ระบบไดเรกทอรีสำหรับเอเจนต์ความปลอดภัยแบบกระจาย

DIRECTORY SYSTEM FOR DISTRIBUTED SECURITY AGENT

นายถนกศักดิ์ รัชปัตย์

นายเจษฎา คำพุทธ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2544

ระบบไดเรกทอรีสำหรับเอเจนต์ความปลอดภัยแบบกระจาย

DIRECTORY SYSTEM FOR DISTRIBUTED SECURITY AGENT

โดย

นายกนกศักดิ์ รัชปัตย์

นายเจษฎา คำพุทธ

อาจารย์ที่ปรึกษา

อาจารย์ธนา หงษ์สุวรรณ

อาจารย์อัครเดช วัชระภูพงษ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2544

ปริญญานิพนธ์ปีการศึกษา 2544

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ระบบไดเรกทอรีสำหรับเอเจนต์ความปลอดภัยแบบกระจาย

DIRECTORY SYSTEM FOR DISTRIBUTED SECURITY AGENT

ผู้จัดทำ

1. นายกนกศักดิ์	รัชปัตย์	รหัสนักศึกษา	42015294
2. นายเจษฎา	คำพุทธ	รหัสนักศึกษา	42015297

อาจารย์ที่ปรึกษา

(นายธนา หงษ์สุวรรณ)

อาจารย์ที่ปรึกษา

(นายอัครเดช วัชระภูพงษ์)

ระบบไดเรกทอรีสำหรับเอเจนต์ความปลอดภัยแบบกระจาย

นาย กนกศักดิ์ รัชปัตย์ 42015294
นาย เจษฎา คำพุทธ 42015297
อาจารย์ ธนา หงษ์สุวรรณ อาจารย์ที่ปรึกษา
อาจารย์ อัครเดช วัชระภูพงษ์ อาจารย์ที่ปรึกษา
ปีการศึกษา 2544

บทคัดย่อ

งานเฝ้าดูสภาพการทำงานและความปลอดภัยของเครือข่ายเป็นสิ่งจำเป็นสำหรับผู้ดูแลระบบเครือข่าย ซึ่งในปัจจุบันมีโปรแกรมในการใช้งานมากมาย ที่ช่วยในการทำงาน เช่น โปรแกรมตรวจจับผู้บุกรุก โปรแกรมตรวจจับแพ็กเก็ต และโปรแกรมป้องกันผู้บุกรุก เป็นต้น ซึ่งโปรแกรมเหล่านี้ติดตั้งลงบนเครื่องคอมพิวเตอร์ต่างๆ ในระบบเครือข่ายการติดตามสถานะและผลการทำงานนั้นผู้ดูแลระบบต้องติดต่อกับโปรแกรมเหล่านั้นในแต่ละเครื่องโดยตรง จุดนี้เองหากมีเครื่องที่ติดตั้งโปรแกรมเหล่านี้เป็นจำนวนมาก ย่อมเสียเวลาในการติดตามดูการทำงานของโปรแกรมในแต่ละเครื่อง อีกทั้งผู้ดูแลระบบเครือข่ายต้องจำด้วยว่า ได้ติดตั้งโปรแกรมเหล่านั้นไว้ที่เครื่องใดบ้าง นับเป็นสิ่งที่ทำให้ไม่ได้รับความสะดวกในการทำงาน

ปริญญานิพนธ์นี้จัดทำขึ้น เพื่อแก้ปัญหาในการติดตามผลการทำงานและสถานะของโปรแกรมต่างๆ ที่ได้กล่าวไปแล้วข้างต้น โดยสร้างระบบที่ใช้ไดเรกทอรีในการเก็บข้อมูลเพื่อใช้ในการติดตามดูสถานะและผลการทำงานของโปรแกรมต่างๆ ซึ่งประกอบไปด้วยรูปแบบของข้อมูลที่ใช้จัดเก็บและ API สำหรับใช้งานในการจัดการข้อมูลที่จัดเก็บในไดเรกทอรีทั้งบนวินโดว์สและยูนิกซ์

DIRECTORY SYSTEM FOR DISTRIBUTED SECURITY AGENT

Mr. Kanoksak Ratchapat
Mr. Jedsada Kumput
Mr. Thana Hongsuwan Advisor
Mr. Akkaradach Watcharapupong Advisor

ABSTARCT

Traffic monitoring and security management of network one necessary tasks for network administrators. Nowaday, there are so many programs to help them to do so, such as Intruder Monitoring Program, Packet Monitoring Program and Intruder Protection Program. Such programs will be installed in any computer sets on network and network administrators must directly connect to these programs in each computer sets to check status and get result. It is time consuming task to monitor each computer sets and network administrators must remember which computer sets one installed these programs. It is inconvenient for network administrators to do so.

This thesis aims to solve network administrators' problems by designing and developing directory system to collect data that necessary to monitor traffic efficiency and program status that cosist type of data to collect in directory and API for management data in directory on windows and unix

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้คงไม่อาจเสร็จได้ด้วยดี หากไม่ได้รับความช่วยเหลือและร่วมมือจากหลายฝ่ายด้วยกัน บุคคลแรกที่ต้องกล่าวถึงเพราะเป็นส่วนสำคัญที่ทำให้วิทยานิพนธ์นี้เสร็จลงได้ก็คือ อาจารย์ ธนา หงษ์สุวรรณ และ อาจารย์ อัครเดช วัชระภูพงษ์ อาจารย์ที่ปรึกษาวิทยานิพนธ์ ที่ให้ความเอาใจใส่แนะนำ และช่วยเหลือเสมอมา ซึ่งต้องขอขอบพระคุณเป็นอย่างมาก

ขอบคุณพี่เมย์ที่ช่วยให้คำแนะนำในการเขียนบทนำและช่วยแปลบทคัดย่อภาษาอังกฤษของปริญญานิพนธ์ฉบับนี้ และขอบขอบคุณ น้องเพ็ญนภา ชิณะวงศ์ ที่ช่วยแปลเอกสารและเป็นกำลังใจมาโดยตลอด

ขอขอบคุณห้องวิจัย ISAG ที่เอื้อเฟื้อสถานและทรัพยากรต่างๆ ในการค้นคว้าและทำโครงงาน และที่จะลืมไม่ได้คือห้องวิจัย OLALA ที่เอื้อเฟื้อสถานที่พักผ่อนแก่ผู้จัดทำ

ขอบคุณเพื่อนๆ และน้องๆ ในห้องวิจัย ISAG ทุกคนที่เอื้อเฟื้อขนมขบเคี้ยวในช่วงระหว่างการดำเนินงานโครงงานตลอดทั้งปีการศึกษารวมทั้งเพื่อนๆ ห้อง OLALA ที่มีผลไม้ตามฤดูกาลมาให้ทานเป็นประจำ

สุดท้ายต้องขอขอบพระคุณบุคคลสำคัญที่สุดที่ทำให้ข้าพเจ้ามีวันนี้ ก็คือบิดามารดาอันเป็นที่เคารพรักยิ่ง ซึ่งได้อบรมเลี้ยงดูผู้จัดทำมาเป็นอย่างดี พร้อมทั้งให้โอกาสในการศึกษาอย่างเต็มที่และยังให้กำลังใจเอาใจใส่เสมอมาในทุกๆ ด้านอันหาที่เปรียบมิได้ ข้าพเจ้าขอระลึกในพระคุณอันสุดประมาณและขอกราบขอบพระคุณมา ณ ที่นี้

กนกศักดิ์ รัชปัตย์
เจษฎา คำพุทธ

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญตาราง

สารบัญรูปภาพ

สารบัญรูปภาพ (ต่อ)

สารบัญรูปภาพ (ต่อ)

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มา

งานเฝ้าดูสภาพการทำงานและความปลอดภัยของเครือข่ายเป็นสิ่งจำเป็นสำหรับผู้ดูแลระบบเครือข่าย ซึ่งในปัจจุบันมีโปรแกรมในการใช้งานมากมาย ที่ช่วยในการทำงานเช่น โปรแกรมตรวจจับผู้บุกรุก โปรแกรมตรวจจับแพ็กเกจ และโปรแกรมป้องกันผู้บุกรุกเป็นต้น ซึ่งโปรแกรมเหล่านี้ติดตั้งลงบนเครื่องคอมพิวเตอร์ต่างๆ ในระบบเครือข่ายการติดตามสถานะและผลการทำงานนั้นผู้ดูแลระบบต้องติดต่อกับโปรแกรมเหล่านั้นในแต่ละเครื่องโดยตรง จุดนี้เองถ้าหากมีเครื่องที่ทำการติดตั้งโปรแกรมเหล่านี้เป็นจำนวนมาก ย่อมทำให้ต้องเสียเวลา ในการที่จะเข้าไปติดตามดู การทำงานของโปรแกรมในแต่ละเครื่อง อีกทั้งผู้ดูแลระบบเครือข่ายต้องจำด้วยว่าได้ติดตั้งโปรแกรมเหล่านั้นไว้ที่เครื่องใดบ้าง นับเป็นสิ่งที่ทำให้ไม่ได้รับความสะดวกในการทำงาน

ปริญญานิพนธ์นี้จัดทำขึ้นเพื่อแก้ปัญหาในการติดตามผลการทำงานและสถานะของโปรแกรมต่างๆที่ได้กล่าวไปแล้วข้างต้น โดยจะสร้างระบบสำหรับเก็บข้อมูลเพื่อใช้ติดตามสถานะและผลการทำงานของโปรแกรมต่างๆ ที่ผู้ดูแลระบบต้องการ ซึ่งข้อมูลต่างๆ นั้นจัดเก็บอยู่ในรูปแบบของบริการไดเรกทอรี

1.2 วัตถุประสงค์

1.2.1 เพื่อศึกษาการใช้งานบริการไดเรกทอรีสำหรับจัดเก็บข้อมูลโดยศึกษาจาก LDAP (Lightweight Directory Access Protocol)

1.2.2 เพื่อออกแบบข้อมูลต้นแบบที่ใช้ในการเฝ้าดูการทำงานและสถานะของโปรแกรมรักษาความ ปลอดภัยเครือข่าย

1.2.3 เพื่อพัฒนาระบบสำหรับเก็บข้อมูลการทำงานและสถานะของโปรแกรมรักษาความปลอดภัย เครือข่ายด้วยบริการไดเรกทอรี

1.2.4 เพื่อพัฒนา API สำหรับใช้งาน LDAP ให้ง่ายต่อการใช้งาน

1.3 ขอบเขต

1.3.1 สร้างรูปแบบข้อมูลสำหรับใช้ในการเฝ้าดูการทำงานและสถานะของโปรแกรมรักษาความปลอดภัยเครือข่าย

1.3.2 สร้างระบบสำหรับเก็บข้อมูลการทำงานและสถานะของโปรแกรมรักษาความปลอดภัยเครือข่าย

1.3.3 สร้าง API เพื่อการใช้งาน LDAP

1.4 ขั้นตอนการดำเนินงาน

1.4.1 ศึกษาบริการไดเรกทอรีด้วย LDAP

1.4.2 ศึกษาการออกแบบข้อมูลที่ใช้กับ LDAP

1.4.3 ศึกษาการใช้งาน API ของ LDAP

1.4.4 ออกแบบข้อมูลที่ใช้ในการเฝ้าดูการทำงานและสถานะของโปรแกรมรักษาความปลอดภัยเครือข่าย

1.4.5 ออกแบบระบบสำหรับจัดเก็บข้อมูลการทำงานและสถานะของโปรแกรมรักษาความปลอดภัย

เครือข่าย

1.4.6 ศึกษาการเขียนโปรแกรมบนเครือข่าย

1.4.7 สร้างรูปแบบข้อมูลสำหรับใช้ในการเฝ้าดูการทำงานและสถานะของโปรแกรมรักษาความปลอดภัยเครือข่าย

1.4.8 พัฒนาระบบสำหรับเก็บข้อมูลการทำงานและสถานะของโปรแกรมรักษาความปลอดภัยเครือข่าย

1.4.9 ทดสอบการทำงานและแก้ไขข้อผิดพลาดต่างของโปรแกรม

บทที่ 2
บริการไดเรกทอรี

2.1 ไดเรกทอรี (Directory) คืออะไร

ไดเรกทอรีคือลิสต์ของข้อมูลซึ่งมีการจัดลำดับและมีรายละเอียดปลีกย่อยต่างๆ ลงไปอีก เช่น สมุดโทรศัพท์ก็ถือว่าเป็นไดเรกทอรี ในสมุดโทรศัพท์มีชื่อรายนามและหมายเลขโทรศัพท์หรืออาจมีหมายเลขเพจเจอร์ ซึ่งสมุดโทรศัพท์จะเรียงลำดับตามรายชื่อผู้ใช้งานหรือบัตรห้องสมุดที่ไว้สำหรับค้นหาชื่อผู้แต่งหนังสือก็ถือเป็นไดเรกทอรีได้เช่นกันเพราะเป็นลิสต์ของชื่อผู้แต่งที่ลำดับตามค่าๆ หนึ่งแล้วยังมีรายละเอียดปลีกย่อยอีกเช่น เลข ISBN เป็นต้น

ในความหมายของทางคอมพิวเตอร์ ไดเรกทอรีคือโครงสร้างข้อมูลชนิดพิเศษที่เก็บชนิดข้อมูลเป็น ออบเจ็กต์ (object) และมีการลำดับไว้ ดังนั้นการระบุชนิดข้อมูลเช่น พรินเตอร์มีค่าต่างๆ ที่เราเข้าถึงได้เช่น ค่าจำนวนหน้าที่พิมพ์ได้ต่อนาที (เช่น 5 หน้าต่อนาทีหรือ 10 หน้าต่อนาที) หรือค่าพรินต์สตรีม (Print Stream) ที่ รองรับ เช่น ASCII หรือ PostScript เป็นต้น นอกจากนี้ไดเรกทอรียอมให้มีการค้นหาข้อมูลได้ เช่นค้นหา พรินท์เตอร์ที่สามารถพิมพ์ ASCII ได้หรือไดเรกทอรีที่เก็บ แอพพลิเคชันเซิร์ฟเวอร์ (Application Server) เราสามารถค้นหาเซิร์ฟเวอร์ (Server) ที่สามารถทำระบบบิลลิ่งได้เป็นต้น

บางครั้งความหมายของสมุดโทรศัพท์หน้าเหลืองหรือสมุดรายนามผู้ใช้โทรศัพท์สามารถช่วยให้เราเข้าใจการทำงานของไดเรกทอรี มากขึ้นถ้าเรารู้จักชื่อของออบเจ็กต์ เช่น ชื่อคน เราก็จะสามารถค้นหาจากสมุดรายนามผู้ใช้โทรศัพท์และได้ค่าที่อยู่มาในที่สุด แต่บางครั้งเป็นการยากที่เราจะรู้ชื่อที่เฉพาะเจาะจงลงไป ไดเรกทอรีสามารถจะค้นหารายละเอียดปลีกย่อยลงไปตามหมวดหมู่เช่นเดียวกับสมุดโทรศัพท์หน้าเหลือง แต่จะยืดหยุ่นมากกว่าเพราะไม่ เจาะจงหมวดหมู่เหมือนสมุดโทรศัพท์หน้าเหลือง

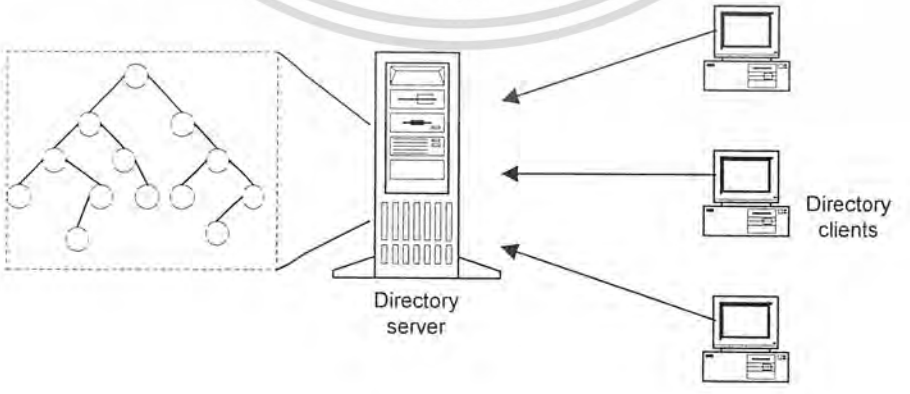
2.2 ไดเรกทอรีต่างจาก ดาต้าเบสอย่างไร

1. ไดเรกทอรีคือโครงสร้างข้อมูลชนิดพิเศษแต่ไม่ใช่ดาต้าเบสอย่างเช่น DB2 , Oracle เพราะไดเรกทอรีสร้างขึ้นมาเพื่อมีลักษณะเฉพาะบางอย่างเพิ่มขึ้นมาและที่มีบางส่วนเหมือนดาต้า ลักษณะที่สำคัญของไดเรกทอรีคือสร้างมาเพื่อเน้นการเข้าถึง (อ่านและค้นหา) แต่ไม่ได้ทำมาสำหรับการอัพเดตบ่อยๆ เช่น ข้อมูลเบอร์โทรของคนอาจจะมีคนเข้ามาค้นหาวันละเป็นพันครั้ง ดังนั้นไดเรกทอรีจึงเหมาะสำหรับการค้นหาบ่อยๆ แต่การอัพเดตจะเป็นหน้าที่ของ ผู้ดูแลระบบหรือเจ้าของข้อมูลนั้น ซึ่งมีเพียงไม่กี่คนที่สามารถ อัพเดตข้อมูลเหล่านั้นได้
2. ไดเรกทอรีเหมาะกับการเก็บข้อมูลที่มีการเปลี่ยนแปลงไม่บ่อย (Static Data) ไม่เหมาะกับการเก็บข้อมูลที่เปลี่ยนแปลงบ่อย (Dynamic Data) เช่น ข้อมูลความสามรถในการพิมพ์ ควรเก็บไว้ในไดเรกทอรี แต่ลำดับการทำงาน (Job Queue) ของ พรินเตอร์ที่มีการเปลี่ยนแปลงบ่อยไม่ควรเก็บไว้ใน ไดเรกทอรี

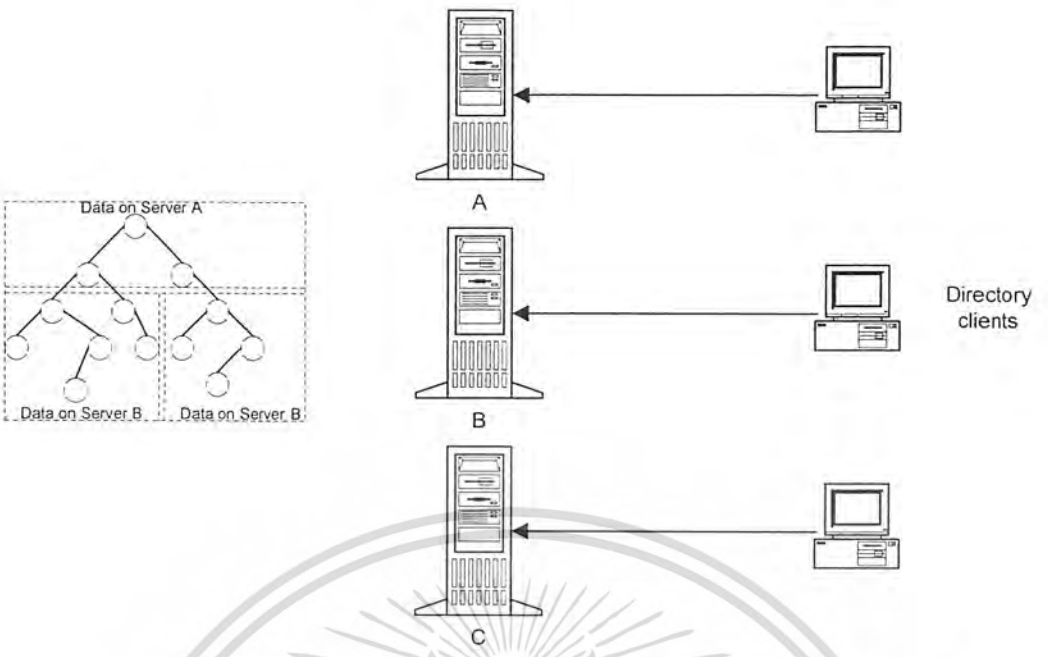
3. ไดเรกทอรีไม่รองรับอะตอมมิคทรานแซกชัน (Atomic Transaction) ถ้าจะทำต้องทำทั้งหมด ถ้าไม่ทำก็ไม่ทำเลย เพราะเนื่องจากไดเรกทอรีเน้นการอ่านมากกว่าเขียน ดังนั้นเพื่อให้อ่านได้เร็วขึ้น จึงจำเป็นต้องเสียส่วนนี้ไปซึ่งต่างจากดาต้าเบส นอกจากนี้ไดเรกทอรีก็ไม่สนับสนุน ไอโซเลชัน (Isolation) ซึ่งคือการการเปลี่ยนแปลงข้อมูลพร้อมๆกัน เพราะเป็นข้อมูลที่มีเพียงไม่กี่คนที่เปลี่ยนแปลงข้อมูลได้ (มี ไดเรกทอรีของผู้ผลิตบางราย ที่สนับสนุนการทำ ทรานแซกชันบ้างแล้ว)
4. ข้อมูลในไดเรกทอรี บางครั้งอาจเป็นข้อมูลที่ไม่ถูกต้อง (ในแง่ของผู้ใช้ ไม่ใช่ระบบ) เช่น ที่อยู่ของคนอาจไม่ถูกต้องเพราะคนคนนี้ย้ายที่อยู่ไปแล้ว ดังนั้นไม่ควรเก็บข้อมูลที่เปลี่ยนแปลงง่าย (เช่น ยอด Balance ของบัญชี)ไว้ที่ไดเรกทอรี
5. การใช้ดาต้าเบส เก็บข้อมูลของแอพพลิเคชัน เช่น การเก็บข้อมูลของลูกค้าในระบบบิลลิ่งเมื่อเราเพิ่มระบบอื่นเข้าไปเช่นเมล์ ทำให้ผู้ดูแลระบบยุ่งยากเช่นอาจจะต้องเพิ่มคอลัมน์เข้าไปในตารางที่เก็บข้อมูลลูกค้า แต่ถ้าใช้ไดเรกทอรีจะง่ายต่อการขยาย
6. ดาต้าเบสจะใช้ SQL แต่ไดเรกทอรีจะมีภาษาที่ต่างไปและจะช่วยให้เข้าถึงได้เร็วขึ้น ความแตกต่างระหว่างดาต้าเบสและไดเรกทอรี ทำให้ผู้ออกแบบระบบต้องคำนึงถึงและเข้าใจในข้อจำกัดของความแตกต่างทั้งสองเพื่อจะได้แบ่งแยกได้ถูกว่าข้อมูลชนิดไหนควรจะเก็บลงดาต้าเบสหรือไดเรกทอรี

2.3 ดิสทริบิวเต็ดไดเรกทอรี (Distributed Directory)

ดิสทริบิวเต็ดไดเรกทอรีหมายถึงการเก็บข้อมูลโดยใช้ไดเรกทอรีหลายตัวช่วยกันเก็บข้อมูล หรือนำข้อมูลไว้หลายที่ ส่วนเซ็นทรัลไลซ์ไดเรกทอรี(Centralized Directory) หมายถึงการเก็บข้อมูลไว้ที่เดียว ดังรูปที่ 2-1 และถ้าเราเก็บข้อมูลแบบดิสทริบิวเต็ดไดเรกทอรีเราก็จะแบ่งได้อีกว่าเป็นแบบพาร์ติชัน (Partition) คือข้อมูลที่แบ่งกันเก็บนั้นไม่ซ้ำกันดังรูปที่ 2-2 แต่ถ้าหากข้อมูลที่เก็บนั้นซ้ำกันก็เรียกว่าเป็นแบบเรพลิเคเต็ด (Replicated)



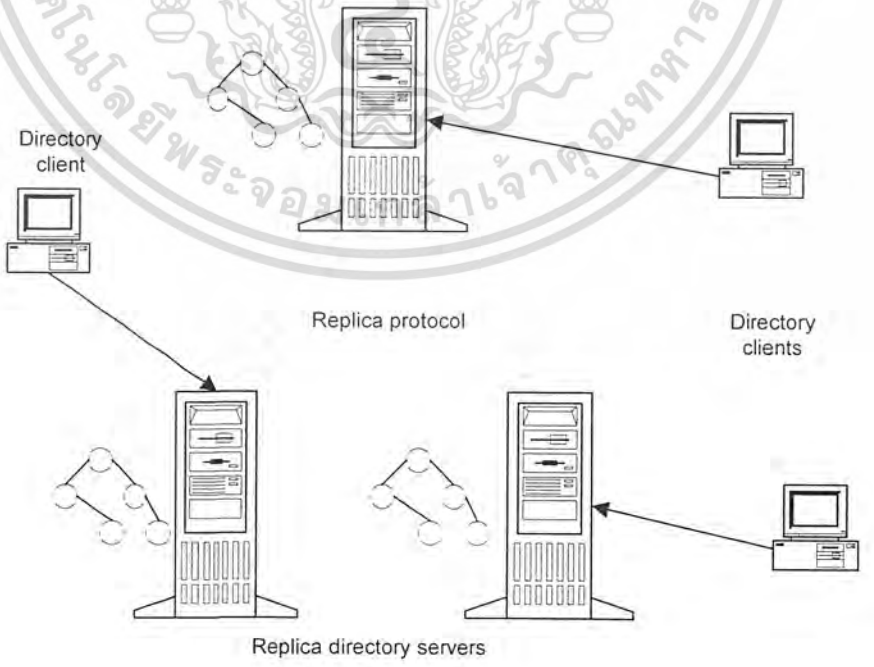
รูปที่ 2-1 แสดงให้เห็นถึงลักษณะการเก็บข้อมูลของเซ็นทรัลไลซ์ไดเรกทอรี



รูปที่ 2-2 แสดงให้เห็นถึงลักษณะการเก็บข้อมูลของดิสทริบิวเต็ดไดเรกทอรี

2.4 เรพลิเคเต็ดไดเรกทอรี (Replicated Directory)

แรพพลิเคตเต็ดไดเรกทอรี จะเก็บข้อมูลบางส่วนที่ซ้ำกัน การเก็บข้อมูลแบบดิสทริบิวเต็ด แรพพลิเคตเต็ด ไดเรกทอรี (Distributed Replicated Directory) จะดีในแง่ถ้ามีไดเรกทอรีใดไม่ทำงานก็จะสามารถใช้ข้อมูลจากอีกตัวได้แต่ก็มีข้อเสียที่เราจะต้องจัดการให้ข้อมูลทั้งสองที่เหมือนกัน รูปที่ 2-3 แสดงให้เห็นถึงรูปแบบการเก็บข้อมูลแบบเรพลิเคเต็ดไดเรกทอรี



รูปที่ 2-3 รูปแบบการเก็บข้อมูลแบบแรพพลิเคตเต็ดไดเรกทอรี

2.5 ไดเรกทอรีอีนาเบิลแอพพลิเคชัน (Directory Enabled Application)

ถ้าเราสร้าง แอพพลิเคชันจองห้องประชุม ในกรณีที่แย่ที่สุดถ้าเราไม่ใช้ระบบไดเรกทอรีเลย เมื่อผู้ใช้จองห้องจะต้องจำว่าห้องประชุมนั้นจุคนได้เท่าใด มีเครื่องอำนวยความสะดวกอะไรบ้าง คนที่มาจองนั้นมี อีเมล์อะไร ผู้เข้าประชุมคนใดบ้างที่ได้รับการแจ้งข่าวแล้วทำให้ยากต่อการใช้งาน แต่ถ้าเราใช้ระบบไดเรกทอรีจะช่วยให้เราจัดการได้ง่ายขึ้น มีไดเรกทอรีของห้องประชุมที่มีข้อมูลต่างๆ (เช่น ขนาด สถานที่ตั้ง อุปกรณ์ต่างๆในห้องประชุม) มีไดเรกทอรีของคนที่เก็บข้อมูลต่างๆ (เช่น ชื่อ หมายเลขโทรศัพท์ ที่อยู่) โดยแอพพลิเคชันนี้สามารถเข้าถึงข้อมูลต่างๆ ได้อย่างมีประสิทธิภาพ

บทที่ 3
แนะนำ LDAP

3.1 LDAP คืออะไร

LDAP เป็นชื่อของโพรโตคอล เป็นแอปพลิเคชั่นโพรโตคอลที่ใช้เข้าถึงข้อมูลที่เป็นชนิดไดเรกทอรี เป็นไลต์เวทโพรโตคอล ซึ่งหมายความถึงความมีประสิทธิภาพของมันความกะทัดรัดและอิมพลีเมนต์ใช้งานได้ง่าย โดยยังคงมีความเป็นฟังก์ชันนอลสูง โพรโตคอลนี้จะรันอยู่บน TCP/IP และสามารถใช้งานทุกแพลตฟอร์ม

มาตรฐานของ LDAP จะมีการกำหนดโมเดล ที่เกี่ยวข้องกับการใช้งานอยู่ 4 โมเดลด้วยกันดังนี้

- The LDAP Information Model,จะกำหนดชนิดของข้อมูลที่เราสามารถใส่ลงไปใน ไดเรกทอรีได้
- The LDAP Naming Model,จะกำหนดว่าเราจะจัดเรียงรวมถึงการเข้าถึงข้อมูลในไดเรกทอรีได้อย่างไร
- The LDAP Functional Model,จะกำหนดว่าสามารถเข้าถึงและ เปลี่ยนแปลงข้อมูลที่อยู่ในไดเรกทอรีได้อย่างไร
- The LDAP Security Model, จะกำหนดว่าเราจะมีการป้องกันข้อมูลในไดเรกทอรีจากบุคคลภายนอกได้อย่างไร

LDAP ยังได้กำหนด LDIF (LDAP Data Interchange Format) ขึ้นมาซึ่งเป็นรูปแบบของ เท็กซ์เบส (Text-based) ที่ได้อธิบายถึงข้อมูลของไดเรกทอรี โดยที่ LDIF สามารถบอกรายละเอียดของกลุ่มของเอนทรี(entry) ที่ต้องการที่จะ เปลี่ยนแปลงข้อมูลเหล่านั้นลงไปในไดเรกทอรี ดังนั้น LDIF ก็คือกลุ่มของเอนทรีที่เราเขียนขึ้นมาให้อยู่ในรูปของไฟล์ ซึ่งเราสามารถนำข้อมูลที่ได้นั้น เปลี่ยนแปลงข้อมูลในไดเรกทอรีและไดเรกทอรีก็ยังสามารถเอ็กซ์พอร์ต (export) ข้อมูลที่อยู่ในไดเรกทอรีออกมาให้อยู่ในรูปแบบของ LDIF ได้ เพื่อที่จะส่งไปยัง LDAP Server ตัวอื่น ปกติแล้วเราใช้ LDIF กับการเพิ่มหรือเปลี่ยนแปลงข้อมูลแบบ Offline

