are permitted. Each attribute value identifies a class of object which can exist at the level below, and defines a set of mandatory and optional object classes. This can be considered as defining a (private) object class implied by the combination of these classes.

For each type of object, the attribute types permitted in the RDN are also listed. This is not checked in the current version of QUIPU. The directory knows about the treeStructure attribute and will ensure consistency. When creating an entry, the DSA must check that it conforms to the treeStructure attribute of the parent entry. When removing information from a treeStructure attribute, the directory will check that all of the children conform to the modified attribute.

14.7.3 Photograph Attributes

The data that a DSA can hold does not have to be limited to data that can be represented by printable strings. Any attribute a QUIPU DSA does not know a string syntax for, will be held as a block of ASN.1. One such attribute