เครื่องมือซอฟต์แวร์ซึ่งอ้างอิงในแอมสำหรับสร้างเอ็กซ์เอ็มแอลสกีมา

NIAM CONCEPTUAL SCHEMA BASED SOFTWARE TOOL FOR XML SCHEMA GENERATION

ณฤดล จันทร์ควง

NARUDOL CHANKUANG

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์

บัณฑิตวิทยาลัย

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ.2548

ISBN 974-15-1955-9

NIAM CONCEPTUAL SCHEMA BASED SOFTWARE TOOL FOR XML SCHEMA GENERATION

NARUDOL CHANKUANG

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF

MASTER OF ENGINEERING IN COMPUTER ENGINEERING

SCHOOL OF GRADUATE STUDIES

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2005

ISBN 974-15-1955-9

สารบัญ(ต่อ)

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญรูป

รูปที่ หน้า

สารบัญรูป(ต่อ)

รูปที่ หน้า

สารบัญรูป(ต่อ)

รูปที่ หน้า

สารบัญรูป(ต่อ)

รูปที่ หน้า

สารบัญรูป(ต่อ)

รูปที่ หน้า

สารบัญตาราง

ตารางที่ หน้า

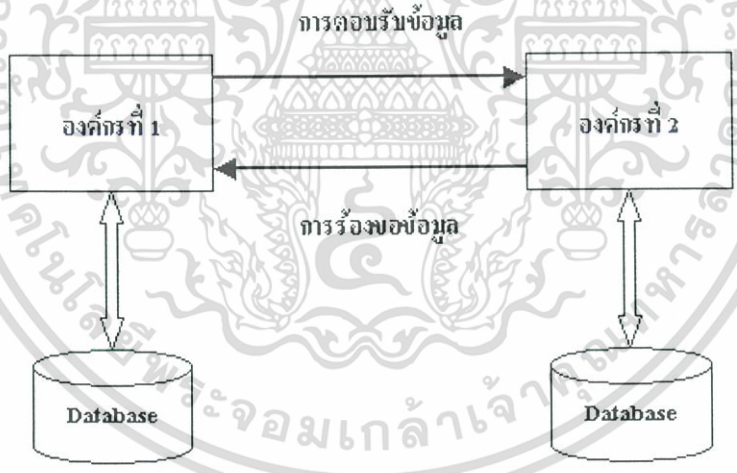
บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ปัจจุบันการสื่อสารข้อมูลผ่านระบบอินเตอร์เน็ตมีการนำภาษาเอ็กซ์เอ็มแอลมาใช้กันอย่างแพร่หลายเนื่องจากภาษาเอ็กซ์เอ็มแอลเป็นภาษาที่มีคุณสมบัติเด่นหลายประการ [1] ในการสื่อสารข้อมูลเช่น

ภาษาเอ็กซ์เอ็มแอลเป็นภาษาที่ใช้สำหรับสร้างข้อมูลที่สามารถอธิบายความหมายของตัวมันเองได้(Self –Description Data) ที่สามารถอ่านได้ทั้งมนุษย์(Human readable) และโปรแกรม(Machine readable) และ

เป็นภาษาที่ใช้สำหรับแลกเปลี่ยนข้อมูล(Data Exchange) เพราะภาษาเอ็กซ์เอ็มแอลเป็นภาษาที่มีลักษณะเป็นไฟล์ข้อความธรรมดา(Text file) จึงทำให้มีคุณสมบัติที่สามารถสื่อสารได้กับทุกแพลตฟอร์ม(Platform Independent)เช่น Window ,Unix และแพลตฟอร์มอื่นๆ รูปที่1.1 แสดงให้เห็นถึงลักษณะการแลกเปลี่ยนข้อมูลระหว่างองค์กร ซึ่งข้อมูลที่นำมาแลกเปลี่ยนกันอาจเป็นข้อมูลทั่วๆไป หรือเป็นข้อมูลระหว่างฐานข้อมูล



รูปที่ 1.1 แสดงรูปแบบการแลกเปลี่ยนข้อมูลระหว่างองค์กร

จากคุณสมบัติของภาษาเอ็กซ์เอ็มแอลที่กล่าวมาข้างต้น จะพบว่าภาษาเอ็กซ์เอ็มแอลมีบทบาทสำคัญในระบบอินเตอร์เน็ต และระบบการสื่อสารข้อมูลระหว่างองค์กรทางธุรกิจ(Business to Business) [2] การสื่อสารข้อมูลที่ต้องการความถูกต้องของข้อมูลจะต้องมีการกำหนดโครงสร้าง

ของภาษาเอ็กซ์เอ็มแอล(XML Schema) เอ็กซ์เอ็มแอลสกีมาจะเป็นสิ่งที่ประกันความถูกต้องของข้อมูลในเอกสารเอ็กซ์เอ็มแอล(XML Document)

การออกแบบเอ็กซ์เอ็มแอลสกีมาในปัจจุบันยังไม่มีการออกแบบในระดับแนวความคิด (Conceptual Level) [3] จึงทำให้เอ็กซ์เอ็มแอลสกีมาที่ได้ขาดความสมบูรณ์ครบถ้วน ไม่สะดวกในการการตรวจสอบความถูกต้องของสกีมา และที่สำคัญคือไม่สามารถประกันความซ้ำซ้อนของข้อมูลได้จึงทำให้การสื่อสารข้อมูลระหว่างองค์กร หรือระหว่างฐานข้อมูลขาดความถูกต้องสมบูรณ์

1.2 ความมุ่งหมายและวัตถุประสงค์ของงานวิจัย

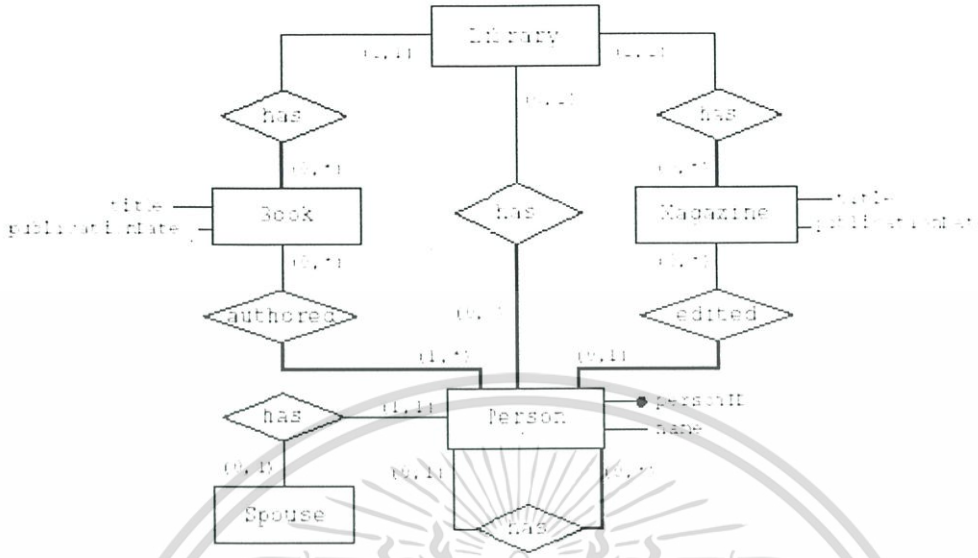
ความมุ่งหมายและวัตถุประสงค์ของงานวิจัยนี้คือการนำเสนอกรอบแนวคิดในการออกแบบเอ็กซ์เอ็มแอลสกีมา โดยใช้ไนแอมสกีมา(NIAM Schema) [4,6] มาเป็นกรอบแนวคิด เพื่อให้การออกแบบเอ็กซ์เอ็มแอลสกีมามีการกระทำอย่างเป็นขั้นตอน สามารถตรวจสอบได้ง่าย และที่สำคัญคือทำให้ได้เอ็กซ์เอ็มแอลสกีมาที่มีความถูกต้อง ครบถ้วนสมบูรณ์ นอกจากนี้แล้วยังประกันได้ว่าเอ็กซ์เอ็มแอลสกีมาที่ได้ไม่มีความซ้ำซ้อนของข้อมูลทำให้ข้อมูลเอ็กซ์เอ็มแอลที่สื่อสารระหว่างองค์กร หรือระหว่างฐานข้อมูลมีความน่าเชื่อถือสูง เพราะคุณสมบัติเด่นของไนแอมสกีมาเมื่อทำการแปลงเป็นรีเลชันแนลสกีมาแล้วจะทำให้ได้รูปแบบของนอร์มัลฟอร์มที่ 5 (5NF) [5,6] ซึ่งขอบเขตของงานวิจัยนี้มุ่งเน้นการประยุกต์ใช้งานเอ็กซ์เอ็มแอลสำหรับงานด้านการแลกเปลี่ยนข้อมูลระหว่างฐานข้อมูล(Data Exchange) จึงพิจารณาในเรื่องการกำจัดความซ้ำซ้อนของข้อมูลเป็นประเด็นสำคัญ

1.3 สมมุติฐานของการศึกษา และงานวิจัยที่เกี่ยวข้อง

ปัจจุบันมีงานวิจัยที่ศึกษาค้นคว้าเกี่ยวกับเอ็กซ์เอ็มแอลสกีมาจำนวนมาก และหลากหลายรูปแบบ และมีบริษัทผลิตซอฟต์แวร์ทูลที่ทำการสร้างเอ็กซ์เอ็มแอลสกีมาออกมาจำหน่ายจำนวนมากทั้งที่เป็นแบบไฟล์ข้อความ(Text Base) และเป็นแบบกราฟฟิค(Graphical Modeling) ซึ่งจะขอกล่าวถึงเฉพาะหัวข้อเรื่องที่เกี่ยวข้องกับงานวิจัยนี้คือการศึกษาค้นคว้าเกี่ยวกับการแปลงคอนเซ็บชวลสกีมาเป็นเอ็กซ์เอ็มแอลสกีมาเช่น การแปลงอีอีอาร์โมเดลเป็นเอ็กซ์เอ็มแอลสกีมา[7] การแปลงยูเอ็มแอลโมเดลเป็นเอ็กซ์เอ็มแอลสกีมา[8] การแปลงรีเลชันแนลสกีมาเป็นเอ็กซ์เอ็มแอลสกีมา[9] และเอ็กซ์เอ็มแอลสปาย(XML Spy) [10] ซึ่งเป็นซอฟต์แวร์ทูลที่นิยมใช้ในขณะนี้

การแปลงอีอีอาร์โมเดลเป็นเอ็กซ์เอ็มแอลสกีมา จะกล่าวถึงการขยายขีดความสามารถของอีอาร์โมเดลซึ่งเป็นโมเดลที่นิยมใช้ในการออกแบบฐานข้อมูล [11] เพื่อให้สามารถนำมาใช้แปลงเป็นเอ็กซ์เอ็มแอลสกีมาได้โดยการเพิ่มสัญลักษณ์บางตัวจากสัญลักษณ์ของอีอาร์โมเดลเดิม(Extend E-R) โดยในขั้นแรกจะทำการออกแบบอีอาร์โมเดลในระดับแนวคิดก่อน แล้วทำการแปลงจากอีอาร์

โมเดลเป็นเอ็กซ์เอ็มแอลสกีมาตามอัลกอริทึมของงานวิจัยนี้รูปที่ 1.2 แสดงอีอาร์โมเดล และ รูปที่ 1.3 แสดงเอ็กซ์เอ็มแอลสกีมาที่ได้จากการแปลงอีอาร์โมเดลรูปที่ 1.2

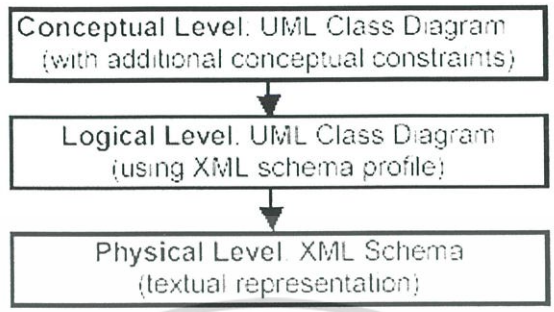


รูปที่ 1.2 แสดงการนำเสนอแนวคิดโดยใช้อีอาร์โมเดล

```
<schema xmlns:t='http://www.w3.org/namespace/'>
 <element name='Library'>
  <complexType>
   <sequence>
    <element ref='t:Book' minOccurs='0' maxOccurs='unbounded'/>
    <element ref='t:Magazine' minOccurs='0' maxOccurs='unbounded'/>
    <element ref='t:Person' minOccurs='0' maxOccurs='unbounded'/>
   </sequence>
  </complexType>
 </element>
 <element name='Book'>
  <complexType>
   <sequence>  <element ref='t:EMPTY'/>  </sequence>
   <attribute name='title' type='string' use='required'/>
   <attribute name='authors' type='IDREFS' use='required'/>
   <attribute name='publicationDate' type='date' use='required'/>
  </complexType>
 </element>
 <element name='Magazine'>
  <complexType>
   <sequence>  <element ref='t:EMPTY'/>  </sequence>
   <attribute name='title' type='string' use='required'/>
   <attribute name='editor' type='IDREF' use='optional'/>
   <attribute name='publicationDate' type='date' use='optional'/>
  </complexType>
 </element>
 <element name='Person'>
  <complexType>
   <sequence>
    <element ref='t:Spouse' minOccurs='0' maxOccurs='1/>
    <element ref='t:Person' minOccurs='0' maxOccurs='unbounded'/>
   </sequence>
   <attribute name='personID' type='ID' use='required'/>
   <attribute name='name' type='string' use='optional'/>
  </complexType>
 </element>
 <element name='Spouse'> <complexType mixed='true'/> </element>
</schema>
```

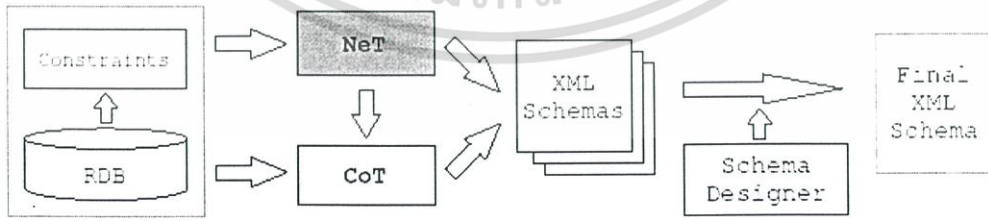
รูปที่ 1.3 แสดงเอ็กซ์เอ็มแอลสกีมาที่ได้จากการแปลงอีอาร์โมเดลรูปที่ 1.2

การแปลงยูเอ็มแอลโมเดลเป็นเอ็กซ์เอ็มแอลสกีมา จะกล่าวถึงการแปลงระหว่างยูเอ็มแอลโมเดลซึ่งเป็นโมเดลที่นิยมใช้ในการออกแบบซอฟต์แวร์เป็นเอ็กซ์เอ็มแอลสกีมา จุดเด่นของงานวิจัยนี้คือการแสดงให้เห็นถึงการออกแบบเอ็กซ์เอ็มแอลสกีมาอย่างเป็นขั้นตอน ประกอบด้วย 3 ขั้นตอนแสดงดังรูปที่ 1.2



รูปที่ 1.4 แสดงขั้นตอนการออกแบบ UML เป็น XML Schema

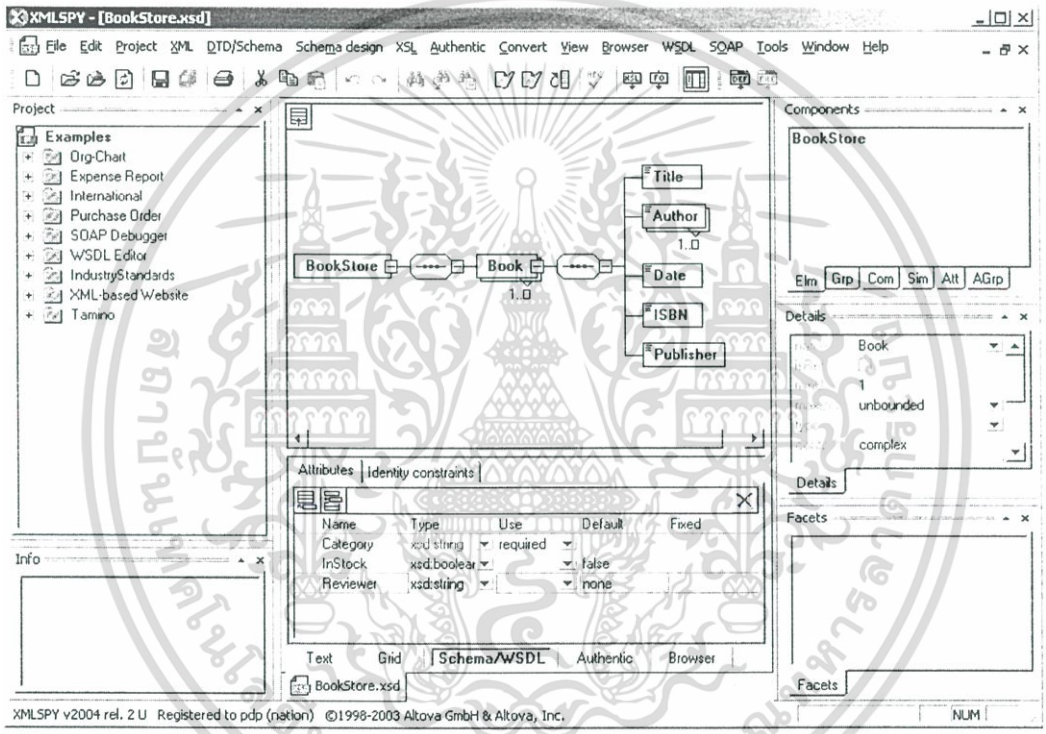
การนำข้อมูลที่อยู่ในฐานข้อมูลเชิงสัมพันธ์(Relational Database) มาสร้างเป็นเอกสารเอ็กซ์เอ็มแอลในปัจจุบันมีการกระทำอย่างแพร่หลาย แต่เป็นการกระทำในทางโครงสร้าง(Structure aspect) ไม่มีการกระทำเชิงความหมาย(Semantic aspect) การกระทำเชิงโครงสร้างอย่างเดียวทำให้ข้อมูลที่เป็นเอกสารเอ็กซ์เอ็มแอลขาดความถูกต้องสมบูรณ์เพราะสูญเสียในเชิงความหมาย การแปลงรีเลชันแนลสกีมาเป็นเอ็กซ์เอ็มแอลสกีมาเป็นแนวคิดที่ทำการแปลงข้อมูลที่เก็บอยู่ในฐานข้อมูลเชิงสัมพันธ์ให้เป็นข้อมูลเอกสารเอ็กซ์เอ็มแอลโดยการแปลงจากรีเลชันแนลสกีมาเป็นเอ็กซ์เอ็มแอลสกีมาก่อนเพื่อให้ได้มาทั้งโครงสร้างและความหมายของโครงสร้างฐานข้อมูลเชิงสัมพันธ์แล้วจึงนำเอ็กซ์เอ็มแอลสกีมาที่ได้มาทำการกำหนดโครงสร้างของเอกสารเอ็กซ์เอ็มแอล ทำให้ได้เอกสารเอ็กซ์เอ็มแอลที่มีทั้งโครงสร้างและความหมายที่กำหนดในฐานข้อมูลเชิงสัมพันธ์เพื่อเป็นการรับประกันความถูกต้องของเอกสารเอ็กซ์เอ็มแอลที่ส่งผ่านระหว่างฐานข้อมูลเชิงสัมพันธ์ รูปที่ 1.3 อธิบายภาพรวมของขั้นตอนการแปลงรีเลชันแนลสกีมาเป็นเอ็กซ์เอ็มแอลสกีมา



รูปที่ 1.5 แสดงขั้นตอนการแปลงรีเลชันแนลสกีมาเป็นเอ็กซ์เอ็มแอลสกีมา

จากรูปที่ 1.3 จะเห็นว่ามีอยู่ 2 อัลกอริทึมหลักคือ NeT และ CoT โดยที่ NeT (Nesting based Translation) คือส่วนที่ทำการแปลงรีเลชันแนลสกีมาเป็นเอ็กซ์เอ็มสกีมาเชิงโครงสร้าง ส่วน CoT (Constraint based Translation) คือส่วนที่ทำการแปลงรีเลชันแนลสกีมาเป็นเอ็กซ์เอ็มสกีมาเชิงความหมาย