3.2 LDAP สามารถทำอะไรให้เราได้บ้าง

ถ้าเราเป็นผู้ดูแลระบบ

- เราสามารถเป็นตัวกลางในการบริหาร users, groups, devices และ ข้อมูลต่างๆได้
- ช่วยให้เราลดภาระเกี่ยวกับการบริหารการแบ่งงานของแอปพลิเคชั่นไดเรกทอรี

ถ้าเราเป็นผู้ตัดสินใจทางด้านไอที

- ช่วยให้เราหลีกเลี่ยงการผูกมัดทางการค้ากับผู้ขายเพียงเจ้าเดียว และแพลตฟอร์ม (Platform) เดียว
- ช่วยลดค่าใช้จ่ายรวมของความเป็นเจ้าของโดยลดจำนวนของไดเรกทอรีที่ผู้ร่วมงานหรือผู้ใช้ในองค์กรให้อยู่ในขอบข่ายและความจำเป็น

ถ้าเราเป็นผู้พัฒนาซอฟต์แวร์

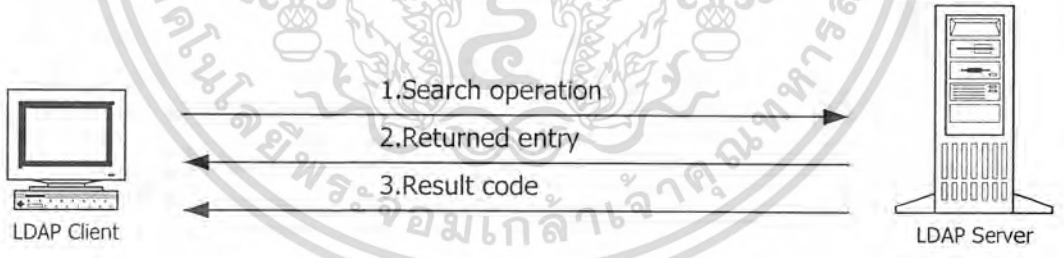
- ช่วยให้เราหลีกเลี่ยงการผูกมัดทางการค้ากับผู้ขายเพียงเจ้าเดียว และแพลตฟอร์มเดียว
- ช่วยให้เราลดเวลาในการพัฒนาโดยหลีกเลี่ยงความต้องการของผู้ใช้ในองค์กร และการจัดการฐานข้อมูลแบบกลุ่ม

3.3 LDAP ทำงานอย่างไร

โพรโตคอลที่เรียกว่า ไคลเอนต์เซิร์ฟเวอร์โพรโตคอล (Client/Server protocol) หมายถึงเครื่องไคลเอนต์ เครื่องหนึ่งรันโปรแกรมเพื่อร้องขอความต้องการ โดยส่งมันผ่านเน็ตเวิร์คมายังเครื่องเซิร์ฟเวอร์ เมื่อเครื่องเซิร์ฟเวอร์รับการร้องขอมาจากเครื่องไคลเอนต์ จากนั้นก็กระทำการอย่างใดอย่างหนึ่งเพื่อหาผลลัพธ์ เมื่อได้ผลลัพธ์แล้วก็จะส่งไปให้เครื่องไคลเอนต์ ตัวอย่างเช่นโพรโตคอล TCP/IP

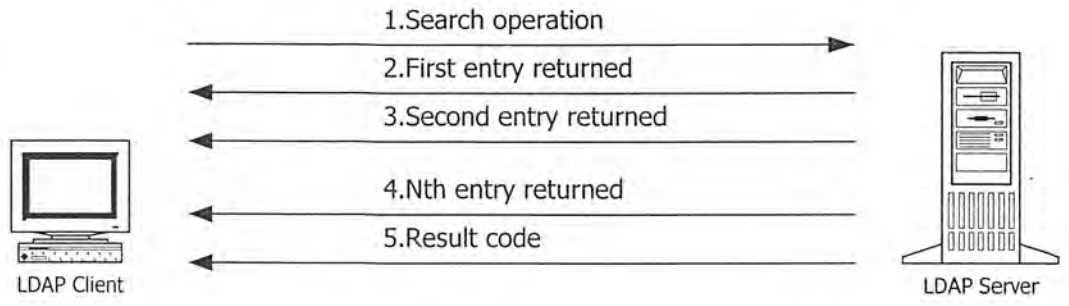
จากหลักการเบื้องต้นเราก็ได้แนวคิดและนำมาใช้กับแมสเซจโอเรียนเต็ดโพรโตคอล (Message Oriented Protocol) อย่าง LDAP ซึ่งเมื่อไคลเอนต์สร้าง LDAP แมสเซจซึ่งขอความต้องการไปยังเซิร์ฟเวอร์ จากนั้นเซิร์ฟเวอร์ก็ปฏิบัติกับคำร้องขอและเมื่อได้คำตอบก็จะส่งกลับไปให้กับไคลเอนต์ แบบอนุกรมในรูปแบบของ LDAP แมสเซจ

ตัวอย่างเช่นเมื่อ LDAP ไคลเอนต์จะค้นหาเอนทรีในไดเรกทอรี มันจะสร้างแมสเซจที่ขอการค้นหา และส่งไปยังเซิร์ฟเวอร์ จากนั้นเซิร์ฟเวอร์จะส่งเอนทรีจากฐานข้อมูลของมันและส่งมันไปให้กับไคลเอนต์ในรูปของ LDAP แมสเซจและจะส่งโค้ดผลลัพธ์เพื่อเป็นการแสดงว่าได้ส่งผลลัพธ์ไปให้แล้ว ดังรูปที่ 3-1



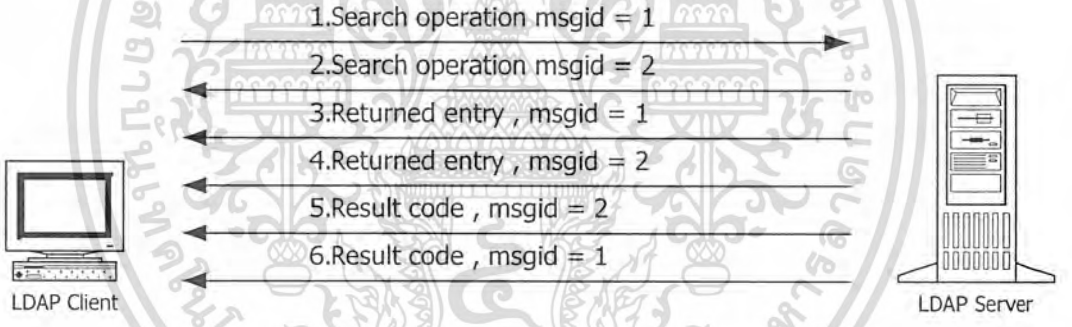
รูปที่ 3-1 เครื่องไคลเอนต์ได้รับ Single entry จากไดเรกทอรี

เมื่อไคลเอนต์ ค้นหาในไดเรกทอรี และเจอผลลัพธ์หลายเอนทรี จากนั้นเอนทรีเหล่านั้นก็จะถูกส่งไปยังไคลเอนต์แบบอนุกรมโดยใช้ LDAP message ซึ่งเก็บผลลัพธ์ทั้งหมดที่ได้จากการค้นหาดังรูปที่ 3-2



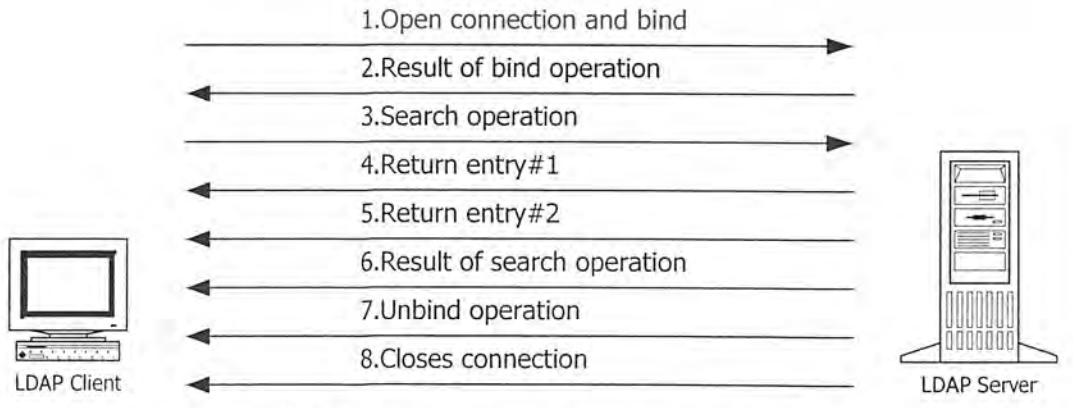
รูปที่ 3-2 ไคลเอนต์ค้นหาไดเรกทอรีและได้รับผลลัพธ์ซึ่งเป็น เอนทรีหลายตัวกลับคืนมา

เพราะว่า โพรโตคอล LDAP เป็นโพรโตคอลแบบ Message-base มันจึงยอมให้ไคลเอนต์ร้องขอไปมากกว่าหนึ่งการร้องขอ ต่อการส่งแมสเซจไปในแต่ละครั้ง สมมติว่าไคลเอนต์ต้องร้องขอสองการค้นหาในเวลาเดียวกัน มันจะสร้าง generate message ID สำหรับแต่ละการร้องขอ และการส่งผลลัพธ์กลับมาก็จะมี Message ID ติดมากับแต่ละผลลัพธ์ด้วย และอนุญาตให้ไคลเอนต์เรียงลำดับผลลัพธ์ที่ส่งไปให้ได้ด้วย ดังตัวอย่างในรูปที่ 3-3



รูปที่ 3-3 ไคลเอนต์มีการร้องขอมากกว่าหนึ่งการร้องขอ ในเวลาเดียวกัน

ตัวอย่างของ LDAP ไคลเอนต์/เซิร์ฟเวอร์ ที่สมบูรณ์แบบอีกตัวอย่างหนึ่งดังรูป



รูปที่ 3-4 ตัวอย่างการแลกเปลี่ยน message ของ LDAP

จากรูปเราสามารถอธิบายขั้นตอนการทำงานได้ดังนี้

ขั้นตอนที่ 1 ไคลเอนต์เปิดการเชื่อมต่อ TCP ไปยัง LDAP เซิร์ฟเวอร์และผูกการเชื่อมต่อซึ่ง Bind operation จะรวมชื่อของ เอนทรีที่ไคลเอนต์ต้องการรับรองและพาสเวิร์ดมาด้วย

ขั้นตอนที่ 2 หลังจากที่ได้มีการพิสูจน์ความเป็นบุคคลของไคลเอนต์แล้วเซิร์ฟเวอร์ก็จะส่งผลลัพธ์กลับมาที่ไคลเอนต์ว่าได้มีการพิสูจน์แล้ว

ขั้นตอนที่ 3 ไคลเอนต์ส่งการร้องขอการค้นหามายังเซิร์ฟเวอร์

ขั้นตอนที่ 4 และ 5 เซิร์ฟเวอร์จะดำเนินการกับการร้องขอและส่งผลลัพธ์ที่ตรงกับที่ขอมาสองเอนทรีด้วยกัน

ขั้นตอนที่ 6 เซิร์ฟเวอร์ส่งผลลัพธ์ไปให้ไคลเอนต์ เพื่อบอกว่าได้ส่งผลลัพธ์ ไปให้เรียบร้อยแล้ว

ขั้นตอนที่ 7 ไคลเอนต์จะส่งการร้องขอการยกเลิกการเชื่อมต่อซึ่งแสดงให้เซิร์ฟเวอร์เห็นว่าไคลเอนต์ต้องการที่จะยกเลิกการเชื่อมต่อ

ขั้นตอนที่ 8 เซิร์ฟเวอร์ ก็จะปิดการเชื่อมต่อ

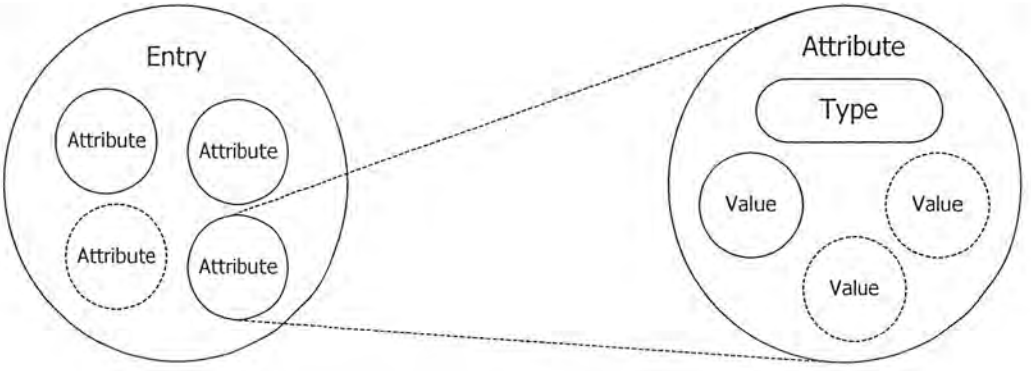
3.4 โมเดล LDAP

LDAP จะกำหนดโมเดลพื้นฐาน 4 อย่างด้วยกันและอธิบายการทำงานว่าข้อมูลอะไรที่สามารถเก็บไว้ใน LDAP ไดเรกทอรีได้และเราสามารถทำอะไรกับข้อมูลเหล่านั้นได้บ้างซึ่งจะอธิบายแต่ละโมเดลดังต่อไปนี้

3.4.1 อินฟอร์เมชั่นโมเดล (Information Model)

จะกำหนดชนิดของข้อมูลและหน่วยพื้นฐานของข้อมูลที่เราสามารถเก็บในไดเรกทอรีได้ นอกจากนั้นยังอธิบายถึงการสร้างบล็อกที่เราสามารถใช้สร้างไดเรกทอรีได้

หน่วยพื้นฐานของข้อมูลในไดเรกทอรีก็คือเอนทรี ซึ่งเป็นกลุ่มของข้อมูลที่อยู่ในรูปของออบเจ็กต์ บ่อยครั้งที่ข้อมูลใน เอนทรีจะอธิบายถึง ออบเจ็กต์ที่มีอยู่จริง เช่น มนุษย์ หรือ สิ่งของอื่นๆ ซึ่งเป็นการง่ายต่อการที่เราจะค้นหาสิ่งใดสิ่งหนึ่งที่แยกตาม ออบเจ็กต์ที่เราคุ้นเคยและสามารถใช้ได้ในองค์กรของเรา และใน เอนทรี จะประกอบด้วยหลายๆ แอตทริบิวต์ซึ่งแต่ละอันก็จะอธิบายลักษณะเฉพาะของแต่ละ ออบเจ็กต์ โดยแต่ละแอตทริบิวต์ก็จะมีไทป์ (type) และ แวลู (value) ซึ่ง แวลูนั้นอาจจะมีค่าเดียว หรือหลายค่าก็ได้ โดยที่ไทป์ของแต่ละ ออบเจ็กต์ ก็จะอธิบายชนิดของข้อมูลที่เก็บอยู่ในแอตทริบิวต์และแวลูนั้น จะเก็บข้อมูลจริงของแอตทริบิวต์นั่นเอง



รูปที่ 3-5 เอนทรี ,แอตทริบิวต์และแวลู

ตัวอย่างที่จะแสดงให้เห็นนี้เป็นการมองเข้าไปในเอนทรีของบุคคล ซึ่งก็จะมีแอตทริบิวต์ของบุคคล เช่น person's full name (cn), surname (sn), telephone number และ email address

Cn:	Babara Jensen Babs Jensen
Sn:	Jensen
Telephonenumber:	+1 408 555 1212
Mail:	Babs@airius.com

LDIF

จากโค้ดข้างล่างนี้ เราจะเห็นไดเรกทอรีเอนทรี ที่แสดงอยู่ในรูปแบบของ LDIF (LDAP Data Interchange Format) ซึ่งมันเป็นมาตรฐานของ LDAP อีกอย่างหนึ่งที่ใช้เสนอ ข้อมูลของไดเรกทอรี ที่อยู่ในรูปแบบ เท็กซ์ ซึ่งจะนำมันมาใช้ เมื่อเราเอกซ์พอร์ต (export) ข้อมูล และ อิมพอร์ตข้อมูลจากไดเรกทอรีเซิร์ฟเวอร์ภายนอก โดย LDIF ไฟล์ จะประกอบด้วยตัวหนังสือที่เป็นไปตามมาตรฐานของแอสกีที่ถูกต้องตามรูปแบบมาตรฐาน คือมีชื่อไม่เกิน 8 ตัวอักษร ดังตัวอย่าง

```
dn: uid=bjensen, dc=airius, dc=com
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: InetOegPerson
cn: Babara Jenseb
```

cn: Babs Jensen

sn: Jensen

mail: bjensen@airius.com

telephoneNumber: +1 408 555 1212

description: A big sailing fan.

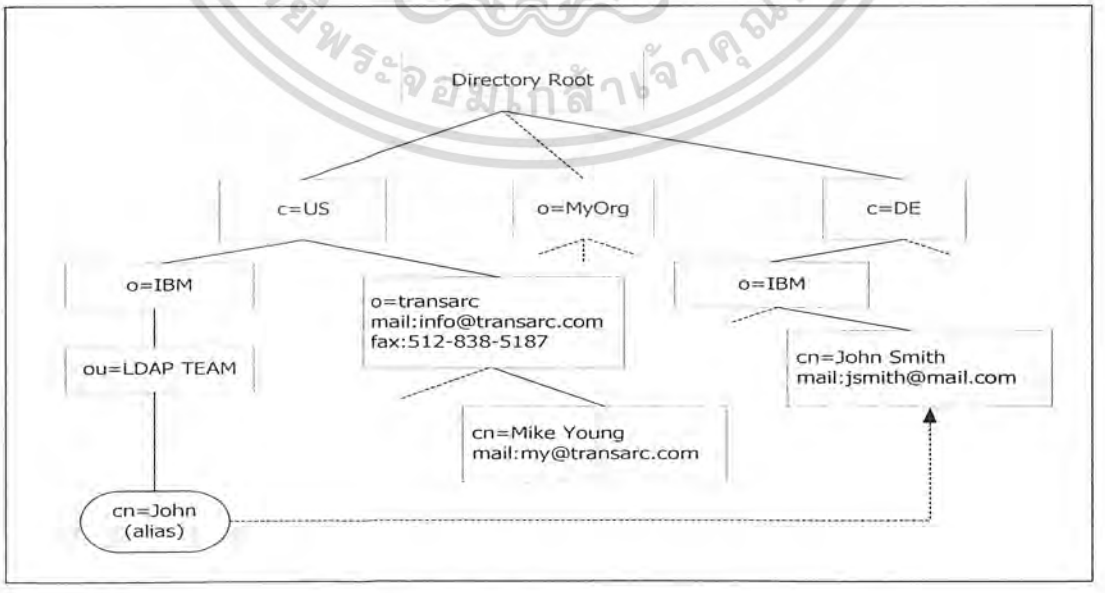
LDIF เอนทรีนั้นจะประกอบด้วยเส้นบรรทัดที่ต่อกันไปเรื่อยๆ ซึ่งมันจะเริ่มด้วย dn: ซึ่งหมายถึงชื่อเฉพาะของเอนทรีนั้นๆ ซึ่งในบรรทัดนั้นจะเป็นชื่อทั้งหมด หลังจากนั้นก็จะเป็นแอตทริบิวต์ของเอนทรี ซึ่งหนึ่งบรรทัดจะมีหนึ่งแอตทริบิวต์ ซึ่งค่าของแอตทริบิวต์แต่ละแอตทริบิวต์นั้นจะคั่นด้วยเครื่องหมายโคลอน (:) ดังตัวอย่างข้างต้น

3.4.2 เนมมิ่งโมเดล (Naming Model)

จะกำหนดว่าเอนทรี จะมีการจัดระเบียบและเรียกใช้งานอย่างไร ซึ่งเอนทรีเหล่านี้จะจัดเรียงกันในทรีเช่นเดียวกับสตรักเจอร์(structure) เราเรียกว่า Directory Information Tree (DIT) โดยเอนทรีเหล่านั้นจะเรียงกันอยู่ใน DIT โดยยึด DN เป็นหลัก

DN เป็นชื่อที่ไม่ซ้ำกันซึ่งเป็นชื่อที่เราระบุให้กับเอนทรีโดย DN นั้นจะทำให้เกิด Relative distinguished names (RDNs) โดยแต่ละ RDN ใน DN จะมีลักษณะเช่นเดียวกันและต่อกิ่งก้านออกมาจากไดเรกทอรี เอนทรี

แต่ละ RDN จะสืบทอดมาจากแอตทริบิวต์ของไดเรกทอรีเอนทรี เราจะเห็นในตัวอย่างด้านล่างแต่ละบล๊อก จะแทนแต่ละไดเรกทอรีเอนทรีและไดเรกทอรีรูทไม่ได้มีอยู่จริงเป็นเพียงการสมมติขึ้น เพื่อให้เรามองเห็นภาพเท่านั้นเอง



รูปที่ 3-6 ตัวอย่าง Directory Information Tree (DIT)

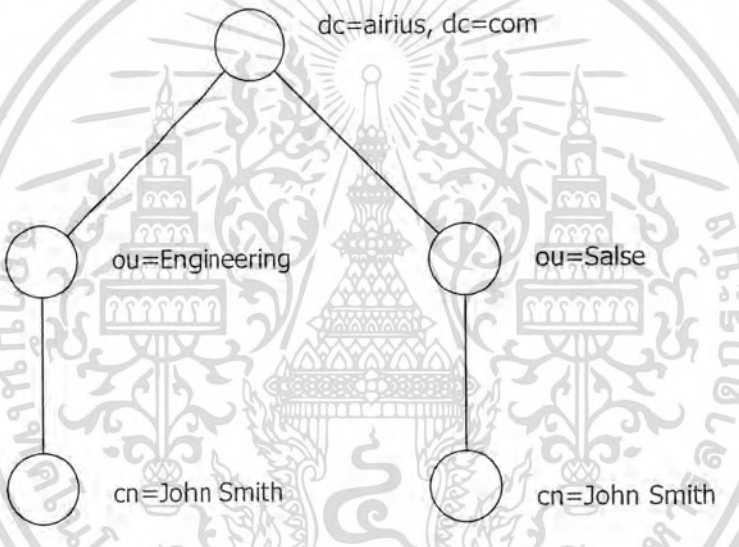
ทำไมเนมมิ่งโมเดลจึงมีความสำคัญ

เนมมิ่งโมเดลมีความจำเป็น เป็นอย่างยิ่ง เพราะเราจะต้องให้ทุกๆเอนทรีมีชื่อที่ไม่ซ้ำกัน เช่นเดียวกันกับ ไฟล์ของระบบ DN ก็เปรียบกับชื่อไฟล์และพาร์ตของไฟล์ทั้งหมดที่นับจากรูทและ RDN ก็หมายถึงชื่อไฟล์นั่นเอง ตัวอย่างของ DN มีดังนี้

Uid=bjensen, ou=people, dc=airius, dc=com

Uid=bjensen,ou=people,dc=airius,dc=com

ในแต่ละเอนทรีนั้น เราจะเลือกให้มี RDN ของแต่ละเอนทรีอยู่ซึ่งแน่นอนว่าชื่อของแต่ละ RDN จะไม่เหมือนกันเพราะ เวลาเรามีโอเปอเรชันที่เกี่ยวกับการค้นหาหรือการแก้ไขต่างๆ มันจะนำเอา RDN เป็นตัวที่ใช้ในการเปรียบเทียบ เพราะฉนั้นแต่ละ RDN จะต้องไม่ซ้ำกันโดยเด็ดขาด แต่ยอมให้มีชื่อเหมือนกันได้ ถ้าอยู่คนละพาร์ตของทรี ดังรูป



รูปที่ 3-7 เอนทรีที่มี RDN เหมือนกัน จะต้องอยู่คนละพาร์ตกัน

3.4.3 ฟังก์ชันนอลโมเดล (Functional Model)

เมื่อเราเข้าใจโมเดลทั้งสองแบบข้างต้นแล้ว เราจำเป็นที่จะต้องหาทางที่จะเข้าถึงข้อมูลที่เก็บอยู่ในไดเรกทอรีทรีซึ่งฟังก์ชันนอลโมเดลจะอธิบายถึงโอเปอเรชันที่เราสามารถทำได้ในไดเรกทอรีโดยใช้โพรโตคอล LDAP

ฟังก์ชันนอลโมเดลจะประกอบด้วยกลุ่มของโอเปอเรชันที่เราสามารถแบ่งเป็นกลุ่มได้ 3 กลุ่มด้วยกัน ดังนี้

Query	โอเปอเรชันนี้จะอนุญาตให้เราค้นหาข้อมูลในไดเรกทอรี และสามารถเอาข้อมูลออกมาได้ โอเปอเรชันที่เกี่ยวข้องคือ search , compare
Update	โอเปอเรชันนี้จะอนุญาตให้เรา เพิ่ม ลบ แก้ไข และเปลี่ยนแปลงข้อมูลใน directory entry โอเปอเรชันที่เกี่ยวข้องคือ add, delete, modify และ modify RDN
Authentication	โอเปอเรชันนี้จะอนุญาตให้ไคลเอนต์ เชื่อมต่อการทำงาน และควบคุมการเชื่อมต่อ ด้วยตัวเอง โอเปอเรชันที่เกี่ยวข้องคือ bind, unbind และ abandon

3.4.3.1 การคิวรี (Query Operation)

Search Operation

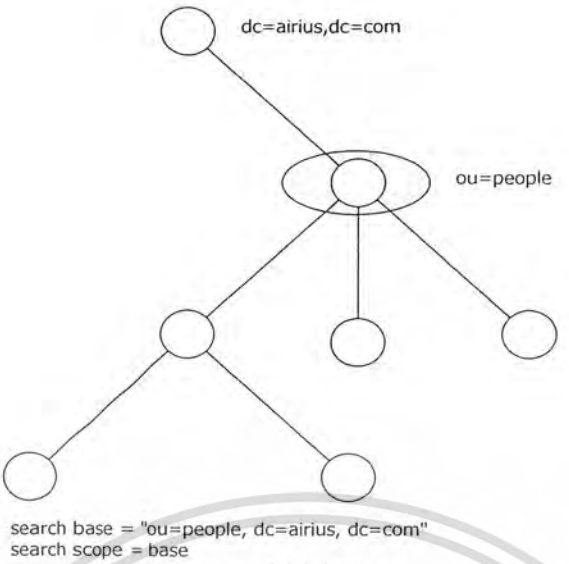
โอเปอเรชันค้นหาจะอนุญาตให้ไคลเอนต์ขอ LDAP Server เข้าถึง DIT สำหรับค้นหาข้อมูลตามเงื่อนไขเพื่ออ่านผลลัพธ์ออกไปใช้งาน โดยที่โอเปอเรชันค้นหาจะอนุญาตให้ใส่จุดเริ่มต้นที่อยู่ใน DIT, ความลึกของชั้นที่ต้องการค้นหา, แอตทริบิวต์ที่ต้องการค้นหา ใน LDAP จะไม่มีโอเปอเรชันที่เกี่ยวกับการอ่าน ดังนั้นเมื่อเราต้องการอ่านข้อมูลในเอนทรีเราต้องใช้โอเปอเรชันของการค้นหา เพื่อให้ได้ข้อมูลที่เราต้องการ

โอเปอเรชันในการค้นหาของ LDAP นั้นมีพารามิเตอร์อยู่ทั้งหมด 8 ตัวด้วยกัน ดังนี้

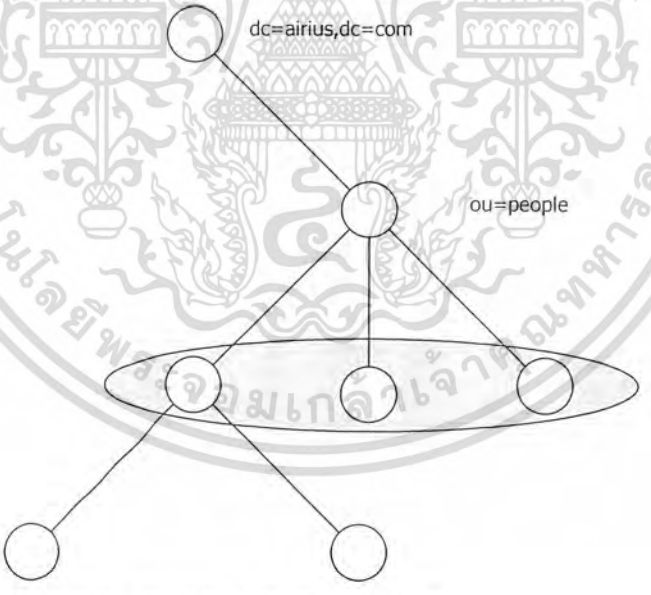
พารามิเตอร์ตัวแรกคือเบสออบเจกต์ (base object) ที่ต้องการค้นหาพารามิเตอร์ตัวนี้ก็คือ DN ซึ่งเป็นส่วนของทรีส่วนบนสุดที่เราต้องการจะค้นหา

พารามิเตอร์ตัวที่สองก็คือสโคบ (scope) ซึ่งสโคบจะมีอยู่ 3 ประเภทด้วยกัน คือ

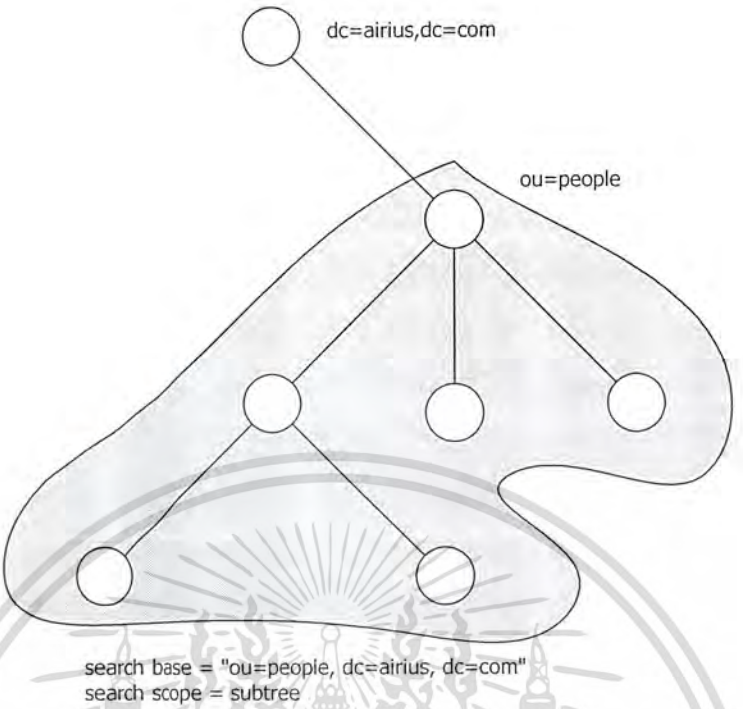
- subtree : เราจะระบุ ก็ต่อเมื่อเราต้องการค้นหา ทุกระดับที่อยู่ใต้ออบเจ็กต์ที่เราระบุ
- onelevel : เมื่อเราระบุสโคบชนิดนี้มันจะค้นหาให้เรานับตั้งแต่ออบเจ็กต์ที่เราระบุเพียงหนึ่งระดับเท่านั้น
- base : เมื่อเราต้องการค้นหาข้อมูลของเบสออบเจกต์



