Chapter 1

Introduction

Overview

Novell Directory Services™ (NDS™), an object-oriented, distributed, hierarchical database, is based on a set of rules defining the particular types of objects that can exist in a Directory tree. This set of rules, governing the structure of the Directory tree, is called the *schema*.

The schema defines the rules for adding and managing objects and object attributes in the Directory. These rules are specified through a data dictionary that provides a standard set of data types from which objects can be created. Every object in the Directory belongs to an object class that specifies what attributes can be associated with the object. All attributes are based on a set of standard attribute types, which in turn, are based on standard attribute syntaxes.

The Directory schema controls not only the structure of individual objects, but also the relationship among objects in the Directory tree. In controlling this relationship the schema specifies subordination among object classes. That is, every object definition specifies a group of object classes from which subordinate objects can be formed.

This chapter presents the general concepts, components, and structure of the Directory schema. The subsequent chapters present the specific rules that govern the structure of the Directory tree.

Schema Components

The set of rules that controls the creation of a particular object type is called an *Object Class.* Each object class is defined in terms of *attributes* or *properties*. An attribute is a specific piece of information that can exist for an object. Attributes, in turn, are defined in terms of a base set of data types called *Attribute Syntaxes.* The attribute syntaxes define the primary data types for values stored in the Directory. The schema thus dictates the requirements, limits, and relationships of objects and attributes of objects that can be created and used in the Directory tree.

Base Schema

This schema specification describes the object classes, the attributes, and attribute syntaxes that are installed when NDS is installed. The schema of every system running NDS will have these components. They cannot be deleted or modified.

Extended Schema

The schema is extensible in that you can define new object classes and attributes of objects. However, these classes and attributes must be defined in terms of the existing data types. That is, defining new attribute syntaxes is not allowed.

Novell, as it has enabled applications to use NDS, has extended the schema. Products such as LDAP, NAL, and Border Manager add object classes and attributes to the schema.

The Schema Manager utility, which ships with NetWare, enables network administrators to manage the schema in their NDS trees. It includes operations for browsing, viewing, modifying, printing, comparing, and diagnosing the object classes and attributes.This application can also be used by application developers to create new object classes and attributes.

General Schema Structure

The schema definitions have a relationship with one another that could be referred to as a structure. This schema structure is established through the following relationships:

- Attribute syntaxes to attribute types
- Attribute types to object classes
- Object classes to object structure rules

Attribute syntaxes define the standard data types for the values within specified attribute types stored in the Directory. An attribute syntax consists of a single data type (such as boolean or time) for which syntax matching rules and qualifiers have been specified. Matching rules indicate the characteristics that are significant when comparing two values of the same syntax. There are three primary matching rules: equality, substrings, and ordering. The qualifiers or directives that are defined for comparison include ignoring case, dashes, and spaces. Other qualifiers allow only digits or only printable characters.

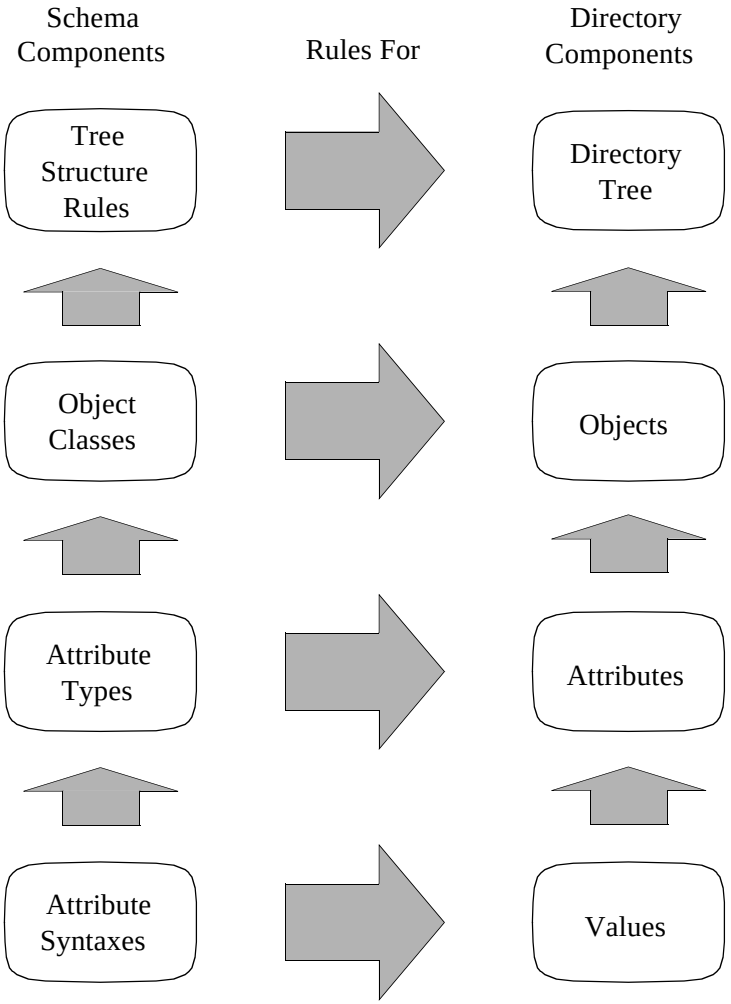
Attribute types are constructed from the attribute syntaxes to define the categories of information that can be associated with objects. The syntax must be selected from the set of predefined attribute syntaxes. The constraints can specify whether the attribute type is single- or multi-valued, as well as any range or size limits for attribute values. In addition, attribute types can be constrained to synchronize immediately. When a synchronize-immediate attribute value is added to an object, NDS immediately attempts to update all replicas in the Directory that include that object rather than waiting for the next scheduled synchronization.