เอ็กซ์เอ็มแอลสปาย(XML Spy) เป็นเครื่องมือซอฟต์แวร์ประเภทไอดีอี(IDE:Integrated Development Environment) สำหรับสร้างงานต่างๆที่เกี่ยวข้องกับภาษาเอ็กซ์เอ็มแอล เช่น การสร้างเอ็กซ์เอ็มแอลสกีมา เอกสารเอ็กซ์เอ็มแอล และสามารถตรวจสอบความถูกต้องของเอ็กซ์เอ็มแอลสกีมา และเอกสารเอ็กซ์เอ็มแอล(Well-formedness และ Valid) ตลอดจนความสามารถอื่นเช่นการติดต่อกับฐานข้อมูล การแปลงข้อมูลรูปแบบอื่นๆ มาเป็นเอกสารเอ็กซ์เอ็มแอลเป็นต้น เอ็กซ์เอ็มแอลสปายเป็นซอฟต์แวร์ทูลที่สามารถแสดงได้ทั้งเท็กซ์เบส และกราฟฟิกโหมดแสดงดังรูปที่ 1.3



รูปที่1.6 แสดงหน้าจอซอฟต์แวร์ทูล XML Spy Version 2004

นอกจากเอ็กซ์เอ็มแอลสปายแล้วในปัจจุบันยังมีซอฟต์แวร์ทูลอีกหลายบริษัท เช่น BizTalk Editor (ของ Microsoft) ,Xml Authority(ของ Extensibility) และ Near and Far Designer(ของ Open Text) เป็นต้น ซึ่งซอฟต์แวร์ทูลที่กล่าวมานี้นำเสนอการสร้างเอ็กซเอ็มแอลสกีมาแบบกราฟฟิกโหมด โดยใช้กราฟฟิกในรูปแบบของต้นไม้ (Tree) การนำเสนอรูปแบบข้อมูลในรูปแบบของต้นไม้มีข้อเสียหลายประการเช่น การนำเสนอรูปแบบคอนสเตรน(Constraints) ของฐานข้อมูลแบบรีเล

ชันแนลบางคอนสเตรนทำได้ยากลำบาก ไม่สามารถประกันความซ้ำซ้อนของข้อมูลได้ และที่สำคัญคือซอฟต์แวร์ทูลเหล่านี้ไม่มีการออกแบบเอ็กซ์เอ็มแอลสกีมาในระดับแนวคิด

1.4 ทฤษฎีหรือแนวคิดที่ใช้ในการวิจัย

ทฤษฎีที่ใช้สำหรับงานวิจัยนี้ประกอบด้วย ทฤษฎีไนแอมโมเดล[4,6,12] หรือไนแอมคอนเซ็บชวลสกีมา ทฤษฎีภาษาเอ็กซ์เอ็มแอล[1,13,14,15,16] ทฤษฎีฐานข้อมูลเชิงสัมพันธ์(Relational Database Model) [17,18] และทฤษฎียูเอ็มแอลโมเดล [19,20]

ทฤษฎีไนแอมโมเดลนำมาใช้ในการกำหนดกรอบแนวคิดในการออกแบบเอ็กซ์เอ็มแอลสีมา เพื่อให้ได้เอ็กซ์เอ็มแอลสกีมาที่มีความสมบูรณ์ครบถ้วน ตรวจสอบง่าย มีความน่าเชื่อถือสูง และสามารถใช้ในการสื่อสารระหว่างฐานข้อมูลกับฐานข้อมูลได้อย่างมีประสิทธิผล เนื่องจากไนแอมโมเดลเป็นโมเดลที่สร้างง่าย นำเสนอเป็นแบบรูปภาพ มีกฎบังคับความถูกต้องที่หลากหลายและนำเสนอกฎบังคับความถูกต้องของฐานข้อมูลเชิงสัมพันธ์ได้อย่างครบถ้วน สามารถทดลองป้อนข้อมูลได้ทันทีเมื่อต้องการตรวจสอบความถูกต้องของการสร้างสกีมา และที่สำคัญคือคุณสมบัติเด่นของไนแอมสกีมาเมื่อทำการแปลงเป็นรีเลชันแนลสกีมาแล้วจะทำให้ได้รูปแบบของนอร์มัลฟอร์มที่ 5 (5NF) ซึ่งทำให้สามารถรับประกันเรื่องความซ้ำซ้อนของข้อมูลได้

ทฤษฎีภาษาเอ็กซ์เอ็มแอลมีอยู่หลายส่วน แต่ที่นำมาใช้ในงานวิจัยนี้จะเน้นหนักในส่วนของเอ็กซ์เอ็มแอลสกีมา เอกสารเอ็กซ์เอ็มแอล และการตรวจสอบความถูกต้องของทั้งเอ็กซ์เอ็มแอลสกีมา และเอกสารเอ็กซ์เอ็มแอล โดยอ้างอิงมาตรฐานที่เผยแพร่โดยองค์กร W3C

ทฤษฎีฐานข้อมูลเชิงสัมพันธ์ (Relational Database Model) เป็นทฤษฎีที่นำมาใช้ในการสร้างฐานข้อมูลสำหรับเก็บข้อมูลรายละเอียดของไนแอมโมเดล ฐานข้อมูลที่สร้างขึ้นได้จากแปลงจากไนแอมเมต้าสกีมาเป็นรีเลชันแนลสกีมาซึ่งเป็นรีเลชั่นแนลสกีมาที่อยู่ในรูปแบบของนอร์มัลฟอร์มที่ 5

ทฤษฎียูเอ็มแอลโมเดลเป็นทฤษฎีที่นิยมใช้สำหรับงานออกแบบซอฟต์แวร์ สำหรับงานวิจัยนี้นำมาใช้ในขั้นตอนการออกแบบซอฟต์แวร์ทูล ไดอะแกรมของยูเอ็มแอลที่นำมาใช้ประกอบด้วยยูเอ็มแอลในส่วนของโครงสร้าง ได้แก่ คลาสไดอะแกรม คอมโพเน็นไดอะแกรม ดีพลอยเม็นไดอะแกรม และ ยูเอ็มแอลในส่วนของพฤติกรรมได้แก่ ยูสเคสไดอะแกรม ซีแควนซ์ไดอะแกรม

1.5 ขอบเขตการวิจัย

ประกอบด้วย

1.5.1 ศึกษาภาษาเอ็กซ์เอ็มแอล

1.5.2 ศึกษาทฤษฎีไนแอมโมเดล

1.5.3 พัฒนาอัลกอริทึมในการแปลงไนแอมสกีมาเป็นเอ็กซ์เอ็มแอลสกีมา

1.5.4 ศึกษาทฤษฎียูเอ็มแอลโมเดลเพื่อนำมาใช้ในการออกแบบซอฟต์แวร์โปรแกรม

1.5.5 พัฒนาซอฟต์แวร์ทูล

1.5.6 สรุปผลการวิจัยและข้อเสนอแนะ

1.6 โครงสร้างของงานวิจัย

รายละเอียดโครงสร้างของงานวิจัยนี้ประกอบด้วย บทนำ ทฤษฎีภาษาเอ็กซ์เอ็มแอล ทฤษฎีไนแอม กระบวนการการแปลงไนแอมสกีมาเป็นเอ็กซ์เอ็มแอลสกีมา การพัฒนาซอฟต์แวร์ทูล และสรุปผลการวิจัยและข้อเสนอแนะ

บทที่ 2
ทฤษฎีภาษาเอ็กซ์เอ็มแอล

บทนี้จะกล่าวถึงทฤษฎีของภาษาเอ็กซ์เอ็มแอลที่นำมาใช้สำหรับงานวิจัยนี้ซึ่งมีเนื้อหาประกอบด้วย ทฤษฎีพื้นฐานของภาษาเอ็กซ์เอ็มแอล การกำหนดโครงสร้างเอกสารด้วย DTD และ XML Schema และการประยุกต์ใช้งานภาษาเอ็กซ์เอ็มแอล

2.1 ทฤษฎีพื้นฐานของเอ็กซ์เอ็มแอล

2.1.1 ประวัติความเป็นมาของภาษาเอ็กซ์เอ็มแอล

ภาษาเอ็กซ์เอ็มแอลเป็นภาษาที่ถูกเผยแพร่โดยสมาคม W3C (World Wide Web Consortium) [13,114,15,16] จุดประสงค์เพื่อสร้างภาษาที่สามารถนิยามข้อมูลได้(Data Definition) เอ็กซ์เอ็มแอลจัดเป็นภาษาที่อยู่ในตระกูลของภาษาที่ใช้อธิบายและ/หรือให้คำจำกัดความของข้อมูล (XML :Extensible Markup Language) ภาษาเอ็กซ์เอ็มแอลเป็นภาษาที่มีรากฐานมาจากภาษา GML(Generalize Markup Language) และต่อมาได้มีการพัฒนาภาษา SGML (Standard Generalize Markup Language) แต่ภาษา SGML มีข้อจำกัดในการใช้งานจึงได้มีการพัฒนาภาษา HTML (Hyper Text Markup Language) สำหรับใช้ในแสดงผลข้อมูลบนเว็บไซต์ ต่อมาได้มีการนำข้อดีของภาษา SGML และ HTML มาพัฒนาเป็นภาษาเอ็กซ์เอ็มแอล ซึ่งเป็นภาษาที่มีความคล่องตัวในการใช้งานและสามารถนิยามข้อมูลได้ จุดเด่นของภาษาเอ็กซ์เอ็มแอล ที่ได้รับความนิยมและแพร่หลายในปัจจุบันคือเป็นภาษาที่แสดงผลได้หลายแพลตฟอร์ม(Platform Independent)

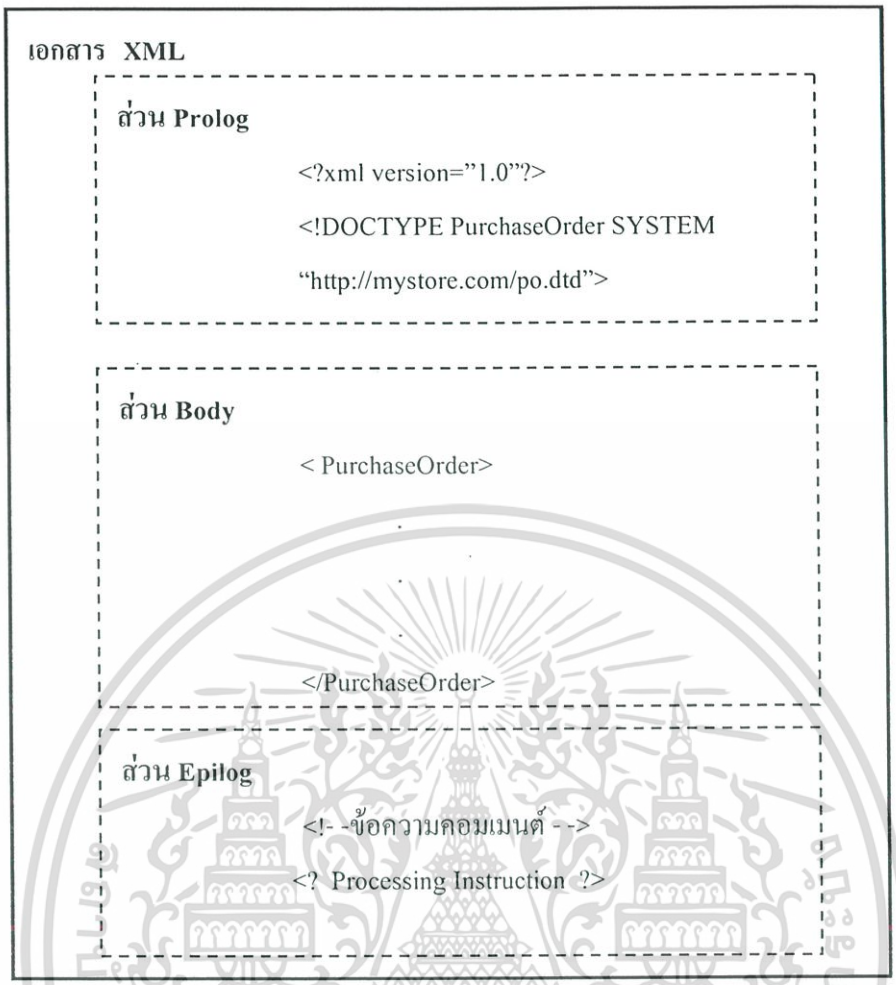
2.1.2 โครงสร้างเอกสารเอ็กซ์เอ็มแอล

เอกสารเอ็กซ์เอ็มแอลมีโครงสร้างประกอบด้วย 3 ส่วนหลักๆ [1] ดังนี้

Prolog ส่วนนี้ประกอบด้วย 2 ส่วนย่อยคือ XML Declaration และ Document Type Declaration

Body เป็นส่วนของเนื้อหาเอกสารซึ่งประกอบด้วยแท็กที่นิยามข้อความหรือข้อมูล และข้อมูลภายในแท็ก

Epilog คือส่วนที่เป็นข้อความของคอมเมนต์(Comment) และ PI(Processing Instruction) การแสดงตำแหน่งของ Epilog ที่แสดงในรูปที่ 2.1 แสดงในส่วนท้ายของเอกสารเพื่อต้องการแยกเอกสารออกเป็นสัดส่วนชัดเจนแต่ในทางปฏิบัติจริงนั้นส่วนของคอมเมนต์จะสอดแทรกอยู่ตามเนื้อหาเอกสาร



รูปที่2.1 แสดงโครงสร้างของเอกสารเอ็กซ์เอ็มแอล

โครงสร้างของเอกสารเอ็กซ์เอ็มแอลมีข้อมูลที่สำคัญที่ต้องให้ความสนใจ 4 ประเภทคือ การประกาศเอกสารเอ็กซ์เอ็มแอล (XML Declaration) การบอกประเภทของเอกสารเอ็กซ์เอ็มแอล (Document Type Declaration) ข้อความคอมเมนต์(Comment) และการแทรกส่วนประมวลผล PI(Processing Instruction)

2.1.2.1 การประกาศเอกสารเอ็กซ์เอ็มแอล (XML Declaration)

การประกาศเอกสารเอ็กซ์เอ็มแอล เป็นการประกาศให้ทราบว่าเอกสารนี้เป็นเอกสารเอ็กซ์เอ็มแอลไม่ใช่ไฟล์ข้อความธรรมดา นอกจากเอกสารเอ็กซ์เอ็มแอลแล้วเอกสารภาษาอื่นๆที่มีรากฐานมาจากภาษาเอ็กซ์เอ็มแอลก็ต้องมีการประกาศความเป็นภาษานั้นๆด้วยเช่นภาษา XHTML, ภาษา WML ที่ใช้กับ WAP(Wireless Application Protocol) รูปแบบการประกาศเอกสารเอ็กซ์เอ็มแอลแสดงดังรูปที่ 2.2

```
<?xml version="1.0" encoding="ชุดอักษร" standalone="yes|no"?>
```

รูปที่ 2.2 แสดงรูปแบบการประกาศเอกสารเอ็กซ์เอ็มแอล

ตัวอักษรทุกตัวของแต่ละแอตทริบิวต์ในรูปแบบการประกาศเอกสารเอ็กซ์เอ็มแอล จะต้องเป็นตัวพิมพ์เล็ก(lower case) ห้ามเป็นตัวอักษรพิมพ์ใหญ่(Uppercase) และแต่ละแอตทริบิวต์ที่ปรากฏในรูปที่ 2.2 ต้องเรียงลำดับกันตามที่แสดงเท่านั้น ความหมายของแต่ละแอตทริบิวต์มีรายละเอียดดังนี้

version เป็นแอตทริบิวต์ที่ต้องระบุไว้เสมอ เพื่อบ่งบอกรุ่นของเอ็กซ์เอ็มแอล รูปแบบที่แสดงดังรูปที่ 2.2 เป็นเอ็กซ์เอ็มแอลเวอร์ชัน 1.0 ในอนาคตอาจเปลี่ยนแปลงได้

encoding เป็นแอตทริบิวต์ตัวเลือก(Optional)ที่ระบุเมื่อจำเป็นต้องใช้เท่านั้น โดยบ่งบอกชุดรหัสอักษรที่ใช้ในเอกสารเอ็กซ์เอ็มแอลเพื่อให้ตัวแปลเอกสาร(XML Parser) สำหรับค่าปกติคือ utf-8

standalone เป็นแอตทริบิวต์ตัวเลือก(Optional)ที่ระบุเมื่อจำเป็นต้องใช้เท่านั้นแอตทริบิวต์นี้จะบอกให้รู้ว่าเอกสารเอ็กเอ็มแอลขึ้นอยู่กับเอกสารอื่นหรือไม่ซึ่งมีค่าที่เป็นไปได้เพียง ค่าเดียวคือ yes หรือ no ถ้าเป็น yes หมายถึงเอกสารเอ็กซ์เอ็มแอลนั้นไม่ขึ้นกับเอกสารอื่น แต่ถ้าเป็น no หมายถึงเอกสารเอ็กซ์เอ็มแอลนั้นขึ้นกับเอกสารอื่น เช่น เอกสารการกำหนดโครงสร้างของเอ็กซ์เอ็มแอลแบบ DTD หรือ XML Schema ซึ่งรายละเอียดของการกำหนดโครงสร้างของเอกสารเอ็กซ์เอ็มแอลจะกล่าวถึงในหัวข้อ 2.2 สำหรับค่าปกติของแอตทริบิวต์นี้คือ no

2.1.2.2 การบอกประเภทของเอกสารเอ็กซ์เอ็มแอล (Document Type Declaration)

การบอกประเภทของเอกสารเอ็กซ์เอ็มแอล คือการประกาศชนิดของเอกสารเอ็กซ์เอ็มแอล และไฟล์ของ DTD ที่กำหนดโครงสร้างของเอกสาร ตัวอย่างแสดงได้ดังรูปที่ 2.3

```
<DOCTYPE PurchaseOrder SYSTEM http://mystore.com/po.dtd>
```

รูปที่ 2.3 ตัวอย่างการประกาศเอกสารเอ็กซ์เอ็มแอล และไฟล์ DTD

ตัวอย่างในรูปที่ 2.3 ประกาศให้ทราบว่าเอกสารเอ็กซ์เอ็มแอลนี้มีอิลิเมนต์ PurchaseOrder เป็นรูตอิลิเมนต์ และมีไฟล์ DTD ชื่อ po.dtd อยู่ที่ http://mystore.com/po.dtd สำหรับคีย์เวิร์ด SYSTEM ต้อง

ระบุไว้เพื่อให้ทราบว่าเป็นการอ้างอิงตำแหน่งของไฟล์ด้วย URL ในกรณีที่อ้างอิงตำแหน่งของไฟล์แบบ Local จะต้องใช้คีย์เวิร์ดเป็น PUBLIC เช่นไฟล์ที่อยู่ในเครื่องคอมพิวเตอร์เดียวกัน

การประกาศประเภทของเอกสารเอ็กซ์เอ็มแอลสามารถประกาศได้ 2 แบบ แบบที่หนึ่งประกาศแบบภายใน (Internal DTD)เป็นการประกาศโดยการแทรก DTD ในเอกสารเอ็กซ์เอ็มแอล แบบที่สองประกาศแบบภายนอก(External DTD) เป็นการประกาศโดยการแยกไฟล์ออกมาต่างหาก ตัวอย่างในรูปที่ 2.3 เป็นการประกาศแบบภายนอกเพราะแยก DTD เป็นไฟล์ต่างหากคือ po.dtd

การบอกประเภทของเอกสารเอ็กซ์เอ็มแอล(Document Type Declaration) ต่างกับการกำหนดโครงสร้างของเอ็กซ์เอ็มแอลแบบ DTD(Document Type Definition) เนื่องด้วยเพราะ DTD เป็นส่วนหนึ่งของการบอกประเภทของเอกสารเอ็กซ์เอ็มแอล

2.1.2.3 ข้อความคอมเมนต์ (Comment)

การเขียนข้อความที่เป็นคอมเมนต์ของเอกสารเอ็กซ์เอ็มแอล ข้อความจะอยู่ภายในเครื่องหมาย <!--และ -->

ข้อห้ามของการเขียนข้อความคอมเมต์มีข้อกำหนดดังนี้

1. ห้ามเขียนข้อความคอมเมนต์ส่วนบนของเอกสารเพราะถูกกำหนดไว้ให้กับการบอกประเภทของเอกสารเอ็กซ์เอ็มแอล