รูปที่ 3-8 base indicates



รูปที่ 3-9 onelevel indicates



รูปที่ 3-10 subtree indicates

พารามิเตอร์ของการค้นหาตัวที่สามก็คือ derefAliases ซึ่งสามารถแสดงค่าทั้งหมด 4 ค่าด้วยกันคือ neverderefAliases, derefInSearching, derefFindingBaseObject และ derefAlways ตามลำดับ

พารามิเตอร์ของการค้นหาตัวที่สี่ก็คือ size limit โดยพารามิเตอร์นี้จะบอกเซิร์ฟเวอร์ว่าไคลเอนต์มีความสนใจที่จะรับเอนทรีในแต่ละครั้งไม่เกินเท่าใด

พารามิเตอร์ของการค้นหาตัวที่ห้าก็คือ time limit โดยพารามิเตอร์นี้จะบอกเซิร์ฟเวอร์ว่าเวลาสูงสุดเท่าใดที่ควรจะใช้ค้นหาซึ่งถ้าเกินเวลาที่กำหนดไว้เซิร์ฟเวอร์ก็จะหยุดการดำเนินการ และส่ง LDAP_TIMELIMIT_EXCEEDED ซึ่งเป็น result code ไปให้กับเครื่องไคลเอนต์

พารามิเตอร์ของการค้นหาตัวที่หกก็คือ attrsOnly จะเป็นข้อมูลชนิดบูลีนถ้ามันถูกเซ็ตเป็น true เซิร์ฟเวอร์ก็จะส่งเฉพาะ ชนิดของแอตทริบิวต์ ไปให้ไคลเอนต์เท่านั้น โดยมันจะไม่ส่งค่าของแอตทริบิวต์ไปให้ แต่ถ้าพารามิเตอร์ถูกเซตให้เป็น false เซิร์ฟเวอร์ก็จะส่งทั้งชนิดและค่าของแอตทริบิวต์ไปให้ไคลเอนต์

พารามิเตอร์ของการค้นหาตัวที่เจ็ดก็คือ search filter จะเป็นตัวกรองชนิดของเอนทรีที่จะคืนให้กับไคลเอนต์

พารามิเตอร์ของการค้นหาตัวที่แปดก็คือ การแสดงแอตทริบิวต์ที่จะคืนให้สำหรับแต่ละ เอนทรีที่ตรงเงื่อนไข เราสามารถระบุได้ว่า มันควรจะคืนผลลัพธ์มาให้เราทั้งหมดหรือระบุแอตทริบิวต์ที่เราต้องการให้คืนมาก็ได้

Equality Filters

Equality filter จะอนุญาตให้เราค้นหา เอนทรีที่ตรงกับค่าที่เรากำหนด เช่น (sn=smith)

Substring Filter

เมื่อเราต้องการค้นหาข้อมูลของเอนทรีใดๆ ที่เราสามารถส่งพารามิเตอร์ไปส่วนหนึ่ง หรือเป็นสับสตริงหนึ่งไปมันก็จะสามารถหาค่าที่อยู่ในเอนทรีที่มีสับสตริงนั้นอยู่ ตัวอย่างเช่น (sn=smith*), (sn=*smith),(sn=smi*th) เมื่อค้นหาตามตัวอย่างแรก ผลลัพธ์ที่ได้ที่อยู่ใน เงื่อนไขดังกล่าวคือ smith,smithers,smithsonian เป็นต้น

Approximate Filters

การค้นหาข้อมูลประเภทนี้ จะเป็นการประมาณค่าของ ค่าที่เราต้องการ ซึ่งคำที่ได้อาจเป็นคำใกล้เคียง ตัวอย่างเช่น (sn~=jensen) ค่าที่ได้ก็อาจจะเป็น jenson ก็ได้

"Graeater than or Equal To" and "Less Than or Equal To" Filter

LDAP เซิร์ฟเวอร์จะสนับสนุนการใช้เครื่องหมาย <=, >=, >, <, ! ตัวอย่างเช่น (sn<=smith) ก็คือจะรีเทิร์นทุกคนที่เท่ากับ smith หรือน้อยกว่า ส่วนเครื่องหมาย ! หมายถึง not ตัวอย่างเช่น (age>21) จะมีความหมายเหมือนกับ (!(age<=21)) เป็นต้น

Presense Filters

การ filter แบบนี้มันจะแมตช์กับทุกๆ เอนทรีถ้าในเอนทรีนั้นมีอย่างน้อยหนึ่งค่า ตัวอย่างเช่น (telephjoneNumber=*) เป็นต้น

Combining Filter Terms

เป็นการให้เงื่อนไขที่เกี่ยวกับการ And กับ Or ด้วย เพราะจะมีเงื่อนไขมากกว่าหนึ่งอย่างในการร้องขอ ตัวอย่างเช่น (|(sn=Smith)(sn=Jones)) ก็หมายถึงต้องการคนที่มีชื่ออย่างใดอย่างหนึ่งก็ได้และ (&(mail=*)(!(telephoneNumber=*))) ก็หมายถึง หาคนที่มีเมล์แต่ไม่มีเบอร์โทรศัพท์ เป็นต้น

Compare Operation

โอเปอเรชันในการเปรียบเทียบนั้นจะเป็นการเทียบกันระหว่างค่าสองค่าที่อยู่ในไดเรกทอรีถ้ามีค่าเท่ากัน LDAP Server ก็จะส่งค่า true มาให้กับ ไคลเอนต์ แต่ถ้าไม่เท่ากัน มันก็จะส่งค่า false มาให้กับไคลเอนต์

3.4.3.2 การอัฟเดต (Update Operation)

ในการทำงานของโอเปอเรชันอัฟเดตนั้น จะมีอยู่ด้วยกันทั้งหมดสี่โอเปอเรชันด้วยกันคือ add, delete, rename(modify DN) และ modify ซึ่งทั้งสี่โอเปอเรชันนี้ก็จะมีข้อกำหนดในการที่จะให้เราใช้มันเพื่อให้ไปเปลี่ยนแปลงข้อมูลใน ไดเรกทอรี

Add operation

จะอนุญาตให้เราสร้างไดเรกทอรีเอนทรีใหม่ขึ้นมาได้ ซึ่งมีพารามิเตอร์อยู่สองตัวที่ต้องมีก็คือ ชื่อเอนทรีและแอตทริบิวต์ซึ่งการเพิ่มข้อมูลจะสำเร็จได้ต้องมีเงื่อนไขสี่เงื่อนไขด้วยกัน ดังนี้

- พาเรนต์ (Parent) ของ เอนทรีใหม่จะต้องมีอยู่ในไดเรกทอรี
- จะต้องไม่เป็น เอนทรีที่มีชื่อเหมือนกัน
- เอนทรีใหม่จะต้องตรวจสอบสกีมา (schema) และถูกต้อง
- การควบคุมการเข้าถึง (Access control) จะต้องอนุญาตให้มีโอเปอเรชัน

Delete operation

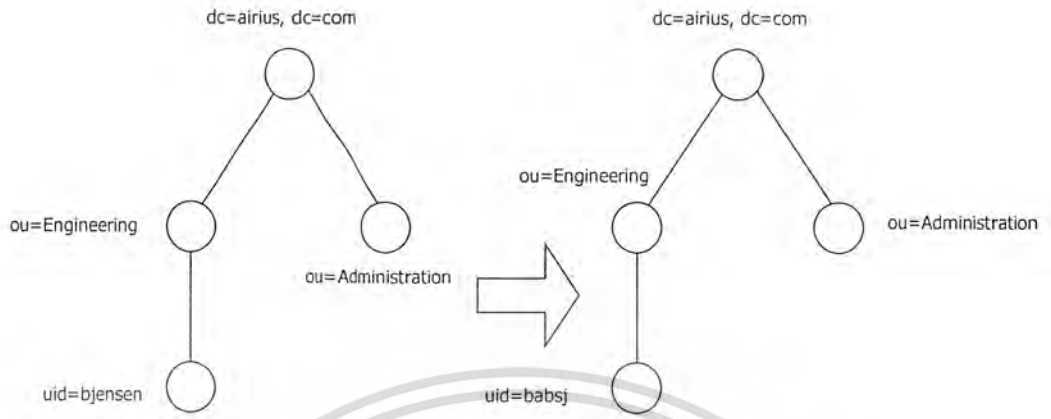
จะลบเอนทรีออกจากไดเรกทอรี ซึ่งมีพารามิเตอร์แค่ตัวเดียวเท่านั้นคือ DN ซึ่งมันจะสำเร็จได้ต้องมีเงื่อนไขดังนี้

- เอนทรีที่ต้องการลบต้องมีอยู่
- มันจะต้องไม่มีลูก (children)
- การควบคุมการเข้าถึงจะต้องอนุญาตให้ เอนทรีถูกลบ

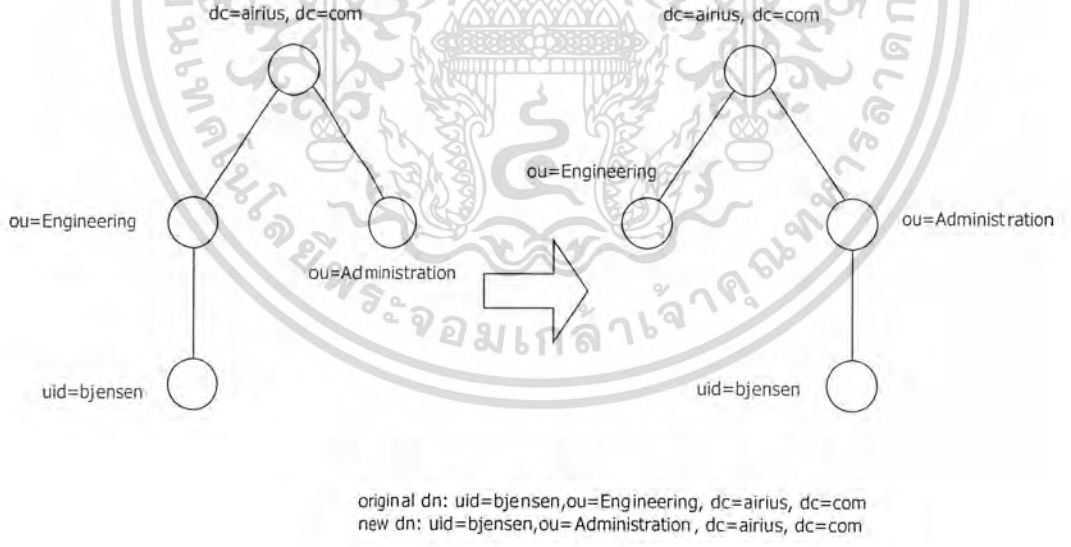
Modify DN operation

ใช้ในการเปลี่ยนแปลงชื่อ หรือย้ายเอนทรีที่อยู่ในไดเรกทอรี มีพารามิเตอร์อยู่สี่ตัวด้วยกันคือ ชื่อ DN ที่ต้องการเปลี่ยนแปลง, ชื่อ RDN ใหม่ที่ต้องการให้เปลี่ยน, ตัวเลือกที่บอกว่าสามารถให้เอนทรีนี้มีพาเรนต์ใหม่หรือไม่และdelete-old-RDN flag ซึ่งการเปลี่ยนแปลงจะสำเร็จได้จะต้องมีเงื่อนไขดังนี้

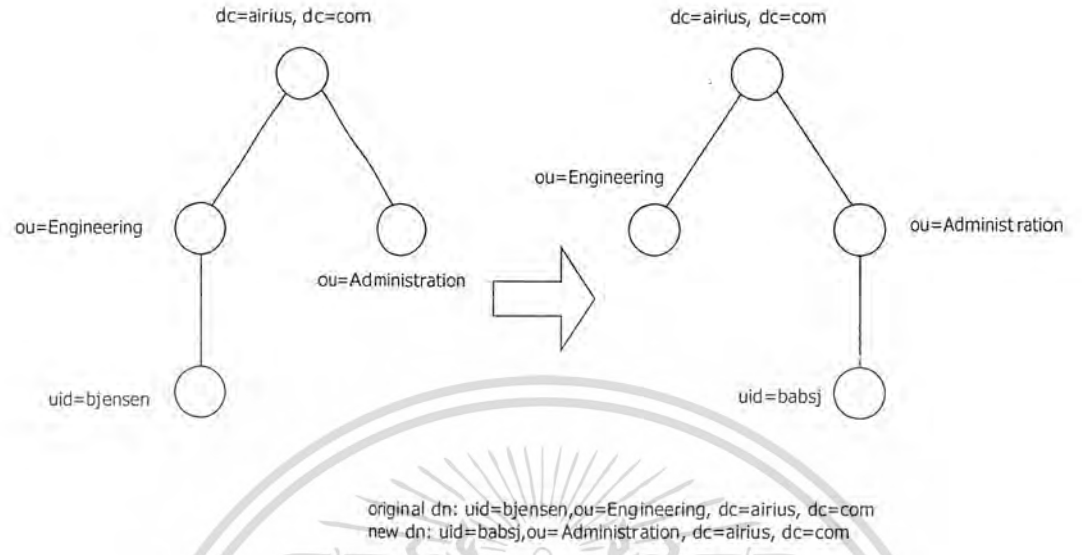
- เอนทรีที่ต้องการเปลี่ยนแปลงจะต้องมีอยู่จริง
- ชื่อใหม่ของเอนทรีจะต้องไม่ซ้ำกับ เอนทรีอื่นๆที่อยู่ภายใต้ พาร์ตเดียวกัน
- การควบคุมการเข้าถึงจะต้องอนุญาตให้มีโอเปอเรชัน



รูปที่ 3-11 การเปลี่ยนแปลงชื่อโดยไม่มีการเคลื่อนย้าย



รูปที่ 3-12 การย้ายเอนทรีและไม่ได้เปลี่ยน RDN



รูปที่ 3-13 การย้ายเอนทรี และเปลี่ยน RDN

3.4.3.3 การพิสูจน์สิทธิ์และการควบคุม (Authentication and Control Operations)

โอเปอเรชันในการออเทนทิเคท (Authenticate) นี้มีอยู่สองอย่างด้วยกันคือ bind และ unbind และมี คอนโทรลโอเปอเรชัน คือ abandon ซึ่งโอเปอเรชันทั้งหมดนี้ จะใช้ในการเชื่อมต่อ ระหว่าง LDAP client กับ LDAP server และยกเลิกการเชื่อมต่อ เมื่อทำ โอเปอเรชันอื่นๆเสร็จเรียบร้อยแล้ว

Bind Operation

จะใช้ในการออเทนทิเคทของตัวไคลเอนต์เข้าไปยังไดเรกทอรี ซึ่งจะต้องส่ง ชื่อ Distinguished Name (DN) และ Credential ซึ่งจะอยู่ในรูปของ พาสเวิร์ด และ user และเมื่อเชื่อมต่อเรียบร้อย ก็จะ success code ไปยังไคลเอนต์

Unbind Operation

ซึ่งโอเปอเรชันนี้ จะไม่มี พารามิเตอร์ที่ส่งมา เมื่อเซิร์ฟเวอร์ ได้ โอเปอเรชันนี้มา ก็จะยกเลิกการเชื่อมต่อจากไคลเอนต์ และปิด TCP connection ทันที

Abandon Operation

จะมีหนึ่งพารามิเตอร์ที่ใช้ ก็คือ message ID ของ LDAP operation โดยที่ทางไคลเอนต์จะใช้โอเปอเรชันนี้ ก็ต่อเมื่อไม่มีความสนใจ หรือต้องการยกเลิก โอเปอเรชัน ที่ได้ทำกับ ไดเรกทอรี ก่อนหน้านั้น

3.4.4 โมเดลความปลอดภัย (Security Model)

SSL and TLS

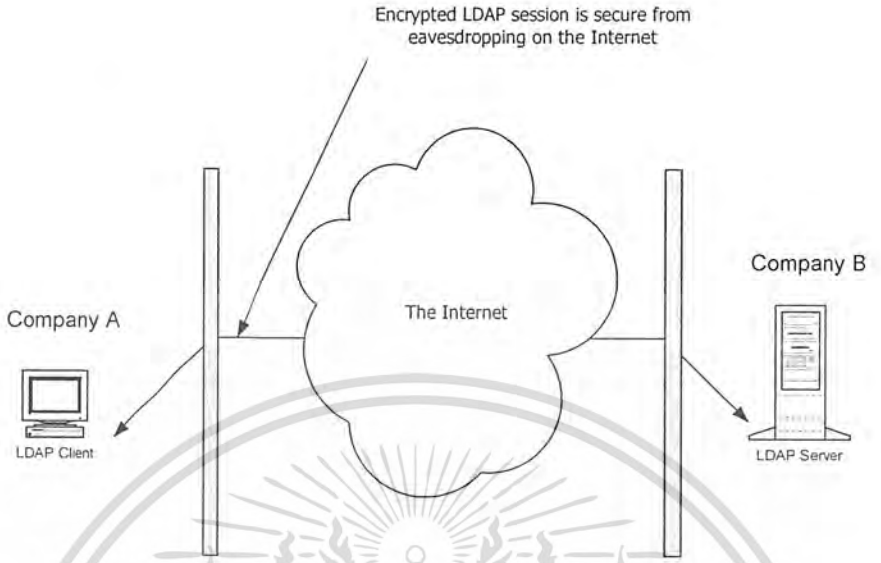
SSL และ TLS เป็นเทคโนโลยีใหม่ ของระบบความปลอดภัย ซึ่งมันจะเข้ารหัส ข้อมูลทั้งหมดก่อนที่จะส่งกัน ระหว่าง ไคลเอนต์และเซิร์ฟเวอร์ ซึ่ง SSL จะกำเนิดมาก่อน TLS เป็นเทคโนโลยีที่ใช้กันมากบน World Wide Web และทั้ง e-commerce ต่างๆ รวมทั้งทรานส์แซกชันต่างๆที่ขึ้นอยู่กับการสื่อสารข้อมูล ที่ป้องกันการดักฟัง SSL นั้นได้รับการยอมรับอย่างกว้างขวางบน world wide web ในการใช้สำหรับตรวจสอบและเข้ารหัสลับการติดต่อสื่อสารระหว่าง client และ server หน้าที่ของ SSL จะแบ่งออกเป็นสามส่วนใหญ่ๆคือ

การตรวจสอบเซิร์ฟเวอร์ว่าเป็นตัวจริง: ตัวโปรแกรมไคลเอ็นต์ที่มีขีดความสามารถในการสื่อสารแบบ SSL จะสามารถตรวจสอบเครื่องเซิร์ฟเวอร์ที่ตนกำลังจะไปเชื่อมต่อได้ว่าเซิร์ฟเวอร์นั้นเป็นเซิร์ฟเวอร์ตัวจริงหรือไม่ โดยใช้เทคนิคการเข้ารหัสแบบ public key ในการตรวจสอบใบรับรอง (certificate) และ public ID ของเซิร์ฟเวอร์นั้น หน้าที่นี้ของ SSL เป็นหน้าที่ที่สำคัญโดยเฉพาะอย่างยิ่งในกรณีที่ไคลเอ็นต์ต้องการที่จะส่งข้อมูลที่เป็นความลับ ให้กับเซิร์ฟเวอร์ซึ่งไคลเอ็นต์จะต้องตรวจสอบก่อนว่าเซิร์ฟเวอร์เป็นตัวจริงหรือไม่

การตรวจสอบว่าไคลเอ็นต์เป็นตัวจริง: เซิร์ฟเวอร์ที่มีขีดความสามารถในการสื่อสารแบบ SSL จะใช้เทคนิคเช่นเดียวกับในหัวข้อที่แล้วในการตรวจสอบไคลเอ็นต์หรือผู้ใช้ว่าเป็นตัวจริงหรือไม่ โดยจะตรวจสอบใบรับรองและ public ID (ที่มีองค์กรที่เซิร์ฟเวอร์เชื่อถือเป็นผู้ออกให้) ของไคลเอ็นต์หรือผู้ใช้นั้นหน้าที่นี้ของ SSL จะมีประโยชน์ในกรณีเช่น ธนาคารต้องการที่จะส่งข้อมูลลับทางการเงินให้แก่ลูกค้าของตนผ่านทางเครือข่ายอินเทอร์เน็ต

การเข้ารหัสลับการเชื่อมต่อ: ในกรณีนี้ ข้อมูลทั้งหมดที่ถูกส่งระหว่างไคลเอ็นต์และเซิร์ฟเวอร์จะถูกเข้ารหัสลับโดยโปรแกรมที่ส่งข้อมูลเป็นผู้เข้ารหัสและโปรแกรมที่รับข้อมูลเป็นผู้ถอดรหัส นอกจากการเข้ารหัสลับในลักษณะนี้แล้ว SSL ยังสามารถปกป้องความถูกต้องสมบูรณ์ของข้อมูลได้อีกด้วย กล่าวคือ ตัวโปรแกรมรับข้อมูลจะทราบได้หากข้อมูลถูกเปลี่ยนแปลงไปในขณะกำลังเดินทางจากผู้ส่งไปยังผู้รับ

ในส่วนของ TLS ก็จะใช้ร่วมกับ SSL ซึ่งเป็น emerging Internet standard โดยที่ LDAP เสนอมาตรฐานที่ได้กล่าวมานี้ให้กับ ไคลเอนต์ สำหรับการเข้ารหัสข้อมูลเพื่อส่ง มา และได้รับ จาก LDAP เซิร์ฟเวอร์โดยใช้ TSL ซึ่งเพียงแต่ SSL ทำการ อีนาเบิล class application บนเว็ป (web) ของมัน , TLS ก็จะอีนาเบิลให้ใช้เทคโนโลยีของไดเรกทอรีไปพร้อมๆ กัน



รูปที่ 3 -14 TLS จะอนุญาตให้ใช้ secure transmission สำหรับข้อมูลใน ไดเรกทอรีผ่านทางอินเทอร์เน็ต

นอกจากนี้มันยังอนุญาตให้มีการเข้ารหัสของข้อมูลที่มีขนาดใหญ่ที่ใช้ในการส่งระหว่างไคลเอนต์เซิร์ฟเวอร์ (Client/Server) ได้อีกด้วย ทั้งสองยังอนุญาตให้มีการออเทนทิเคท (Authenticate) โดยใช้ Strong Cryptography ได้อีกด้วย

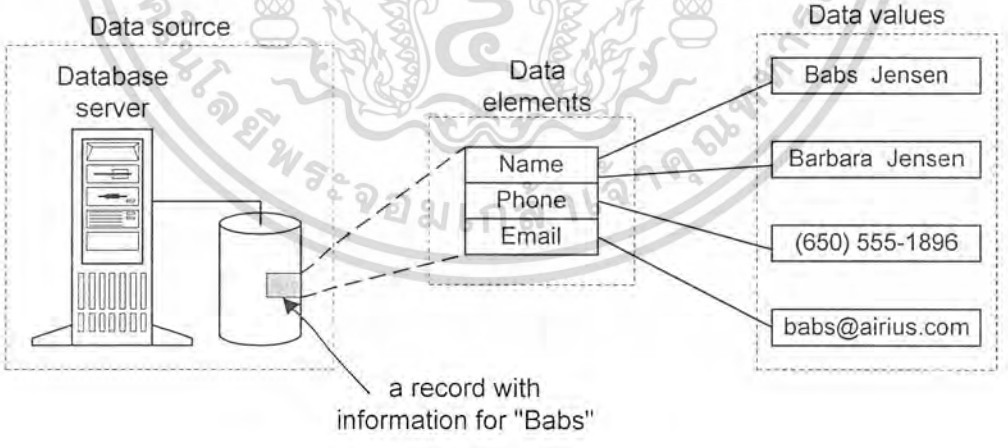
บทที่ 4
การออกแบบข้อมูล

4.1 แนะนำการออกแบบข้อมูล

เนื่องจากว่าบริการไดเรกทอรีต้องเกี่ยวข้องกับการเก็บข้อมูล ดังนั้นเราจึงต้องมีการวางแผนและออกแบบว่าข้อมูลชนิดใดที่เราต้องการที่จัดเก็บในไดเรกทอรีเพื่อความสะดวกในการใช้งานและการดูแลข้อมูล

ในเรื่องของการออกแบบข้อมูล มีองค์ประกอบที่สำคัญดังนี้ ส่วนของดาต้าหมายถึงข้อมูลข่าวสารทั้งหมดซึ่งเก็บอยู่ในบริการไดเรกทอรี ซึ่งความต้องการในการเก็บข้อมูลของแต่ละคนอาจเหมือนหรือแตกต่างกัน ส่วนของดาต้าเอเลเมนต์ หมายถึงส่วนของข้อมูลและส่วนของดาต้าซอส หมายถึงระบบซึ่งเก็บรวบรวมดา-ต้าเอเลเมนต์ซึ่งในบางครั้งเราอาจเรียกว่า เรสโพสิทอรี (respositories) ซึ่งใน LDAP นั้น ดาต้าเอเลเมนต์มีลักษณะเช่นเดียวกันกับแอททริบิวไทป์ (attribute type) ยกตัวอย่างเช่น ชื่อเต็มของคน ขนาดความจุของกระดาษของพรินท์เตอร์หรือรูปแบบของซีพียู ซึ่งสิ่งที่คอยระบุตัวแทนของดาต้าเอเลเมนต์เราเรียกว่าดาต้า

เอเลเมนต์แวลู (data element value) หรือ ดาต้าแวลู (data value) ใน LDAP เราเรียกดาต้าแวลูว่าแอททริบิวแวลู (attributevalue) ซึ่งในหัวข้อของการออกแบบข้อมูลนี้เป็นการอธิบายถึงการออกแบบดาต้าเอเลเมนต์ ในรูปที่ 4-1 แสดงให้เห็นถึงความสัมพันธ์กันระหว่างดาต้าซอส, ดาต้าเอเลเมนต์และดาต้าแวลู



รูปที่ 4-1 แสดงความสัมพันธ์กันระหว่าง ดาต้าซอส, ดาต้าเอเลเมนต์และดาต้าแวลู

ปัญหาทั่วๆไปเกี่ยวกับข้อมูล

- ข้อมูลที่มากเกินไป (ซึ่งทำให้ยากที่จะหาข้อมูลที่เราต้องการ)
- ข้อมูลที่ไม่เพียงพอ (ซึ่งทำให้ข้อมูลที่เราต้องการไม่เพียงพอ)
- ข้อมูลคุณภาพต่ำหรือข้อมูลที่ผิดพลาด (อาจทำให้เราใช้ผิดทาง)
- ข้อมูลที่หมดอายุแล้ว (เราคิดว่าได้ข้อมูลที่ใหม่แต่กลับเป็นของในอดีต)

แอปพลิเคชั่นต่างๆ ที่ใช้บริการไดเรกทอรี จะมีความเชื่อมั่นในข้อมูลที่อยู่ในบริการไดเรกทอรีก็ต่อเมื่อข้อมูลนั้นมีการปรับปรุงให้ถูกต้องอยู่เสมอ ส่วนหัวใจสำคัญของการทำความเข้าใจและหลีกเลี่ยงปัญหาเกียวกับข้อมูลนั้นก็คือการสร้างความเข้าใจที่ดีระหว่างผู้ใช้และแอปพลิเคชั่นที่ใช้งานบริการไดเรกทอรี

ปัญหาอื่นที่อาจเกิดขึ้นได้ก็คือปัญหาดาต้ารีดันแดนซี (data redundancy) ซึ่งคือการที่เรามีดาต้าเอเลเมนต์และค่าที่เก็บในดาต้าเอเลเมนต์เหมือนกันในดาต้าซอสแต่ละที่ซึ่งทำให้ต้องมีการอัพเดตข้อมูลที่อยู่ในแต่ละที่ให้เหมือนกันซึ่งค่อนข้างยุ่งยาก แนวทางในการแก้ปัญหานี้ก็คือการลดดาต้าซอสที่เก็บข้อมูลนั้นๆ หรือใช้การทำซิงโครไนซ์ (synchronize) ระหว่างข้อมูล

4.2 การกำหนดนโยบายข้อมูล

ในการวางแผนว่าควรเก็บข้อมูลชนิดใดในบริการไดเรกทอรี จำเป็นต้องมีแนวทางในการเก็บข้อมูลนั้นๆซึ่งแนวทางเหล่านี้เป็นตัวทำความเข้าใจร่วมกันระหว่างผู้ใช้ข้อมูล ผู้ออกแบบ และผู้ดูแลระบบไดเรกทอรี

การกำหนดนโยบายข้อมูลควรครอบคลุมหัวข้อต่อไปนี้

1. ต้องบอกแนวทางสำหรับการกำหนดว่าข้อมูลอะไรบ้างที่ควรเก็บหรือไม่เก็บไว้ในบริการไดเรกทอรี ตัวอย่างเช่น อาจกำหนดว่าข้อมูลที่จัดเก็บต้องเป็นข้อมูลซึ่งมีแอพพลิเคชันใช้งานร่วมกันมากกว่าหนึ่งแอพพลิเคชันขึ้นไป หรืออาจกำหนดว่าข้อมูลที่จัดเก็บ ต้องมีขนาดไม่เกิน 10K
2. ต้องบอกแนวทางการเข้าถึงข้อมูลไดเรกทอรี ซึ่งเป็นสิ่งที่สำคัญมากถ้าหากว่าข้อมูลที่เราจัดเก็บเป็นข้อมูลที่สำคัญ เราจำเป็นต้องระบุว่าจะใช้รูปแบบในการยืนยันการใช้งานแบบใดและจะใช้การเข้ารหัสข้อมูลแบบใด
3. ต้องบอกแนวทางในการปรับปรุงข้อมูลข้อมูลไดเรกทอรี คือต้องกำหนดว่าข้อมูลใดบ้างที่อนุญาตให้เปลี่ยนแปลงได้โดยต้องมีการกำหนดรูปแบบของการยืนยันการใช้งานและการเข้ารหัสข้อมูลเมื่อมีการเปลี่ยนแปลงข้อมูลด้วย
4. พิจารณาในเรื่องกฎข้อบังคับต่างๆ คือต้องระบุว่าข้อมูลที่จัดเก็บ ต้องไม่ขัดแย้งต่อกฎและข้อบังคับขององค์กร เช่นข้อมูลที่ไม่ต้องการให้ผู้อื่นรู้ก็ไม่ควรที่เก็บไว้
5. ต้องบอกแนวทางในการดูแลข้อมูลซึ่งเก็บในที่ต่างๆ ซึ่งมากกว่าหนึ่งที่ เนื่องจากในบางครั้งเราอาจจำเป็นต้องเก็บข้อมูลไว้ในที่ต่างๆดังนั้นเราจึงควรกำหนดด้วยว่าข้อมูลที่อยู่ต่างที่กันนั้นมีวิธีการและขั้นตอนในการดูแลและจัดการอย่างไร