Object classes are constructed from the attribute types, and these classes define the rules for creating Directory objects. Object classes have structure rules, the constraints that control the construction and hierarchy of the Directory tree. Naming rules for each object class specify the attribute(s) used in the Relative Distinguished Name (RDN) for objects of that class. The values of the components assigned to an object class vary and expand to include other class components, depending on the class's superclass.

Schema and Directory Components

The schema components and the Directory components are interrelated. Figure 1.1 illustrates this relationship.

Figure 1.1
Relationship of Schema and Directory Components



The vertical arrows indicate the structure dependencies from the basic building blocks up to the Directory schema and the Directory tree, respectively. The horizontal arrows denote the schema rules that apply to the respective Directory components.

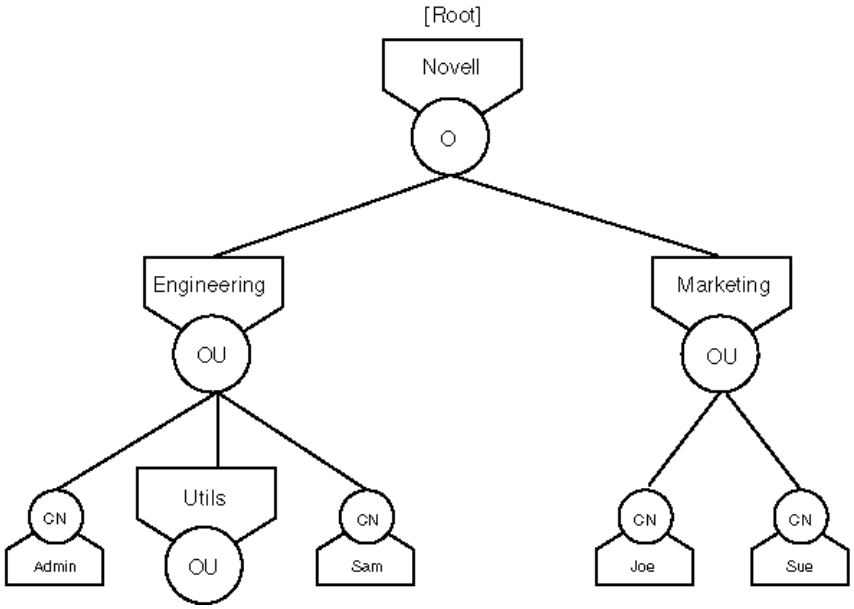
The attribute syntaxes define the primary data types for values stored in the Directory. Attribute types are defined from the attribute syntaxes and define the possible attributes an object can have. Object classes are defined using a subset of the possible attributes and determine the types of objects that can be in the Directory tree. The tree structure rules define how the object classes can be organized and nested in the tree and therefore determine the tree's structure.

Schema and Hierarchical Structure

Each object in the Directory tree belongs to a particular object class defined in the schema. Each object must follow the rules pertaining to members of the object class to which it belongs. Objects, therefore, are connected in a hierarchical tree structure. Objects that can contain other objects are called *container objects.* Container objects are the building blocks of the Directory tree. Objects that cannot contain other objects are known as *non-container* or *leaf objects*. Leaf objects comprise the actual network resources that perform some function in the Directory tree.

Figure 1.2 illustrates a simple NDS tree, showing the logical, hierarchical structure that the tree structure rules create. This NDS tree has four container objects (Novell, Engineering, Marketing, and Utils) and four leaf objects (Admin, Sam, Joe, and Sue).

Figure 1.2
NDS Tree



O (Organization) and OU (Organizational Unit) are class objects that can contain leaf objects. CN stands for common name and is used to designate the leaf objects in this tree.