2. ห้ามแทรกข้อความคอมเมนต์อยู่ในแท็กตัวอย่างแสดงดังรูปที่ 2.4

```
<Element <!- - ข้อความคอมเมนต์ --> > Value</Element>
```

รูปที่ 2.4 แสดงตัวอย่างข้อห้ามการแทรกข้อความคอมเมนต์ในแท็ก

3. ห้ามมีเครื่องหมายลบ 2 ตัวติดกัน (--) ซึ่งจะทำให้มีลักษณะเหมือนกับเครื่องหมายเปิด – ปิด ของคอมเมนต์ เพราะจะทำให้ตัวแปลภาษาเอ็กซ์เอ็มแอลทำงานสับสนและเกิดข้อผิดพลาดได้

ประโยชน์ของคอมเมนต์โดยพื้นฐานก็คือ เป็นข้อความสำหรับให้ใครอ่านก็ได้ อาจเป็นข้อความเพื่อเน้นย้ำ แนะนำ หรือเตือนความจำ เป็นต้น โดยที่ตัวแปลภาษาเอ็กซ์เอ็มแอลจะละเว้นข้อความคอมเมนต์ไว้ไม่แปลเหมือนข้อความอื่น ๆ ในเอกสารเอ็กซ์เอ็มแอล และ

สามารถอาศัยคุณสมบัติของคอมเมนต์มาช่วยตัดงานบางส่วนออกไปชั่วคราวโดยไม่ต้องลบทิ้งเพื่อมิให้ตัวแปลภาษาเอ็กซ์อ็มแอลแปลงานในส่วนนั้น

2.1.2.4 การแทรกส่วนประมวลผลด้วย Processing Instruction :PI

การแทรกส่วนประมวลผลด้วย Processing Instruction เป็นส่วนที่โปรแกรมตัวแปลภาษาเอ็กซ์อีมแอลจะต้องแปลหรือประมวลผลข้อความส่วนที่เป็น Processing Instruction ขึ้นต้นด้วยเครื่องหมาย <? และลงท้ายด้วย ?> รูปที่ 2.5 แสดงตัวอย่างการแทรกส่วนประมวลผลด้วย Processing Instruction

```
<? xml-stylesheet type = “text/css” href =”mystyle.css” ?>
```

รูปที่ 2.5 แสดงตัวอย่างตัวอย่างการแทรกส่วนประมวลผลด้วย Processing Instruction

รูปที่ 2.5 เป็นตัวอย่างการแทรกส่วนประมวลผลด้วย Processing Instruction โดย Processing Instruction จะบอกโปรแกรมตัวแปลภาษาเอ็กซ์เอ็มแอลที่ฝังอยู่ใน Internet Explore ให้ใช้ สไตล์ชีตจากไฟล์ชื่อ mystyle.css มากำหนดรูปแบบการแสดงผลเอกสารเอ็กซ์เอ็มแอลทางเบราเซอร์

2.1.3 กฎพื้นฐานของภาษาเอ็กซ์เอ็มแอล

กฎของภาษาเอ็กเอ็มแอลเบื้องต้นได้แก่กฎไวยากรณ์ของอิลิเมนต์ เมื่ออิลิเมนต์ (Element) มีความหมายครอบคลุมตั้งแต่ตัวแท็ก (Tag) และข้อมูลที่อยู่ภายในแท็ก ขณะที่แท็ก หมายถึงส่วนที่เป็นแท็กจริง ๆ ไม่รวมข้อมูลภายในแท็กนั้น ไวยากรณ์ที่เกี่ยวกับอิลิเมนต์ของภาษาเอ็กซ์เอ็มแอลประกอบด้วย

1. เอกสารเอ็กซ์เอ็มแอลหนึ่งๆ จะมีรูตอิลิเมนต์ (Root Element) ได้เพียงรูตเดียวซึ่งจะทำหน้าที่คลุมอิลิเมนต์อื่นๆ ทั้งหมด รูปที่ 2.6 ตัวอย่าง <Paper> รูตอิลิเมนต์

```
<Paper>
        <title> An Object and XML Database Schemas Design Tool </title>
        < Author> Narudol Chankuang  and  Suphamit Chittayasothorn </Author>
</Paper>
```

รูปที่ 2.6 แสดงตัวอย่าง <Paper> รูตอิลิเมนต์

2 แท็กเปิดและแท็กปิดต้องเหมือนกัน แตกต่างกันเพียงแท็กปิดต้องมีเรื่องหมายแสลช (/) นำหน้าชื่อแท็ก

3. การระบุแท็กห้ามมีการเหลื่อมกัน (Overlap) หมายถึงแท็กที่เปิดก่อนต้องปิดทีหลัง รูปที่ 2.7 แสดงตัวอย่างการระบุแท็กที่เหลื่อมกัน

<book><title> An Object and XML Database Schemas Design Tool </book></title>

รูปที่ 2.7 แสดงตัวอย่างการระบุแท็กที่เหลี่ยมกัน

4. ชื่อแท็กมีคุณสมบัติเป็น case-sensitive คือตัวอักษรพิมพ์เล็กตัวอักษรพิมพ์ใหญ่จะมีความแตกต่างกัน เช่น แท็ก <City> , <CITY> และ แท็ก <city> จะเป็นคนละแท็กกัน

5. แท็กว่าง (Empty Tag) หมายถึงแท็กที่ไม่มีข้อมูลอยู่ภายในแท็ก รูปแบบการเขียนแท็กว่างเขียนได้ 2 รูปแบบ แบบที่หนึ่ง <title/> แบบที่สอง <title></title>

6. ค่าของแอตทริบิวต์ต้องอยู่ในเครื่องหมายคำพูดแบบ Double quote (“) หรือ Single quote (‘) อย่างใดอย่างหนึ่งเช่น < book ISBN =”974-86631-0-3” /> จากตัวอย่างนี้ยังแสดงให้เห็นว่า อิลิเมนต์ book เป็นอิลิเมนต์ว่างหรือแท็กว่างซึ่งมีอิลิเมนต์ได้เหมือนกับอิลิเมนต์หรือแท็กปกติทุกอย่าง

กรณีที่ในค่าของแอตทริบิวต์ มีเครื่องหมายคำพูดแบบ Double quote หรือ Single quote อยู่ด้วยจะต้องหลีกเลี่ยงการใช้เครื่องหมายที่แตกต่างกัน เช่น <link Onclick= “ location.href = ‘main.xml’ ”> หรือ <link Onclick= ‘location.href = “main.xml” ’ >

7.คำสงวนในภาษาแอ็กซ์เอ็มแอล ประกอบด้วย 5 อักขระได้แก่ < , & , > , “ , และ ‘ ซึ่งอักขระทั้ง5ตัวนี้ ถูกสงวนไว้เพื่อใช้เป็นส่วนประกอบตามโครงสร้างของภาษา ดังนั้นหากเนื้อความเอกสารจำเป็นต้องมีอักขระที่เป็นคำสงวนก็จะต้องระบุเป็นชุดอักษรพิเศษ (Entity Reference) แทนคำสงวนทั้ง5ตัว การระบุชุดอักษรพิเศษแทนคำสงวนทั้ง5ตัวแสดงได้ดังตารางที่ 2.1

ตารางที่ 2.1 ตารางแสดงคำสงวนและอักษรพิเศษ

คำสงวน	อักษรพิเศษ
<	<
&	&
>	>
“	"
‘	'

ตัวอย่างการใช้อักษรพิเศษ เช่น หากต้องการให้ข้อความ คุกกี้ราคา < 25 บาททุกชิ้นที่ S&P. อยู่ในแท็ก <promotion> ก็จะต้องเขียนดังรูปที่ 2.8

```
<promotion>คุกกี้ราคา &lt; 25 บาททุกชิ้นที่ S&amp;P. </promotion>
```

รูปที่ 2.8 แสดงตัวอย่างการการใช้อักษรพิเศษ

8. การตั้งชื่อแท็กมีหลักเกณฑ์ดังนี้

- ชื่อแท็กต้องขึ้นด้วยตัวอักษรหรือเครื่องหมาย underscore (_) เท่านั้น
- อักษรตัวถัดไปต้องเป็นตัวอักษร ตัวเลข เครื่องหมายจุด (.) เครื่องหมายยัติภังค์ (-) เครื่องหมาย underscore (_) หรือ เครื่องหมาย colon (:) เท่านั้น แต่สำหรับเครื่องหมาย colon จะมีปัญหาเรื่อง namespace ดังนั้นควรจะหลีกเลี่ยงการใช้
- ชื่อแท็กมีคุณสมบัติ case sensitive ดังได้กล่าวไว้ในไวยากรณ์ที่ 4
- อักษร 3 ตัวแรกของแท็กห้ามเป็นค่า XML ทั้งตัวเล็กและตัวใหญ่

ตัวอย่างชื่อแท็กที่ถูกต้องแสดงดังรูปที่ 2.9 และ ตัวอย่างแท็กที่ไม่ถูกต้องแสดงได้ดังตารางที่ 2.2

```
<narudol />
<_ narudol />
< narudol _chank />
< naru_dol -chank />
<_ narudol. chank />
< naru.dol. chank />
< narudol 1 />
< narudol _2 />
```

รูปที่ 2.9 แสดงตัวอย่างชื่อแท็กที่ถูกต้อง

ตารางที่ 2.2 ตารางแสดงตัวอย่างแท็กที่ไม่ถูกต้อง

ชื่อแท็กที่ไม่ถูกต้อง	คำอธิบายสาเหตุที่ไม่ถูกต้อง
<- narudol />	ขึ้นต้นด้วยเครื่องหมายยัติภังค์
<1 narudol />	ขึ้นต้นด้วยตัวเลข
<. narudol />	ขึ้นต้นด้วยเครื่องหมายจุด
<: narudol />	ขึ้นต้นด้วยเครื่องหมายโคลอน
<? narudol />	ขึ้นต้นด้วยเครื่องหมายคำถาม
< narudol? />	มีเครื่องหมายคำถาม
< narudol * />	มีเครื่องหมายดอกจัน
<xmlnarudol />	ขึ้นต้นด้วย xml
<XMLnarudol />	ขึ้นต้นด้วย XML
<Xml narudol />	ขึ้นต้นด้วย Xml

2.1.4 มาตรฐานที่เกี่ยวข้องกับภาษาเอ็กซ์เอ็มแอล

มาตรฐานเสริมศักยภาพของการทำงานของเอ็กซ์เอ็มแอลที่ถูกเสนอต่อ W3C จำนวนมาก ได้แก่ XSL , Xlink และ Xpointer ซึ่งต่างก็เป็นมาตรฐานที่ถูกนำเสนอเพื่อสนับสนุนเอ็กซ์เอ็มแอล ในด้านของ สไตล์ชีต (Style Sheet) การทำไฮเปอร์ลิงก์ (HyperLinks) และคุณสมบัติด้านอื่น ๆ นอกจากนั้นยังมีมาตรฐานอีกหลายรายการเช่น HyTime และ DSSL (Document Style Semantics and Specification Language) ที่ได้รับการปรับปรุงสนับสนุนเอ็กซ์เอ็มแอล

มาตรฐานต่าง ๆ ที่เกี่ยวข้องกับเอ็กซ์เอ็มแอลที่ได้ถูกเสนอให้ W3C รับรองตัวอย่างเช่น

2.1.4.1 Namespace

Namespace ทำหน้าที่ช่วยในการกำหนดขอบเขตให้กับชื่อของอิลิเมนต์ และ แอตทริบิวท์ เพื่อหลีกเลี่ยงการถูกเรียกใช้ซ้ำกันระหว่างเอกสารเอ็กซ์เอ็มแอล ที่เกิดจากแหล่งกำเนิดต่างกัน Namespace แต่ละชุดเป็นการรวมของชื่อต่าง ๆ ของเอ็กซ์เอ็มแอลอิลิเมนต์ และ แอตทริบิวต์ที่ถูกระบุพร้อมกับ URL (Uniform Resource Identifier) ที่เป็นแหล่งกำเนิดของเอกสารเอ็ซ์เอ็มแอลนั้น

2.1.4.2 XSL (Extensible Stylesheet Language)

XSL เป็นภาษาในการกำหนดสไตล์ชีตให้กับข้อมูลเอ็กซ์เอ็มแอล (XML Data) สำหรับการนำเสนอในเว็บบราวเซอร์ หรือสื่อนำเสนออื่น ๆ XLS มีศักยภาพสูงกว่า CSS (Cascading Style Sheet) ที่ใช้เป็นสไตล์ชีตสำหรับ HTML มาก

2.1.4.3 Xlink

Xlink มีพื้นฐานการทำงานมาจาก HyTime และ XLinkเป็นการปรับปรุงการเชื่อมโยงใน HTML โดยสนับสนุนการเชื่อมโยงแบบสองทาง (Bi-Direction) แบบหนึ่งต่อกลุ่ม (One-to-many) และแบบแยกตามประเภท (Typed Links) เช่น การอ้างอิง footnote และ glossary เป็นต้น

2.1.4.4 Xpointer

Xpointer มีพื้นฐานการทำงานมาจาก Text Encoding Initiative (TEI) Xpointer อนุญาติให้สามารถชี้ไปที่จุดหรือตำแหน่งใดก็ได้ในเอกสารเป้าหมายโดยไม่ต้องกำหนดการเชื่อมโยงแบบ HTML ไว้ล่วงหน้า

2.1.4.5 RDF (Resource Description Framework)

RDF เป็น Metadata สำหรับเอกสารเอ็กซ์เอ็มแอลคล้ายกับ Metadata Tag ของ HTML เช่น ผู้แต่ง ลิขสิทธ์ และวันที่จัดพิมพ์ เป็นต้น

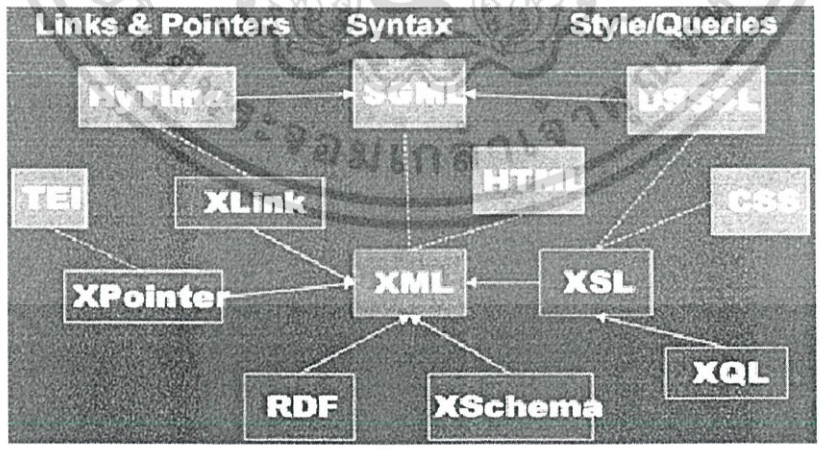
2.1.4.6 Xschema

Xschema เข้ามาแทนที่ DTD สำหรับการกำหนดประเภทข้อมูลของเอ็กซ์เอ็มแอล Xschema เป็นการประยุกต์หลายเทคโนโลยีเข้าด้วยกัน ได้แก่ XData , SOX , DCD (Document Content Description) และ DDML

2.1.4.7 XQL (XML Query Language)

XQL เป็นส่วนขยายเพิ่มเติมของ XSL สำหรับใช้อ้างถึงและกลั่นกรองส่วนที่เป็นอิลิ้มนต์ และแท็กของเอกสารเอ็กซ์เอ็มแอล XQL ช่วยกำหนดรูปแบบที่ชัดเจนเข้าใจง่ายในการเข้าถึงอิลิเมนต์ที่เฉพาะเจาะจง และสำหรับการค้นหาโหนดที่มีคุณลักษณะพิเศษต่าง ๆ

ความสัมพันธ์ระหว่าง 7 มาตรฐานของเอ็กซ์เอ็มแอล แสดงได้ดังรูปที่ 2.9



รูปที่ 2.10 แสดงความสัมพันธ์ระหว่าง 7 มาตรฐานของเอ็กซ์เอ็มแอล

นอกจากมาตรฐานทั้ง 7 ของเอ็กซ์เอ็มแอลที่กล่าวมานั้น ยังมีการกำหนดมาตรฐานด้าน Application Programming Interface (API) เพื่อที่จะให้แอพลิเคชันสามารถเข้าถึงเอกสารเอ็กซ์เอ็มแอลได้ทันที เมื่อเอกสารเอ็กซ์เอ็มแอลนั้นเข้าสู่กระบวนการ Parsing โดยมี APIs 2 กลุ่มที่ได้รับความนิยมได้แก่

2.1.4.8 DOM (Document Object Model)

DOM เป็น Tree-based API ที่ทำการประมวลผลโครงสร้างเอกสารเอ็กซ์เอ็มแอล ให้เป็นโครงสร้างแบบต้นไม้ที่ประกอบไปด้วยลำต้น กิ่งก้านสาขา และใบ เพื่อให้แอพลิเคชันสามารถเข้าหาจุดต่าง ๆ ของโครงสร้างต้นไม้นั้นได้ โดยที่ขอบเขตจะถูกจำกัดโดยหน่วยความจำที่เรียกใช้ได้ในขณะนั้น

2.1.4.9 SAX (Simple API for XML)

SAX เป็น Event-based API ที่รายงานตามเหตุการณ์ของ Parsing เช่นจุดเริ่มต้นและจุดสิ้นสุดของอิลิเมนต์ต่าง ๆ ไปให้ แอพลิเคชันผ่านทางการร้องขอ โดยปกติจะไม่มีการสร้างโครงสร้างแบบต้นไม้ขึ้นมา เนื่องจากการทำงานเป็นแบบ Event-based API การเข้าถึงเอกสารเอ็กซ์เอ็มแอลจึงทำได้ง่ายไม่ซับซ้อน และที่สำคัญผู้ใช้สามารถทำ parsing เอกสารที่มีขนาดใหญ่มากกว่าปริมาณหน่วยความจำที่เรียกใช้ได้และผู้ใช้สามารถสร้างโครงสร้างข้อมูลของเอกสารอยู่ในรูปแบบที่ตนเองต้องการได้

2.1.5 ประโยชน์ของภาษาเอ็กซ์เอ็มแอล

ประโยชน์ของเอ็กเอ็มแอลที่สำคัญคือการเป็นแม่แบบหรือต้นแบบในการนิยามข้อมูลเพื่อใช้ในงานต่าง ๆ ซึ่งสามารถแบ่งรายละเอียดได้ดังนี้

1. ใช้สำหรับสร้างข้อมูลที่สามารถอธิบายความหมายของตัวเองได้ (Self-describe data)

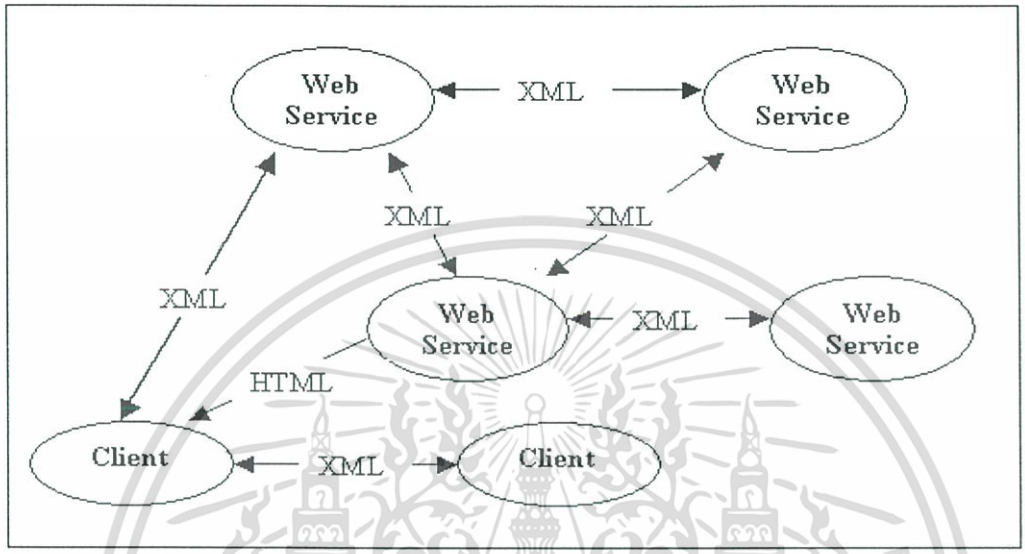
จากความสามารถในการสร้างแท็กขึ้นมาอธิบายข้อมูลที่อยู่ภายในแท็กทำให้ข้อมูลนั้นมีความหมายอยู่ในตัวมันเอง เป็นข้อมูลที่สามารถเขียนโปรแกรมมาดึงข้อมูลไปใช้งานได้โดยง่าย และแม้นแต่มนุษย์ก็สามารถอ่านได้ ดังนั้นจึงสามารถกล่าวได้ว่าเอกสารเอ็กซ์เอ็มแอล มีคุณสมบัติครบทั้งแบบ machine readable และแบบ human readable