6. ต้องบอกแนวทางสำหรับข้อยกเว้นในนโยบายที่กำหนดด้วย เนื่องจากว่าไม่มีนโยบายใดที่ครอบคลุมทุกๆอย่าง ดังนั้นเราจึงควรกำหนดข้อยกเว้นสำหรับในบางกรณีไว้ด้วย

การกำหนดนโยบายข้อมูลนั้น ต้องกำหนดให้ชัดเจนและครอบคลุมข้อมูลที่ต้องการใช้งานดังนั้นในการกำหนดนโยบายนั้น จึงจำเป็นต้องอาศัยความร่วมมือกันระหว่างกลุ่มต่างๆที่เกี่ยวข้องกับการใช้งานข้อมูล

4.3 การกำหนดดาต้าเอเลเมนต์ที่ต้องการ

การกำหนดดาต้าเอเลเมนต์ที่ต้องการก็คือการสำรวจความต้องการว่าในแต่ละแอพพลิเคชันที่ใช้งานมีข้อมูลใดบ้างที่ต้องการใช้งานจากนั้นก็ให้เขียนออกมา ซึ่งโดยส่วนใหญ่แล้วถ้าหากว่าเป็นซอฟต์แวร์ที่เป็นซอฟต์แวร์ทางการค้าซึ่งรองรับการใช้งาน LDAP มีการบอกดาต้าเอเลเมนต์ที่ต้องใช้งานมาให้ในเอกสารอยู่แล้ว แต่ถ้าเป็นซอฟต์แวร์ที่ไม่ใช่ซอฟต์แวร์ทางการค้า ในเอกสารของการออกแบบก็ต้องมีการบอกไว้เช่นกัน

Application	Vendor	Class of Information	Data Element
Authenticated Webserver	Netscape	People Groups	User ID, Password Group name,Description, list of members owner
Electronic Mail system	Netscape	People Groups	User ID,Password,email Address, mail host Vacation message,Deliveryoption, Forwarding address Group name,Description, list of members owner
Company Phone book	Developed In-house	People	Name,email address,phone numbers,user ID,password,mailing address,department, Manager,home page URL
Organizational Chart generation Utillity	Oblix	People	Manager,employee type Job title

Asset Management Applicatipn	Developed In-house	Computers	Owner,make,model, Processor,speed storage capacity,IP address,host name

ตารางที่ 4-1 แอพพลิเคชัน และดาต้าเอเลเมนต์

4.4 คุณสมบัติทั่วไปของดาต้าเอเลเมนต์

รูปแบบ (Format)

ดาต้าเอเลเมนต์สามารถจัดแบ่งเป็นกลุ่มของข้อมูลได้เช่น ชื่อของคนมักเป็นรูปแบบที่เป็นตัวอักษรในภาษาต่างๆ เบอร์โทรศัพท์มักเป็นรูปแบบของตัวเลขที่มีการแบ่งช่วงด้วยช่องว่าง หรือเครื่องหมาย "-" ตารางที่ 4-2 แสดงให้เห็นถึงบางส่วนของรูปแบบข้อมูลและตัวอย่างของดาต้าเอเลเมนต์

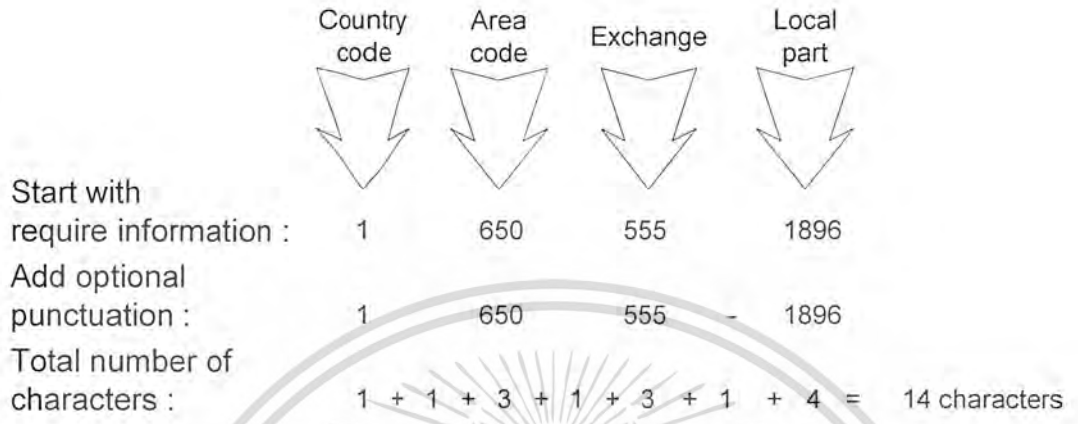
General Format Element	Common Variations	Example Data
Text string	Case sensitive,case insensitive	Person's name, Printer's name, URL
Multiline text string	Case sensitive,case insensitive	Postal address, Description
Phone number	Local,internation	Work phone number,fax number
Numeric	Integer,floating point	Employee number,cost
Multimedia	Image,sound,movie	Photograph,musical sample
Binary	(Many variations)	Digital certificate, Preferences data

ตารางที่ 4-2 แสดงรูปแบบข้อมูล

ขนาดของดาต้าแวลู (Size of data values)

ขนาดในที่นี้หมายถึงจำนวนของตัวอักษรหรือไบท์ซึ่งดาต้าเอเลเมนต์ใช้ในการเก็บข้อมูล ซึ่งขนาดที่เราประมาณไว้นั้น ช่วยทำให้เราได้ทราบว่าต้องใช้เนื้อที่ในการเก็บข้อมูลประมาณเท่าไหร่ ถึงแม้ว่าในการประมาณขนาดของดาต้าแวลูนั้นค่อนข้างทำได้ยากหากเราต้องการให้ได้ขนาดที่พอดีกับขนาด

ของดาต้าแวลู ดังนั้นเราจึงต้องกำหนดช่วงของขนาดเผื่อเอาไว้ รูปที่ 4-2 แสดงตัวอย่างของการประมาณขนาดของหมายเลขโทรศัพท์ที่ใช้ในระบบโทรศัพท์ของอเมริกาเหนือ



รูปที่ 4-2 ขนาดของหมายเลขโทรศัพท์

จำนวนของค่าข้อมูลที่เก็บ (Number of occurrences)

จำนวนของข้อมูลในหนึ่งดาต้าเอเลเมนต์ว่าจะเก็บข้อมูลไว้เท่าใด เช่น ดาต้าเอเลเมนต์ ของบุคคลซึ่งมักมีค่า user ID เอาไว้ซึ่งค่า user ID นี้ในหนึ่งคนก็จะมีเพียงแค่ค่าเดียว แต่ถ้าเป็นหมายเลขโทรศัพท์อาจมีมากกว่าหนึ่งหมายเลขได้ เพราะฉนั้นข้อมูลในส่วนนี้ก็จะช่วยในการวางแผนในเรื่องของขนาดพื้นที่ที่ใช้ในการเก็บข้อมูลด้วย

เจ้าของข้อมูล (Data Ownership)

ความเป็นเจ้าของข้อมูลในเรื่องของการเข้าถึงข้อมูล,ความปลอดภัย,ความเป็นส่วนตัว รวมสิทธิในการเปลี่ยนแปลงแก้ไขข้อมูล

ผู้ที่ใช้งาน (Consumers)

ได้แก่แอพพลิเคชันที่ใช้งานดาต้าเอเลเมนต์นั้น

รูปแบบข้อมูลแบบไดนามิก (Dynamic) หรือ สแตติก (static)

ข้อมูลแบบไดนามิก (dynamic) คือข้อมูลที่มีการเปลี่ยนแปลง ส่วนข้อมูลแบบสแตติก (static) คือข้อมูลที่ไม่มีการเปลี่ยนแปลง ซึ่งข้อมูลในส่วนนี้จะเป็นส่วนสำคัญในการออกแบบโทโพโลยี (topology) ของ

ไดเรกทอรีว่าจะใช้แบบใด

ความสัมพันธ์กับดาต้าเอเลเมนต์อื่น (Relationship with other data elements)

ความสัมพันธ์กันของดาต้าเอเลเมนต์เป็นตัวช่วยในการจัดกลุ่มของดาต้าเอเลเมนต์ที่เกี่ยวข้องกันไว้เป็นกลุ่มเดียวกัน

บทที่ 5
การออกแบบสกีมา

5.1 ความหมายของสกีมา

สกีมา ก็คือกลุ่มของกฎเกณฑ์ที่กำหนดว่ามีข้อมูลอะไรบ้างที่สามารถเก็บไว้ในฐานข้อมูล (Database) หรือไดเรกทอรีได้ โดยสกีมาค่อนข้างมีความสำคัญเพราะมันช่วยให้เรารักษาความเป็นอันหนึ่งอันเดียวกัน รวมถึงเป็นตัวที่คอยจัดการดูแลในเรื่องของความถูกต้องและคุณภาพของข้อมูลที่เก็บอยู่ในไดเรกทอรี

นอกจากนั้นสกีมายังช่วยลดการเกิดสำเนาของข้อมูลรวมถึงการคาดเดาทางที่จะเข้าถึงและเปลี่ยนแปลงกลุ่มของไดเรกทอรีออบเจกต์ (Directory Objects) ให้กับไดเรกทอรีเอ็มเบ็ทเด็ดแอพพลิเคชัน (Directory-Enabled Application) ได้

5.2 จุดประสงค์ของสกีมา

จากที่ได้กล่าวไปแล้วว่าสกีมาคือเซ็ตของกฎที่คอยกำหนดว่าอะไรบ้างที่สามารถเก็บลงในไดเรกทอรี และไดเรกทอรีเซิร์ฟเวอร์กับไดเรกทอรีไคลเอนจะทำงานกับข้อมูลได้อย่างไรในตอนที่ไดเรกทอรีเซิร์ฟเวอร์จะเก็บข้อมูลหรือเปลี่ยนแปลงข้อมูลของเอนทรีมันจะตรวจสอบคอนเทนท์ (content) ของเอนทรีกับสกีมารูล (schema rule) ก่อนว่าถูกต้องหรือไม่และในการเปรียบเทียบข้อมูลระหว่างแอตทริบิวต์มันจะใช้สกีมาในการหาว่าจะใช้ อัลกอลิทึม (algorithm) ใดในการเปรียบเทียบ

สกีมายังทำหน้าที่ในการกำหนดขนาด, ขอบเขตและรูปแบบของข้อมูลที่เก็บลงในไดเรกทอรี ตัวอย่างเช่น ที่อยู่อีเมล์ซึ่งข้อมูลของที่อยู่อีเมล์ ประกอบไปด้วยเซตของตัวอักษรและมีรูปแบบเฉพาะ เช่น *addr@domain* ในบางสกีมาก็มีการกำหนดกฎอย่างง่ายๆ ไว้ด้วยเช่น แวลูต้องเป็นจำนวนเต็ม สุดท้ายสกีมายังช่วยกำหนดในเรื่องของความปลอดภัยของการเข้าถึงข้อมูลอีกด้วย

5.3 ส่วนประกอบของ LDAP Schema

ใน LDAP ตัวสกีมาประกอบไปด้วย แอตทริบิวต์ไทป์ (attribute type), แอตทริบิวต์ซินแท็ก (attribute syntaxes) และออบเจ็กต์คลาส (object class)

5.3.1 แอตทริบิวต์

ในไดเรกทอรีเอนทรีประกอบไปด้วยจำนวนของแอตทริบิวต์ไทป์และแวลู ซึ่งตัวแอตทริบิวต์ไทป์นั้นเป็นส่วนที่บอกลักษณะของข้อมูลเช่น ชื่อคน, เบอร์โทร เป็นต้น ใน LDAP มีข้อกำหนดของแอตทริบิวต์-ไทป์ดังนี้

- ชื่อที่ใช้ระบุแอตทริบิวต์ไทป์ต้องไม่ซ้ำกัน
- ออปเจกต์ไอเดนติไฟเออร์ (object identifier (OID)) ต้องไม่ซ้ำกัน

- ต้องมีตัวบอกว่า แอตทริบิวต์แวลู (attribute value) จะเป็นแบบซิงเกิลแวลู (single value) หรือเป็นแบบ มัลติแวลู (multi valued)
- แอตทริบิวต์ซินแท็กและเซ็ตของแมทซ์ชิ่งรูล (matching rules) ต้องมีความสัมพันธ์กัน
- การใช้อินดิเคเตอร์ (indicator) สำหรับแอพพลิเคชัน (application) หรือ โอเปอเรชั่น (operation) ของบริการไดเรกทอรี
- ข้อจำกัดของขอบเขต หรือ ขนาดของแวลูที่เก็บในแอตทริบิวต์

ในส่วนของชื่อแอตทริบิวต์ (attribute name) จะใช้ชื่อที่สั้นและค่อนข้างเข้าใจง่ายมีคุณสมบัติดังนี้

- ชื่อแอตทริบิวต์จะไม่มีความแตกกันระหว่างตัวอักษรตัวใหญ่กับตัวเล็ก เช่น *CN* กับ *cn* จะเหมือนกัน
- ตัวอักษรที่ใช้ต้องเป็น ASCII และตัว "-" ซึ่ง ชื่อของแอตทริบิวต์ต้องขึ้นต้นด้วยตัวอักษรภาษาอังกฤษ
- ชื่อแอตทริบิวต์ต้องไม่ซ้ำกัน

ตัวอย่างของชื่อแอตทริบิวต์ที่ถูกต้องเช่น cn,telephoneNumber,postalAddress,one-way และตัวอย่างของชื่อแอตทริบิวต์ที่ไม่ถูกต้องได้แก่ last#,2for2,my.boss และ favorite_drink ในบางมาตรฐานแอตทริบิวต์ อาจมีชื่อทั้งแบบยาวและแบบสั้น (commonName,cn) แต่ส่วนใหญ่นิยมใช้แบบสั้น

ในส่วนของ OID (object identifier) เป็นตัวเลขเฉพาะที่ไม่ซ้ำกันส่วนใหญ่จะเขียนในรูปแบบของเลขจำนวนเต็มโดยใช้จุดในการคั่นระหว่างตัวเลข เช่น OID ของ potalAddress คือ 2.5.4.16 OID เป็นสิ่งที่จำเป็นถ้า นำไปใช้กับ X.500 directory เพราะว่ามันเป็นตัวที่ใช้ในการระบุแอตทริบิวต์ไทป์ ถึงแม้ว่า LDAP ไคลเอ็นต์และเซิร์ฟเวอร์สามารถใช้ OID ในส่วนของชื่อแอตทริบิวต์ได้ก็ตามแต่ในทางปฏิบัติชื่อแอตทริบิวต์เพียงอย่างเดียวก็ยังเป็นที่นิยมใช้มากกว่าเพราะว่ามันง่ายในการใช้งานมากกว่า OID นั่นเอง

ในเอนทรีที่ถูกเก็บในไดเรกทอรีนั้น ตัวแอตทริบิวต์จะมีแอตทริบิวต์แวลูหนึ่งค่าหรือมากกว่าค่าที่มันเก็บไว้นั้นโดยตัวมันเองแล้วจะไม่มีความหมายอะไรถ้าเราดูแต่เพียงแค่ค่านั้นเช่นค่า ตัวเลข 12 ถ้าเราดูแต่ตัวเลขจะไม่รู้ว่าหมายความว่าอย่างไรแต่ถ้าเรารู้ว่ามันเป็นค่าของ pagePerMinute เราก็จะรู้ว่ามันหมายความว่าอย่างไร ซึ่งในที่นี้ก็คือจำนวนหน้าที่พิมพ์ได้ต่อนาทีของพรินเตอร์นั่นเองดังนั้นแอตทริบิวต์ไทป์เป็นตัวบอกความหมายของค่านั้นๆ ในข้อกำหนดของแอตทริบิวต์ต้องมีการระบุด้วยว่าเป็นแอตทริบิวต์ไทป์แบบซิงเกิล-
แวลูหรือเป็นแบบมัลติแวลู

แอตทริบิวต์ยูสเซสอินดิเคเตอร์ (attribute usage indicator) มักเว้นไว้ในการกำหนดข้อกำหนดของ แอตทริบิวต์ เนื่องจากว่ามันจะถูกดีฟอลท์ (default) โดยยูสเซอร์แอพพลิเคชัน (user application) อยู่แล้ว ซึ่งก็หมายความว่าตัวแอพพลิเคชันสามารถกำหนดความต้องการในการใช้งานได้เอง ในกรณีที่แอตท

ริบิวต์ไทป์เป็นแอตทริบิวต์ไทป์ของระบบ จะมีการกำหนดในส่วนของแอตทริบิวต์ยูสเซสอินดิเคเตอร์เป็นโอเปอเรชั่น-นอล (operational)

โอเปอเรชั่นนอลแอตทริบิวต์ (operational attribute) มักถูกใช้โดยตัวของบริการไดเรกทอรีเพื่อใช้สำหรับงาน ดูแลระบบหรืองานที่เกี่ยวข้องกับระบบ แอตทริบิวต์เหล่านี้จะถูกดูแลจัดการโดย ไดเรกทอรีเซิร์ฟเวอร์และแอตทริบิวต์ไม่สามารถมองเห็นได้จากไดเรกทอรีไคลเอนต์ นอกจากว่ามีการร้องขอ ส่วนใหญ่แล้ว โอเปอเรชั่นนอลแอตทริบิวต์มักไม่สามารถเปลี่ยนแปลงได้โดยไคลเอนท์ตัวอย่างของ โอเปอเรชั่นนอลแอตทริบิวต์บางตัวดูได้จากตาราง 4-1

Attribute Name	Where Found	Description
ModifyTimeStemp	Any enrtry	Date/time an entry was last modified
ModifiersName	Any entry	Name(DN) of entry that made the last modification
NameingContexts	LDAPv3 root DSE	Partition of the directory held in a server
SupportedLDAPVersion	LDAPv3 root DSE	Version of the LDAP protocol supported by server
aci	Any entry	Access control information (Netscape-specific)

ตารางที่ 5-1 ตัวอย่างของโอเปอเรชั่นนอลแอตทริบิวต์

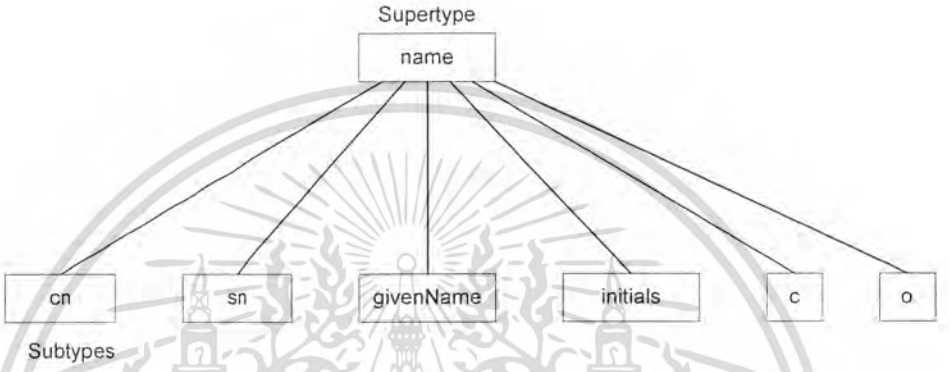
โอเปอเรชั่นนอลแอตทริบิวต์จะไม่ส่งค่ากลับเมื่อไคลเอนต์ร้องขอข้อมูลจากแอตทริบิวต์แบบปกติ แต่ถ้าไคล์เอนท์ต้องการก็สามารถที่รับข้อมูลของอเปอเรชั่นนอลแอตทริบิวต์ได้ โดยการอ้างถึงชื่อของ
โอเปอเรชั่นนอลแอตทริบิวต์ที่ต้องการ

ลำดับขั้นของแอตทริบิวต์ (Attribute Hierarchies)

ในการใช้งาน LDAP บางงานโดยเฉพาะงานที่มีลักษณะใกล้เคียงกับ มาตรฐาน X.500 มันจะรองรับ แอตทริบิวต์ไทป์ ซึ่งสับไทป์ (subtype) มักใช้กำหนดลำดับขั้นของแอตทริบิวต์ โดยทั่วไปแล้วจะใช้เป็นแบบในการกำหนดไทป์ (type) ตัวอย่างเช่น X.500 กำหนดให้ แอตทริบิวต์ cn เป็นซับไทป์ของแอตทริบิวต์ไทป์ name ในทำนองเดียวกัน แอตทริบิวต์ไทป์ sn ก็เป็นซับไทป์ของ name เช่นเดียวกัน แอตทริบิวต์ไทป์ name

เราเรียกว่าเป็นซุปเปอร์ไทป์ (supertype) ของ cn และ cn เราเรียกว่าเป็น ตัวที่ถูกดีไรฟด์จาก (derived from) name แอตทริบิวต์ที่เป็นสับไทป์จะสืบทอดคุณสมบัติของซุปเปอร์ไทป์ (supertype) มาและสามารถใช้ในการค้นหาและรับค่าได้เหมือนกับ แอตทริบิวต์ที่มันดีไรฟด์มา ตัวอย่างเช่น cn, sn ซึ่งเป็น สับไทป์ของ name ในการค้นหา ถ้าเราสั่งให้ค้นหาแบบหาค่าทั้งหมดสำหรับแอตทริบิวต์ที่มีรูปแบบ name มันจะ คืนค่าทั้งหมดมาคือ name, cn, sn และสับไทป์อื่นๆ ของ name ด้วย รูปที่ 4-1 แสดงลำดับขั้นของแอตทริบิวต์ของ

แอตทริบิวต์ไทป์ name



รูปที่ 5-1 แสดงลำดับขั้นของแอตทริบิวต์ของ แอตทริบิวต์ไทป์ name

ถึงแม้ว่าแอตทริบิวต์ซับไทป์ (attribute subtype) จะเป็นที่หน้าสนใจ แต่ว่ามันก็เป็นส่วนที่ทำให้สับสน ดังนั้นในการใช้งาน LDAP ส่วนใหญ่แล้วไม่นิยมใช้ แอตทริบิวต์ซับไทป์

แอตทริบิวต์ซินแท็กและแมตชิ่งรูล (Attribute Syntaxes and Matching Rules)

ในแต่ละแอตทริบิวต์ไทป์จะเกี่ยวข้องกับแอตทริบิวต์ซินแท็กซึ่งเป็นตัวกำหนดว่ารูปแบบของดาต้าแวลูจะเป็นรูปแบบใดและเป็นตัวบอกด้วยว่าในการเปรียบเทียบต้องเป็นแบบใด ซึ่งใน LDAP ส่วนใหญ่แล้วดาต้าแวลู มักเป็นในรูปแบบของสตริงเป็นส่วนใหญ่ (ไม่เหมือนกับ X.500) ส่วนกฎที่ใช้ในการเปรียบเทียบค่าของแอตทริบิวต์เราเรียกว่าแมตชิ่งรูล รายละเอียดของแอตทริบิวต์ซินแท็กและแมตชิ่งรูล ดูได้จากเอกสาร RFC 2252

5.3.2 ออบเจกต์คลาส

ออบเจกต์คลาสใน LDAP ก็คือกลุ่มของข้อมูลที่มีความสัมพันธ์กันซึ่งรูปแบบจะคล้ายๆ กับออบเจกต์ ในความเป็นจริงเช่น คน, พรินเตอร์หรือ อุปกรณ์เน็ตเวิร์ก ต่างๆ ในแต่ละเอนทรีจะอยู่ใน ออบเจกต์คลาสหนึ่งคลาสหรือมากว่า ซึ่งออปเจกต์คลาสจะเกี่ยวข้องกับสิ่งเหล่านี้

- การกำหนดว่าแอตทริบิวต์ไทป์ใดที่จำเป็นต้องมีในเอนทรี
- การกำหนดว่าแอตทริบิวต์ไทป์ใดที่มีหรือไม่มีก็ได้ในเอนทรี
- จัดหาวิธีการที่สะดวกสำหรับไดเรกทอรีไคลเอนต์ ในการค้นหาซับเซ็ตของเอนทรี

ตัวอย่างเช่น ออปเจกต์คลาสที่ถูกออกแบบมาให้เก็บข้อมูลเกี่ยวกับคน จำเป็นต้องมีชื่อและข้อมูลอื่นๆที่เกี่ยวข้องกับคนๆ นั้นเป็นแบบออปชั่นนอลแอตทริบิวต์ ถ้าเกิดว่าออปเจกต์คลาสนี้ชื่อว่า person ในการค้นหาข้อมูลของไดเรกทอรีไคลเอนต์ สามารถหาข้อมูลเฉพาะเอนทรีที่เกี่ยวข้องกับคนได้ โดยการเพิ่มฟิลเตอร์ ในการค้นหาโดยระบุว่า *objectclass=person*

ในการกำหนดออปเจกต์คลาส ใน LDAP ต้องประกอบด้วย

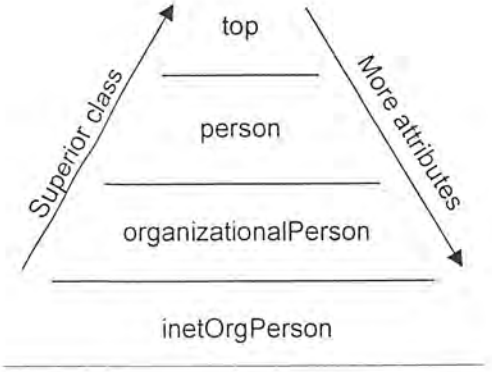
- ชื่อของออปเจกต์คลาส ซึ่งต้องไม่ซ้ำกันในแต่ละคลาส
- OID ซึ่งต้องไม่ซ้ำกันในแต่ละคลาส
- เซ็ตของแอตทริบิวต์ไทป์ที่จำเป็นต้องมี
- เซ็ตของแอตทริบิวต์ไทป์ที่อนุญาตให้มี
- ชนิด (structural,auxiliary,หรือ abstract)

ตัวอย่างของออปเจกต์คลาสแสดงให้เห็นในรูปที่ 5-2

รูปที่ 5-2 ตัวอย่างของออบเจกต์คลาส person

การสืบทอดของออบเจกต์คลาส (object class inheritance)

ออบเจกต์คลาส สามารถสืบทอดคุณสมบัติจากออบเจกต์คลาสหนึ่งไปสู่อีกออบเจกต์คลาสหนึ่งได้โดยที่ออบเจกต์คลาส ที่สืบทอดมานั้นมีคุณสมบัติเหมือนกับออบเจกต์คลาสต้นแบบทุกอย่าง คือมีแอตทริบิวต์ ที่ต้องการและแอตทริบิวต์ที่เป็นออปชั่นเหมือนกันและยังสามารถเพิ่มแอตทริบิวต์ในส่วนที่ต้องการเพิ่มเข้าไปได้ด้วย ออบเจกต์คลาสที่เป็นต้นแบบเราเรียกว่าเป็น ซุพีเรียคลาส (superior class) หรือ ซุปเปอร์คลาส (superclass) รูปที่ 5-3 แสดงตัวอย่างการสืบทอดของออบเจกต์คลาส



รูปที่ 5-3 แสดงตัวอย่างการสืบทอดของออบเจกต์คลาส

5.4 รูปแบบของไดเรกทอรีสกีมา (Directory Schema Format)

รูปแบบของไดเรกทอรีสกีมานั้นที่ใช้อยู่ในปัจจุบันมีอยู่ด้วยกัน 3 รูปแบบ คือ slapd.conf, ASN.1, LDAPv3 ซึ่งในที่นี้จะขอกล่าวถึงเพียง 2 แบบ คือ แบบ slapd.conf และ แบบ LDAPv3 เนื่องจากว่าแบบ ASN.1 นั้นไม่ค่อยเป็นที่นิยมในการเขียนสกีมาเนื่องจากค่อนข้างยากต่อการทำความเข้าใจ

<u>การกำหนดสกีมา แบบ slapd.conf</u>

รูปแบบนี้เป็นการเขียนลงในไฟล์ slapd.conf ซึ่งเป็นคอนฟิกกูเรชั่นไฟล์ (configuration file) ของ LDAP การเขียนในรูปแบบ slapd.conf นี้เป็นการเขียนสกีมาแบบที่ง่ายที่สุดและเข้าใจง่ายมากที่สุดและเป็นแบบที่ใช้ใน Netscape Directory Server, U-M LDAP

รูปแบบการกำหนดแอตทริบิวต์ไทป์มีรูปแบบดังนี้

```
attribute NAME [ALIASES] [OID] SYNTAXID [OPTIONS]
```

รูปที่ 5-4 แสดงรูปแบบการกำหนดแอตทริบิวต์ไทป์แบบ slapd.conf

NAME คือ ชื่อของแอตทริบิวต์ไทป์ เช่น cn

ALIASES คือ ชื่อของแอตทริบิวต์ไทป์ชื่ออื่น เช่น commonName

OID คือ หมายเลขของออปเจกต์ของแอตทริบิวต์ไทป์

SYNTAXID คือ ตัวแสดงรูปแบบของซินแท็ก

OPTION คือ เป็นส่วนเพิ่มเติมที่บอกว่าแอตทริบิวต์นั้นเป็นแบบโอเปอเรชั่นนอล แอตทริบิวต์หรือไม่และเป็นแบบซิงเกิลหรือไม่ ถ้าเรากำหนดว่าเป็นโอเปอเรชั่นนอลแอตทริบิวต์ ก็หมายความว่าแอตทริบิวต์นั้นถูกใช้งานโดยตัวบริการไดเรกทอรี ถ้าไม่กำหนดก็จะดีฟอลท์เป็นแอตทริบิวต์แบบที่ใช้งานธรรมดา แต่ถ้าเรากำหนดเป็นซิงเกิล หมายความว่าแอตทริบิวต์นั้นมี

ค่าที่เก็บได้เพียงค่าเดียวและถ้าเราไม่ได้กำหนดก็จะดีฟอลท์ว่าแอตทริบิวต์นั้นเป็นแบบมัลติแวลู คือสามารถเก็บค่าได้หลายค่า

รูปแบบของซินแท็กที่ใช้ในการกำหนดแอตทริบิวต์ไทป์แบบ slapd.conf ซึ่งเป็นรูปแบบที่ใช้ใน Netscape Directory Server 4.0,U-M LDAP 3.3 แสดงให้เห็นในตารางที่ 5-2

Slapd.conf Syntax ID	X.500 Equivalent	Description
Cis	DirectoryString; CaseIgnoreMatch	Text string; case of letters and leading trailing,and multiple spaces are ignored during comparison.
Ces	DirectoryString; CaseExactMatch	Text string; case of letters is significant during comparison; leading trailing and multiple spaces are ignored.
Tel	PrintableString; TelephoneNumberMatch	Text string that represents A phone number; like cis except space and hyphen characters are also ignored during comparisons.
Int	Integer; IntegerMatch	Integer number;comparisons follow rules for comparing integers.
Dn	DistinguishedName; DistinguishedNameMatch	Directory names (a pointer to another entry); comparisons follow special rules for comparing distinguished names (DNs).
Bin	OctetString OctetStringMatch	Arbitrary binary data; byte-by-byte comparisons.