2. ใช้สำหรับการแลกเปลี่ยนข้อมูล (Data Exchange)

เนื่องด้วยเอกสารเอ็กเอ็มแอลมีลักษณะเป็นไฟล์ข้อความธรรมดา ดังนั้นจึงทำให้เอกสารเอ็กซ์เอ็มแอลเป็นภาษากลางที่สามารถใช้ได้ในทุกแพลตฟอร์ม เช่น แพลตฟอร์มวินโดว์ ยูนิกซ์ หรืออื่น ๆ ดังนั้นจึงทำให้เอกสารเอ็กซ์เอ็มแอลมีความสามารถในการแลกเปลี่ยนเอกสารข้ามแพลตฟอร์มกันได้

3. เป็นรูปแบบข้อความในการสื่อสาร (Messaging Format) ระหว่างแอปพลิเคชันหรือโปรแกรม เป็นแนวคิดที่กำลังได้รับความนิยมเป็นอย่างมากสำหรับประโยชน์ของเอ็กซ์เอ็มแอล

ข้อนี้ เนื่องด้วยตั้งแต่ปีคศ.2000 เป็นต้นมาแนวคิดนี้เป็นแนวคิด Web Service ที่บริษัทไมโครซอฟต์รียกว่า .NET ซึ่งแนวความคิดนี้ก็คือการเตรียมซอฟแวร์สำหรับให้บริการทำงานบางอย่างอยู่ในเซิร์ฟเวอร์เครื่องใดเครื่องหนึ่งภายในเครื่อข่ายอินเตอร์เน็ต โดยผู้ใช้งานทั่วไปสามารถเรียกใช้บริการนั้นได้ ดังรูปที่ 2.10 ซึ่งจะเห็นได้ว่าเอ็กซ์เอ็มแอลเป็นรูปแบบการสื่อสารระหว่างองค์ประกอบต่าง ๆ ตามแนวคิดของ Web Service



รูปที่ 2.11 ข้อความที่ใช้สื่อสารระหว่างองค์ประกอบต่าง ๆของ Web Service มีรูปแบบเป็น XML

4. ประโยชน์ในเชิงเทคโนโลยีอินเตอร์เน็ตและการพัฒนาเว็บ

ประโยชน์ในเชิงเทคโนโลยีอินเตอร์เน็ตและการพัฒนาเว็บในเมืองไทยยังไม่เห็นประโยชน์มากนัก เพราะลักษณะงานยังไม่มีความจำเป็นให้นำเอ็กซ์เอ็มแอลไปพัฒนา แต่งานบางอย่างเช่นการพัฒนาเว็บสามารถที่จะนำเอ็กซ์เอ็มแอลมาช่วยเพิ่มประสิทธิภาพได้

5. เป็นรากฐานของภาษาใหม่ ๆ ในการพัฒนาเว็บ

ภาษาใหม่ ๆ ในการพัฒนาเว็บ เช่น ภาษา XHTML , MathML , VML , WML เป็นต้น

6. ใช้ในแวดวงธุรกิจแบบ B2B (Business to Business)

ในกรณีนี้จะต้องใช้ภาษาเฉพาะอย่างเช่น cXML (Commerce XML) , xCBL (XML Common Business Language) เป็นต้น

และในต่างประเทศมีโครงการนำ XML มาใช้แทนระบบ EDI (Electronic Data Interchange) ซึ่งเป็นการแลกเปลี่ยนเอกสารอิเลกทรอนิกระหว่างองค์กร

2.2 การกำหนดโครงสร้างเอกสารเอ็กซ์เอ็มแอลแบบ DTD และ XML Schema

การกำหนดโครงสร้างหรือสกีมาของเอกสารเอ็กซ์เอ็มแอลเป็นการระบุว่าเอกสารเอ็กซ์เอ็มแอลมีโครงสร้างอย่างไร มีอิลิเมนต์ มีแอตทริบิวต์อะไรบ้าง และมีชนิดข้อมูลของอิลิเมนต์ แอตทริบิวต์อย่างไร เอกสารเอ็กซ์เอ็มแอลที่ถูกกฎพื้นฐานนั้นเรียกว่าเอกสารเอ็กซ์เอ็มแอลที่มีคุณสมบัติ well-formed [1,13,14,16] แต่เอกสารเอ็กซ์เอ็มแอลที่มีความน่าเชื่อถือนั้นต้องเป็นเอกสารเอ็กซ์เอ็มแอลที่มีการกำหนดโครงสร้าง เอกสารเอ็กซ์เอ็มแอลที่มีการอ้างอิงโครงสร้างเป็นเอกสารเอ็กซ์เอ็มแอลที่มีคุณสมบัติ Valid การกำหนดโครงสร้างของเอกสารเอ็กซ์เอ็มแอลนั้นมีประโยชน์สำคัญสองประการคือ

เป็นการกำหนดรูปแบบโครงสร้างของเอกสารเอ็กซ์เอ็มแอล หรือเรียกว่าแบบจำลองข้อมูล (Data Model)ซึ่งเป็นการกำหนดว่าเอกสารเอ็กซ์เอ็มแอลมีแท็กอะไรบ้าง มีแอตทริบิวต์อะไรบ้าง ค่าของแอตทริบิวต์เป็นอะไรบ้าง เป็นต้น และ

เป็นสัญญาระหว่างองค์กรที่ใช้เอกสารเอ็กซ์เอ็มแอลในการแลกเปลี่ยนข้อมูล เพื่อให้รูปแบบของเอกสารเอ็กซ์เอ็มแอลที่ใช้มีรูปแบบที่ตรงกัน และเพื่อความถูกต้องในการนำเอกสารเอ็กซ์เอ็มแอลไปใช้งาน

การกำหนดโครงสร้งของเอกสารเอ็กซ์เอ็มแอลทำได้หลายวิธีเช่น DCD(Document Content Description) RELAX/TREX , BizTalk, XDR(XML-Data Reduced), DTD และ XML Schema เป็นต้น แต่ที่กล่าวถึงในงานวิจัยนี้มีสองวิธีคือ แบบ DTD และแบบ XML Schema ซึ่งจะกล่าวในหัวข้อ 2.2.1 และ 2.2.2 ตามลำดับ ซึ่งทั้งสองแบบนี้ถูกเผยแพร่โดยสมาคม W3C

2.2.1 การกำหนดโครงสร้างเอกสารเอ็กซ์เอ็มแอลแบบ DTD

การวางโครงสร้างของเอกสารเอ็กซ์เอ็มแอลแบบ DTD นั้นสืบทอดมาจากภาษา SGML ปัจจุบันมีแนวโน้มของการใช้งานน้อยลงแต่ก็ควรศึกษาเพื่อเป็นพื้นฐานในการออกแบบโครงสร้างของเอกสารเอ็กซ์เอ็มแอล เพราะมีบางเทคโนโลยียังใช้งาน DTD อยู่บ้างเช่น เทคโนโลยี WAP(Wireless Application Protocol) ซึ่งเป็นการเรียกดูข้อมูลอินเตอร์เน็ตผ่านโทรศัพท์มือถือ ถูกสร้างด้วยภาษา WML(Wireless Markup Language) เป็นภาษาที่อยู่ในตระกูลของภาษา XML แต่มีการกำหนดโครงสร้างแบบ DTD

2.2.1.1 ลักษณะของการกำหนดโครงสร้างเอกสารเอ็กซ์เอ็มแอลแบบ DTD

ลักษณะของการกำหนดโครงสร้างเอกสารเอ็กซ์เอ็มแอลแบบ DTD แสดงดังรูปที่ 2.11

```
<!ELEMENT BookCataloge (Book)*>
<!ELEMENT Book (Title,Author,Date ,ISBN,Publisher)>
<!ELEMENT Title (#PCDATA)>
<!ELEMENT Author (#PCDATA)>
<!ELEMENT Date (#PCDATA)>
<!ELEMENT ISBN (#PCDATA)>
<!ELEMENT Publisher (#PCDATA)>
```

รูปที่ 2.12 แสดงลักษณะของเอกสาร DTD

จากการกำหนดโครงสร้างของเอกสารเอ็กซ์เอ็มแอลในรูปที่ 2.11 ทำให้เราสามารถสร้างเอกสารเอ็กซ์เอ็มแอลที่อ้างอิงกับโครงสร้างได้ดังรูปที่ 2.12

```
<?xml version =”1.0”?>
<BookCataloge>
    <Book>
        <Title> Conceptual Schema Relational Database Design</Title>
        <Author> Terry Helpin</Author>
        <Date> 1 Febuary 1995</Date>
        <IDBN> 0 13 355702 2</ISBN>
        <Publisher> Hall of Australia</Puplisher>
    </Book>
</BookCataloge>
```

รูปที่ 2.13 เอกสารเอ็กซ์เอ็มแอลที่อ้างอิงกับโครงสร้างแบบ DTD(รูปที่ 2.11)

2.2.1.2 การประกาศโครงสร้างเอกสารแบบ DTD

การประกาศโครงสร้างเอกสารแบบ DTD สามารถทำได้สองวิธีคือ การประกาศแบบภายใน(Internal DTD) และการประกาศแบบภายนอก(External DTD) การประกาศแบบภายในนั้นจะประกาศไว้ในส่วนของ Document Type Declaration ซึ่งเป็นส่วนหนึ่งของเอกสารเอ็กซ์เอ็มแอล การประกาศโครงสร้างของเอกสาร DTD แบบภายในแสดงได้ดังรูปที่ 2.13 (บริเวณที่เป็นตัวหนา)

```
<?xml version ="1.0"?>
<!DOCTYPE BookCataloge [
        <ELEMENT BookCataloge (Book)*>
        <ELEMENT Book (Title, Author, Date, ISBN,
                Publisher)>
        <ELEMENT Title (#PCDATA)>
        <ELEMENT Date (#PCDATA)>
        <ELEMENT ISBN (#PCDATA)>
        <ELEMENT Publisher (#PCDATA)>
]>
<BookCataloge>
  <Book>
        <Title> Conceptual Schema Relational Database Design</Title>
        <Author> Terry Helpin</Author>
        <Date> 1 Febuary 1995</Date>
        <IDBN> 0 13 355702 2</ISBN>
        <Publisher> Hall of Australia</Puplisher>

    </Book>
</BookCataloge>
```

รูปที่ 2.14 การประกาศโครงสร้างเอกสาร DTD แบบภายใน

การประกาศโครงสร้างเอกสารDTD แบบภายนอกจะกระทำโดยแยกเป็นไฟล์ต่างหาก โดยไม่รวมกับส่วนของเอกสารเอ็กซ์เอ็มแอล เพียงแต่มีการอ้างถึงเท่านั้นไฟล์ของโครงสร้างเอกสารจะมีนามสกุล(.dtd) แสดงดังรูปที่ 2.14

```
<?xml version =”1.0”?>
<!DOCTYPE BookCataloge SYSTEM “bookcataloge.dtd”>
<BookCataloge>
    <Book>
        <Title> Conceptual Schema Relational Database Design</Title>
        <Author> Terry Helpin</Author>
        <Date> 1 Febuary 1995</Date>
        <ISBN> 0 13 355702 2</ISBN>
        <Publisher> Hall of Australia</Puplisher>

    </Book>
</BookCataloge>
```

รูปที่ 2.15 แสดงการประกาศโครงสร้างเอกสาร DTD แบบภายนอก

2.2.1.3 การประกาศ อิลิเมนต์ และแอตทริบิวต์

การกำหนดโครงสร้างของเอกสารเอ็กซ์เอ็มแอลแบบ DTD สิ่งที่ผู้กำหนดจำเป็นต้องทราบมีอยู่สองประเภทหลักๆคือการประกาศ อิลิเมนต์ และแอตทริบิวต์

I. การประกาศอิลิเมนต์

ภายในเอกสารเอ็กซ์เอ็มแอล จะประกอบด้วยอิลิเมนต์จำนวนมากตั้งแต่รูตอิลิเมนต์และอิลิเมนต์ต่างๆประกอบกันเป็นเอกสารเอ็กซ์เอ็มแอล ซึ่งการประกาศอิลิเมนต์จึงถือว่าเป็นสิ่งสำคัญที่จะต้องกล่าวถึง รูปแบบการประกาศอิลิเมนต์แสดงดังรูปที่ 2.15

```
<!ELEMENT ชื่ออิลิเมนต์ (เนื้อหาของอิลิเมนต์)>
```

รูปที่ 2.16 แสดงรูปแบบการประค่าอิลิเมนต์

ชื่ออิลิเมนต์นั้นมีข้อกำหนดตามที่กล่าวไว้ในหัวข้อกฎพื้นฐานของเอ็กซ์เอ็มแอล ส่วนเนื้อหาของอิลิเมนต์มีโอกาศเป็นไปได้คือ เนื้อหาของอิลิเมนต์ประกอบด้วยอิลิเมนต์อื่นๆ เนื้อหาอิลิเมนต์ประกอบด้วยข้อความปกติ และเนื้อหาของอิลิเมนต์อาจไม่มีอะไรอยู่เลย(เป็นแท็กว่าง) หรือภายในอิลิเมนต์จะประกอบด้วยอิลิเมนต์ปนอยู่กับข้อความก็ได้จากที่กล่าวมาสามารถแสดงตัวอย่างได้ดังรูปที่ 2.16 – 2.23

```
<!ELEMENT ชื่ออิลิเมนต์(อิลิเมนต์1,อิลิเมนต์2,อิลิเมนต์3,...,อิลิเมนต์ n)
```

รูปที่ 2.17 รูปแบบการประกาศอิลิเมนต์ที่ภายในประกอบด้วยอิลิเมนต์

```
<Book>
    <Title> Conceptual Schema Relational Database Design</Title>
    <Author> Terry Helpin</Author>
    <Date> 1 Febuary 1995</Date>
    <ISBN> 0 13 355702 2</ISBN>
    <Publisher> Hall of Australia</Puplisher>
</Book>
```

รูปที่ 2.18 แสดงอิลิเมนต์ที่ประกอบด้วยอิลิเมนต์อื่นๆอยู่ภายใน

จากรูปที่ 2.17 จะแสดงให้เห็นว่าอิลิเมนต์ Book จะประกอบด้วยอิลิเมนต์อื่นๆ ภายในเช่น Title Authour Date ISBN และ Publisher

```
<!ELEMENT ชื่ออิลิเมนต์(#PCDATA) >
```

รูปที่ 2.19 แสดงรูปแบบการประกาศอิลิเมนต์ที่ภายในประกอบด้วยข้อความ

รูปที่ 2.18 แสดงรูปแบบการประกาศอิลิเมนต์ที่ภายในประกอบด้วยข้อความธรรมดา โดยที่ PCDATA(Parsed Character Data) เป็นชนิดของข้อมูลของโครงสร้างเอกสารเอ็กซ์เอ็มแอลแบบ DTD

```
<Title> Conceptual Schema Relational Database Design</Title>
```

รูปที่ 2.20 แสดงอิลิเมนต์ที่ภายในประกอบด้วยข้อความ

```
<!ELEMENT ชื่ออิลิเมนต์ EMPTY>
```

รูปที่ 2.21 แสดงรูปแบบการประกาศอิลิเมนต์ว่าง

```
<Student ID =”44061635”>
```

รูปที่ 2.22 แสดงอิลิเมนต์ที่ไม่มีอะไรอยู่ หรือ(แท็กว่าง)

รูปที่ 2.21 แสดงอิลิเมนต์ที่ภายในไม่มีข้อความ ไม่มีอิลิเมนต์ มีเพียงแอตทริบิวต์ อิลิเมนต์ลักษณะนี้เป็นอิลิเมนต์ว่าง

```
<!ELEMENT ชื่ออิลิเมนต์(#PCDATA,อิลิเมนต์1,อิลิเมนต์2,...,อิลิเมนต์ n)>
```

รูปที่ 2.23 แสดงรูปแบบการประกาศอิลิเมนต์ที่ปนกับข้อความ

```
<Book>
    This is essential book for conceptual database design
    <Title> Conceptual Schema Relational Database Design</Title>
</Book>
```

รูปที่ 2.24 แสดงอิลิเมนต์ที่ปนอยู่กับข้อความ

นอกจากที่กล่าวมาแล้วรูปแบบการประกาศอิลิเมนต์ยังมีอีกหลายรูปแบบ เช่น รูปแบบการประกาศอิลิเมนต์แบบให้เลือกด้วยไปท์(|) และรูปแบบการประกาศอิลิเมนต์แบบระบุ (+,*,?) แสดงดังรูปที่ 2.24 และ 2.25 ตามลำดับ

```
<! ELEMENT ชื่ออิลิเมนต์ (อิลิเมนต์1|อิลิเมนต์2 |อิลิเมนต์3)>
```

รูปที่ 2.25 แสดงรูปแบบการประกาศอิลิเมนต์แบบเลือก

การใช้เครื่องหมายไปป์(|) คั่นระหว่างอิลิเมนต์หมายความว่าให้เลือกเพียงอิลิเมนต์อิลิเมนต์หนึ่งเท่านั้น

```
<! ELEMENT Book (ISBN,Title,Author+,Editor*,(Review|Example)?)>
```

รูปที่ 2.26 แสดงรูปแบบการประกาศอิลิเมนต์แบบระบุ

จากรูปที่ 2.25 สามารถอธิบายได้ดังนี้ เมื่อ อิลิเมนต์ Book หมายถึงรูตอิลิเมนต์ อิลิเมนต์ ISBN หมายถึงอิลิเมนต์ที่สามารถปรากฏได้เพียงอิลิเมนต์เดียว และต้องมีปรากฏในเอกสารไม่มีไม่ได้ อิลิเมนต์ Title หมายถึงอิลิเมนต์ที่ปรากฏต่อจากอิลิเมนต์ ISBN มีได้เพียงอิลิเมนต์เดียวและไม่มีไม่ได้ อิลิเมนต์ Author หมายถึงอิลิเมนต์ที่ปรากฏต่อจาก Title มีได้ตั้งแต่ 1 อิลิเมนต์ขึ้นไป และต้องมีปรากฏไม่มีไม่ได้ อิลิเมนต์ Editor หมายถึงอิลิเมนต์ที่ปรากฏต่อจาก Author มีได้ตั้งแต่ 1 อิลิเมนต์ขึ้นไปหรือไม่มีปรากฏเลยก็ได้ อิลิเมนต์ Review หรือ Example เป็นอิลิเมนต์ที่ปรากฏต่อจาก Editor จะมีหรือไม่ก็ได้ ถ้ามีจะมีปรากฏได้เพียงอิลิเมนต์ใดอิลิเมนต์หนึ่งเท่านั้น

2. การประกาศแอตทริบิวต์

การประกาศแอตทริบิวต์มีความสำคัญไม่น้อยไปกว่าการประกาศอิลิเมนต์ เนื่องจากดอกสารเอ็กซ์เอ็มแอลโดยทั่วไปจะประกอบด้วยข้อมูลที่ถูกเก็บอยู่ภายในอิลิเมนต์ และข้อมูลที่บรรจุอยู่ภายในแอตทริบิวต์แล้วแต่ผุออกแบบโครงสร้างเอกสารจะเป็นผู้กำหนด รูปแบบการประกาศแอตทริบิวต์แสดงได้ดังรูปที่ 2.26 ตัวอย่างการประกาศแอตทริบิวต์แสดงดังรูปที่ 2.27

```
<! ATTLIST ชื่ออิลิเมนต์ ชื่อแอตทริบิวต์ ชนิดข้อมูลของแอตทริบิวต์
(#REQUIRED|#IMPLIED|#FIXED ค่าปกติของแอตทริบิวต์) >
```