ตารางที่ 5-2 รูปแบบของซินแท็กที่ใช้ในการกำหนดแอตทริบิวต์ไทป์แบบ slapd.conf

ตัวอย่างการกำหนดแอทริบิวต์ไทป์ (attribute type)

```
attribute cn commonName 2.5.4.3 cis
attribute labeledURI 1.3.6.1.4.1.250.1.57 ces
attribute seeAlso 2.5.4.34 dn
attribute supportedLDAPVersion 1.3.6.1.4.1.1466.101.120.15 int
attribute ntUserHomeDir 1.2.840.113556.1.4.44 cis single
attribute passwordExpirationTime 2.16.840.1.113730.3.1.91 cis operational
```

รูปที่ 5-5 แสดงตัวอย่างการกำหนดแอทริบิวต์ไทป์

รูปแบบการกำหนดออบเจกต์คลาส (Object Class Definition) มีรูปแบบดังนี้

objectclass *NAME* [oid *OID*] [superior *SUP*] [requires *REQATTRS*] [allows *ALLOWATTRS*]

รูปที่ 5-6 แสดงรูปแบบการกำหนดออบเจกต์คลาสแบบ slapd.conf

NAME คือ ชื่อของออบเจกต์คลาส

OID คือ ออบเจกต์ไอเดนติไฟเออร์ (object identifier) ของคลาส

SUP คือ ซุพีเรีย (superior) ของคลาส

REQATTRS คือ แอตทริบิวต์ที่ต้องมี

ALLOWATTRS คือ แอตทริบิวต์ที่มีหรือไม่มีก็ได้

**แอตทริบิวต์ไทป์ที่ใช้ใน REQATTRS และ ALLOWATTRS ต้องถูกกำหนดไว้ก่อนหน้าแล้ว

ตัวอย่างของการกำหนดออบเจกต์คลาส

```
objectclass country
    oid 2.5.6.2
    superior top
    requires
        c
    allows
        serchGuide,
        description
```

รูปที่ 5-7 แสดงตัวอย่างของการกำหนดออบเจกต์คลาส

จากตัวอย่างเป็นการกำหนดออบเจกต์คลาสที่ชื่อว่า country ซึ่งเป็นออบเจกต์คลาสของข้อมูลประเทศโดยกำหนดว่าเป็นออบเจกต์คลาสที่สืบทอดมาจากออบเจกต์คลาสที่ชื่อ top มี c (ชื่อประเทศ) เป็นแอตทริบิวต์ที่จำเป็นต้องมี โดยมี searchGuide และ description เป็นแอตทริบิวต์ที่จะมีหรือไม่มีก็ได้

การกำหนดสกีมาโดยใช้รูปแบบ slapd.conf นี้ถึงแม้จะมีข้อด้อยในเรื่องของซอฟต์แวร์ที่รองรับการใช้งานที่มีน้อย (มีเพียงซอฟต์แวร์ที่สืบทอดมาจาก U-M LDAP เท่านั้น) แต่ว่ารูปแบบนี้เป็นรูปแบบที่เข้าง่ายที่สุดและเวลากำหนดสกีมาก็ทำได้ง่ายไม่ยุ่งยาก

<u>การกำหนดสกีมา (schame) แบบ LDAPv3</u>

รูปแบบการกำหนดแอตทริบิวต์ไทป์ (Attribute type Definition) มีรูปแบบดังนี้

```
(ATOID NAME ATNAME
        [ DESC ATDESC ]
        [ SUP SUPOID ]
        [ EQUALITY EQMATCHOID ]
        [ ORDERING ORDMATCHOID ]
        [ SUBSTR SUBMATCHOID ]
        [ SYNTAX SYNOID ]
        [ SINGLE-VALUE ]
)
```

รูปที่ 5-8แสดงการกำหนดแอตทริบิวต์ไทป์แบบ LDAPv3

ATOID คือ ออบเจกต์ไอเดนติไฟเออร์ของแอตทริบิวต์

ATNAME คือ ชื่อของแอตทริบิวต์

ATDESC คือ คำอธิบาย

SUPOID คือ ออบเจกต์ไอเดนติไฟเออร์ของแอตทริบิวต์ที่สืบทอดมา

EQMATCHOID,ORDMATCHOID,SUBMATCHOID คือ แมทช์ชิ่งรูล

SYNOID คือ ออปเจกต์ไอเดนติไฟเออร์ของแอตทริบิวต์ซินแท็ก

SINGLE-VALUE คือ ตัวที่บอกว่าแอตทริบิวต์นี้สามารถเก็บค่าได้เพียงค่าเดียว ถ้าไม่ใส่บรรทัดนี้จะมีค่าดีฟอลท์เป็นมัลติแวลู คือสามารถเก็บค่าได้หลายค่า

แมทช์ชิ่งรูล

คือ สิ่งที่คอยระบุฟิลเตอร์ในการค้นหาแล้วเปรียบเทียบข้อมูลในเอนทรี โดยจะมีฟิเตอร์ทั้งหมด 3 รูปแบบด้วยกันคือ

- อีควอลิตี้ฟิลเตอร์ (Equality filter)

- ออร์เดอร์ริ่งฟิลเตอร์ (Ordering filter)
- สับสตริงฟิลเตอร์ (Substring filter)

แอตทริบิวต์ซินแท็ก

คือสิ่งที่กำหนดรูปแบบของข้อมูลของแอตทริบิวต์ ว่าจะมีรูปแบบของข้อมูลเป็นแบบใด ซึ่งใน LDAPv3

LDAPv3 Name	OID	General name
Directory String	1.3.6.1.4.1.1466.115.121.1.15	CaseIgnoreString
IA5 String	1.3.6.1.4.1.1466.115.121.1.26	CaseExactString
Integer	1.3.6.1.4.1.1466.115.121.1.27	Integer
Boolean	1.3.6.1.4.1.1466.115.121.1.7	Boolean

ตารางที่ 5-3 แสดงแอตทริบิวต์ซินแท็กที่เลือกใช้งานในโครงงาน

แอตทริบิวต์ซินแท็กแต่ละตัวจะต้องมีการกำหนดแมทช์ชิ่งรูลสำหรับแต่ละฟิเตอร์ที่ต้องการจะใช้งาน ในการค้นหาและเปรียบเทียบเอนทรี ซึ่งแต่ละตัวก็จะมีแมทช์ชิ่งรูลที่แตกต่างกัน

Attribute Syntax	Filter	Matching Rules
Directory String	Equality	CaseIgnoreMatch
	Ordering	CaseIgnoreOrderingMatch
	Substring	CaseIgnoreSubstringsMatch
IA5 String	Equality	CaseExactIA5Match
	Ordering	CaseIgnoreOrderingMatch
	Substring	CaseIgnoreSubstringsMatch
Integer	Equality	IntegerMatch
	Ordering	CaseIgnoreOrderingMatch
	Substring	CaseIgnoreSubstringsMatch
Boolean	Equality	CaseExactIA5Match

ตารางที่ 5.4 แสดงแอตทริบิวต์ซินแท็กและแมทช์ชิ่งรูลที่ใช้ในโครงงาน

ตัวอย่างของการกำหนดแอตทริบิวต์ไทป์ (attribute type)

```
attributetype ( 1.3.6.1.4.4.1 NAME 'AgentType'
        EQUALITY caseIgnoreMatch
        ORDERING caseIgnoreOrderingMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
```

รูปที่ 5-9 แสดงตัวอย่างของการกำหนดแอตทริบิวต์ไทป์

รูปแบบการกำหนดออปเจกต์คลาส (Object Class Definition) มีรูปแบบดังนี้

```
(OCOID NAME OCNAME
        [ DESC  OCDESC ]
        [ SUP  SUPOID ]
        [ OCKIND ]
        [ MUST REQATSET ]
        [ MAY  ALLOWATSET ]
)
```

รูปที่ 5-10 แสดงรูปแบบการกำหนดออบเจกต์คลาสแบบ LDAPv3

OCOID คือ ออบเจกต์ไอเดนติไฟเออร์ของออบเจกต์คลา

OCNAME คือ ชื่อของออบเจกต์คลาส

OCDESC คือ คำอธิบาย

OCKIND คือชนิดของออบเจกต์คลาส ABSTRACT, STRUCTURAL, AUXILIARY ค่าดีฟอลท์ คือ STRUCTURAL

REQATSET คือ ชื่อหรือออบเจกต์ไอเดนติไฟเออร์ของแอตทริบิวต์ที่จำเป็นต้องมี

ALLOWATSET คือ ชื่อหรือออบเจกต์ไอเดนติไฟเออร์ของแอตทริบิวต์ (attribute) ที่มีหรือไม่มีก็ได้

ตัวอย่างของการกำหนดออบเจกต์คลาส

```
objectclass (1.3.6.1.4.3.8 NAME 'FirewallRule'
        SUP top STRUCTURAL
        MUST (Rule$Source$Destination$Service$Action)
        MAY description)
```

รูปที่ 5-11 แสดงตัวอย่างของการกำหนดออบเจกต์คลาส

การกำหนดสกีมาแบบ LDAPv3 นี้จะเห็นได้ว่าค่อนข้างยุ่งยากกว่าการกำหนดแบบ slapd.conf มากแต่จะมีความสารถมารถในการใช้งานที่ค่อนข้างหลากหลายกว่าเช่น สามารถกำหนดรูปแบบของแมทซ์ชิ่งรูล ได้หลากหลายรูปแบบมากกว่า อีกทั้งยังกำหนดแอตทริบิวต์ซินแท็กได้หลากหลายกว่าด้วย

5.5 การตรวจสอบสกีมา (Schema Checking)

เมื่อเราทำการเพิ่มเอนทรีหรือทำการเปลี่ยนแปลงข้อมูลในเอนทรีจะมีการทำการตรวจสอบสกีมาเพื่อทำการตรวจสอบข้อมูลในเอนทรีว่าตรงตามที่กำหนดไว้ในสกีมาหรือไม่ ถ้าหากข้อมูลในเอนทรีไม่ตรงตามที่กำหนดไว้ในสกีมา ไดเรกทอรีเซิร์ฟเวอร์จะทำการยกเลิกการเพิ่มหรือการแก้ไขเอนทรีนั้น แล้วทำการรายงานให้กับไดเรกทอรีไคลเอนทราบ

ขั้นตอนในการตรวจสอบสกีมาจะมีขั้นตอนดังนี้

1. ตรวจสอบค่าที่เพิ่มเข้ามาหรือค่าที่มีการเปลี่ยนแปลงว่าถูกต้องตามซินแท็กหรือไม่
2. ตรวจสอบว่ามีการกำหนดออบเจกต์คลาสในเอนทรีหรือไม่ซึ่งอย่างน้อยจะต้องกำหนดไว้หนึ่งออบเจกต์คลาส
3. แอตทริบิวต์ที่เป็นแอตทริบิวต์ที่จำเป็นต้องมี ที่กำหนดไว้ในในออบเจกต์คลาสจะต้องมีค่าอย่างน้อยหนึ่งค่าเสมอ
4. ตรวจสอบว่าแอตทริบิวต์ที่เป็นแบบซิงเกิลแวลูมีค่าในแอตทริบิวต์เพียงค่าเดียวตามที่กำหนดไว้หรือไม่

บทที่ 6
การออกแบบเนมสเปซ

การออกแบบเนมสเปซเป็นสิ่งที่สำคัญมากที่สุดอีกอย่างหนึ่งก็ว่าได้โดยไดเรกทอรีเนมสเปซ จะเตรียมความหมายพื้นฐานซึ่งใช้อ้างอิงข้อมูลในไดเรกทอรีซึ่งการออกแบบนั้นจะขึ้นอยู่กับความเหมาะสมของการใช้งาน โดยวัตถุประสงค์ของการออกแบบนั้นจะนำไปสู่

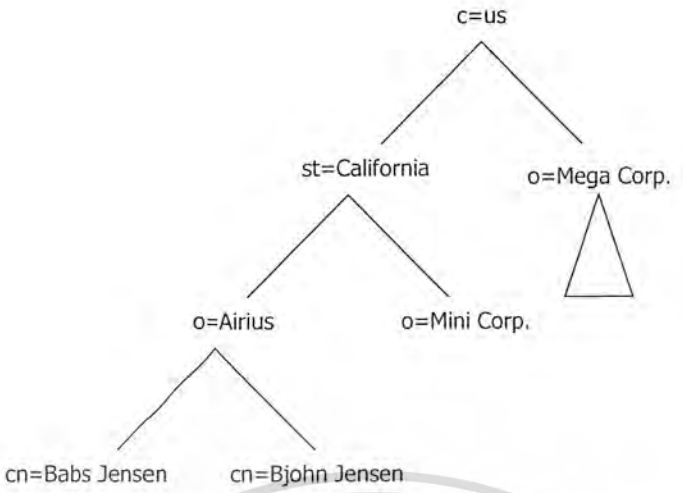
- ง่ายต่อการดูแลรักษา
- ง่ายต่อการจัดการ การควบคุมการเข้าถึงข้อมูลในไดเรกทอรี และง่ายต่อการกำหนดนโยบายในการเรพลิเคชัน
- มีความสามารถในการทำแอพลิเคชัน ได้หลากหลายและครอบคลุมกับความต้องการ
- ทั้งผู้ใช้ และ ผู้บริหารระบบ สามารถใช้ระบบไดเรกทอรีได้เป็นอย่างดี

ในทางกลับกันแล้ว ถ้าเราออกแบบ เนมสเปซ ได้ไม่ดี ก็จะนำไปสู่การจัดการที่ขัดแย้งกันเมื่อมีการเปลี่ยนแปลงชื่อของเอนทรี, การเรพลิเคชันหรือควบคุมการเข้าถึงข้อมูลหรือผู้ใช้พยายามค้นหาข้อมูล ในกรณีที่แย่ที่สุดที่จะเกิดขึ้นก็คือต้องออกแบบใหม่ซึ่งถือว่ามีความจำเป็นและสำคัญในการแก้ปัญหา นอกจากนั้น ผลกระทบของการออกแบบที่ไม่ดี ก็คือ ทำให้ผู้ใช้ ใช้งานอย่างไม่เป็นที่พอใจ รวมถึง ทำให้ผู้ดูแลระบบ ทำงานหนักขึ้นอีกด้วย

6.1 โครงสร้างของ เนมสเปซ

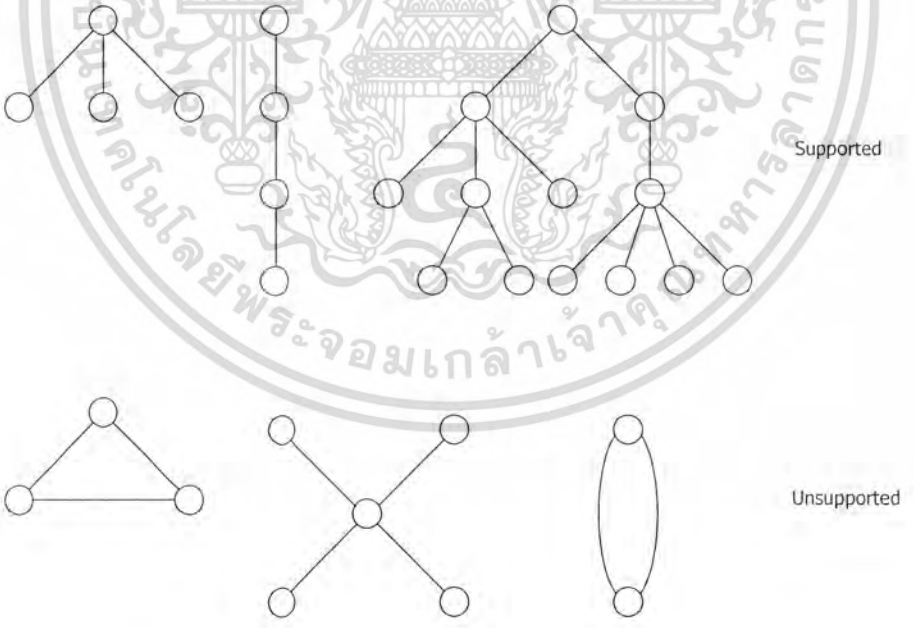
LDAP โมเดล ได้กำหนด เนมสเปซ เฟรมเวิร์ค (Namespace Framework) ที่ค่อนข้างยืดหยุ่น ซึ่งหมายถึงว่า เราสามารถ ออกแบบ พื้นที่การใช้งานได้ตามความพอใจ และตามความต้องการ โดยไม่สำคัญว่าข้อมูลนั้นจะเป็นอะไร แต่อย่างไรก็ตาม มันก็ยังหมายรวมถึงว่า เรามีทางเลือกในการสร้างมากกว่าที่เราชอบ

LDAP เนมสเปซ โมเดล (LDAP Namespace Model) ได้สืบทอดมาจากมาตรฐาน ไดเรกทอรี X.500 ซึ่งตั้งใจเอาไว้ใช้สำหรับระบบที่มีความยืดหยุ่น และใช้กันอย่างกว้างขวาง รวมถึง ระบบไดเรกทอรีแบบลำดับขั้นอีกด้วย



รูปที่ 6-1 แสดง X.500 เนมสเปซ

LDAP โมเดล พื้นฐาน ก็คือ แบบลำดับขั้น (hierarchical) หรือ โครงสร้างทรี (Tree-Structured) โดยลำดับขั้นที่มีแค่ระดับเดียวเราเรียกว่า แฟลตเนมสเปซ (Flat Namespace) โดย LDAP ไม่สนับสนุน เนมสเปซแบบโครงสร้างกราฟ ดังรูป



รูปที่ 6-2 แสดงตัวอย่างโครงสร้างของ เนมสเปซที่ถูกต้อง และไม่ถูกต้อง

หลังจากเราได้กำหนด ความสัมพันธ์ของลำดับขั้น ระหว่างเอนทรีแล้ว ขั้นตอนต่อไปก็คือ เราจะเลือกชื่อของแต่ละเอนทรี ใน แต่ละลำดับขั้นนั้นอย่างไร โดยที่ได้กล่าวมาในบทที่ 3 ซึ่งเราทราบแล้วว่า แต่ละเอนทรี ของ LDAP ไดเรกทอรีจะมีหลายแอตทริบิวต์โดยชื่อของเอนทรีจะถูกกำหนดขึ้นมาโดยการ

เลือกชื่อของแอตทริบิวต์มา หนึ่งชื่อ หรือมากกว่านั้น เราเรียกว่า RDN (Relative Distinguished Name) ซึ่งบางแอตทริบิว อาจมีหลายค่า แต่เราจะเลือกแอตทริบิวต์ที่มีค่าเดียวเท่านั้น

cn: Barbara Jensen cn: Babs Jensen sn: Jensen title: Director uid: babs mail: babs@airius.com	cn: Barbara Jensen uid: babs uid: babs + sn = Jensen mail: babs@airius.com
Entry	Examples.of RDNS

รูปที่ 6-3 แสดงตัวอย่างของ RDN

สาเหตุที่เราไม่เลือกใช้ Mutivalue RDNs ก็เพราะว่ามันไม่มีความจำเป็นที่จะต้องมีความซับซ้อน ซึ่งจะทำให้มีผลเสีย กับสมรรถภาพ (Performance) และบ่อยครั้งที่เราไม่สามารถแก้ปัญหานี้ได้

6.2 จุดประสงค์ของ เนมสเปซ

เนมสเปซ เป็นการเตรียมวิธีที่จะให้ข้อมูลในไดเรกทอรีใช้เป็นชื่อ และตัวอ้างอิงได้ โดยไดเรกทอรี จำเป็นที่จะต้องมีชื่อที่ใช้ในการอ้างอิง โดยเหตุที่ต้องมีการอกแบบ เนมสเปซ มีดังต่อไปนี้

6.2.1 Data Reference

เนมสเปซ จะหาวิธีการที่จะให้ ไดเรกทอรี ใช้ในการอ้างอิง ซึ่งมีเหตุผลสำคัญอยู่ 2 อย่าง ด้วยกัน ดังต่อไปนี้

1. เป็นทางสำหรับไดเรกทอรีไคลเอนต์ ได้อ้างอิงได้อย่างชัดเจนไปถึงไดเรกทอรีเอนทรี เมื่อเราต้องการเก็บ หรือเปลี่ยนแปลงข้อมูล
2. ชื่อของไดเรกทอรีจะหาเส้นทางที่กระทัดรัดและมีประสิทธิภาพที่จะรองรับกลุ่มของไดเรกทอรีเอนทรี

6.2.2 Data Organization

เนมสเปซ จะหาหนทางที่จะจัดระเบียบข้อมูล เราอาจจะแทนที่ เอนทรี ทั้งหมด ให้สอดคล้องกับอุปกรณ์ที่คล้ายๆกันให้เป็นส่วนหนึ่งของ เนมสเปซ นอกจากนี้ การแบ่งองค์กรออกเป็นย่อยๆก็มีความเป็นไปได้

บางทีก็มีพื้นฐานบนลักษณะตามธรรมชาติ หรือตามข้อมูลขององค์กร ดังนั้น องค์กรสามารถช่วยให้การค้นหาข้อมูลในไดเรกทอรีง่ายขึ้น ตัวอย่างเช่น Printing Application ทำให้ผู้ใช้ สามารถเลือกข้อมูลของพรินเตอร์ ที่มีคุณสมบัติใกล้เคียงกันได้ เช่น ความเร็ว สี ที่ใกล้เคียงกัน เป็นต้น

6.2.3 Data Partitioning

เนมสเปซ ทำให้ข้อมูลในไดเรกทอรี สามารถแบ่งออกเป็นส่วนๆได้ หรือสามารถแบ่งระหว่างเซิร์ฟเวอร์หลายๆตัวได้ (Multiple Servers) โดยเฉพาะอย่างยิ่ง การแบ่งออกเป็นส่วนๆ นั้นสามารถกระทำได้เฉพาะเนมสเปซที่มีการกำหนดมาเป็นอย่างดีเท่านั้นถ้าเนมสเปซไม่ได้มีการกำหนดจุดในการแบ่งแล้วเราก็ไม่สามารถแบ่งข้อมูลออกเป็นส่วนๆได้

6.2.4 Data Replication

ถึงแม้ว่าจะไม่มีความเกี่ยวข้องกับการเรพลิเคชันโดยตรง แต่การเลือกเนมสเปซสามารถบังคับการเรพลิเคชันของไดเรกทอรีให้เป็นไปในทิศทางเดียวกันได้ ซึ่งการเรพลิเคชันนั้นส่วนมากแล้ว ต้องกาการแบ่งส่วนข้อมูล ที่มีการออกแบบมาเป็นอย่างดีแล้ว

6.2.5 Access Control

เช่นเดียวกับการเรพลิเคชันโดย Access Control ก็มีผลกระทบต่อเนมสเปซ ซึ่งบางผลิตภัณฑ์จะอนุญาตให้ตั้งค่า การควบคุมการเข้าถึงข้อมูลในกิ่งของทรีเท่านั้น

6.2.6 Application Support

ในการออกแบบเนมสเปซนั้นแน่นอนว่าเราจะต้องออกแบบให้แอพพลิเคชันที่ใช้งานสามารถใช้งานได้อย่างเป็นที่น่าพอใจ และ ง่ายต่อการเข้าใจด้วย รวมถึงรองรับการใช้งานที่เกิดขึ้นในอนาคตด้วย

6.3 การวิเคราห์ความต้องการของเนมสเปซ

ในขั้นแรกที่เราจะออกแบบ เนมสเปซ ก็คือ ความเข้าใจว่า ความต้องการคืออะไร ต้องการที่จะออกแบบแฟลตเนมสเปซ (Flat Namespace) หรือ แบบลำดับขั้น จากนั้นก็ดูว่าแอตทริบิวต์อะไรบ้างที่เราต้องใช้เป็นชื่อของเอนทรี แล้วเรามีการเรพลิเคชันหรือมีการแบ่งข้อมูลหรือไม่ ซึ่งจะมีผลร้ายต่อการออกแบบเนมสเปซ นอกจากนี้เราต้องดูว่า การเข้าถึงข้อมูลเป็นแบบใด รวมทั้งดูว่าแอพลิเคชันที่ไดเรกทอรีรองรับคืออะไรและมีการเปลี่ยนแปลงของข้อมูลมากน้อยขนาดไหน ซึ่งเราจะต้องทราบข้อมูลทั้งหมดก่อนที่เราจะออกแบบเนมสเปซ

6.3.1 การเลือก Suffix

Suffix คือชื่อของเอนทรี ที่อยู่บนสุดของ DIT ที่เราสร้าง โดยการเลือกใช้ suffix นั้น เราจะต้องดูข้อมูลในองค์กรก่อน เพื่อนให้สอดคล้องกับการใช้งาน

6.3.2 แฟลตและลำดับขั้น

อย่างหนึ่งที่จำเป็นจะต้องพิจารณาก่อนที่จะออกแบบเนมสเปซ ก็คือเราจะทำเป็นแบบแฟลต หรือลำดับขั้น โดยหลักการของการออกแบบแล้ว เราจะต้องทำให้เป็นแฟลต มากที่สุดเท่าที่จะเป็นไปได้ เพราะจะได้มีการเปลี่ยนแปลงที่น้อยที่สุด และไม่เป็นภาระกับผู้ดูแลระบบ

6.3.3 เนมมิ่ง แอตทริบิวต์

แอตทริบิวต์ที่ใช้งานควรจะขึ้นกับชนิดของเอนทรีที่เราตั้งชื่อขึ้นมา ทุกแอตทริบิวต์ก็จะเป็นข้อมูลที่เกี่ยวข้องกับเอนทรีนั้นๆ เพื่อนให้ข้อมูลที่ได้ ไปในทิศทางเดียวกัน และไม่ปนกัน

6.3.4 การพิจารณา แอพพลิเคชัน

ระบบไดเรกทอรีที่ติดต่อกับแอพลิเคชันภายนอกนั้น ซึ่งมันจะอ้างอิงผ่านเนมสเปซ ซึ่งแน่นอนว่าเราจะต้องออกแบบให้แอพลิเคชันใช้งานได้ง่ายและเป็นธรรมชาติ เพื่อให้การติดต่อระหว่างแอพลิเคชันและ ระบบไดเรกทอรี เป็นไปได้โดยสะดวก

6.3.5 การพิจารณาการบริหาร

ในการออกแบบเนมสเปซนั้นจะต้องคำนึงถึงผลกระทบที่มีต่องานด้านบริหารระบบด้วย เช่น เมื่อมีการเพิ่ม เปลี่ยนแปลง หรือ ลบ ข้อมูล แล้ว มันมีผลเสียหรือขัดกับกฎของระบบไดเรกทอรีหรือเปล่า หรือมีผลเสียที่ทำให้ประสิทธิภาพการทำงานของระบบไดเรกทอรีช้าลงหรือไม่ ซึ่งเราจะต้องมีการออกแบบเนมสเปซให้ดีและควรออกแบบเผื่อกรณีที่ยังไม่เกิดขึ้นด้วย เพราะถ้าเราทำแบบนี้แล้ว ก็จะทำให้ งานของ การบริหารระบบ ซึ่งมีผู้บริหารระบบรับผิดชอบอยู่ ก็จะเบาลงได้

6.3.6 การพิจารณาความเป็นส่วนตัว

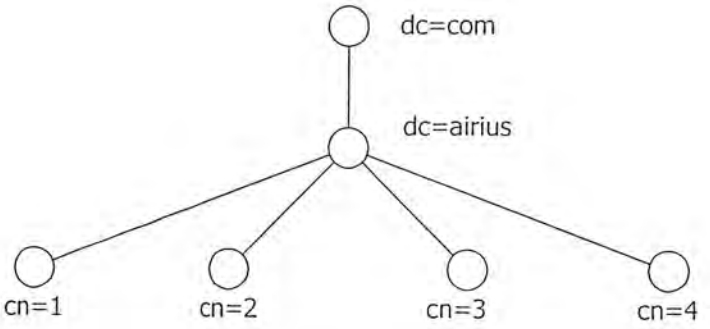
จำเป็นอย่างยิ่งที่เราจะต้องระมัดระวังข้อมูลที่เป็นส่วนตัวของแต่ละบุคคล โดยจะต้องออกแบบการควบคุมการเข้าถึงข้อมูลตามระดับที่ควรจะเป็นเพื่อให้ข้อมูลที่เป็นความลับและเป็นส่วนตัวไม่สามารถล่วงละเมิดได้

6.4 ตัวอย่างของ เนมสเปซ

หลังจากที่เราได้อธิบายจุดประสงค์ของเนมสเปซและการตัดสินใจหลักที่ต้องทำในการออกแบบแต่ละครั้ง เราลองมาดูตัวอย่างของเนมสเปซ ทั้งแบบแฟลต เนมสเปซ และ แบบลำดับขั้น

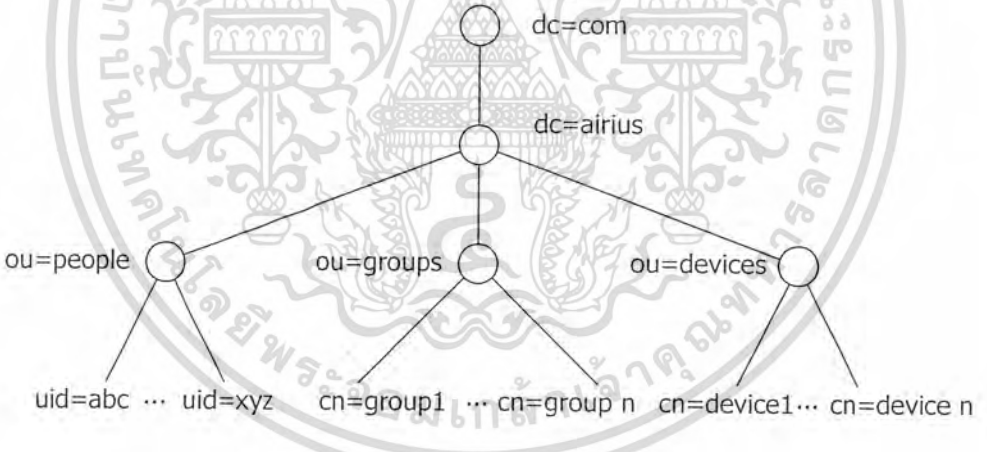
6.4.1 ตัวอย่างของ แฟลต เนมสเปซ

ในตัวอย่างแรกจะเป็นแบบแฟลตเนมสเปซโดยชื่อของแอตทริบิวต์จะเลือก cn มาเพื่อให้ดูง่าย และแต่ละเอนทรี จะใช้ ค่าแอตทริบิวต์ และค่าตัวเลขที่สร้างขึ้นมาเรียงกันเป็นเรื่อยๆ ตามรูป



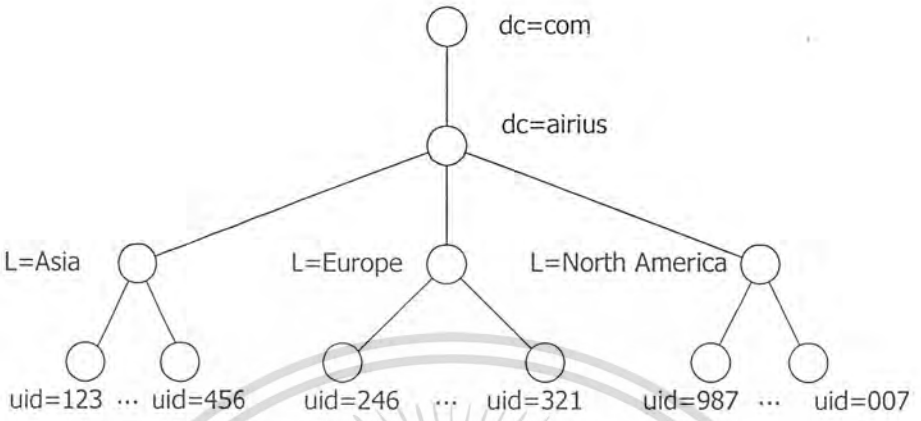
รูปที่ 6-4 แสดงตัวอย่างของแฟลตเนมสเปซ

ตัวอย่างต่อมาเป็นตัวอย่างของแฟลตเนมสเปซและแบบลำดับขั้นรวมกัน โดยแบบแฟลตเนมสเปซ จะอยู่ด้านล่าง แบบลำดับขั้น ดังรูป



รูปที่ 6-5 แสดงตัวอย่างของ แฟลตเนมสเปซแบบลำดับขั้น

6.4.2 ตัวอย่างของเนมสเปซแบบลำดับขั้น



รูปที่ 6-6 แสดงตัวอย่างของ เนมสเปซแบบลำดับขั้น

บทที่ 7
การพัฒนาโครงงาน

7.1 การออกแบบข้อมูล

ขั้นตอนในการออกแบบข้อมูลสำหรับเอเจนต์แต่ละตัวนั้น เราจะทำการเลือกว่าจะเก็บข้อมูลใดบ้างและข้อมูลที่จัดเก็บนั้นเป็นข้อมูลแบบใดและมีข้อมูลใดบ้างที่ใช้รูปแบบเดียวกันได้ ข้อมูลของแต่ละเอเจนต์นั้นแสดงให้เห็นในตาราง

โปรแกรมไฟล์วอลและเอเจนต์ฉลาด (Firewall with Intelligent Agent)

Element	Description	Syntax
AgentType	ชนิดของเอเจนต์	CaseIgnoreString
AgentID	ชื่อรหัสของเอเจนต์	CaseExactString
IP	หมายเลขไอพีของเครื่องที่ติดตั้งเอเจนต์	CaseIgnoreString
Starttime	เวลาที่เอเจนต์เริ่มทำงาน	CaseIgnoreString
Refresh	เวลาที่เอเจนต์รีเฟรส	CaseIgnoreString
Source	ไอพีต้นทาง	CaseIgnoreString
Destination	ไอพีปลายทาง	CaseIgnoreString
Service	รูปแบบของบริการ	CaseIgnoreString
Action	การอนุญาตหรือไม่อนุญาตให้ใช้งานบริการ	CaseIgnoreString

ตารางที่ 7-1 ข้อมูลและรูปแบบข้อมูลของโปรแกรมไฟล์วอลและเอเจนต์ฉลาด

โปรแกรมวิเคราะห์แพ็กเก็ตแบบสเตตฟูล (Stateful Packet Analyzer)

Element	Description	Syntax
AgentType	ชนิดของเอเจนต์	CaseIgnoreString
AgentID	ชื่อรหัสของเอเจนต์	CaseExactString
IP	หมายเลขไอพีของเครื่องที่ติดตั้งเอเจนต์	CaseIgnoreString
Starttime	เวลาที่เอเจนต์เริ่มทำงาน	CaseIgnoreString
Refresh	เวลาที่เอเจนต์รีเฟรส	CaseIgnoreString
IPCap	หมายเลขไอพีที่จะดักข้อมูล	CaseIgnoreString
PortCap	หมายเลขพอร์ตที่จะดักข้อมูล	CaseIgnoreString

ตารางที่ 7-2 ข้อมูลและรูปแบบข้อมูลของโปรแกรมวิเคราะห์แพ็กเก็ตแบบสเตตฟูล

โปรแกรมตรวจจับผู้บุกรุกเครือข่าย (Network Intrution Detection)

Element	Description	Syntax
AgentType	ชนิดของเอเจนต์	CaseIgnoreString
AgentID	ชื่อรหัสของเอเจนต์	CaseExactString
IP	หมายเลขไอพีของเครื่องที่ติดตั้งเอเจนต์	CaseIgnoreString
Starttime	เวลาที่เอเจนต์เริ่มทำงาน	CaseIgnoreString
Refresh	เวลาที่เอเจนต์รีเฟรส	CaseIgnoreString

Element	Description	Syntax
Source	ไอพีต้นทาง	CaseIgnoreString
Destination	ไอพีปลายทาง	CaseIgnoreString
AttackType	รูปแบบการโจมตี	CaseIgnoreString
Atime	เวลาที่ถูกโจมตี	CaseIgnoreString

ตารางที่ 7-3 ข้อมูลและรูปแบบข้อมูลของโปรแกรมโปรแกรมตรวจจับผู้บุกรุกเครือข่าย

7.2 การออกแบบสกีมา

หลังจากที่ออกแบบข้อมูลที่ต้องการจะจัดเก็บแล้วเราจะนำมาออกแบบเป็นสกีมาโดยจะแบ่งการออกแบบเป็น 2 ส่วนคือ การออกแบบออบเจกต์คลาสและการออกแบบแอตทริบิวต์ ซึ่งการออกแบบแอตทริบิวต์นั้นเราก็เพียงแต่นำข้อมูลที่ได้ออกแบบไว้ข้างต้นมากำหนดในรูปแบบของไฟล์สกีมา ในส่วนของการออกแบบออบเจกต์คลาส ก็จะเป็นการจัดกลุ่มของแอตทริบิวต์ตามชนิดของเอเจนต์ การกำหนดสกีมาจะต้องทำการเขียนแล้วจัดเก็บเป็นไฟล์จากนั้นถ้าจะใช้งานก็จะต้องกำหนดในไฟล์คอนฟิกของ LDAP ซึ่งก็คือไฟล์ slapd.conf โดยกำหนดโดยใช้ include แล้วตามด้วยชื่อไดเรกทอรีและชื่อไฟล์สกีมา

```
include /usr/local/etc/openldap/core.schem
```

รูปที่ 7-1 แสดงการเรียกใช้งานไฟล์สกีมา

<u>ส่วนกำหนดแอตทริบิวต์</u>

<u>แอตทริบิวต์ของแต่ละโปรแกรมที่มีรูปแบบข้อมูลเหมือนกัน</u>

แอตทริบิวต์ AgentType

```
attributetype ( 1.3.6.1.4.4.1 NAME 'AgentType'
        EQUALITY caseIgnoreMatch
        ORDERING caseIgnoreOrderingMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
```

รูปที่ 7-2 แสดงรูปแบบของแอตทริบิวต์ AgentType

แอตทริบิวต์ AdentID

```
attributetype (1.3.6.1.4.4.2 NAME 'AgentID'
        EQUALITY caseExactIA5Match
        ORDERING caseIgnoreOrderingMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)
```

รูปที่ 7-3 แสดงรูปแบบของแอตทริบิวต์ AgentID

แอตทริบิวต์ IP

```
attributetype (1.3.6.1.4.4.4 NAME 'IP'
        EQUALITY caseIgnoreMatch
        ORDERING caseIgnoreOrderingMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)
```

รูปที่ 7-4 แสดงรูปแบบของแอตทริบิวต์ IP

แอตทริบิวต์ StartTime

```
attributetype (1.3.6.1.4.4.6 NAME 'StartTime'
        EQUALITY caseIgnoreMatch
        ORDERING caseIgnoreOrderingMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)
```

รูปที่ 7-5 แสดงรูปแบบของแอตทริบิวต์ StartTime

แอตทริบิวต์ Refresh

```
attributetype (1.3.6.1.4.4.5 NAME 'Refresh'
        EQUALITY caseIgnoreMatch
        ORDERING caseIgnoreOrderingMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)
```

รูปที่ 7-6 แสดงรูปแบบของแอตทริบิวต์ Refresh

แอตทริบิวต์ที่มีรูปแบบเหมือนกันของโปรแกรมไฟล์วอลและโปรแกรมตรวจจับผู้บุกรุกเครือข่าย

แอตทริบิวต์ Source

```
attributetype (1.3.6.1.4.4.7 NAME 'Source'
        EQUALITY caseIgnoreMatch
        ORDERING caseIgnoreOrderingMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)
```

รูปที่ 7-7 แสดงรูปแบบของแอตทริบิวต์ Source

แอตทริบิวต์ Destination

```
attributetype (1.3.6.1.4.4.8 NAME 'Destination'
        EQUALITY caseIgnoreMatch
        ORDERING caseIgnoreOrderingMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)
```

รูปที่ 7-8 แสดงรูปแบบของแอตทริบิวต์ Destination

<u>แอตทริบิวต์ของโปรแกรมไฟล์วอล</u>

แอตทริบิวต์ Service

```
attributetype (1.3.6.1.4.4.9 NAME 'Service'
        EQUALITY caseIgnoreMatch
        ORDERING caseIgnoreOrderingMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)
```

รูปที่ 7-9 แสดงรูปแบบของแอตทริบิวต์ Service

แอตทริบิวต์ Action

```
attributetype (1.3.6.1.4.4.10 NAME 'Action'
        EQUALITY caseIgnoreMatch
        ORDERING caseIgnoreOrderingMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)
```

รูปที่ 7-10 แสดงรูปแบบของแอตทริบิวต์ Action

<u>แอตทริบิวต์ของโปรแกรมวิเคราะห์แพ็กเก็ตแบบสเตตฟูล</u>

แอตทริบิวต์ IPCap

```
attributetype (1.3.6.1.4.4.11 NAME 'IPCap'
        EQUALITY caseIgnoreMatch
        ORDERING caseIgnoreOrderingMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)
```

รูปที่ 7-11 แสดงรูปแบบของแอตทริบิวต์ IPCap

แอตทริบิวต์ PortCap

```
attributetype (1.3.6.1.4.4.12 NAME 'PortCap'
        EQUALITY caseIgnoreMatch
        ORDERING caseIgnoreOrderingMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)
```

รูปที่ 7-12 แสดงรูปแบบของแอตทริบิวต์ PortCap

แอตทริบิวต์ของโปรแกรมตรวจจับผู้บุกรุกเครือข่าย

แอตทริบิวต์ Atime

```
attributetype (1.3.6.1.4.4.15 NAME 'Atime'
        EQUALITY caseIgnoreMatch
        ORDERING caseIgnoreOrderingMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)
```

รูปที่ 7-13 แสดงรูปแบบของแอตทริบิวต์ Atime

แอตทริบิวต์ AttackType

```
attributetype (1.3.6.1.4.4.16 NAME 'AttackType'
        EQUALITY caseIgnoreMatch
        ORDERING caseIgnoreOrderingMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)
```

รูปที่ 7-14 แสดงรูปแบบของแอตทริบิวต์ AttackType

<u>ส่วนกำหนดออบเจกต์คลาส</u>

ออบเจกต์คลาส AttackType

```
objectclass (1.3.6.1.4.3.1 NAME 'AgentGroup'
        SUP top STRUCTURAL
        MUST (AgentType)
        MAY description)
```

รูปที่ 7-15 แสดงรูปแบบของออบเจกต์คลาส AttackType

ออบเจกต์คลาส Firewall

```
objectclass (1.3.6.1.4.3.3 NAME 'Firewall'
        SUP top STRUCTURAL
        MUST (AgentID$IPSStartTime$Refresh$Source$Destination$Service$Action)
        MAY description)
```

รูปที่ 7-16 แสดงรูปแบบของออบเจกต์คลาส Firewall

ออบเจกต์คลาส Stateful

```
objectclass (1.3.6.1.4.3.6 NAME 'Stateful'
        SUP top STRUCTURAL
        MUST (AgentID$IPSStartTime$Refresh$IPCap$PortCap)
        MAY description)
```

รูปที่ 7-17 แสดงรูปแบบของออบเจกต์คลาส Stateful

ออบเจกต์คลาส NIDS

```
objectclass (1.3.6.1.4.3.7 NAME 'NIDS'
        SUP top STRUCTURAL
        MUST (RefreshSAgentIDSIPSStartTimeSSourceSDestinationSAttackTypeSAtime)
        MAY description)
```

รูปที่ 7-18 แสดงรูปแบบของออบเจกต์คลาส NIDS

7.3 การออกแบบโปรแกรมช่วยสร้างไฟล์สกีมา (Schema Tool)

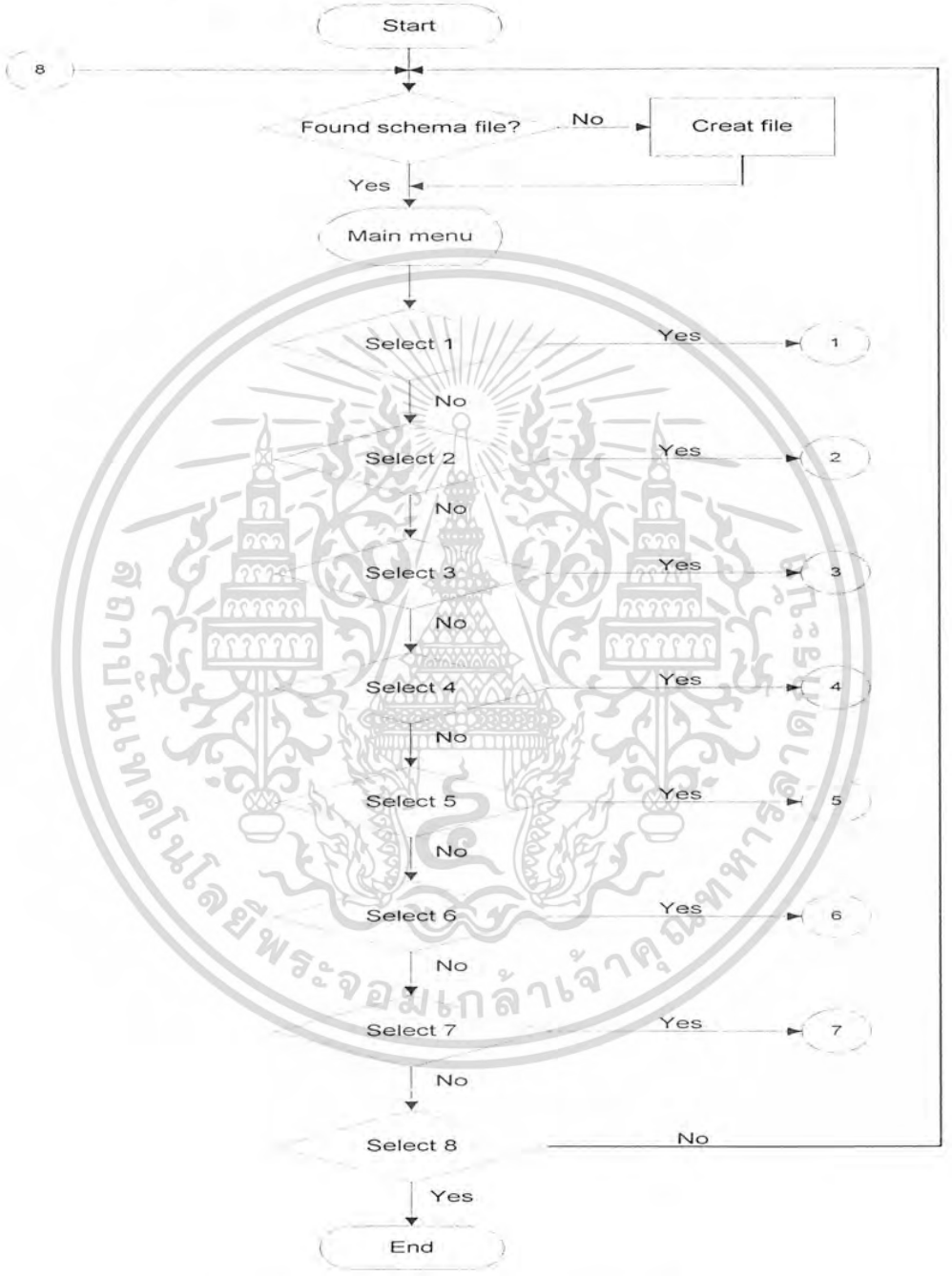
โปรแกรมช่วยสร้างไฟล์สกีมาจะเป็นตัวช่วยในการสร้างไฟล์สกีมาโดยจะอำนวยความสะดวกในการจัดการเพิ่มลบแอตทริบิวต์และออบเจอต์คลาสต่างๆ ในสกีมาโดยจะมีส่วนการทำงานหลักๆ ดังนี้

- ส่วนแสดงเมนูให้เลือกใช้งาน
- ส่วนเพิ่มแอตทริบิวต์ในสกีมา
- ส่วนเพิ่มออบเจกต์คลาสในสกีมา
- ส่วนลบแอตทริบิวต์ออกจากสกีมา
- ส่วนลบออบเจกต์คลาสออกจากสกีมา
- ส่วนเพิ่มแอตทริบิวต์ในออบเจกต์คลาส
- ส่วนลบแอตทริบิวต์ออกจากออบเจกต์คลาส
- ส่วนแสดงแอตทริบิวต์และออบเจกต์คลาสในสกีมา

1. ส่วนแสดงเมนูให้เลือกใช้งาน

ทำหน้าที่แสดงเมนูและตรวจสอบการเลือกใช้งานดังนี้

- ตรวจสอบว่ามีไฟล์สกีมาหรือไม่ ถ้าไม่มีทำการสร้างไฟล์สกีมา
- ทำการแสดงเมนูให้เลือก



รูปที่ 7-19 โฟลว์ชาร์ต แสดงการทำงานของเมนูหลัก

2. ส่วนเพิ่มแอตทริบิวต์ในสกีมา

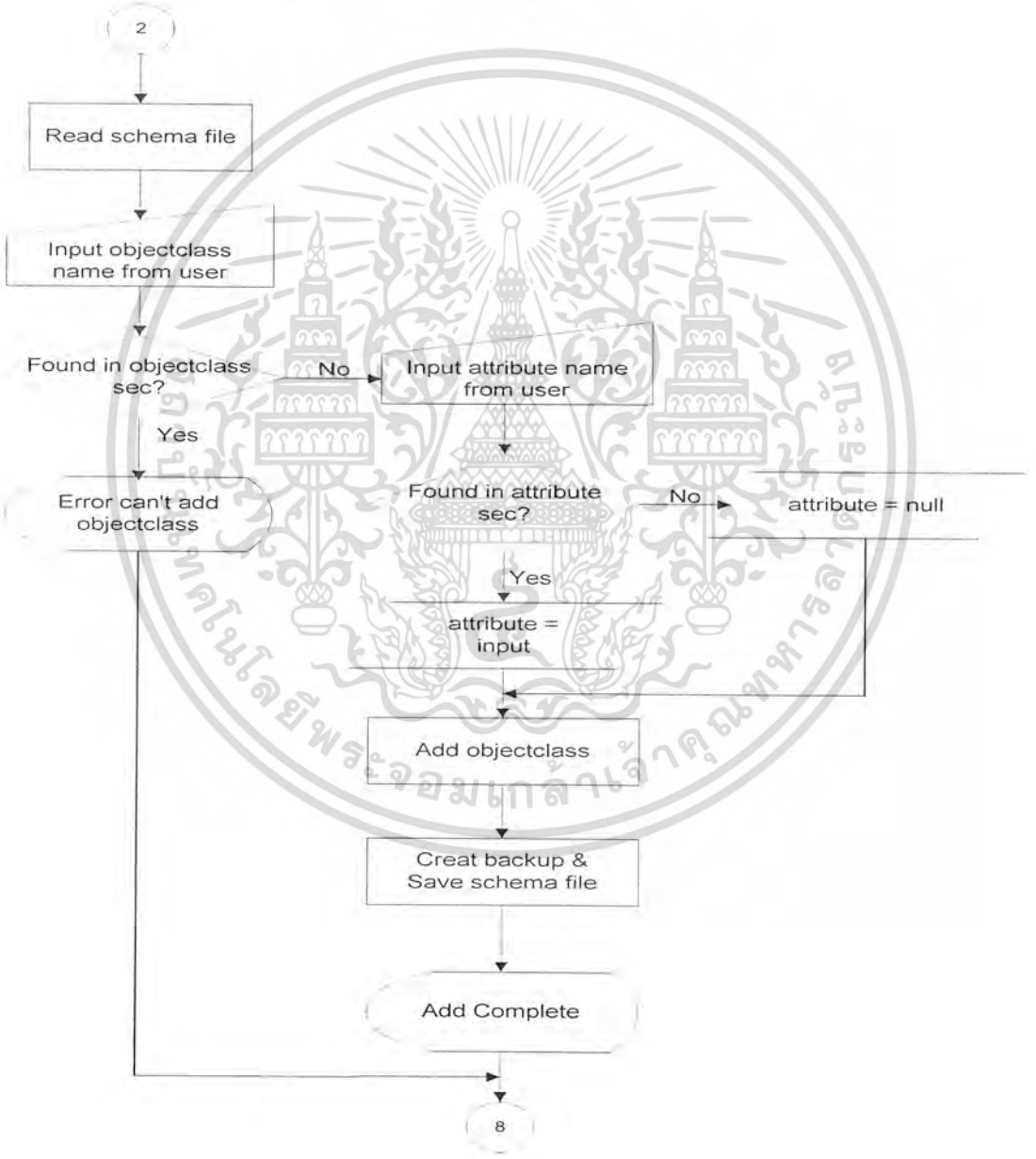
ทำหน้าที่รับเพิ่มแอตทริบิวต์ดังนี้

- อ่านไฟล์สกีมาจากนั้นรับชื่อแอตทริบิวต์และซินแท็กซ์จากผู้ใช้
- ตรวจสอบชื่อแอตทริบิวต์ว่าได้มีการกำหนดไว้แล้วหรือไม่ ถ้ามีการกำหนดไว้แล้วจะแจ้งว่าไม่สามารถเพิ่มแอตทรบิวต์นั้นได้ แต่ถ้าไม่มีการกำหนดไว้ก็จะทำการเพิ่มแอตทริบิวต์นั้นด้
- เพิ่มแอตทริบิวต์
- สร้างแบ็กอัพและเซฟสกีมาแล้วกลับสู่เมนูหลัก

รูปที่ 7-20 โฟลว์ชาร์ต แสดงการทำงานของส่วนการเพิ่มแอตทริบิวต์ในสกีมา

3. ส่วนเพิ่มออบเจกต์คลาสในสกีมา

- อ่านไฟล์สกีมาจากนั้นรับชื่อออบเจกต์คลาสจากผู้ใช้
- ตรวจสอบชื่อออบเจกต์คลาสว่าได้มีการกำหนดไว้แล้วหรือไม่ ถ้ามีการกำหนดไว้แล้ว จะแจ้งว่าไม่สามารถเพิ่มออบเจกต์คลาสนั้นได้ แต่ถ้าไม่มีการกำหนดไว้ก็จะทำการรับค่าแอตทริบิวต์จากผู้ใช้ที่ต้องการให้เป็นสมาชิกของออบเจกคลาส โดยจะทำการตรวจสอบด้วยว่าแอตทริบิวนั้นได้กำหนดไว้หรือไม่ ถ้าไม่ได้กำหนดไว้จะไม่เพิ่มในออบเจกต์คลาส
- เพิ่มออบเจกคลาส
- สร้างแบ็กอัพและเซฟสกีมาแล้วกลับสู่เมนูหลัก



รูปที่ 7-21 โฟลว์ชาร์ต แสดงการทำงานของส่วนเพิ่มออบเจกต์คลาสในสกีมา

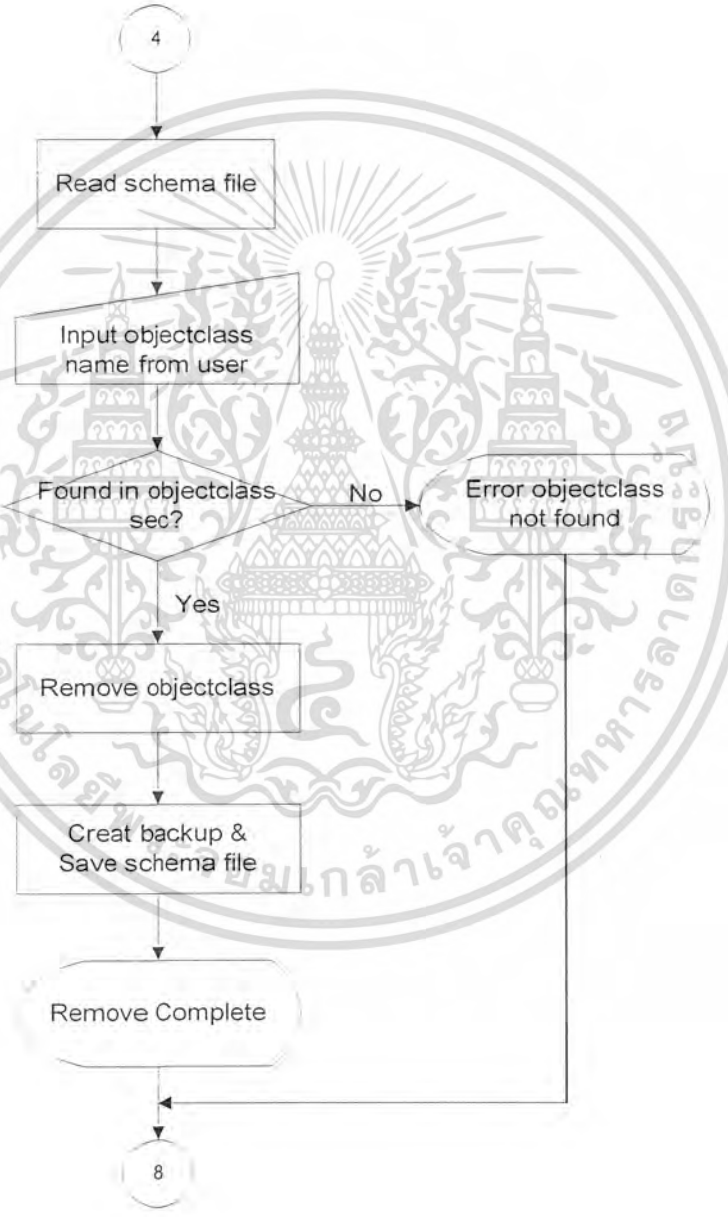
4. ส่วนลบแอตทริบิวต์ออกจากสกีมา

- อ่านไฟล์สกีมาจากนั้นรับชื่อแอตทริบิวต์จากผู้ใช้
- ตรวจสอบชื่อแอตทริบิวต์ว่าได้มีการกำหนดไว้แล้วหรือไม่ ถ้ามีการกำหนดไว้แล้วจะทำการลบออกจากส่วนกำหนดแอตทริบิวต์และส่วนกำหนดออบเจกต์คลาส แต่ถ้าไม่มีการกำหนดไว้ก็จะแจ้งว่าไม่พบแอตทริบิวต์ที่ต้องการลบแล้วกลับสู่เมนูหลัก
- สร้างแบ็กอัพและเซฟสกีมาแล้วกลับสู่เมนูหลัก