รูปที่ 2.27 แสดงรูปแบบการประกาศแอตทริบิวต์

เมื่อชื่ออิลิเมนต์หมายถึง ชื่อของอิลิเมนต์ที่มีแอตทริบิวต์ปรากฏอยู่ ชื่อแอตทริบิวต์ หมายถึงชื่อของแอตทริบิวต์ที่ปรากฏ ชนิดข้อมูลของแอตทริบิวต์ หมายถึงชนิดของข้อมูลที่กำหนด ซึ่งชนิดของข้อมูลถูกกำหนดโดยสมาคม W3C #REQUIRED หมายถึงแอตทริบิวต์นั้นจำเป็นต้องมีปรากฏ #IMPLIED หมายถึงแอตทริบิวต์นั้นจะมีปรากฏหรือไม่ก็ได้ #FIXED ค่าปกติของแอตทริบิวต์ หมายถึงเมื่อมีการระบุคีย์เวิร์ดนี้จะต้องระบุค่าปกติต่อท้ายเสมอ

```
<! ATTLIST Student ID CDATA #REQUIRED >
```

รูปที่ 2.28 แสดงตัวอย่างการประกาศแอตทริบิวต์

จากรูปที่ 2.27 สามารถอธิบายรายละเอียดได้ดังนี้ เมื่อ Student หมายถึงชื่อของอิลิเมนต์ ID หมายถึงชื่อของแอตทริบิวต์ CDATA หมายถึงชนิดของข้อมูล #REQUIRED หมายถึงการระบุว่าต้องมีแอตทริบิวต์นี้ปรากฏทุกครั้ง และการกำหนดอิลิเมนต์แสดงดังรูปที่ 2.21

2.2.2 การกำหนดโครงสร้างเอกสารเอ็กซ์เอ็มแอลแบบ XML Schema

การกำหนดโครงสร้างเอกสารเอ็กซ์เอ็มแอลแบบ DTD ที่กล่าวมาแล้วนั้นเป็นรูปแบบของการกำหนดโครงสร้างเอกสารของภาษา SGML ยังมีข้อด้อยที่ไม่เหมาะสมกับภาษาเอ็กซ์เอ็มแอลหลายประการเช่น มีชนิดของข้อมูลน้อย และมีรูปแบบการเขียน(Syntax) ที่ไม่เหมือนกับภาษาเอ็กซ์เอ็มแอลเป็นต้น ทำให้ต้องใช้ตัวแปรภาษาคนละตัวกับภาษาเอ็กซ์เอ็มแอล ด้วยเหตุผลที่กล่าวมาข้างต้นทางสมาคม W3C จึงได้ออกมาตรฐานการกำหนดโครงสร้างเอกสารเอ็กซ์เอ็มแอลแบบเอ็กซ์เอ็มแอลสกีมา(XML Schema) มาตรฐานของเอ็กซ์เอ็มแอลสกีมาแบ่งออกเป็น 3 ส่วนคือ ส่วนแรกเป็นข้อมูลเบื้องต้น(Primer) ส่วนที่สองเป็นส่วนของโครงสร้าง(Structure) และส่วนที่สามเป็นชนิดของข้อมูล(Datatypes) [13,14,15]

2.2.2.1 ลักษณะของเอ็กซ์เอ็มแอลสกีมา

เพื่อให้เห็นภาพลักษณ์ของเอ็กซ์เอ็มแอลสกีมาที่ชัดเจนจะอธิบายการกำหนดโครงสร้างแบบ DTD ควบคู่ไปกับ XML Schema แสดงดังรูปที่ 2.28 และ 2.29 ตามลำดับ

```
<!ELEMENT BookStore (Book)*>
<!ELEMENT Book(Title,Author,Date,ISBN,Publisher)>
<!ELEMENT Title(#PCDATA)>
<!ELEMENT Author(#PCDATA)>
<!ELEMENT Date(#PCDATA)>
<!ELEMENT ISBN(#PCDATA)>
<!ELEMENT Publisher(#PCDATA)>
```

รูปที่ 2.29 แสดงการกำหนดโครงสร้างแบบ DTD

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://www.books.org"
            xmlns="http://www.books.org"
            elementFormDefault="qualified">
  <xsd:element name="BookStore">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Book" minOccurs="1" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Book">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Title" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Author" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Date" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="ISBN" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Publisher" minOccurs="1" maxOccurs="1"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Title" type="xsd:string"/>
  <xsd:element name="Author" type="xsd:string"/>
  <xsd:element name="Date" type="xsd:string"/>
  <xsd:element name="ISBN" type="xsd:string"/>
  <xsd:element name="Publisher" type="xsd:string"/>
</xsd:schema>
```

รูปที่ 2.30 แสดงการกำหนดโครงสร้างแบบ XML Schema

เมื่อเปรียบเทียบการกำหนดโครงสร้างของเอกสารแบบ DTD ในรูปที่ 2.28 กับการกำหนดโครงสร้างแบบ XML Schema ในรูปที่ 2.29 แล้วจะเห็นได้ว่าการกำหนดโครงสร้างแบบเอ็กซ์เอ็มแอลสกีมานั้นมีความซับซ้อนกว่าแบบ DTD พอสมควรการกำหนดอิลิเมนต์ต่างๆในเอ็กซ์เอ็มแอลสกีมาจะกล่าวถึงในหัวข้อถัดไป

2.2.2.2 ประเภทอิลิเมนต์ของโครงสร้างแบบ XML Schema

การกำหนดโครงสร้างของเอกสารเอ็กซ์เอ็มแอลแบบเอ็กซ์เอ็มแอลสกีมาที่แสดงในรูปที่ 2.29 จะประกอบด้วยการกำหนดอิลิเมนต์ต่างๆจำนวนมาก อิลิเมนต์ที่กล่าวถึงในเอ็กซ์เอ็มแอล

สกีมาแบ่งออกเป็นสองชนิดคือ อิลิเมนต์แบบธรรมดา(Simple Type) และอิลิเมนต์แบบซับซ้อน (Complex Type)

อิลิเมนต์แบบธรรมดาคืออิลิเมนต์ที่มีข้อมูลภายในเป็นข้อมูลพื้นฐาน เช่น สตริง ตัวเลข และวันที่เป็นต้น ชนิดข้อมูลของอิลิเมนต์แบบธรรมดาถูกกำหนดในมาตรฐานเอ็กซ์เอ็มแอลสกีมาส่วนที่ 3นอกจากนี้แล้วแอตทริบิวต์ก็จัดว่าเป็นอิลิเมนต์แบบธรรมดาเช่นกัน รูปแบบการประกาศอิลิเมนต์แบบธรรมดาแสดงดังรูปที่ 2.30 และตัวอย่างการประกาศอิลิเมนต์แบบธรรมดาแสดงดังรูปที่ 2.31

```
<xsd: element name= "ชื่ออิลิเมนต์" type = "ชื่อชนิดข้อมูล"/>
```

รูปที่ 2.31 แสดงรูปแบบการประกาศอิลิเลมนต์แบบธรรมดา

```
<xsd: element name= "Title" type = "xsd:string"/>
```

รูปที่ 2.32 ตัวอย่างการประกาศอิลิเมนต์แบบธรรมดา

อิลิเมนต์แบบซับซ้อนคืออิลิเมนต์ที่มีข้อมูลภายในเป็นอิลิเมนต์ หรืออิลิเมนต์ที่มีแอตทริบิวต์ปรากฏอยู่ รูปแบบการประกาศอิลิเมนต์แบบซับซ้อนแสดงดังรูปที่ 2.32 และตัวอย่างของการประกาศอิลิเมนต์แบบซับซ้อนแสดงดังรูปที่ 2.33

```
<xsd: element name= "ชื่ออิลิเมนต์" >
  <xsd:complexType>
  [<xsd:sequence> ,<xsd:choice>]
    <element 1>
    <element 2>
    <element 3>
          .
          .
          .
    <element n>
  [</xsd:sequence>,</xsd:choice>]
  </xsd:complexType>
```

รูปที่ 2.33 แสดงรูปแบบการประกาศอิลิเมนต์แบบซับซ้อน

```
<xsd: element name= “Book” >
    <xsd:complexType>
    <xsd:sequence>
      <xsd:element name =”Tile” type = “xsd:string”>
      <xsd:element name =”Author” type = “xsd:string”>
      <xsd:element name =”Publish” type = “xsd:string”>
    </xsd:sequence>
   </xsd:complexType>
```

รูปที่ 2.34 ตัวอย่างการประกาศอิลิเมนต์แบบซับซ้อน

คีย์เวิร์ด “<xsd:sequence>” xsd(XML Schema Definition) หมายถึงการบ่งบอกว่าเป็นเอกสารข้อมูลของเอ็กซ์เอ็มแอลสกีมาและไฟล์ข้อมูลของเอ็กซ์เอ็มแอลสกีมามีนามสกุล(.xsd) sequence หมายถึงการเรียงลำดับของอิลิเมนต์ลูกที่อยู่ภายในอิลิเมนต์แบบซับซ้อน และอิลิเมนต์ที่ปรากฏในเอกสารเอ็กซ์เอ็มแอลที่อ้างถึงต้องเรียงลำดับตามที่กำหนด และคีย์เวิร์ดอีกตัวคือ (<xsd:choice>) หมายถึงการระบุว่าอิลิเมนต์ลูกตัวใดตัวหนึ่งเท่านั้นที่ปรากฏได้ในเอกสารเอ็กซ์เอ็มแอลที่อ้างถึง

การกำหนดโครงสร้างของเอกสารเอ็กซ์เอ็มแอลแบบเอ็กซ์เอ็มแอลสกีมาสามารถกำหนดจำนวนอิลิเมนต์ที่สามารถปรากฏได้ในเอกสารเอ็กซ์เอ็มแอลด้วยการใช้แอตทริบิวต์ minOccurs เป็นแอตทริบิวต์ที่ระบุจำนวนต่ำสุดที่จำนวนอิลิเมนต์สามารถปรากฏได้ค่าที่ระบุส่วนมาก 0 กับ 1 และ maxOccurs เป็นอิลิเมนต์ที่ระบุค่าสูงสุดของจำนวนอิลิเมนต์ที่สามารถปรากฏได้ในเอกสารเอ็กซ์เอ็มแอลสามารถระบุค่าได้หลายค่าตั้งแต่ 1 จนถึง “unbounded” การระบุค่าสูงสุดเป็น unbounded หมายความว่าสามารถระบุค่าของอิลิเมนต์นั้นได้โดยไม่จำกัดจำนวน ในกรณีที่เราไม่ระบุแอตทริบิวต์ minOccurs และ maxOccurs ก็จะถือเอาค่าปกติมากำหนดจำนวนอิลิเมนต์ ค่าปกติมีค่าเป็น 1 หมายความว่าจำนวนอิลิเมนต์ปรากฏได้ต่ำสุด และสูงสุดเท่ากับ 1 และต้องมีอิลิเมนต์ปรากฏถ้าไม่มีจะเกิดข้อผิดพลาด(Error) ตัวอย่างการใช้แอตทริบิวต์ minOccurs และ maxOccurs แสดงในรูปที่ 2.29 รูปแบบการประกาศอิลิเมนต์ของการกำหนดโครงสร้างเอกสารแบบเอ็กซ์เอ็มแอลสกีมายังมีอีกหลายรูปแบบสามารถหาข้อมูลอ้างอิงได้ที่มาตรฐานเอ็กซ์เอ็มแอลสกีมาของ W3C ในส่วนที่ 2

2.2.2.3 การประกาศแอตทริบิวต์

รูปแบบการประกาศแอตทริบิวต์ของการกำหนดโครงสร้างเอกสารเอ็กซ์เอ็มแอลแบบเอ็กซ์เอ็มแอลสกีมา มีรูปแบบที่เหมือนกับรูปแบบการประกาศอิลิเมนต์แบบธรรมดา เพราะแอตทริบิวต์จัดอยู่ในอิลิเมนต์ประเภทธรรมดา รูปแบบการประกาศแอตทริบิวต์แสดงดังรูปที่ 2.34

```
<xsd: attribute  name= "ชื่อแอตทริบิวต์" type = "ชื่อชนิดข้อมูล"
[use default fixed ค่าปกติ] />
```

รูปที่ 2.35 รูปแบบการประกาศแอตทริบิวต์

เมื่อ ชื่อแอตทริบิวต์ หมายถึงชื่อของแอตทริบิวต์ ชื่อชนิดข้อมูล หมายถึงชื่อชนิดของข้อมูลเช่น สตริง นัมเบอร์ และวันที่ เป็นต้น นอกจากนี้ยังมีแอตทริบิวต์อื่นๆให้เลือกคือ แอตทริบิวต์ use หมายถึง การระบุระดับความจำเป็นของแอตทริบิวต์นี้ว่าต้องมีหรือไม่ ค่าของแอตทริบิวต์นี้มี 3 ค่า คือ required (จำเป็นต้องมี) option(มีหรือไม่ก็ได้) และ prohibited(ไม่ต้องมี) แอตทริบิวต์ default ใช้กำหนดค่าปกติของแอตทริบิวต์ แอตทริบิวต์ fixed ใช้กำหนดค่าของแอตทริบิวต์ แอตทริบิวต์ทั้ง 3 คือ use default และ fixed จะมีหรือไม่ก็ได้เนื่องจากเป็นแอตทริบิวต์ทางเลือก ตัวอย่างการประกาศแอตทริบิวต์แสดงดังรูปที่ 2.35