รูปที่ 7-22 โฟลว์ชาร์ต แสดงการทำงานของส่วนลบแอตทริบิวต์ออกจากสกีมา

5. ส่วนลบออบเจกต์คลาสออกจากสกีมา

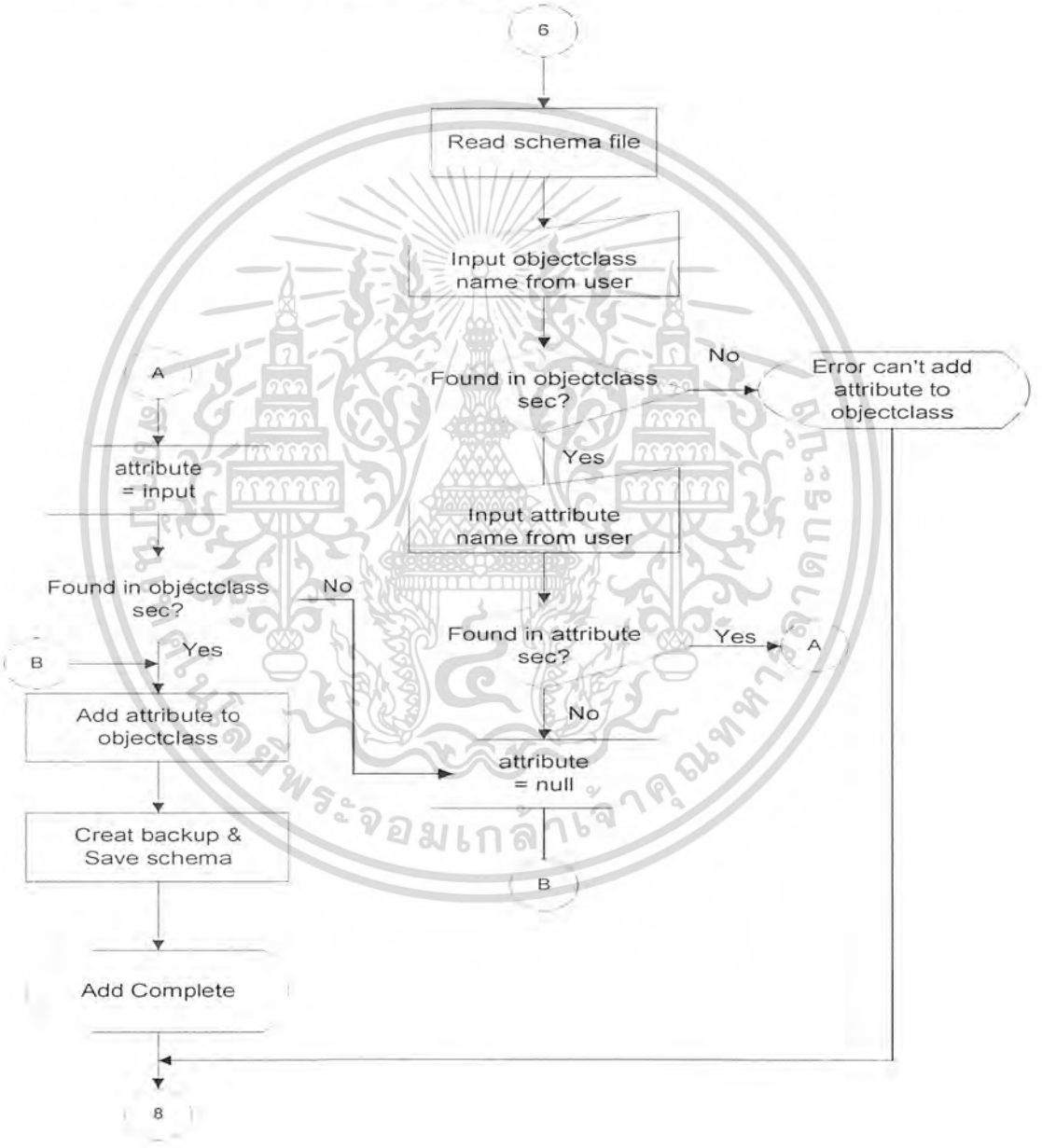
- อ่านไฟล์สกีมาจากนั้นรับชื่อออบเจกต์คลาสจากผู้ใช้
- ตรวจสอบชื่อออบเจกต์คลาสว่าได้มีการกำหนดไว้แล้วหรือไม่ ถ้ามีการกำหนดไว้แล้วจะทำการลบออกจากส่วนกำหนดออบเจกต์คลาส แต่ถ้าไม่มีการกำหนดไว้ก็จะแจ้งว่าไม่พบออบเจกต์คลาสที่ต้องการลบแล้วกลับสู่เมนูหลัก
- สร้างแบ็กอัพและเซฟสกีมาแล้วกลับสู่เมนูหลัก



รูปที่ 7-23 โฟลว์ชาร์ต แสดงการทำงานของส่วนลบออบเจกต์คลาสออกจากสกีมา

6. ส่วนเพิ่มแอตทริบิวต์ในออบเจกต์คลาส

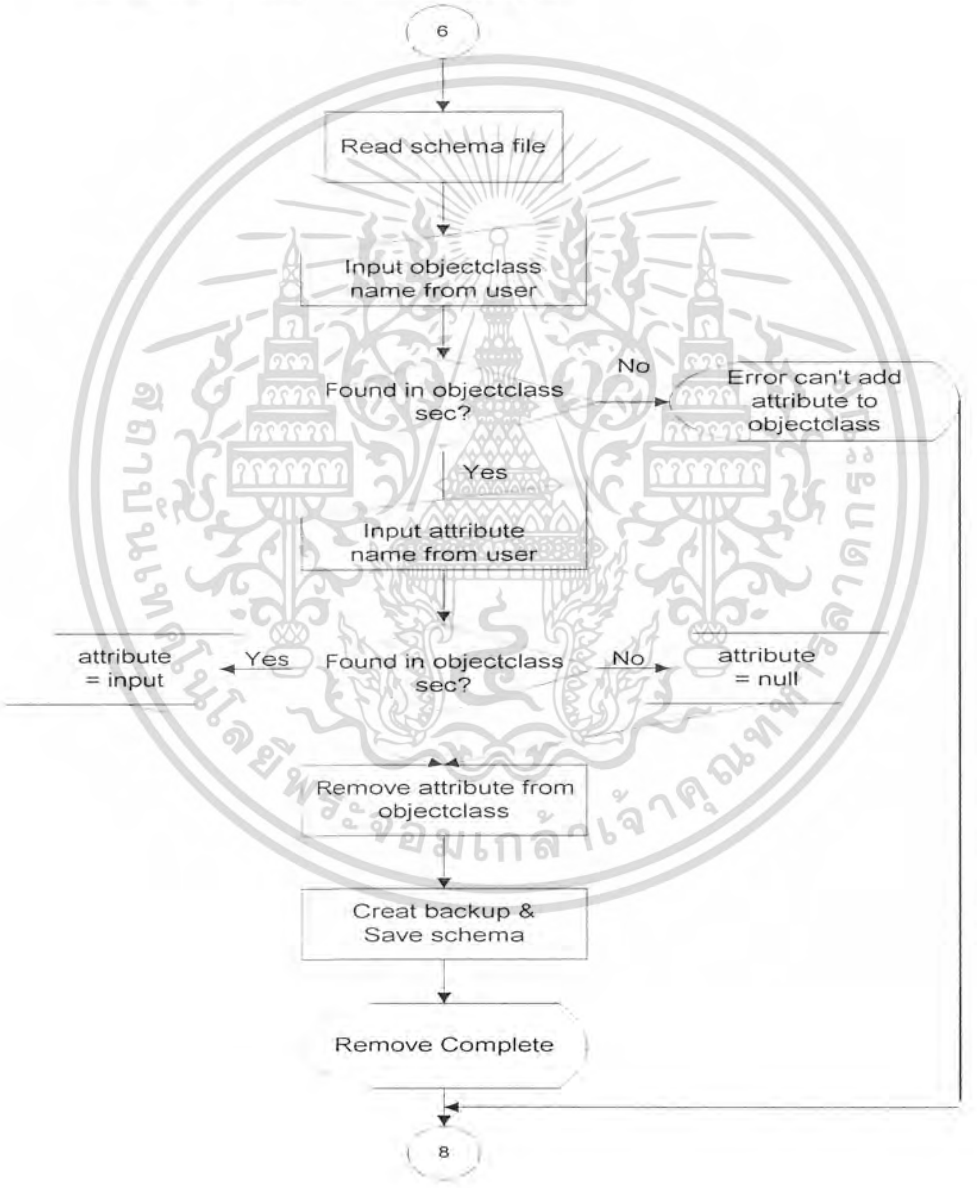
- อ่านไฟล์สกีมาจากนั้นรับชื่อออบเจกต์คลาสจากผู้ใช้
- ตรวจสอบชื่อออบเจกต์คลาสว่าได้มีการกำหนดไว้แล้วหรือไม่ ถ้ามีการกำหนดไว้ก็จะทำการรับค่าแอตทริบิวต์จากผู้ใช้ที่ต้องการให้เป็นสมาชิกของออบเจกคลาส โดยจะทำการตรวจสอบด้วยว่าแอตทริบิวนั้นได้กำหนดไว้หรือไม่ ถ้าไม่ได้กำหนดไว้จะไม่เพิ่มในออบเจกต์คลาส และในกรณีที่ไม่ได้กำหนดออบเจกต์คลาสไว้จะทำการแจ้งเตือนแล้วกลับสู่เมนูหลัก
- สร้างแบ็กอัพและเซฟสกีมาแล้วกลับสู่เมนูหลัก



รูปที่ 7-24 โฟลว์ชาร์ต แสดงการทำงานของส่วนเพิ่มแอตทริบิวต์ในออบเจกต์คลาส

7. ส่วนลบแอตทริบิวต์ออกจากออบเจกต์คลาส

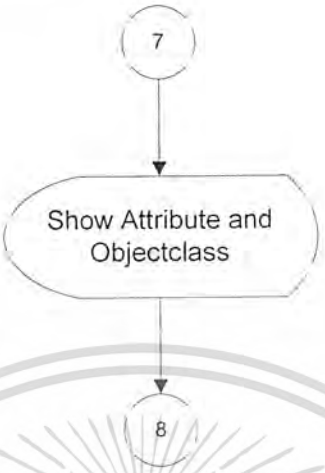
- อ่านไฟล์สกีมาจากนั้นรับชื่อออบเจกต์คลาสจากผู้ใช้
- ตรวจสอบชื่อออบเจกต์คลาสว่าได้มีการกำหนดไว้แล้วหรือไม่ ถ้ามีการกำหนดไว้ก็จะทำการรับค่าแอตทริบิวต์ที่ต้องการลบออกจากออบเจกคลาส โดยจะทำการตรวจสอบด้วยว่าแอตทริบิวต์นั้นได้กำหนดไว้ในออบเจกต์คลาส หรือไม่ถ้ากำนดไว้ก็จะทำการลบออก แต่ถ้าไม่ได้กำหนดไว้จะทำการแจ้งเตือนว่าไม่พบแล้วกลับสู่เมนูหลัก
- สร้างแบ็กอัพและเซฟสกีมาแล้วกลับสู่เมนูหลัก



รูปที่ 7-25 โฟลว์ชาร์ต แสดงการทำงานของส่วนลบแอตทริบิวต์ออกจากออบเจกต์คลาส

8. ส่วนแสดงแอตทริบิวต์และออบเจกต์คลาสในสกีมา

- แสดงแอตทริบิวต์และออบเจกต์คลาสในสกีมาแล้วกลับสู่เมนูหลัก



รูปที่ 7-26 โฟลว์ชาร์ต แสดงการทำงานของส่วนลบแอตทริบิวต์ออกจากออบเจกต์คลาส

7.4 หลักการการออกแบบ API

ความต้องการของผู้ใช้

ในการออกแบบ API นั้นจำเป็นอย่างยิ่งที่เราต้องทราบความต้องการการใช้งานของผู้ใช้เพื่อลดความยุ่งยากในการใช้ API ของ LDAP โดยตรง ในการใช้งานของผู้ใช้นั้น สำหรับระบบไดเรกทอรีแล้วการใช้งานส่วนมากที่ใช้กันบ่อยในส่วนของผู้ใช้ ก็คือ การเพิ่มข้อมูลและการเปลี่ยนแปลงข้อมูลเพราะแต่ละเอเจนต์จะรายงานสถานะของตัวเอง เมื่อมีการใช้งาน ว่าขณะนี้สถานะ Active อยู่หรือถ้ามีการเลิกใช้งานก็ต้องเปลี่ยนสถานะของตัวเองให้เป็น Down เป็นต้น

การกำหนดรูปแบบของฟังก์ชัน

เป็นการกำหนดชื่อของฟังก์ชัน, พารามิเตอร์, การคืนค่ารวมทั้งขอบเขตของข้อมูลที่สามารถส่งไปได้ ว่าต้องมีลักษณะอย่างไร ชื่อของฟังก์ชันจะต้องสื่อกับการใช้งานและการใช้งานโดยรวมต้องง่ายไม่มีความซับซ้อนของข้อมูลที่เราได้คืนมาจากฟังก์ชัน

Code Optimization

การอิมพลีเมนต์ฟังก์ชันการใช้งานจะต้องมีการใช้ทรัพยากรที่ประหยัดซึ่งยังผลให้เวลาในการทำงานโดยรวมของฟังก์ชันใช้เวลาน้อย นอกจากนี้อัลกอริทึมในการคำนวณก็จะต้องมีความกระทัดรัดไม่มีความซ้ำซ้อนของแต่ละคำสั่งเพื่อให้ฟังก์ชันโดยรวมมีความเร็วและโค้ดน้อยที่สุด

ความถูกต้องของฟังก์ชัน

ความถูกต้องในที่นี้คือสามารถทำได้ตามหน้าที่โดยไม่มีผลกระทบต่อส่วนอื่นและเมื่อมีการคำนวณ จะต้องได้ผลลัพธ์ที่ถูกต้องและแม่นยำ

ความคงทน

เมื่อมีการส่งพารามิเตอร์ที่ไม่ถูกต้องเข้าไปในฟังก์ชันจะต้องไม่มีการทำให้ระบบเสียหายหรือทำให้เครื่องแฮงค์ได้ ซึ่งปัญหาที่เกิดขึ้นบ่อยครั้งในการใช้โปรแกรมภาษาซีก็คือพอยน์เตอร์ซึ่งสามารถแก้ปัญหานี้ได้โดย เมื่อมีการใช้งานกับพอยน์เตอร์ทุกครั้งเราจะต้องให้มันทำงานในส่วนของ Heap memory พยายามหลีกเลี่ยงการใช้ตัวแปรพอยน์เตอร์แบบสแตติกโดยเด็ดขาด และเมื่อเลิกใช้งาน ตัวแปรชนิดไดนามิกแล้วก็จะต้องคืนพื้นที่ให้กับระบบ เพื่อที่ระบบจะสามารถนำไปใช้กับโพรเซส อื่นได้

7.5 การสร้างฟังก์ชันที่ใช้งานจากการสำรวจหน้าที่การใช้งานของผู้ใช้

ระบบไดเรกทอรีได้นำมาใช้เก็บข้อมูลบางส่วนของเอเจนต์เป็นข้อมูลที่มีการเปลี่ยนแปลงน้อย โดยในส่วนของเอเจนต์ที่มีการใช้งานจะใช้ในส่วนของเพิ่มเอนทรีซึ่งมีรายละเอียดของตัวเองเข้าไปในระบบไดเรกทอรีเมื่อทีการติดตั้งและจะมีการแก้ไขข้อมูล เช่นสถานะของตัวเอง ดังนั้นฟังก์ชันที่ใช้ในส่วนของเอเจนต์นี้ก็คือ Addentry(), ModifyEntry() และ ModifyRDN()

ในส่วนของตัวเมเนเจอร์ (manager) จะมีความสามารถมากกว่าตัวเอเจนต์ นั่นคือสามารถลบ เพิ่ม แก้ไขและค้นหาข้อมูลที่อยู่ในระบบไดเรกทอรีได้เราจึงได้ฟังก์ชันที่เพิ่มขึ้นมาจากส่วนของเอเจนต์คือ DelEntry(), SearchEntry() และ CompareEntry() รวมทั้งหมด 6 ฟังก์ชันด้วยกัน

ติดตั้งโปรแกรม และ คอนฟิกูเรชันในส่วนที่จำเป็น

เมื่อเราดาวโหลด โปรแกรม Open LDAP มาติดตั้งเรียบร้อยแล้ว (รายละเอียด การติดตั้งจะอยู่ภาคผนวก) ก็จะต้องเซ็ตค่า ในคอนฟิก ไฟล์ พร้อมทั้งออกแบบ สกีมา ตามที่เราต้องการให้เรียบร้อย พร้อมทั้งทดสอบการทำงานเบื้องต้นของ LDAP เซิร์ฟเวอร์ จากทูลที่ให้มา เพื่อให้แน่ใจว่า สามารถใช้งานได้

การศึกษาฟังก์ชันพื้นฐาน ของ LDAP APIs

เมื่อได้กำหนดฟังก์ชันที่จำเป็นในการใช้งานเรียบร้อยแล้วเราก็ต้องศึกษา API ของ LDAP ให้ครอบคลุมทุกฟังก์ชันเสียก่อนโดยสามารถดาวน์โหลดโปรแกรมตัวอย่างได้จาก www.openldap.org แล้วนำมาศึกษาทีละฟังก์ชัน ว่า มีการทำงานอย่างไรบ้าง ผลที่ได้ตรงกับข้อมูลที่ส่งไปหรือไม่ เมื่อศึกษาการทำงาน และตัวเลือกของแต่ละฟังก์ชันอย่างละเอียดรวมถึงเห็นปัญหาที่เกิดขึ้นแล้ว ก็สามารถที่จะทดลองอิมพลีเมนต์ API เพื่อใช้งานได้

7.6 ส่วนประกอบของ API functions

ในการอิมพลีเมนต์ API ที่ใช้งานจะมีฟังก์ชันที่ใช้งานทั้งหมด 8 ฟังก์ชันด้วยกันโดยจะอธิบายรูปแบบ และ การทำงานของแต่ละฟังก์ชัน ดังนี้

7.6.1 AddEntry

เพิ่ม entry เข้าไปในระบบไดเรกทอรีโดยผู้ใช้จะต้องส่งข้อมูลที่ต้องการเก็บให้ครบตามที่ได้กำหนดไว้ในสกีมาให้ครบ ซึ่งในแต่ละประเภทก็จะแตกต่างกันไป

รูปแบบ

```
int AddEntry (
    int agtype,
    char *str);
```

พารามิเตอร์

agtype

ชนิดของเอเจนต์ที่เราต้องการเพิ่มข้อมูลโดยในการออกแบบสกีมาเพื่อการใช้งานขั้นต้นมีอยู่ด้วยกัน 3 ประเภท คือ Firewall, Ststeful และ NIDS สามารถใส่พารามิเตอร์ 1, 2, 3 ตามลำดับ ถ้านอกจากนี้ฟังก์ชันจะแสดงข้อผิดพลาด

str

เป็นสตริงข้อมูลที่ใช้เพิ่มลงไปในระบบไดเรกทอรีโดยรูปแบบของการใส่สตริงคือจะเริ่มด้วยชื่อของแอตทริบิวต์ตามด้วยค่าของแอตทริบิวต์นั้น ค่าทั้งสองจะคั่นด้วยเครื่องหมาย (:) ในส่วนของเครื่องหมายที่ใช้แบ่งระหว่าง แอตทริบิวต์คือ (,) ดังตัวอย่าง

AgentID:FW1,IP: 161.246.7.10, Destination: 161.246.9.15

ในกรณีที่มีหลายค่าในแอตทริบิวต์เดียวจะคั่นแต่ละค่าด้วยเครื่องหมาย (:) สามารถเขียนได้ดังนี้

IP:161.246.5.11:161.246.5.12

การคืนค่า

0 succes

> 0 error , เราสามารถตรวจสอบข้อผิดพลาดที่เป็นสตริงได้โดยนำค่า integer ที่คืนมา ผ่านฟังก์ชัน err2string() อีกทีหนึ่ง

7.6.2 ModifyEntry

เปลี่ยนแปลงค่าของแอตทริบิวต์ในระบบไดเรกทอรีที่เราต้องการ โดยจะมีโหมดของการเปลี่ยนแปลง อยู่ 3 ประเภทด้วยกัน คือ การเพิ่ม, การลบและการแทนที่แอตทริบิวต์

รูปแบบ

```
int ModifyEntry(
    char  *rdn,
    int    modtype,
    char  *modi);
```

พารามิเตอร์

rdn

ชื่อของ rdn ของเอนทรี ที่เราต้องการเปลี่ยนแปลง

modtype

เป็นประเภทของเอเจนต์และประเภทของการเปลี่ยนแปลง ซึ่งมี 3 ประเภทด้วยกันคือ replace, add, delete ค่าที่สามารถเป็นไปได้ มีดังนี้

11	:	Firewall-Replace
12	:	Firewall-Add
13	:	Firewall- Delete
21	:	Stateful-Replace
22	:	Stateful-Add
23	:	Stateful-Delete
31	:	NIDS-Replace
32	:	NIDS-Add
33	:	NIDS-Delete

modi

เป็นสตริงข้อมูลที่ใช้เปลี่ยนแปลงในระบบไดเรกทอรีโดยรูปแบบของการใส่สตริงคือ จะเริ่มด้วยชื่อของแอตทริบิวต์ตามด้วยค่าของแอตทริบิวต์นั้น ค่าทั้งสองจะคั่นด้วยเครื่องหมาย (:) ในส่วนของเครื่องหมายที่ใช้แบ่งระหว่าง แอตทริบิวต์คือ (,) ดังตัวอย่าง

AgentID:FW1,IP: 161.246.7.10, Destination: 161.246.9.15

ในกรณีที่มีหลายค่าในแอตทริบิวต์เดียวจะคั่นแต่ละค่าด้วยเครื่องหมาย (:) สามารถเขียนได้ดังนี้

IP:161.246.5.11:161.246.5.12

การคืนค่า

0 succes

> 0 error , เราสามารถตรวจสอบข้อผิดพลาดที่เป็นสตริงได้โดยนำค่า integer ที่คืนมา ผ่านฟังก์ชัน err2string() อีกทีหนึ่ง

7.6.3 DelEntry

ลบ entry ออกจากระบบไดเรกทอรี

รูปแบบ

```
int  DelEntry (
      char *rdn);
```

พารามิเตอร์

rdn

ชื่อของ rdn ของเอนทรีที่เราต้องการเปลี่ยนแปลง

การคืนค่า

0 succes

> 0 error , เราสามารถตรวจสอบข้อผิดพลาดที่เป็นสตริงได้ โดยนำค่า integer ที่คืนมา ผ่านฟังก์ชัน err2string() อีกทีหนึ่ง

7.6.4 SearchEntry

ค้นหา entry ในระบบไดเรกทอรี ตามเงื่อนไขที่เรากำหนดซึ่งต้องเป็นไปตามมาตรฐานที่ ระบบไดเรกทอรีรองรับ

รูปแบบ

```
int SearchEntry(
      char *schBase,
      int   level,
      char *filter,
      char *sort,
      char **result);
```

พารามิเตอร์

schBase

เป็นสตริงข้อมูล ที่ระบุตำแหน่งที่ต้องการเริ่มค้นหา ซึ่งจะเป็นชื่อ DN ของ เอนทรีที่เริ่มต้น Node นั่นเอง

level

เป็นตัวเลขที่ใช้ระบุขอบเขตของการค้นหา โดยจะมีด้วยกัน 3 ระดับ คือ

1. Base คือ ตัวของมันเอง
2. One Level คือ มีการค้นหา ข้อมูล แค่ระดับล่างของตัวมันเอง อีก 1 ระดับเท่านั้น
3. Sub Tree จะค้นหาทุกเอนทรี ที่อยู่ด้านล่างของตัวมันทั้งหมด

filter

เป็นสตริงข้อมูล ที่ใช้ กรองข้อมูล ตามเงื่อนไขที่ได้กำหนด โดยรูปแบบทั้งหมดของการกรองข้อมูล ดูได้จาก บทที่ 3

sort

เป็นสตริงข้อมูล ที่ระบุแอตทริบิวต์ที่ใช้เรียงลำดับข้อมูล ในที่นี้ จะกำหนดเป็น เรียงจากน้อยไปหามาก

result

เป็นสตริงข้อมูล ผลลัพธ์ ที่ได้จากการค้นหา ตามเงื่อนไขดังกล่าว ที่สามารถนำไปแสดงผล ออก มอนิเตอร์ ได้

การคืนค่า

0 succes

> 0 error , เราสามารถตรวจสอบข้อผิดพลาดที่เป็นสตริงได้ โดยนำค่า integer ที่คืนมา ผ่านฟังก์ชัน err2string() อีกทีหนึ่ง

7.6.5 CompareEntry

เปรียบเทียบเอนทรีในไดเรกทอรีตามเงื่อนไขที่กำหนดซึ่งจะใช้ในการตรวจว่ามีเอนทรีนั้นๆอยู่ในระบบไดเรกทอรีหรือไม่ โดยการใส่ข้อมูลของเอนทรีนั้นเข้าไป

รูปแบบ

```
int CompareEntry(
        char *RDN,
        char *attribute,
        char *value,
        char **result);
```

พารามิเตอร์

rdn

ชื่อของ rdn ของเอนทรีที่เราต้องการเปลี่ยนแปลง

attribute

เป็นสตริงข้อมูล ที่ใช้ บอกชื่อของแอตทริบิวต์ที่ต้องการเปรียบเทียบ

value

เป็นสตริงข้อมูลของค่าของแอตทริบิวต์ที่ต้องการเปรียบเทียบในระบบไดเรกทอรี

result

เป็นสตริงข้อมูลผลลัพธ์ที่ได้จากการค้นหาตามเงื่อนไขดังกล่าวที่สามารถนำไปแสดงผลออกมอนิเตอร์ได้

การคืนค่า

0 succes

> 0 error , เราสามารถตรวจสอบข้อผิดพลาดที่เป็นสตริงได้โดยนำค่า integer ที่คืนมา ผ่านฟังก์ชัน err2string() อีกทีหนึ่ง

7.6.6 ModifyRDN

เปลี่ยนแปลง RDN ของเอนทรีแต่มีข้อแม้ว่าตัวที่เปลี่ยนต้องอยู่ Leaf Node เท่านั้น

รูปแบบ

```
int ModifyRDN(
        char *modrdn,
        char *moddn);
```

พารามิเตอร์

modrdn

เป็นสตริงข้อมูลที่ระบุชื่อของ RDN ค่าใหม่ของเอนทรีที่เราต้องการเปลี่ยนแปลง

moddn

เป็นสตริงข้อมูลที่ระบุชื่อของ DN ของเอนทรีที่เราต้องการเปลี่ยนแปลง

การคืนค่า

0 succes

> 0 error , เราสามารถตรวจสอบข้อผิดพลาดที่เป็นสตริงได้โดยนำค่า integer ที่คืนมา ผ่านฟังก์ชัน err2string() อีกทีหนึ่ง

** เอนทรีที่สามารถเปลี่ยนแปลงได้จะต้องเป็นเอนทรีที่อยู่ Leaf Node เท่านั้นเอนทรีที่ไม่ได้อยู่ Leaf Node ไม่สามารถเปลี่ยนแปลง RDN ได้

7.6.7 err2string

เปลี่ยนหมายเลขที่ส่งเข้าไปในฟังก์ชันให้อยู่ในรูปของสตริงแล้วคืนค่ากลับมา

รูปแบบ

```
char *err2string(
    int errcode);
```

พารามิเตอร์

errcode

เป็น errcode ที่ได้จากการคืนค่าจากฟังก์ชันต่างๆ โดยเมื่อมีการติดต่อกับ LDAP เซิร์ฟเวอร์ก็จะมีการคืนค่ามาในรูปแบบของตัวเลข

การคืนค่า

0 – 90 จะคืนค่าสตริง จากฟังก์ชัน ldap_err2string(str);

101 LDAP session initialization failed

102 Not enough memory

103 Invalid agent type / modify type

7.3.8 GetValue

นำค่าของแอตทริบิวต์ของเอนทรีที่เราต้องการออกมา

รูปแบบ

```
char *GetValue(
    char *Base,
    char *filter,
    char *att_name,
    char **result);
```

<u>พารามิเตอร์</u>

Base

เป็นสตริงข้อมูลที่ระบุตำแหน่งที่ต้องการเริ่มค้นหาซึ่งจะเป็นชื่อ DN ของเอนทรีที่เริ่มต้น Node นั่นเอง

filter

เป็นสตริงข้อมูลที่ใช้กรองข้อมูลตามเงื่อนไขที่ได้กำหนดโดยรูปแบบทั้งหมดของการกรองข้อมูล ดูได้จาก บทที่ 3

att_name

เป็นสตริงข้อมูลที่ระบุแอตทริบิวต์ที่ต้องการนำค่าออกมา

result

เป็นสตริงข้อมูลผลลัพธ์ที่ได้จากการค้นหา ตามเงื่อนไขดังกล่าวที่สามารถนำไปแสดงผล ออกมอนิเตอร์ได้

<u>การคืนค่า</u>

ค่าของแอตทริบิวต์ของเอนทรี ที่เราต้องการ จะแปลงให้อยู่ในรูปของสตริง

บทที่ 8
การทดสอบและผลการใช้งาน

8.1 การทดสอบ API functions

เมื่อได้อิมพลีเมนต์ API function แล้ว การทดลองใช้งาน จะจำลอง โปรแกรมมาเรียกฟังก์ชันที่ได้ออกแบบมาทุกฟังก์ชันแล้วดูผลลัพท์ว่าได้ผลที่ถูกต้องหรือไม่พร้อมกับตรวจสอบ error ต่างๆ ว่าตรงกับข้อผิดพลาดที่เกิดขึ้นหรือไม่ โปรแกรมที่ได้สร้างขึ้นมาจะใช้สำหรับระบบปฏิบัติการ UNIX ดังรูป

8.1.1 ฟังก์ชัน AddEntry();

การทดสอบการใช้งานฟังก์ชัน AddEntry() นั้นเราจะทำการเรียกฟังก์ชัน AddEntry() โดยส่งพารามิเตอร์ก็คือประเภทของเอเจนท์และสตริงที่ระบุแอตทริบิวต์ต่างๆ ที่อยู่ในเอนทรีด้วย ที่สำคัญหนึ่งในจำนวนแอตทริบิวต์นั้นจะต้องมีการระบุ ObjectClass เพื่อที่ LDAP เซิร์ฟเวอร์จะตรวจสอบแต่ละแอตทริบิวต์ว่ามีความถูกต้องหรือไม่

```
/*Test AddEntry() */
#include "ldapinit.h"
int main(void)
{
char *str = "AgentID: FW11,IP: 161.246.7.14,StartTime: Mar:9:2002,04:35:57
            Refresh: Mar:9:2002,04:45:58,Source: Any,
            Destination: 161.246.9.20,Service: http,Action: Accept
           ,objectClass: firewall";

printf("\n\nStart test... 'AddEntry(1, str)'"
       "function\nSend parameter:\nstr:  %s\n",str);

printf("\nReturn Value:\n%s\n\n",err2string(AddEntry(1, str)));

return 0;
}
```

รูปที่ 8-1 แสดงโปรแกรมทดสอบฟังก์ชัน AddEntry();

```
Start test...  'AddEntry(1, str)' function
Send parameter:
str:  AgentID:FW11,IP:161.246.7.114,StartTime:Mar-9-2002-04-35-57,Refresh:Ma
ation:161.246.9.20,Service:http,Action:Accept,objectClass:firewall

Return Value:
Success
```

รูปที่ 8-2 แสดงผลการทดสอบฟังก์ชัน AddEntry();

8.1.2 ฟังก์ชัน CompareEntry();