```
<xsd: attribute  name="country" type = "xsd:string" fixed "TH" />
```

รูปที่ 2.36 ตัวอย่างการประกาศแอตทริบิวต์

2.2.3 ความแตกต่างระหว่าง DTD และ XML Schema

การกำหนดโครงสร้างเอกสารเอ็กซ์เอ็มแอลโดยใช้ DTD และ XML Schema มีความแตกต่างกันหลายประการมีรายละเอียดดังนี้

- XML Schema มีรูปแบบการเขียน(Syntax)เหมือนกับเอกสารเอ็กซ์เอ็มแอลทำให้สามารถใช้ตัวแปลภาษาตัวเดียวกันได้ แต่ DTD มีรูปแบบการเขียนต่างจากเอกสารเอ็กซ์เอ็มแอล ทำให้ต้องมีตัวแปลภาษาสำหรับ DTD แยกต่างหากเกิดความสิ้นเปลืองและความยากลำบากในการใช้งาน

- XML Schema สนับสนุนการใช้เนมสเปส(Namespaces) ทำให้สามารถใช้แท็กซ์ที่มีชื่อเหมือนกันในเอกสารเดียวกันแต่มีความหมายแตกต่างกัน สำหรับการกำหนดโครงสร้างแบบ DTD นั้นไม่สามารถทำได้

- XML Schema มีชนิดของข้อมูลมากกว่า DTD และชื่อชนิดของข้อมูลมีชื่อเรียกที่ใช้กันทั่วไปในการเขียนโปรแกรมเช่น สตริง นัมเบอร์ เป็นต้น สำหรับการกำหนดโครงสร้างแบบ DTD นั้นมีชื่อเรียกชนิดของข้อมูลที่ไม่คุ้นเคยเช่น PCDATA NMTOKENS เป็นต้น

- การกำหนดโครงสร้างเอกสารเอ็กซ์เอ็มแอลแบบ XML Schema สามารถกำหนดลำดับของอิลิเมนต์ที่ปรากฏในเอกสารเอ็กซ์เอ็มแอลสกีมาได้ และ

- การกำหนดโครงสร้างเอกสารเอ็กซ์เอ็มแอลแบบ XML Schema สามารถกำหนดรูปแบบชนิดของข้อมูลตามที่ผู้ใช้ต้องการได้เช่น เบอร์โทรศัพท์มีรูปแบบ “dd-ddd-dddd” หมายถึงรูปแบบของเบอร์โทรศัพท์ที่มีตัวเลขข้างหน้า 2 หลักตามด้วยขีดและเลขอีก 3 หลักตามด้วยขีดและเลขอีก 4 หลักเป็นต้น

2.3 การประยุกต์ใช้งานภาษาเอ็กซ์เอ็มแอล

ปัจจุบันมีการนำภาษาเอ็กซ์เอ็มแอลไปประยุกต์ใช้งานหลายด้าน [16] พอจะสรุปเป็นหัวข้อหลักๆได้ดังต่อไปนี้

2.3.1 ใช้สำหรับแยกข้อมูลออกจากภาษา HTML

การใช้ภาษา HTML แสดงผลข้อมูลบนบราวเซอร์ข้อมูลที่จะแสดงถูกเก็บในภาษา HTML แต่เราสามารถใช้ภาษาเอ็กซ์เอ็มแอลร่วมกับ HTML ได้โดยใช้ HTML มีบทบาทสำคัญในการแสดงผลและจัดตำแหน่งการแสดงผล ส่วนเอ็กซ์เอ็มแอลจะเป็นส่วนเก็บข้อมูลที่จะแสดงแยกเก็บในไฟล์(.xml)ต่างหาก ข้อมูลเอ็กซ์เอ็มแอลจะปรากฏเป็นกลุ่มๆในภาษา HTML หรือเรียกว่า “Data Islands”

2.3.2 ใช้สำหรับแลกเปลี่ยนข้อมูลระหว่างแอปพลิเคชั่น

การเก็บข้อมูลในระบบคอมพิวเตอร์ และฐานข้อมูลในปัจจุบันมีหลากหลายรูปแบบทำให้เป็นปัญหาสำคัญสำหรับการกำหนดรูปแบบของการส่งข้อมูลบนระบบอินเตอร์เน็ตในปัจจุบัน การขจัดปัญหานี้คือต้องหารูปแบบของข้อมูลที่ไม่ขึ้นอยู่กับแพลตฟอร์ม และสามารถอ่านได้หลายๆแอปพลิเคชั่น ซึ่งตรงกับคุณสมบัติของภาษาเอ็กซ์เอ็มแอลเพราะข้อมูลในเอกสารเอ็กซ์เอ็มแอลมีลักษณะเป็นไฟล์ข้อมูลธรรมดาที่อ่านได้ทั้งมนุษย์ และโปรแกรมแอปพลิเคชั่น

2.3.3 ใช้สำหรับเก็บข้อมูล

ภาษาเอ็กซ์เอ็มแอลสามารถนำไปประยุกต์ใช้ในการเก็บข้อมูลทั้งในรูปของไฟล์ข้อมูลและการเก็บข้อมูลลงฐานข้อมูล นอกจากนี้ยังมีแอปพลิเคชั่นที่สามารถค้นคืนข้อมูลเอ็กซ์เอ็มแอลมาแสดงผลในแพลทฟอร์มต่างได้

2.3.4 ใช้สำหรับการสร้างภาษาใหม่

เนื่องจากภาษาเอ็กซ์เอ็มแอลเป็นภาษาที่สามารถกำหนดโครงสร้างขึ้นเองได้ จึงมีการนำภาษาเอ็กซ์เอ็มแอลมาเป็นแม่แบบในการสร้างภาษาใหม่ขึ้นมาใช้งานกับแพลตฟอร์มต่างๆเช่น WML(Wireless Markup Language) ใช้สำหรับอุปกรณ์แบบพกพาเช่นโทรศัพท์มือถือเป็นต้น

2.3.5 ใช้สำหรับส่งข้อมูลระหว่างองค์กรธุรกิจ(B2B :Business to Business) [2]

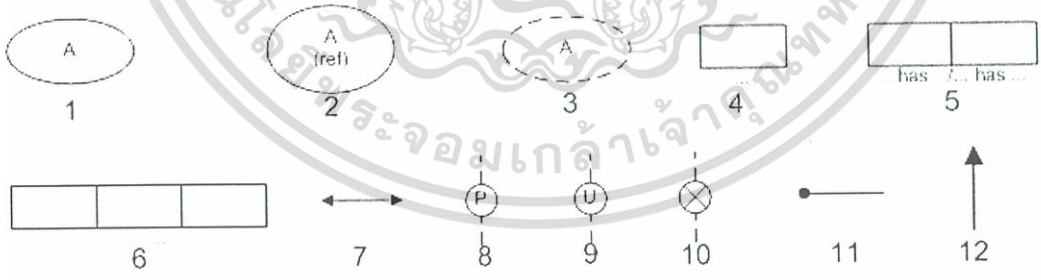
ภาษาเอ็กซ์เอ็มแอลเป็นภาษาที่สามารถกำหนดโครงสร้างของเอกสารที่สร้างขึ้นได้ ดังนั้นเมื่อองค์กรต้องการแลกเปลี่ยนข้อมูลกันจึงต้องมีการกำหนดโครงสร้างที่เหมือนกันจึงจะทำให้เอกสารที่แลกเปลี่ยนกันนั้นมีความถูกต้องและมีความน่าเชื่อถือสูง

บทที่ 3
แบบจำลองข้อมูลไนแอม

แบบจำลองข้อมูลระดับแนวคิดไนแอม(NIAM Conceptual Model หรือ NIAM Schema) นั้น คำว่า NIAM ย่อมาจาก Nijssen's Information Analysis Methodology [4] ซึ่งเป็นรูปแบบหนึ่งของแบบจำลองข้อมูลในระดับแนวคิด ที่ได้มีการคิดค้นมาตั้งแต่ปี ค.ศ. 1977 โดย ศาสตราจารย์ จี เอ็ม ไนเซน ได้เสนอแบบจำลองข้อมูลไนแอม ซึ่งเป็นแบบจำลองที่มีพื้นฐานมาจากโครงสร้างภาษาธรรมชาติ ที่มีรูปประโยคประกอบด้วย ประธาน กริยา และกรรม เป็นแบบจำลองระดับแนวคิด ที่มีความหมายและเครื่องหมาย แสดงความสัมพันธ์ของข้อมูล และข้อจำกัดของข้อมูลได้อย่างชัดเจน นอกจากนี้แล้วยังสามารถแปลง แบบจำลองดังกล่าว เป็นโครงสร้างของฐานข้อมูลเชิงสัมพันธ์ (Relational Database Schema) ซึ่งผลลัพธ์ที่ได้จากการแปลงจะอยู่ในรูปของ Fifth Normal Form(5NF) [18,19] ดังนั้น จึงมีการนำแบบจำลองข้อมูลระดับแนวคิดไนแอม มาใช้ช่วยในการออกแบบฐานข้อมูลเชิงสัมพันธ์กันอย่างแพร่หลาย เพราะมีการใช้สัญลักษณ์ที่แสดงความสัมพันธ์ของข้อมูลอย่างชัดเจน และง่ายต่อการเข้าใจ และในปัจจุบันได้มีการนำแบบจำลองข้อมูลแบบไนแอมไปประยุกต์ใช้กับงานด้านอื่นๆ เช่น งานวิจัยเกี่ยวกับระบบผู้เชี่ยวชาญ งานวิจัยด้านคลังข้อมูล เป็นต้น

3.1 ส่วนประกอบของไนแอมสกีมา

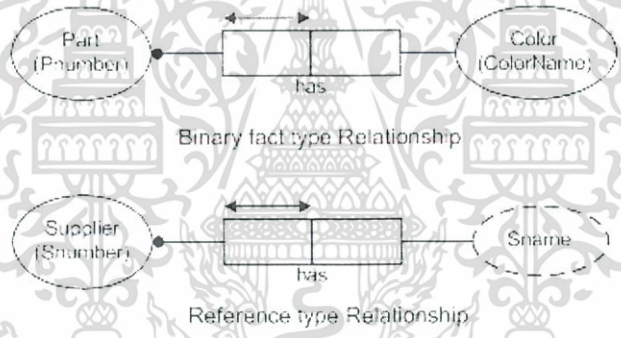
แบบจำลองข้อมูลไนแอมประกอบด้วยสัญลักษณ์พื้นฐาน[12] ที่ใช้ในการสื่อความหมายของประโยคภาษาธรรมชาติ ตามที่ผู้สร้างต้องการนำเสนอ สัญลักษณ์ต่างๆแสดงดังรูปที่ 3.1



รูปที่ 3.1 แสดงสัญลักษณ์พื้นฐานของไนแอมสกีมา

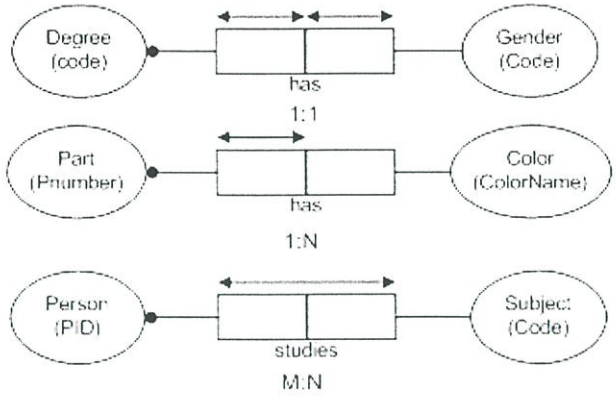
สัญลักษณ์ที่แสดงในรูปที่ 3.1 เป็นสัญลักษณ์พื้นฐานที่นำมาใช้สร้างแบบจำลองข้อมูลไนแอม แต่ละสัญลักษณ์มีความหมายต่างกันคือ สัญลักษณ์หมายเลข 1 – 3 เป็นสัญลักษณ์ที่แทนออบเจ็กต์

ไทป์(Object Type) ซึ่งออบเจ็กต์ไทป์แบ่งออกเป็น 2 ประเภทคือ Entity type(หมายเลข 1,2) และ Value or Label type(หมายเลข 3) หมายเลข 4-6 แสดงถึงสัญลักษณ์ของบทบาทหรือกิริยา(Role) ของออบเจ็ต์ หมายเลข 4 แสดงบทบาทเดียว(Unary role) หมายเลข 5 แสดงสัญลักษณ์ของสองบทบาท(Binary role) และหมายเลข 3 แสดงสัญลักษณ์ของสามบทบาท(Ternary role) หมายเลข 7 ถึง 12 แสดงถึงคอนสเตรนหรือกฎบังคับความถูกต้องแบบต่างๆคือ หมายเลข 7 แสดง Internal Uniqueness Constraint หมายเลข 8 แสดง External Uniqueness Constraint Primary หมายเลข 9 แสดงสัญลักษณ์ของ External Uniqueness Constraint หมายเลข 10 แสดงสัญลักษณ์ของ Exclusion Constraint หมายเลข 11 แสดงสัญลักษณ์ของ Mandatory role และหมายเลข 12 แสดงสัญลักษณ์ของ Subtype สัญลักษณ์ของไนแอมสกีมาที่กล่าวมานี้เป็นเพียงสัญลักษณ์พื้นฐานที่นำมาใช้สำหรับงานวิจัยนี้ แต่ยังมีอีกหลายสัญลักษณ์ที่ไม่ได้กล่าวถึง การสร้างไนแอมสกีมาเกิดจากการนำเอาออบเจ็กต์มาเชื่อมต่อกันหรือมาสัมพันธ์กัน(Relationship) ประเภทของความสัมพันธ์กันระหว่างออบเจ็กต์มี 2 ประเภทคือ Fact type relationship และ Reference type relationship ความสัมพันธ์ทั้งสองแบบแสดงดังรูปที่ 3.2



รูปที่ 3.2 แสดงประเภทความพันธ์ของออบเจ็กต์

โดยที่ Fact type relationship แสดงถึงความสัมพันธ์ระหว่างเอนติทีไทป์กับเอนติทีไทป์ และ Reference type relationship แสดงถึงความสัมพันธ์ระหว่างเอนติทีไทป์กับเลเบลไทป์ นอกจากนี้แล้วประเภทของความสัมพันธ์ยังสามารถแยกได้อีก 3 ประเภทคือความสัมพันธ์แบบ 1:1 1:N และ M:N การแยกประเภทของความสัมพันธ์วิธีนี้จะพิจารณาร่วมกับ Internal Uniqueness Constraint โดยพิจารณาจากจำนวนบทบาทที่ Internal Uniqueness Constraint ครอบคลุม ประเภทของความสัมพันธ์ทั้งสามรูปแบบแสดงดังรูปที่ 3.3



รูปที่ 3.3 แสดงความสัมพันธ์แบบ 1:1 1:N และ M:N

3.2 ขั้นตอนการสร้างไนแอมสกีมา

ขั้นตอนการสร้างไนแอมสกีมา (CSDP :Conceptual Schema Design Procedure) [4,21] มีขั้นตอนในการสร้าง 7 ขั้นตอนดังนี้

1. เรียบเรียงข้อมูลที่จะนำมาใช้สร้างไนแอมสกีมา ข้อมูลที่นำมาสร้างไนแอมสกีมาอาจเป็นข้อมูลที่เป็นตาราง ข้อมูลประโยคคำพูดเชิงพรรณนา หรือเป็นแบบฟอร์มต่างๆ แล้วทำการแปลงข้อมูลให้อยู่ในรูปของอิลิเมนต์ทารีแฟคไทป์(Elementary fact type) หมายถึงเป็นประโยคที่เล็กที่สุดไม่สามารถแยกย่อยลงไปได้อีก

ตัวอย่างเช่น

F1 The Academic with empnr 715 **has** EmpName 'Adams A'

F2 The Academic with empnr 715 **work for** the Dept named 'Computer Science'

F3 The Academic with empnr 715 **occupies** the Room with roomnr '69-301'

F4 The Academic with empnr 715 **uses** the Extension with extnr '2345'

F5 The Extension with extnr '2345' **provides** the AccesLevel with code 'LOC'

F6 The Acadmic with empnr 715 **is contracted till** the Date with mdy-code '01-31/95'

F7 The Academic with empnr 139 **is tenured.**

จากประโยคข้อความทั้ง 7 ประโยคข้างต้นสามารถนำมาเรียบเรียงเป็นประโยคที่กระชับเพื่อสะดวกในการนำไปเขียนไนแอมสกีมา หรือไนแอมไดอะแกรมตัวอักษรหนาแสดงถึง Predicate ที่เชื่อมต่อระหว่างออบเจ็กต์ หรือ Predicate ก็คือบทบาทนั่นเอง และชื่อของออบเจ็กต์จะนำหน้าด้วยตัวอักษรพิมพ์ใหญ่ ตัวอักษรย่อต่างๆ มีความหมายคือ empnr หมายถึง Employee Number EmpName หมายถึง Employee Name Dept หมายถึง Department roomnr หมายถึง Room Number เป็นต้น

F1 Academic **has** EmpName

F2 Academic **work for** Dept

F3 Academic **occupies** Room

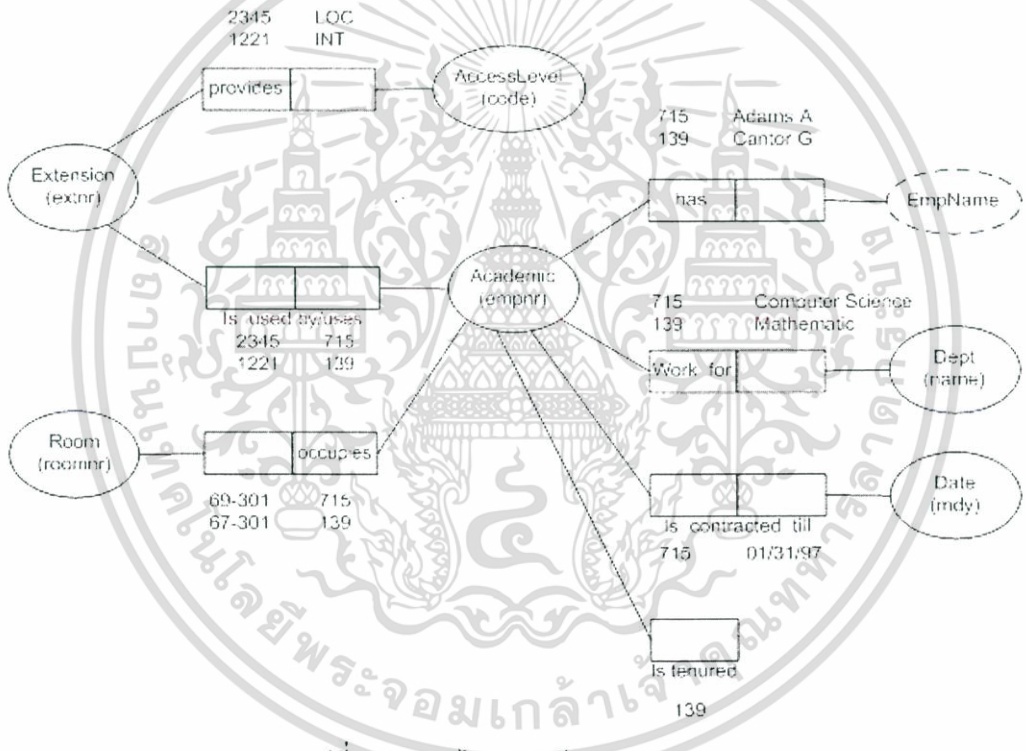
F4 Academic **uses** Extension

F5 Extension **provides** AccessLevel

F6 Academic **is contracted till** Date

F7 Academic **is tenured**.

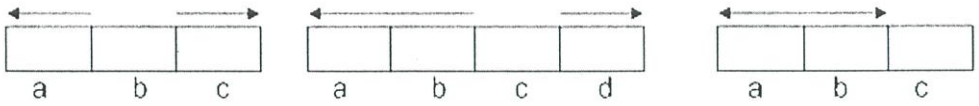
2. วาดไนแอมสกีมาจากประโยคที่เรียบเรียงไว้จากข้อที่ 1 และทดสอบความถูกต้องโดยการใส่ข้อมูลประมาณ 2-3 ข้อมูลดังแสดงในรูปที่ 3.4



รูปที่ 3.4 แสดงไนแอมสกีมาจาก F1- F7

3. ทำการตรวจเช็คว่ามีแฟคไทป์ไหนบ้างที่สามารถคำนวนหาได้จากแฟคไทป์อื่นๆ กรณีที่ต้องการให้แฟคไทป์ที่สามารถคำนวณได้ปรากฏในสกีมาเราจะหมายเหตุไว้เป็นข้อความด้านล่าง โดยใช้เครื่องหทายดอกจันทร์(*) แล้วตามด้วยกฎหรือสูตรของการคิดคำนวณตัวอย่างแสดงในรูปที่ 3.7

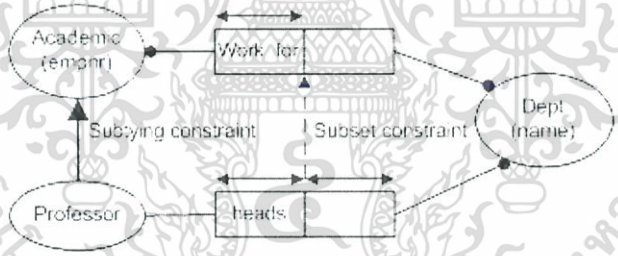
4. ทำการใส่เครื่องหมาย Internal Uniqueness Constraint และตรวจเช็คความสัมพันธ์ระหว่างออบเจ็กต์ว่าเป็นแบบใดเช่นมีความสัมพันธ์แบบ 1:1 1:N หรือ M:N เพื่อจะได้เขียนเครื่องหมาย Internal Uniqueness Constraint ครอบคลุมจำนวนบทบาทได้ถูกต้อง ในกรณีที่มีจำนวนบทบาทมากกว่า 2 หรือตั้งแต่ 3 บทบาทขึ้นไปจำนวนบทบาทที่ Internal Unique Constraint ครอบคลุมจะมีได้ n-1 เมื่อ n คือจำนวนบทบาท ตัวอย่างแสดงดังรูปที่ 3.5



รูปที่ 3.5 แสดง Internal Uniqueness Constraint บน n-ary fact type

5. ใส่เครื่องหมาย Mandatory Role Constraint และทำการตรวจเช็ค Logical Derivation เป็นการตรวจเช็คการได้มา(Derived)ของแฟคไทป์ที่เกิดจากแฟคไทป์อื่นที่ไม่เกี่ยวกับการคำนวณทางคณิตศาสตร์

6. ใส่เครื่องหมายกฎบังคับความถูกต้องอื่นๆเช่น Value constraint Set comparison และ Subtyping constraint ตัวอย่างของ Value constraint คือ {'INT', 'NAT','LOC'} ตัวอย่างของ Subset constraint Subtyping constraint แสดงดังรูปที่ 3.6

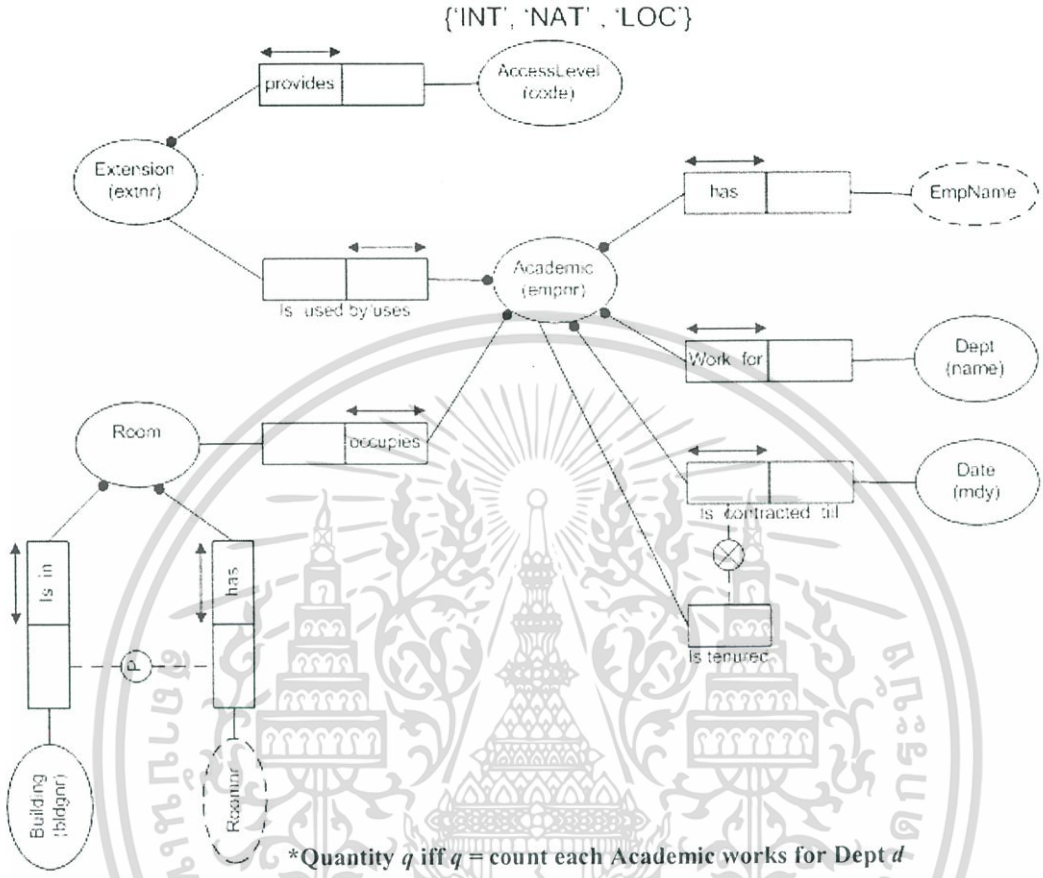


รูปที่ 3.6 แสดง Subtype และ Subset constraint

รูปที่ 3.6 แสดงให้เห็นว่า ออบเจ็กต์ไทป์ Professor เป็น Subtype ของออบเจ็กต์ไทป์ Academic และบทบาทชื่อ head เป็น Subset ของบทบาทชื่อ work for กล่าวได้ว่า "หัวหน้าภาคทำงานที่ภาคนั้นๆ" เช่นหัวหน้าภาควิชาวิศวกรรมคอมพิวเตอร์ก็ทำงานที่ภาควิชาวิศวกรรมคอมพิวเตอร์

7. ทำการเพิ่มคอนสเตรนอื่นๆ และตรวจเช็คความถูกต้องขั้นสุดท้ายเพื่อให้ได้สกีมาที่ครบถ้วนสมบูรณ์ตามที่ผู้สร้างสกีมาต้องการ ไนแอมสกีมามีคอนสเตรนหลายๆคอนสเตรนซึ่งไม่อาจกล่าวได้หมดในงานวิจัยนี้ เนื่องจากไนแอมสกีมามีคอนสเตรนที่หลากหลายจึงทำให้มีนักวิจัยหลายท่านนำไปประยุกต์ใช้กับงานด้านต่างๆ จากที่กล่าวมาแล้วตั้งแต่ข้อ 1 ถึงข้อ 7 สามารถแสดง

ไนแอมสกีมาที่สมบูรณ์ดังแสดงในรูปที่ 3.7 ตัวอักษรย่อ INT หมายถึง International NAT หมายถึง National และ LOC หมายถึง Local

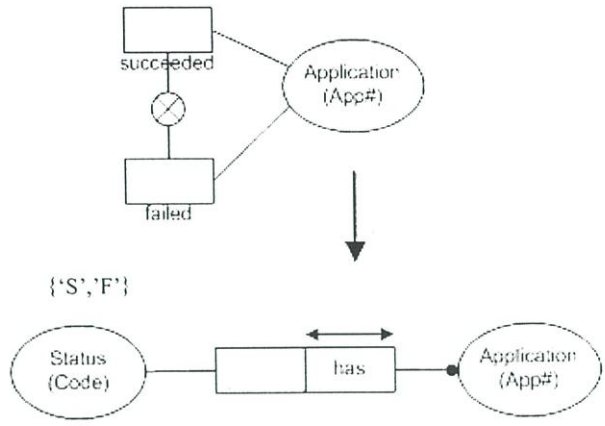


รูปที่ 3.7 แสดงตัวอย่างไนแอมสกีมาที่สมบูรณ์(จากขั้นตอนที่ 1 ถึง 7)

3.3 กระบวนการในการแปลงแบบจำลองระดับแนวคิดไนแอม ให้อยู่ในรูปของโครงสร้างฐานข้อมูลเชิงสัมพันธ์

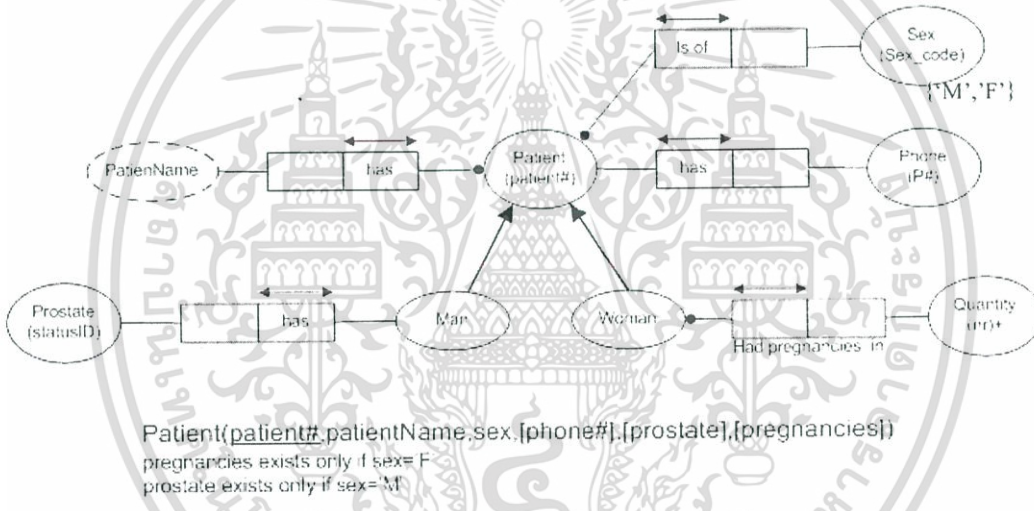
กระบวนการแปลงไนแอมสกีมาเป็นโครงสร้างฐานข้อมูลเชิงสัมพันธ์ [4] หรือรีเลชันแนลสกีมาประกอบด้วยขั้นตอนต่างๆ ดังนี้

1. เปลี่ยนทุกๆ Unary Elementary Fact Type ให้อยู่ในรูปของ Binary Elementary Fact Type ตัวอย่างแสดงดังรูปที่ 3.8



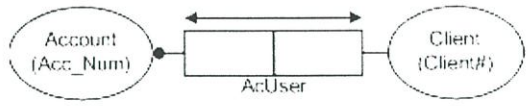
รูปที่ 3.8 แสดงการแปล Unary Fact type เป็น Binary Fact type

2. ทำการย้าย Fact type ที่เชื่อมต่อกับ Subtype มาเชื่อมต่อกับ Supertype แทน



รูปที่ 3.9 การแปลง Subtype constraint เป็น Relational Schema

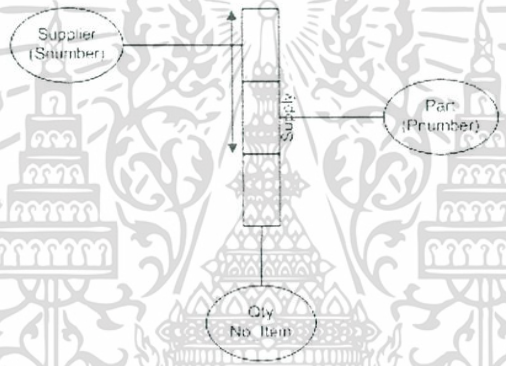
3. สร้างรีเลชันแนลสกีมาสำหรับแต่ละBinary fact type ซึ่งมี Internal uniqueness constraints กำกับอยู่บนทุกบทบาท โดยนำแต่ละ Entity Type ที่เกี่ยวข้องกับ Binary fact type มาสร้างเป็น Attribute ของรีเลชันแนลสกีมานั้น ๆ ทุก Attribute ที่สร้างขึ้นจะถูกกำหนดให้เป็น mandatory และ Attribute ทั้งหมดที่ถูก Internal uniqueness constraints กำกับอยู่นั้น ให้ทำการกำหนดเป็นคีย์หลัก(Primary Key) ของรีเลชัน



AcUser(brance#,client#)

รูปที่ 3.10 แสดงการแปลง Binary Fact type (n:m) เป็น Relational Schema

4. สร้างรีเลชันแนลสกีมาสำหรับแต่ละ n-ary fact type ที่มีมากกว่า 2 บทบาทขึ้นไป ซึ่งมี Internal uniqueness constraints กำกับอยู่ n-1 บทบาท โดยนำแต่ละ Entity Type ที่เกี่ยวข้องกับ n-ary fact type มาสร้างเป็น Attribute ของรีเลชันแนลสกีมานั้น ๆ ทุก ๆ Attribute ที่สร้างขึ้น จะถูกกำหนดให้เป็น mandatory และ Attribute ทั้งหมดที่ถูก internal uniqueness constraints กำกับอยู่นั้น ให้ทำการกำหนดเป็นคีย์หลัก (Primary Key) ของรีเลชันแนลสกีมา

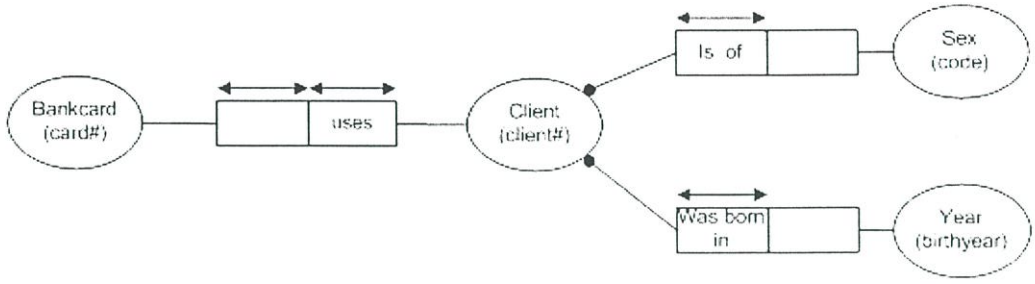


Spupply(Snumber,Pnumber,No_Item)

รูปที่ 3.11 แสดงการแปลง n-ary Fact type เป็น Relational Schema

5. จัดกลุ่ม Binary Fact type ที่มี Internal uniqueness constraints กำกับอยู่ 1 บทบาท อยู่ฝั่งเดียวกัน สำหรับกรณีที่มี Internal uniqueness constraints กำกับอยู่ 1 บทบาททั้งสองฝั่ง ให้ทำการจัดตามฝั่งที่มี mandatory constraints ที่ปรากฏอยู่ หากไม่มี มี mandatory constraints ปรากฏอยู่ทั้งสองฝั่งให้ทำการจัดตามลำดับของตัวอักษร

6. สร้างรีเลชันแนลสกีมาตามกลุ่มที่ได้จัดไว้ในขั้นตอนที่ 5 โดยทำการสร้าง Attribute ขึ้นจาก Entity Type ที่มีอยู่ และทำการกำหนด mandatory constraints ตาม Entity Typeนั้น ๆ ตัวอย่างแสดงดังรูปที่ 3.12



Client(client#,sex,birthyear)
Bankcard(card#,client#)

รูปที่ 3.12 แสดงการแปลง Binary Fact type แบบ 1:1 และ 1:N เป็น Relational Schema

บทที่ 4
กระบวนการการแปลงไนแอมสกีมาเป็นเอ็กซ์เอ็มแอลสกีมา

กระบวนการแปลงไนแอมสกีมา (NIAM Conceptual Schema) เป็นเอ็กซ์เอ็มแอลสกีมา (XML Schema) ที่นำเสนอในบทนี้จะกล่าวถึงขั้นตอนการแปลงจากไนแอมสกีมาเป็นเอ็กซ์เอ็มแอลสกีมา และแสดงรายละเอียดของกระบวนการสร้างไนแอมเมต้าสกีมา จุดประสงค์หลักของงานวิจัยคือการนำไนแอมสกีมามาใช้ในการออกแบบเอ็กซ์เอ็มแอลสกีมาในระดับแนวคิด(Conceptual Level) เพื่อให้ได้เอ็กซ์เอ็มแอลสกีมาที่ครบถ้วนสมบูรณ์ และมั่นใจได้ว่าเอ็กซ์เอ็มแอลสกีมาที่ได้ไม่มีความซ้ำซ้อนของข้อมูล เพื่อความถูกต้องในการแลกเปลี่ยนข้อมูลระหว่างฐานข้อมูล การแปลงจากไนแอมสกีมาเป็นเอ็กซ์เอ็มแอลสกีมายังมีบางคุณสมบัติที่ไม่สามารถแปลงได้เนื่องจากการไม่เข้ากันของทั้งสองทฤษฎีแสดงดังตารางที่ 4.1

ตารางที่ 4.1 ตารางเปรียบเทียบคุณสมบัติของไนแอมสกีมาและเอ็กซ์เอ็มแอลสกีมา

NIAM	XML Schema
Value Type	Simple Type
Fact Type	Complex Type
Mandatory Constraint	Occurrence Constraint
Option Constraint	Occurrence Constraint
Subtype Constraint	Single Inheritance
Multiple Inheritance	No Have
Value and rang Constraint	Simple Type (Enumerate)
No Have	Child element order
No Have	Mix content with element
Sub set Constraint	No Have
Exclusion Constraint	No Have

4.1 การแปลงไนแอมสกีมาเป็นเอ็กซ์เอ็มแอลสกีมา

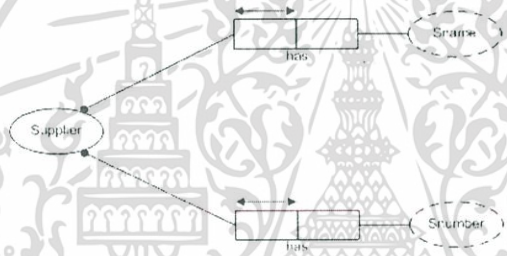
การแปลงไนแอมสกีมาเป็นเอ็กเอ็มแอลสกีมาสำหรับงานวิจัยนี้ในทางปฏิบัติผู้ใช้งานซอฟต์แวร์จะต้องสร้างไนแอมสกีมาในส่วนของไนแอมเอดิเตอร์ก่อน แล้วทำการเก็บข้อมูลของไนแอมสกีมาที่สร้างเสร็จแล้วลงฐานข้อมูล(ไนแอมเมต้าสกีมา) จากนั้นจึงจะสามารถแปลงเป็นเอ็กซ์เอ็มแอลสกีมาอย่างอัตโนมัติ ขั้นตอนการแปลงไนแอมสกีมาเป็นเอ็กซ์เอ็มแอลสกีมา มีกระบวนการดังนี้

4.1.1. กำหนดอิลิเมนต์เริ่มต้นของเอ็กซ์เอ็มแอลสกีมา (Root Element)

การกำหนดอิลิเมนต์เริ่มต้นของเอ็กซ์เอ็มแอลสกีมา มีรูปแบบการกำหนดโดยใช้ชื่อของไนแอมสกีมา หรือชื่อของฐานข้อมูล เพื่อให้สามารถสื่อความหมายได้อย่างเหมาะสม

4.1.2. แปลงแต่ละลาเบลไทป์ (Label Type) เป็น XML Schema Simple type

เนื่องจากลาเบลไทป์ของไนแอมสกีมาแสดงถึงออปเจ็คที่เป็นข้อมูลพื้นฐานเช่น สตริง นัมเบอร์ และ วันที่เป็นต้น ซึ่งตรงกับคุณสมบัติของ XML Schema simple type



รูปที่ 4.1 ไนแอมสกีมานำเสนอลาเบลไทป์

รูปแบบของเอ็กซ์เอ็มแอลสกีมาที่เกิดขึ้น จากการแปลงลาเบลไทป์ของไนแอมสกีมาแสดงได้ดังรูปที่ 4.2