การทดสอบการใช้งานฟังก์ชัน CompareEntry() นั้นเราจะส่งพารามิเตอร์ที่ระบุ DN และเงื่อนไขที่เราต้องการทราบในที่นี้ คือต้องการทราบว่า แอตทริบิวต์ IP ค่า 161.246.203.881 มีอยู่ใน LDAP เซิร์ฟเวอร์หรือไม่ ถ้ามีก็จะคืนค่า True มาให้

```
/*Test CompareEntry() */
#include "ldapinit.h"
int main(void)
{

char *compareDN= "AgentID=FW11,AgentType=Firewall,o=kmitl";
char  *attribute = "IP";
char *value= "161.246.7.114";

printf("\n\nStart test...  'CompareEntry(compareDN, attribute, value)'"
       "function\nSend parameter:\ncompareDN:  %s\nattribute:  %s\nvalue:
        %s\n",compareDN,attribute,value);

printf("\nReturn Value:\n%s\n\n",err2string(CompareEntry( compareDN,
         attribute, &value)));

printf("%s\n",value);

return 0;

}
```

รูปที่ 8-3 แสดงโปรแกรมทดสอบฟังก์ชัน CompareEntry();

```
Start test...  'CompareEntry(1, compareDN, attribute, value)' function
Send parameter:
compareDN:  AgentID=FW11,AgentType=Firewall,o=kmitl
attribute:  IP
value:  161.246.7.114

Return Value:
Compare true
```

รูปที่ 8-4 แสดงผลการทดสอบฟังก์ชัน CompareEntry();

8.1.3 ฟังก์ชัน ModifyEntry();

การทดสอบการใช้งานฟังก์ชัน ModifyEntry() นั้นจะส่งพารามิเตอร์ rdn เพื่อใช้บอกชื่อของเอเจนต์ และ modi สำหรับบอกค่าที่ต้องการเปลี่ยนแปลง ในที่นี้ คือ IP:161.246.7.115 และพารามิเตอร์ตัวสุดท้ายก็คือ ตัวบ่งบอกถึงชนิดของเอเจนต์และโหมดการทำงานของฟังก์ชันในที่นี้ คือ 12 ซึ่งหมายถึงเป็นเอเจนต์แบบไฟร์วอลและโหมดการทำงานเป็นแบบ LDAP_ADD

```
/*Test ModifyEntry()*/
#include "ldapinit.h"
int main(void)
{

char *rdn = "AgentID=FW11", *modi = "IP:161.246.7.115";

printf("\n\nStart test...  'ModifyEntry( 12, rdn, modi)'"
      "function\nSend parameter:\nrdn:  %s\nmodi:  %s\n",rdn,modi);

printf("\nReturn Value:\n%s\n\n",err2string(ModifyEntry(12, rdn,  modi)));

return 0;
}
```

รูปที่ 8-5 แสดงโปรแกรมทดสอบฟังก์ชัน ModifyEntry();

```
Start test...  'ModifyEntry( 12, rdn, modi)' function
Send parameter:
rdn:  AgentID=FW11
modi:  IP:161.246.7.115

Return Value:
Success
```

รูปที่ 8-6 แสดงผลการทดสอบฟังก์ชัน ModifyEntry();

8.1.4 ฟังก์ชัน SearchEntry();

การทดสอบการใช้งานฟังก์ชัน SearchEntry() นั้นจะส่งพารามิเตอร์ schBase ซึ่งจะระบุจุดเริ่มต้นของการค้นหาข้อมูล พารามิเตอร์ตัวต่อไปก็คือฟิลเตอร์ซึ่งเป็นเงื่อนไขของการค้นหาแต่ละครั้ง สุดท้ายก็คือ sort ซึ่งจะใช้กำหนดว่า จะเรียงข้อมูลตามแอตทริบิวต์ไหนและผลลัพธ์ทั้งหมดจะอยู่ที่ result

```
/*Test SearchEntry()*/
#include "ldapinit.h"
int main(void)
{

char *schBase = "o=kmitl", *filter = "(IP=161.246.203.881)", *sort = "AgentID";
char *result;

printf("\n\nStart test... 'SearchEntry( schBase, 1, filter, sort)'"
       "function\nSend parameter:\nschBase:  %s\nfilter:  %s\nsort:
       %s\n",schBase,filter,sort);

printf("\nReturn Value:\n%s\n\n",err2string(SearchEntry( schBase, 1, filter, sort,
       &result)));

printf("%s",result);

return 0;

}
```

รูปที่ 8-7 แสดง โปรแกรมทดสอบฟังก์ชัน SearchEntry();

```
Start test...  'SearchEntry(schBase, 1, filter, sort)' function
Send parameter:
schBase:  o=kmitl
filter:  (AgentID=FW11)
sort:  AgentID

Return Value:
Success


dn: AgentID=FW11,AgentType=Firewall,o=KMITL
    AgentID: FW11
    IP: 161.246.7.114
    IP: 161.246.7.115
    StartTime: Mar-9-2002-04-35-57
    Refresh: Mar-9-2002-04-45-58
    Source: Any
    Destination: 161.246.9.20
    Service: http
    Action: Accept
    objectClass: firewall

Search completed successfully.
Entries  returned: 1
```

รูปที่ 8-8 แสดงผลการทดสอบฟังก์ชัน SearchEntry();

8.1.5 ฟังก์ชัน ModifyRDN();

การทดสอบการใช้งานฟังก์ชัน ModifyRDN() นั้นจะเป็นการเปลี่ยนแปลง RDN โดยส่งพารามิเตอร์ rdn ซึ่งจะระบุค่าใหม่ของ RDN พารามิเตอร์ตัวต่อไปก็คือ modi จะเป็นค่า DN ค่าเดิม

```
/*Test ModifyRDN()*/
#include "ldapinit.h"
int main(void)
{
char *rdn = "AgentID=FW1";
char *modi = "ip:161.246.203.882:161.246.5.89";

printf("\n\nStart test...  'ModifyRDN( modrdn, moddn)'"
      "function\nSend parameter:\nmodrdn:  %s\nmoddn:  %s\n",modrdn,moddn);

printf("\nReturn Value:\n%s\n\n",err2string(ModifyRDN( modrdn, moddn)));

return 0;

}
```

รูปที่ 8-9 แสดงโปรแกรมทดสอบฟังก์ชัน ModifyRDN();

```
Start test...  'ModifyEntry( 12, rdn, modi)' function
Send parameter:
rdn:  AgentID=FW11
modi:  IP:161.246.7.115

Return Value:
Success
```

รูปที่ 8-10 แสดงผลการทดสอบฟังก์ชัน ModifyRDN();

8.1.6 ฟังก์ชัน DelEntry();

การทดสอบการใช้งานฟังก์ชัน DeleteEntry() นั้นจะเป็นการลบเอนทรีออกจากไดเรคทอรีโดยส่งพารามิเตอร์ deleteDN ซึ่งจะระบุค่าของ DN ของเอนทรีที่ต้องการลบ

```
/*Test DelEntry()*/
#include "ldapinit.h"
int main(void)
{
char *deleteDN= "AgentID=FW1,AgentType=Firewall,o=kmitl";

printf("\n\nStart test...  'DelEntry(deleteDN)'"
      "function\nSend parameter:\ndeleteDN:  %s\n",deleteDN);
printf("\nReturn Value:\n%s\n\n",err2string(DelEntry( deleteDN)));

return 0;

}
```

รูป 8-11แสดง โปรแกรมทดสอบฟังก์ชัน DeleteEntry();

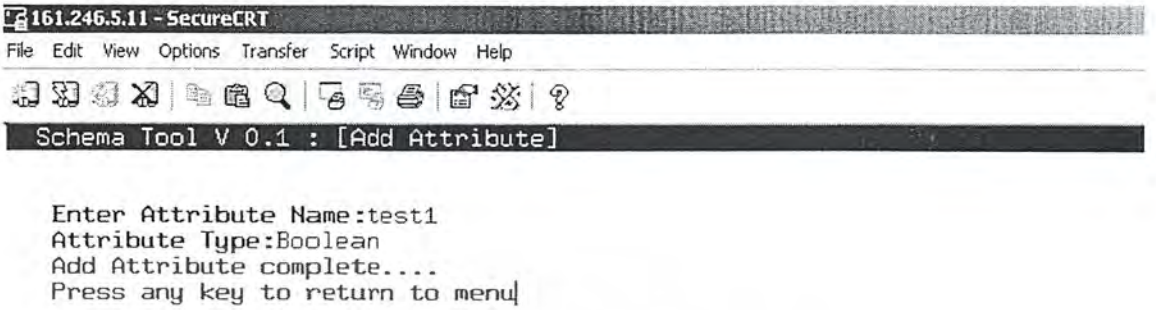
```
Start test...  'DelEntry(1, deleteDN)' function
Send parameter:
deleteDN:  AgentID=FW13,AgentType=Firewall,o=kmitl

Return Value:
Success
```

รูป 8-12แสดงผลการทดสอบฟังก์ชัน DeleteEntry();

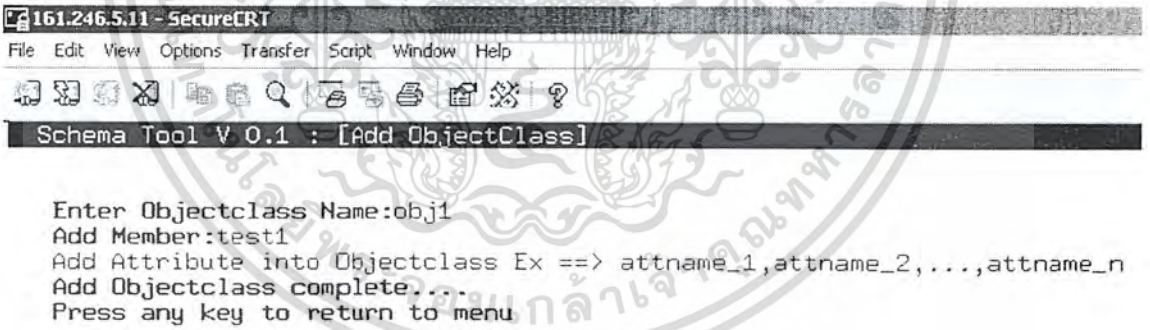
8.2 การทดสอบโปรแกรม Schema Tool

ทดสอบเพิ่มแอตทริบิวต์ test1 เข้าไปในสกีมา



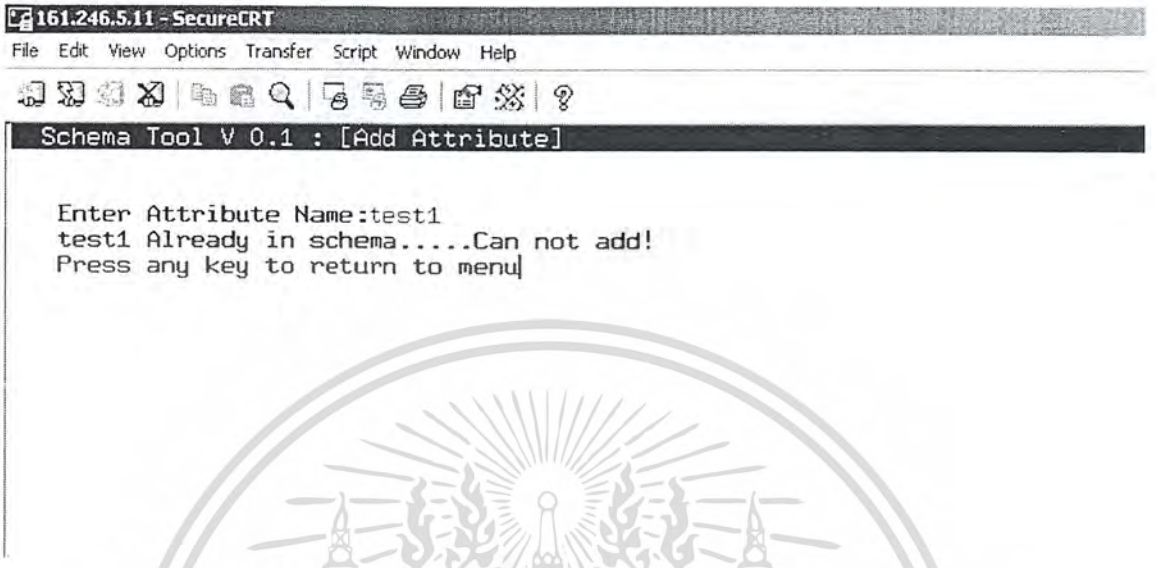
รูปที่ 8-13 แสดงการเพิ่มแอตทริบิวต์ test1 เข้าไปในสกีมา

ทดสอบเพิ่มออบเจกคลาส obj1 เข้าไปในสกีมา



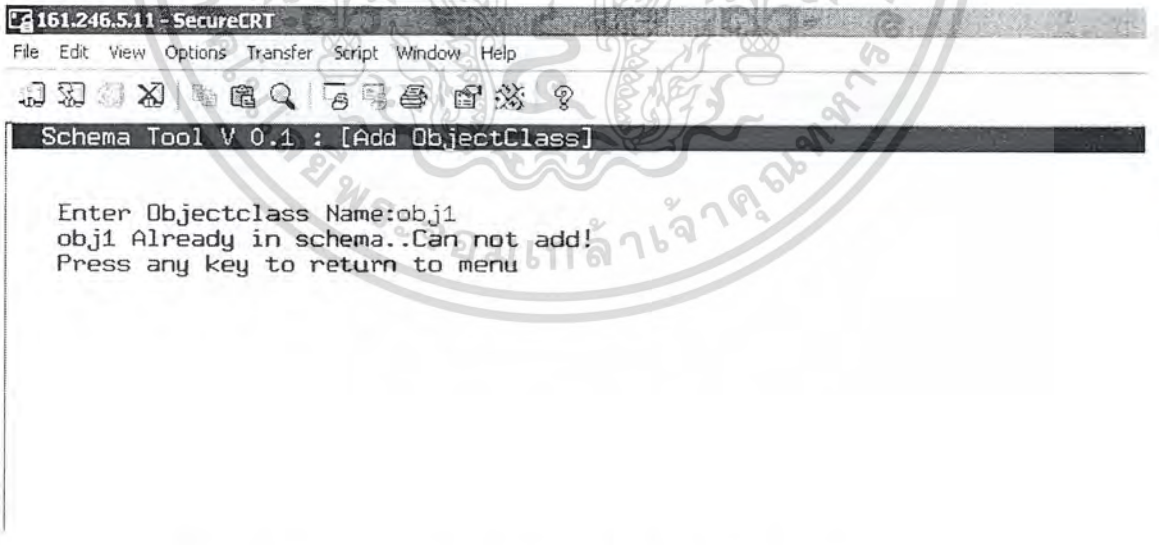
รูปที่ 8-14 แสดงผลการเพิ่มออบเจกต์คลาส obj1 เข้าไปในสกีมา

ทดสอบเพิ่มแอตทริบิวต์ test1 ซ้ำอีกครั้งซึ่งปรากฏว่าไม่สามารถเพิ่มได้เนื่องจากว่าได้ทำการเพิ่มเข้าไปในสกีมาแล้ว



รูปที่ 8-15 แสดงผลการเพิ่มแอตทริบิวต์ test1 ซ้ำกับที่มีในสกีมา

ทดสอบเพิ่มออบเจกต์คลาส obj1 ซ้ำอีกครั้งซึ่งปรากฏว่าไม่สามารถเพิ่มได้เนื่องจากว่าได้ทำการเพิ่มเข้าไปในสกีมาแล้ว



รูปที่ 8-16 แสดงผลการเพิ่มออบเจกต์คลาส obj1 ซ้ำกับที่มีในสกีมา

ผลการเพิ่มแอตทริบิวต์ test1 และออบเจกต์คลาส obj1 จะได้ไฟล์สกีมาที่มีรูปแบบดังนี้

```
#%>Sec1;
#%>LastObjId=1;
#%>LastAttId=1;
#%>ObjNum=1;
#%>AttNum=1;
#%>End Sec1;

#%>Sec2;
#%>PrefixObjId=1.3.6.1.4.3;
#%>PrefixAttId=1.3.6.1.4.4;
#%>End Sec2;

#%>Att Section;

#s_test1;
attributetype (1.3.6.1.4.4.1 NAME 'test1'
        EQUALITY IntegerMatch
        ORDERING caseIgnoreOrderingMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.7)
#e_test1;
#%>End Att;

#%>Obj Section;

#s_obj1;
objectclass (1.3.6.1.4.3.1 Name 'obj1'
        SUP top STRUCTURAL
        MUST (test1)
        MAY description)
#e_obj1;
#%>End Obj;


#%>View Section;
#--------------
#Attribute:Type
#--------------
# test1:Integer
#end_att
#-------------------
#ObjectClass:Member
#-------------------
# obj1:test1
#end_obj
#%>End View;
```

รูปที่ 8-17 แสดงไฟล์สกีมาที่ได้จากการเพิ่มแอตทริบิวต์ test1 และออบเจกต์คลาส obj1

บทที่ 9
บทสรุปและวิจารณ์

9.1 บทสรุปและวิจารณ์

เราสามารถนำระบบไดเรกทอรีมาใช้กับข้อมูลที่มีการเปลี่ยนแปลงน้อยแต่เป็นข้อมูลที่มีความสำคัญโดยเฉพาะต่อระบบรักษาความปลอดภัยได้ อีกทั้งยังทำให้ง่ายต่อการตรวจสอบว่าโฮสต์ (Host) ไหนที่มีการใช้บริการเอเจนต์ความปลอดภัย (Security Agent) อยู่ ซึ่งในโครงงานนี้ได้พัฒนาในระดับของการสร้าง API เพื่อใช้ในการติดต่อกับระบบไดเรกทอรีให้ใช้งานง่ายขึ้นและส่วนของสกีมาต้นแบบในการเก็บข้อมูล ซึ่งในอนาคตหากนำส่วนที่พัฒนาขึ้นมานี้ไปประยุกต์ใช้งานในงานด้านการเก็บข้อมูลต่างๆ ที่ต้องการทราบเกี่ยวกับเอเจนต์ความปลอดภัย ซึ่งสามารถช่วยอำนวยความสะดวกให้กับผู้บริหารระบบและเป็นเครื่องมือที่มีประสิทธิภาพตัวหนึ่งเลยทีเดียว

ในโครงงานนี้เป็นสร้างต้นแบบการนำเอาระบบไดเรกทอรีมาประยุกต์ใช้งาน ซึ่งกว่าจะออกมาให้อยู่ในรูปที่ใช้งานได้ผู้จัดทำจะต้องศึกษาการเก็บข้อมูลในรูปแบบต่างๆ โดยการเปรียบเทียบระบบแบบต่างๆ เช่น XML, SNMP รวมทั้ง LDAP โดยได้เลือกใช้ LDAP เพราะถือว่าสามารถนำมาใช้งานได้อย่างครอบคลุมและสามารถพัฒนาต่อได้ อีกทั้งฟังก์ชันการใช้งานต่างๆ ก็ใช้ง่ายและมีประสิทธิภาพอีกด้วย ซึ่งยอมรับว่ามีข้อมูลที่ใช้อ้างอิง เพื่อที่จะนำมาอิมพลีเมนต์น้อยมาก อย่างไรก็ตามเมื่อได้ทำให้ระบบใช้งานได้ในระดับหนึ่งแล้ว ผู้ที่มาพัฒนาต่อก็สามารถศึกษาและพัฒนาได้อย่างไม่ยากนัก

9.2 ปัญหาที่เกิดขึ้น

1. เอกสารอ้างอิงส่วนใหญ่ที่ค้นคว้าจากอินเตอร์เน็ตมีรายละเอียดค่อนข้างน้อย เนื้อหาบางส่วนที่เป็นพื้นฐานก็ไม่ได้กล่าวถึง หนังสือที่ใช้อ้างอิงเกี่ยวกับ LDAP มีน้อยทำให้การค้นคว้าและหาข้อมูลค่อนข้างเสียเวลา
2. การติดตั้งและใช้งาน Openldap ค่อนข้างยุ่งยากพอสมควร
3. การอิมพลีเมนต์มีปัญหาค่อนข้างมาก เพราะส่วนมากจะใช้ตัวแปรชนิดพอยน์เตอร์ของภาษาซีทำให้ใช้เวลานานในการอิมพลีเมนต์
4. การทำความเข้าใจในเรื่องของการออกแบบสกีมาค่อนข้างจะยุ่งยาก เนื่องจากหาตัวอย่างการใช้งานที่ครอบคลุมการออกแบบทั้งหมดค่อนข้างหายาก ดังนั้นทางผู้จัดทำจึงต้องอาศัยการลองผิดลองดูในการศึกษาการใช้งาน ซึ่งทำให้ค่อนข้างเสียเวลา

9.3 วิธีการแก้ไข

ปัญหาที่เกิดขึ้นส่วนใหญ่มักจะมาจากเรื่องของเอกสารที่ไม่ค่อยจะละเอียดเพียงพอทำให้เสียเวลาในการรวบรวมข้อมูลและเอกสาร แต่ก็อาศัยความอดทน ในการค้นหาข้อมูล และ การทดลองแบบลองผิดลองถูก ซึ่งการทำอย่างนี้เป็นการเสียเวลาอย่างมากและปัญหาหรือข้อมูลบางอย่างก็ยังไม่รู้อย่างละเอียดเพราะขาดเอกสารที่ดีและทันสมัย การแก้ไขในจุดนี้คงต้องหาเอกสารที่ใหม่และทันสมัยมาใช้เพราะเอกสารที่มี ก็ค่อนข้างเก่าและเอกสารที่แจกฟรีก็ไม่ค่อยละเอียดเท่าที่ควร

9.4 แนวทางการพัฒนา

การพัฒนาขั้นต่อไปคือ การนำเอา API ไปใช้งานในการสร้างตัวจัดการ (manager) และนำไปใช้ในตัวเอเจนต์ความปลอดภัยโดยนำไปใช้ในการเก็บข้อมูลสถานะและการทำงานต่างๆ ลงในไดเรกทอรี ตัวจัดการจะมีหน้าที่ในการติดตามการทำงานและสั่งงาน โดยทำการตรวจสอบการทำงานจากข้อมูลที่เก็บไว้ในไดเรกทอรี จากนั้นจะติดต่อและสั่งงานโดยตรงกับตัวโปรแกรมรักษาความปลอดภัย ซึ่งการใช้งานในรูปแบบนี้จะทำให้การจัดการและการสั่งงานโปรแกรมรักษาความปลอดภัยมีประสิทธิภาพมากขึ้น

นอกจากนี้ยังมีในส่วนของการพัฒนาความปลอดภัยในการรับส่งข้อมูลระหว่างโปรแกรมรักษาความปลอดภัยและระบบไดเรกทอรีให้มีการรับส่งข้อมูลที่มีความปลอดภัย

บรรณานุกรม

[1] Timothy A. Howes ,Mark C. Smith : " *Understanding and Deploying LDAP Directory Services* ", Netscape Communication Corporation 1st edition , 1999

[2] Timothy A. Howes : "*LDAP Programming and Implementation* " Netscape Communication Corporation 1st edition , 1999

[3] Heinz Johner , Larry Brown : " *Understanding LDAP* " , International Technical Support Organization : http://www.redbook.ibm.com

[4] Novell, Inc. : " *LDAP Library for C* " , Novell Developer Kit , September 2001 : http://www.novell.com/documentation

[5] Novell, Inc. : " *NDS Schema Reference* " , Novell Developer Kit , September 2001, 101-000222-001 : http://www.novell.com/documentation

[6] http://community.roxen.com/developers/idocs/drafts/draft-ietf-ldapext-ldap-c-api-04.html

[7] http://ldapman.org/articles/intro_to_ldap.html

[8] http://ldapman.org/articles/intro_to_ldap.html

[9] http://ldap-project.berkeley.edu/reports/introduction/LDAPintro.html

[10] http://ldap-project.berkeley.edu/reports/introduction/LDAPintro.html

[11] http://staff.pisoftware.com/bmarshal/publications/ldap_tut.html

[12] http://staff.pisoftware.com/bmarshal/publications/ldap_tut.html

[13] http://support.ca.com/tng22manuals/sdkprog/sdkprch07.html

[14] http://teets.cba.ufl.edu/ism6223f00/rafanarv/left_frame.htm

[15] http://webopedia.internet.com/TERM/L/LDAP.html

[16] http://webopedia.internet.com/TERM/L/LDAP.html

[17] http://www.cis.ksu.edu/~mahadev/ldap/sld001.htm

[17] http://www.cis.ksu.edu/~mahadev/ldap/sld001.htm

[18] http://www.cisco.com/warp/public/535/3.html

[19] http://www.ddri.com/Doc/SNMP_Overview.html

[20] http://www.ietf.org/

[21] http://www.ietf.org/

[22] http://www.ietf.org/ID.html

[23] http://www.ietf.org/ID.html

[24] http://www.ldapcentral.com/

[25] http://www.ldapcentral.com/

[26] http://www.openldap.org/faq/data/cache/74.html

[27] http://www.openldap.org/faq/data/cache/74.html

[28] http://www.openldap.org/faq/data/cache/75.html

[29] http://www.openldap.org/faq/data/cache/75.html

[30] http://www.rad.com/networks/1995/snmp/snmp.htm

[31] http://www.rfc-editor.org/

[32] http://www.umich.edu/~dirsvcs/ldap/

[33] http://www.umich.edu/~dirsvcs/ldap/

[34] RFC 1777: Lightweight Directory Access Protocol

[35] RFC 1778: The String Representation of Standard Attribute Syntaxes

[36] RFC 1779: A String Representation of Distinguished Names (obsolete)

[37] RFC 1823: The LDAP Application Programming Interface

[38] RFC 1959: An LDAP URL Format

[39] RFC 1960: A String Representation of LDAP Search Filters

[40] RFC 2044: UTF-8, a transformation format of Unicode and ISO 10646

[41] RFC 2251: Lightweight Directory Access Protocol (v3)

[42] RFC 2252: LDAPv3: Attribute Syntax Definitions

[43] RFC 2253: LDAPv3: UTF-8 String Representation of Distinguished Names

[44] RFC 2254: The String Representation of LDAP Search Filters

[45] RFC 2255: The LDAP URL Format

ภาคผนวก
การติดตั้ง OpenLdap 2.0.X

ขั้นตอนการติดตั้ง OpenLdap

1. ทำการแตกไฟล์ออกโดยใช้คำสั่ง

   gunzip – c openldap-VERSION.tgz | tar – xvfB

   หรือ

   tar – zxvf openldap-VERSION.tgz

   ไฟล์ทั้งหมดจะถูกเก็บไว้ในไดเรกทอรี openldap-VERSION
2. ทำการรันสคริปต์ configure โดยใช้คำสั่ง

   ./configure
3. ทำการ make โดยใช้คำสั่ง

   make depend

   make
4. ทำการทดสอบโดยใช้คำสั่ง

   make test
5. ทำการติดตั้งซึ่งจะต้องใช้สิทธิ์ของรูทโดยใช้คำสั่ง

   su root –c 'make install'

   ส่วนของโปรแกรมทั้งหมดจะติดตั้งลงในไดเรกทอรี /usr/local

หลังจากติดตั้งแล้วเราจะทำการกำหนดการใช้งาน โดยกำหนดจากไฟล์ slapd.conf ซึ่งปกติจะอยู่ใน /usr/local/etc/openldap ส่วนที่สำคัญที่ควรรู้มีดังนี้

```
include /usr/local/etc/openldap/schema/core.schema
database ldbm
suffix "dc=<MY-DOMAIN>,dc=<COM>"
rootdn "cn=Manager,dc=<MY-DOMAIN>,dc=<COM>"
rootpw secret
directory  /usr/local/var/openldap-ldbm
```

คำอธิบาย

include เป็นส่วนที่ใช้กำหนดไฟล์สกีมาที่เราต้องการจะใช้งานซึ่งสามารถกำหนดได้หลายไฟล์

database กำหนดรูปแบบของดาต้าเบสซึ่งในที่นี้ให้กำหนดเป็น ldbm ซึ่งเป็นค่าดีฟอล์ท

suffix กำหนดค่าของ root ของระบบไดเรกทอรี

rootdn กำหนดชื่อของ rootdn ซึ่งจะใช้ในขั้นตอนการ bind

rootpw กำหนดพาสเวิร์ดของ rootdn

directory กำหนดไดเรกทอรีที่เก็บไฟล์ดาต้าเบส

ให้กำหนดค่าในไฟล์ slapd.conf ดังนี้เพื่อการทดสอบการใช้งาน

include /usr/local/etc/openldap/schema/core.schema

database ldbm

suffix "o=kmitl"

rootdn "cn=admin,o=kmitl"

rootpw kadmin

directory /usr/local/var/openldap-ldbm

การทดสอบการใช้งาน OpenLDAP Server

1. ทำการเริ่มการทำงานของ SLAPD โดยพิมพ์คำสั่ง

su root –c /usr/local/libexec/slapd

2. ทดลองคำสั่ง ldapsearch โดยพิมพ์คำสั่ง

ldapsearch –x –b '' –s base '(objectclass=)' namingContexts*

```
toon@Isag11:~/LDIF$ ldapsearch -x -b '' -s base '(objectclass=*)' namingContexts
version: 2

#
# filter: (objectclass=*)
# requesting: namingContexts
#

#
dn:
namingContexts: o=kmitl

# search result
search: 2
result: 0 Success

# numResponses: 2
# numEntries: 1
toon@Isag11:~/LDIF$
```

รูปที่ 1 แสดงผลของคำสั่ง ldapsearch

3. ทำการเพิ่มเอนทรีเข้าไปในไดเรกทอรีด้วยคำสั่ง ldapadd โดยพิมพ์คำสั่ง

ldapadd –D "cn=admin,o=kmitl" –w "kadmin" –f test.ldif

โดยไฟล์ test.ldif ให้กำหนดดังนี้

dn: o=kmitl

o: kmitl

objectclass: organization

```
toon@Isag11:~/LDIF$ ldapadd -D "cn=admin,o=kmitl" -w "kadmin" -f test.ldif
adding new entry "o=kmitl"

toon@Isag11:~/LDIF$
```

รูปที่ 2 แสดงผลของคำสั่ง ldapadd

4. ทำการค้นหาเอนทรีที่เพิ่มเข้าไปโดยใช้คำสั่ง ldapsearch ดังนี้

ldapsearch –b "o=kmitl" –s sub

```
toon@Isag11:~/LDIF$ ldapsearch -b "o=kmitl" -s sub
version: 2

#
# filter: (objectclass=*)
# requesting: ALL
#

# kmitl
dn: o=kmitl
o: kmitl
objectClass: organization

# search result
search: 2
result: 0 Success

# numResponses: 2
# numEntries: 1
toon@Isag11:~/LDIF$
```

รูปที่ 3 แสดงผลของคำสั่ง ldapsearch

5. จบการทำงานของ SLAPD ด้วยคำสั่ง kill ดังนี้

kill –INT 'cat /usr/local/var/slapd.pid'