```
<simpleType name= “ชื่อลาเบลไทป์”>
   <sp:restriction base="ชนิดข้อมูลของลาเบลไทป์">
        [ <sp:length value="ขนาดความยาวของชื่อลาเบลไทป์"/>]
        [<sp:pattern value="รูปแบบของชื่อลาเบลไทป์"/> ]
     </sp:restriction>
   </sp:simpleType>
```

รูปที่4.2 รูปแบบเอ็กเอ็มแอลสกีมาจากการแปลงลาเบลไทป์ของไนแอมสกีมา

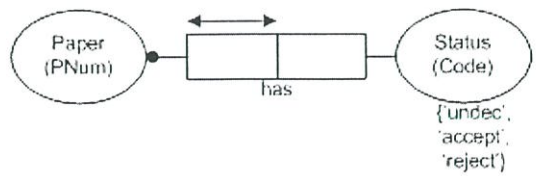
เมื่อ ชื่อลาเบลไทป์ หมายถึงชื่อของลาเบลไทป์(ที่อยู่ในวงรีเส้นประ) ของไนแอมสกีมา ชนิดข้อมูลของลาเบลไทป์ หมายถึงชนิดข้อมูลของลาเบลไทป์ เช่น ข้อมูลแบบสติง(String) หรือ นัมเบอร์ (Number) เป็นต้น ขนาดความยาวของชื่อลาเบลไทป์ หมายถึงความยาวของชื่อลาเบลไทป์ที่กำหนด ตัวอย่างเช่น ถ้ากำหนดชื่อลาเบลไทป์เป็น Sname ดังนั้นขนาดความยาวของชื่อลาเบลไทป์มีค่าเท่ากับ 5 รูปแบบของชื่อลาเบลไทป์ หมายถึงรูปแบบของชื่อลาเบลไทป์ที่กำหนดรูปแบบในการตั้งชื่อลาเบลไทป์ ซึ่งมีรูปแบบดังนี้ [A-Z]{1}-\d{3} หมายถึง ขึ้นต้นด้วยอักษรพิมพ์ใหญ่ และตามด้วยอักษรพิมพ์เล็ก ตัวอย่างเช่นเมื่อชื่อลาเบลไทป์เป็น Sname จากรูปที่4.1 ของไนแอมสกีมาประกอบด้วยลาเบลไทป์ 2 ลาเบลไทป์คือ Saname และ Snumber จะได้เอ็กซ์เอ็มแอลสกีมาจากรูปแบบที่กำหนดรูปที่ 4.2 ดังรูปที่4.3

```
<simpleType name= “Snumber”>
  <sp:restriction base="sp:string">
      <sp:length value="7"/>
      <sp:pattern value="[A-Z]{1}-\d{3}"/>
  </sp:restriction>
</sp:simpleType>
<simpleType name= “Sname”>
  <sp:restriction base="sp:string">
      <sp:pattern value="[A-Z][a-z]*"/>
  </sp:restriction>
</sp:simpleType>
```

รูปที่4.3 ตัวอย่างเอ็กซ์เอ็มแอลสกีมาที่ได้จากการแปลงไนแอมสกีมารูปที่4.1

4.1.3. แปลงไนแอม Value Constrain เป็น XML Schema Simple type

ไนแอม Value Constrain เป็นการกำหนดขอบเขตโดเมนของเอนทิตีไทป์ซึ่งตรงกับคุณสมบัติ Enumerate ของ XML Simple type



รูปที่4.4 ไนแอมสกีมานำเสนอ Value Constraint(Status(Code))

รูปแบบของเอ็กซ์เอ็มแอลสกีมาที่เกิดขึ้น จากการแปลงไนแอม Value Constraint แสดงได้ดังรูปที่ 4.4

```
<simpleType name="ชื่อเอนทิตีไทป์ หรือลาเบลไทป์" base="ชนิดข้อมูล"/>
        <enumeration value="รูปแบบของ Value Constrain ตัวที่1"/>
        <enumeration value="รูปแบบของ Value Constrain ตัวที่ 2"/>
                .
                .
                .
        <enumeration value="รูปแบบของ Value Constrain ตัวที่ n"/>
</simpleType>
```

รูปที่4.5 รูปแบบเอ็กซ์เอ็มแอลสกีมาจากการแปลงValue Constrain ของไนแอมสกีมา

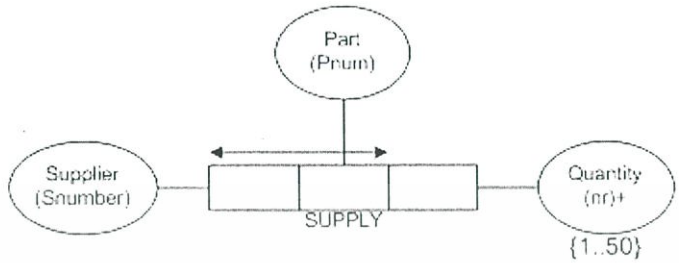
เมื่อ ชื่อเอนทิตีไทป์ หรือลาเบลไทป์ หมายถึงชื่อเอนทิตีไทป์ หรือลาเบลไทป์ของไนแอมสกีมาที่มีรูปแบบของ Value Constrain ชนิดข้อมูล หมายถึงชนิดของข้อมูลของเอนทิตีไทป์ หรือลาเบลไทป์ เช่นมีชนิดข้อมูลเป็นสตริง(String) รูปแบบของ Value Constrain หมายถึง รูปแบบของ Value Constraint ที่กำหนดในไนแอมสกีมา เช่น undec accep และ reject ที่กำหนดในไนแอมสกีมารูปที่4.4 จากรูปที่4.4 ไนแอมสกีมาประกอบด้วย 2 เอนทิตีไทป์คือ Paper(PNum) และ Status(Code) โดยเอนทิตีไทป์ Status(Code) มีการระบุ Value constraint ประกอบด้วย undec accep และ reject จากรูปแบบที่กำหนดในรูปที่ 4.5 จะได้เอ็กซ์เอ็มแอลสกีมาดังรูปที่4.6

```
<simpleType name="StatusCode" base="string"/>
        <enumeration value="undec"/>
        <enumeration value="accept"/>
        <enumeration value="reject"/>
</simpleType>
```

รูปที่4.6 ตัวอย่างเอ็กซ์เอ็มแอลสกีมาที่ได้จากการแปลงไนแอมสกีมารูปที่4.4

4.1.4. แปลงไนแอม Range Constrain เป็น XML Schema Simple type

Range Constrain ของไนแอมสกีมาเป็นการกำหนดขอบเขตโดเมนของเอนทิตีที่มีลักษณะเป็นช่วงของค่าที่เป็นตัวเลข ซึ่งตรงกับคุณสมบัติ min , max ของ XML Schema simple type



รูปที่4.7 ไนแอมสกีมานำเสนอ Range Constraint (Quantity(nr)+{1..50})

รูปแบบของเอ็กซ์เอ็มแอลสกีมาที่เกิดขึ้น จากการแปลงไนแอม Range Constraint รูปที่4.7 แสดงได้ดังรูปที่ 4.8

```
<simpleType name="ชื่อเอนทิตีไทป์ หรือลาเบลไทป์" base="ชนิดข้อมูล">
  <minInclusive value="ค่าต่ำสุดของ Range"/>
  <maxInclusive value="ค่าสูงสุดของ Range"/>
</simpleType>
```

รูปที่4.8 รูปแบบเอ็กซ์เอ็มแอลสกีมาจากการแปลง Range Constrain ของไนแอมสกีมา

เมื่อชื่อเอนทิตีไทป์ หรือลาเบลไทป์ หมายถึงชื่อเอนทิตีไทป์ หรือลาเบลไทป์ของไนแอมสกีมาที่มีรูปแบบของ Range Constrain ชนิดข้อมูล หมายถึงชนิดของข้อมูลของเอนทิตีไทป์ หรือลาเบลไทป์ เช่นมีชนิดข้อมูลเป็นสตริง (String) ค่าต่ำสุดของ Range หมายถึงค่าต่ำสุดของไนแอม Range constraint ค่าสูงสุดของ Range หมายถึงค่าสูงสุดของไนแอม Range constraint จากรูปที่ 4.7 นำเสนอไนแอมสกีมาประกอบด้วย 3 เอนทิตีไทป์คือ Country(Name) MedalKind(Code)และ Quantity(nr)+ โดยเอนทิตีไทป์ Quantity(nr)+ มีการระบุ Range constraint ที่มีค่าต่ำสุดเท่ากับ 1 และค่าสูงสุดเท่ากับ 50 จากรูปแบบที่กำหนดสามารถแปลงเป็นเอ็กซ์เอ็มแอลสกีมาดังรูปที่4.9

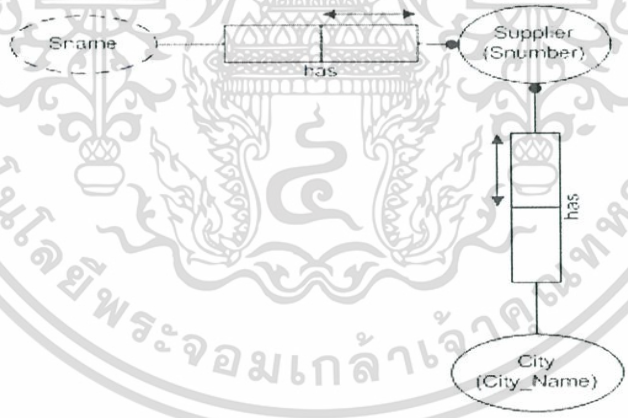
```
<simpleType name="Quantity" base="integer">
  <minInclusive value="1"/>
  <maxInclusive value="50"/>
</simpleType>
```

รูปที่4.9 ตัวอย่างเอ็กซ์เอ็มแอลสกีมาที่ได้จากการแปลงไนแอมสกีมารูปที่4.7

4.1.5 แปลงไนแอมเอนทิตีไทป์หลัก (Main Entity type) เป็น XML Schema Complex type

เมื่อเอนทิตีไทป์หลัก เป็นแอนทิตีไทป์ที่ต้องมีความสัมพันธ์กับลาเบลไทป์ และหรือ แอนทิตีไทป์ อย่างใดอย่างหนึ่งหรือทั้งสอง จากรูปที่ 4.10 ไนแอมสกีมาประกอบด้วย 2 แอนทิตีไทป์ คือ แอนทิตีไทป์ Supplier และ แอนทิตีไทป์ City จากข้อกำหนดที่กล่าวมาจะพบว่าแอนทิตีไทป์ Supplier เป็นแอนทิตีไทป์หลักตามข้อกำหนดเพราะเป็นแอนทิตีไทป์ที่มีความสัมพันธ์กับลาเบลไทป์ Sname และ แอนทิตีไทป์ City

การแปลงเอนทิตีไทป์หลักเป็น XML Schema Complex type ความสัมพันธ์แบบ Many-to-One และแบบ One-to-One ซึ่งเป็นความสัมพันธ์แบบไบนารี จะถูกแปลงเป็นสมาชิกย่อยของ Complex type เช่น



รูปที่4.10 ไนแอมสกีมานำเสนอ Main Entity type (Supplier(Snumber))

รูปแบบของเอ็กซ์เอ็มแอลสกีมาที่เกิดขึ้น จากการแปลงไนแอม Main Entity type รูปที่4.10 แสดงได้ดังรูปที่ 4.11

```
<complexType name= "ชื่อ Main Entity type">
        <element name= "Unique identifier ของเอนทิตีไทป์" type= "Unique identifier ของเอนทิตีไทป์"/>
        [<element name= "ชื่อลาเบลไทป์" type= "ชื่อลาเบลไทป์"/>]
        [<element name= "Unique identifier ของเอนทิตีไทป์" type= "Unique identifier ของเอนทิตีไทป์"/>]
</complexType>
```

รูปที่4.11 รูปแบบเอ็กซ์เอ็มแอลสกีมาจากการแปลง Main Entity type ของไนแอมสกีมา

เมื่อชื่อ Main Entity type หมายถึงชื่อแอนทิตีไทป์หลักของไนแอมสกีมา Unique identifier ของเอนทิตีไทป์ หมายถึง Unique identifier ของเอนทิตีไทป์หลักของไนแอมสกีมา ชื่อลาเบลไทป์ หมายถึงชื่อของลาเบลไทป์ของไนแอมสกีมา Unique identifier ของเอนทิตีไทป์ หมายถึง Unique identifier ของเอนทิตีไทป์ของไนแอมสกีมา

ซึ่ง ลาเบลไทป์ และ Unique identifier ของเอนทิตีไทป์ ที่มีความสัมพันธ์กับ แอนทิตีไทป์หลัก จากรูปแบบที่กำหนดจะพบว่ามีเครื่องหมาย " [ ] " ระบุอยู่เป็นการแสดงถึงทางเลือกของไนแอมสกีมาของแอนทิตีไทป์หลักที่เงื่อนไขกำหนดว่าจะต้องต้องมีความสัมพันธ์กับลาเบลไทป์ และหรือ แอนทิตีไทป์ อย่างใดอย่างหนึ่งหรือทั้งสอง

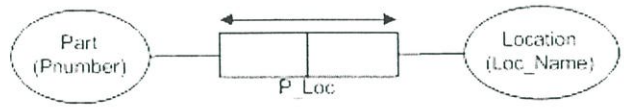
จากรูปที่4.10 นำเสนอไนแอมสกีมาประกอบด้วย 2 เอนทิตีไทป์คือ Supplier(Snumber), City(City_Name)และ 1 ลาเบลไทป์คือ Sname โดยเอนทิตีไทป์ Supplier(Snumber) เป็น Main Entity type จากรูปแบบที่กำหนดสามารถแปลงเป็นเอ็กซ์เอ็มแอลสกีมาดังรูปที่4.12

```
<complexType name= "SupplierType">
        <element name= "Snumber" type= "SnumberType"/>
        <element name= "Sname" type= "SnameType"/>
        <element name= "City_Name" type= "City_NameType"/>
</complexType>
```

รูปที่4.12 ตัวอย่างเอ็กซ์เอ็มแอลสกีมาที่ได้จากการแปลงไนแอมสกีมารูปที่4.10

4.1.6 แปลงไนแอมไบนารี Many-to-Many relationship เป็น XML Schema Complex type

การแปลงไนแอมสกีมาที่มีความสัมพันธ์แบบไบนารี Many-to-Many เป็นรีเลชันแนลสกีมาจะได้ตารางหนึ่งตาราง ซึ่งตรงกับคุณสมบัติ Complex type ของ XML Schema



รูปที่4.13 ไนแอมสกีมานำเสนอไบนารีแฟคไทป์ที่มีความสัมพันธ์แบบ many-to-many

รูปแบบของเอ็กซ์เอ็มแอลสกีมาที่เกิดขึ้น จากการแปลงไนแอมไบนารี Many-to-Many relationship รูปที่ 4.13 แสดงได้ดังรูปที่ 4.14

```
<complexType name= "ชื่อไบนารีแฟคไทป์">
        <Subelement 1>
        <Subelement 2>
</complexType>
```

รูปที่4.14 รูปแบบของเอ็กซ์เอ็มแอลสกีมาที่ได้จากการแปลงไนแอมสกีมารูปที่4.13

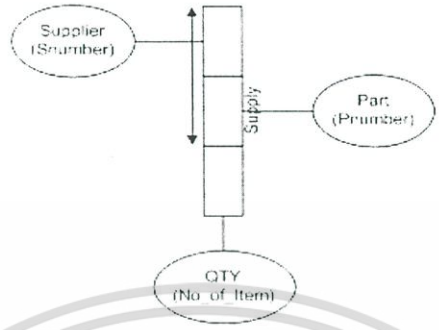
เมื่อชื่อไบนารีแฟคไทป์ หมายถึงชื่อไบนารีแฟคไทป์ของไนแอมสกีมาที่มีความสัมพันธ์แบบmany-to-many และ Subelement หมายถึง เอนทิตีไทป์ หรือลาเบลไทป์ที่สัมพันธ์กับไบนารีแฟคไทป์

```
<complexType name= "P_Loc">
    <element name= "Pnumber" type= "Pnumber"/>
    <element name= "Loc_Name" type= "Loc_Name"/>
</complexType>
```

รูปที่ 4.15 ตัวอย่างเอ็กซ์เอ็มแอลสกีมาที่ได้จากการแปลงไนแอมสกีมารูปที่4.13

4.1.7 แปลงไนแอม n-ary fact type เป็น XML Schema Complex type

การแปลงไนแอมที่มีความสัมพันธ์แบบ n-ary fact type เป็นรีเลชันแนลสกีมาจะได้ตารางหนึ่งตาราง ซึ่งตรงกับคุณสมบัติ Complex type ของ XML Schema



รูปที่4.16 ไนแอมสกีมานำเสนอ n-ary fact type

รูปแบบของเอ็กซ์เอ็มแอลสกีมาที่เกิดขึ้นจากการแปลงไนแอมสกีมารูปที่4.16 แสดงได้ดังรูปที่ 4.17

```
<complexType name= "ชื่อแฟคไทป์แบบ n-aryType">
        <Subelement 1>
        <Subelement 2>
              .
              .
              .
        <Subelement n>
</complexType>
```

รูปที่4.17 รูปแบบของเอ็กซ์เอ็มแอลสกีมาที่ได้จากการแปลงไนแอมสกีมารูปที่4.16

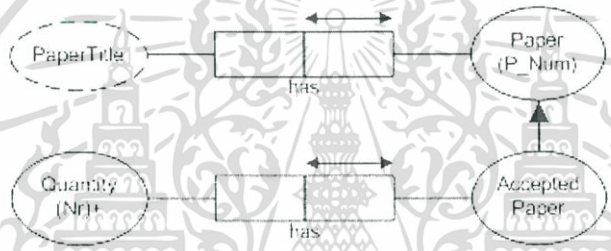
เมื่อ ชื่อแฟคไทป์แบบ n-ary หมายถึงชื่อแฟคไทป์ของไนแอมสกีมาที่มีความสัมพันธ์แบบ n-ary และ Sub element หมายถึง เอนทิตีไทป์ที่สัมพันธ์กับแฟคไทป์แบบ n-ary

```
<complexType name= “Supply”>
        <element name= “Snumber” type= “Snumber”/>
        <element name= “Pnumber” type= “Pnumber”/>
        <element name= “No_of_Item” type= “No_of_Item”/>
</complexType>
```

รูปที่4.18 ตัวอย่างเอ็กซ์เอ็มแอลสกีมาที่ได้จากการแปลงไนแอมสกีมารูปที่4.16

4.1.8 แปลงไนแอม Subtype เป็น XML Schema Complex type ที่สืบทอดมาจาก Supper type

ไนแอมสกีมา Subtype Constraint แสดงถึงคุณสมบัติการสืบทอดจากเอนทิตีไทป์หนึ่งไปยังเอนทิตีไทป์หนึ่ง ซึ่งตรงกับคุณสมบัติ Subtype by extension ของ XML Schema



รูปที่4.19 ไนแอมสกีมานำเสนอ Entity Subtype

รูปแบบของเอ็กซ์เอ็มแอลสกีมาที่เกิดขึ้นจากการแปลงไนแอมสกีมารูปที่4.19 แสดงได้ดังรูปที่ 4.20

```
<complexType name = "ชื่อของ Entity Subtype " base=”ชื่อของ Entity Suppertype” derived By=”extension”>
        <Subelement 1>
        <Subelement 2>
                .
                .
                .
        <Subelement n>
</complexType>
```

รูปที่4.20 รูปแบบของเอ็กซ์เอ็มแอลสกีมาที่ได้จากการแปลงไนแอมสกีมารูปที่4.19

เมื่อชื่อของ Entity Subtype หมายถึงชื่อของ Entity Subtype ของไนแอมสกีมา ชื่อของ Entity Suppertype หมายถึงชื่อของ Entity Suppertype ของไนแอมสกีมา และ Subelement หมายถึง เอนทิตีไทป์ หรือลาเบลไทป์ที่สัมพันธ์กับ Entity Subtype

```
<complexType name = "AcceptedPaper" base="Paper" derivedBy="extension">
        <element name="NrPages" type="NrPages"/>
</complexType>
```

รูปที่4.21 ตัวอย่างเอ็กซ์เอ็มแอลสกีมาที่ได้จากการแปลงไนแอมสกีมารูปที่4.19

4.1.9 The unique identifier ของแต่ละ Main entity type แปลงเป็น XML Schema key

จากรูปที่ 4.10 เมนเอนทีตีไทป์ Supplier มี (Snumber) เป็น unique identifier สามารถแปลงเป็น Key Constraint ของ XML Schema มีรูปแบบดังแสดงในรูปที่ 4.22

```
<key name ="keyname">
        <selector>pathname</selector>
        <field> unique identifier </field>
</key>
```

รูปที่4.22 รูปแบบการแปลงไนแอม unique identifier Main entity type เป็น XML Schema key

เมื่อ keyname หมายถึงชื่อของ XML Schema key pathname หมายถึงเส้นทางการระบุถึงตำแหน่งของ XML Schema key และ unique identifier หมายถึงตำแหน่งของ XML Schema key จากรูปที่ 4.10 สามารถแปลงไนแอม unique identifier เป็น XML Schema key ได้ดังรูปที่4.23

```
<key name ="SumberKey">
        <selector>Supplier_Part/Supplier</selector>
        <field> /Snumber </field>
</key>
```

รูปที่4.23 ตัวอย่างเอ็กซ์เอ็มแอลสกีมาที่ได้จากการแปลงไนแอม unique identifier Main entity type

4.1.10 The uniqueness constraint ของไบนารี one-to-one, many-to-many fact type และ n-ary fact type แปลงเป็น XML Schema unique constraint

การแปลงไนแอม uniqueness constraint ของไบนารี one-to-one, many-to-many fact type และ n-ary fact type แปลงเป็น XML Schema unique constraint แสดงได้ดังรูปที่ 4.24

```
<unique name =”uniquename”>
        <selector>pathname</selector>
        <field> unique identifier 1 </field>
        <field> unique identifier 2 </field>
                .
                .
                .
        <field> unique identifier n </field>
</unique>
```

รูปที่4.24 รูปแบบการแปลงไนแอมสกีมาเป็น XML Schema unique constraint

เมื่อ uniquename หมายถึงชื่อของ XML Schema unique constraint Pathname หมายถึงเส้นทางการระบุถึงตำแหน่งของ XML Schema unique constraint และ unique identifier หมายถึงตำแหน่งของ XML Schema unique constraint

กรณี uniqueness constraint ของไบนารี one-to-one fact type จะระบุ unique identifier เพียงค่าเดียว

กรณี uniqueness constraint ของไบนารี many-to-many fact type จะระบุ unique identifier จำนวน2 ค่า

กรณี uniqueness constraint ของ n-ary fact type จะระบุ unique identifier จำนวน n ค่า

จากรูปที่ 4.16 สามารถแปลงไนแอม uniqueness constraint กรณีของ n-ary fact type เป็น XML Schema unique constraint ได้ดังรูปที่4.25

```
<unique name =" SupplyKey">
        <selector>Supplier_Part/Supplier</selector>
        <field> /Snumber </field>
        <selector>Supplier_Part/Part</selector>
        <field> /Pnumber </field>
</unique>
```

รูปที่4.25 ตัวอย่างเอ็กซ์เอ็มแอลสกีมาที่ได้จากการแปลงไนแอม uniqueness constraint จากรูปที่ 4.16 (n-ary relationship)

4.1.11 แปลง many-to-many, n-ary fact type ระหว่าง Main entity type เป็น XML Schema key reference

การแปลงไนแอม Fact type ระหว่าง Main entity type เป็น XML Schema key reference แสดงได้ดังรูปที่ 4.26 เนื่องจาก Fact type ระหว่าง Main entity type เปรียบเสมือนการอ้างอิงระหว่างกลุ่ม Complex type ของ XML Schema

```
<keyref  name ="keyrefname" refer="keyname ">
        <selector>pathname</selector>
        <field> keyref element </field>
</keyref>
```

รูปที่4.26 รูปแบบการแปลง Fact type ระหว่าง Main entity type เป็น XML Schema key reference

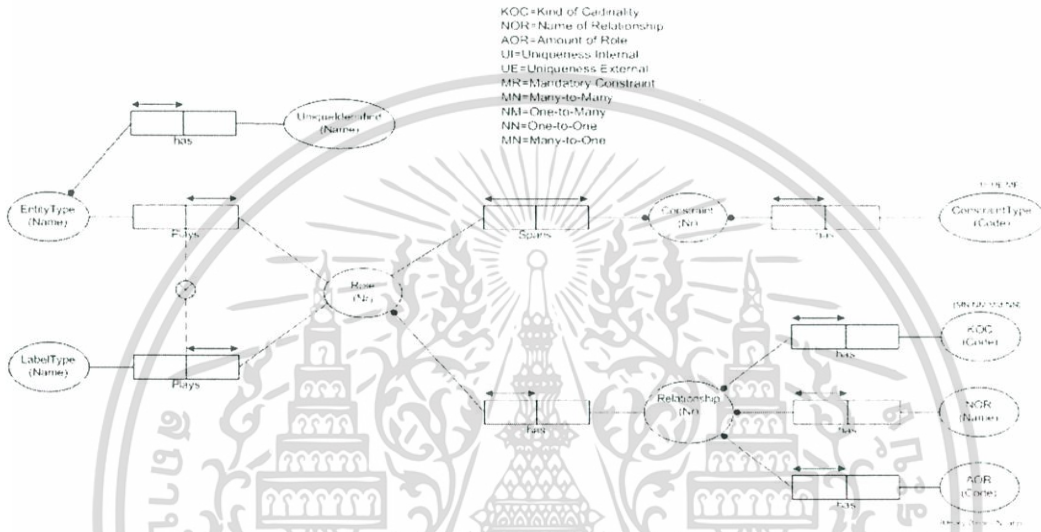
เมื่อ keyrefname หมายถึงชื่อของ keyref name Keyname หมายถึง XML Schema key ของ Main entity type ที่อ้างถึง Pathname หมายถึงเส้นทางที่ระบุถึงตำแหน่งของ keyref name และ Keyref element หมายถึงตำแหน่งของอิลิเมนต์ย่อยที่เป็น XML Schema key reference

```
<keyref  name ="P_Loc_PartRef" refer="PnumberKey ">
        <selector> Supplier_Part/P_Loc</selector>
        <field> /Pnumber </field>
</keyref>
```

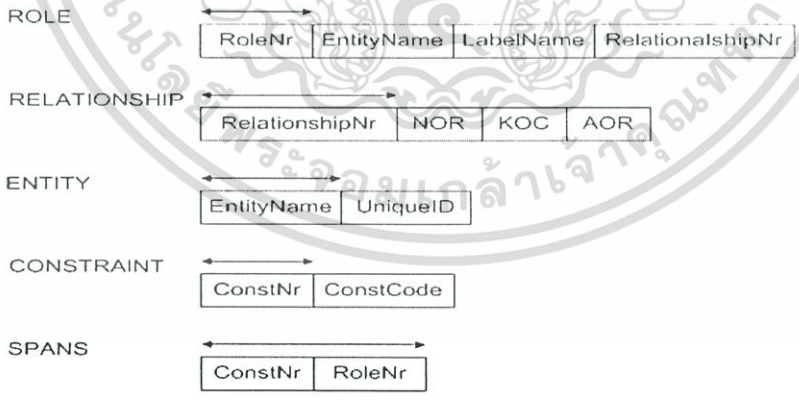
รูปที่4.27 ตัวอย่างการแปลง Fact type ระหว่าง Main entity type เป็น XML Schema key reference

4.2 การสร้างไนแอมเมต้าสกีมา

การสร้างไนแอมเมต้าสกีมา เมื่อผู้ใช้สร้างไนแอมสกีมาจากหน้าจอไนแอมเอดิเตอร์ของซอฟต์แวร์ทูล อิลิเมนต์ทุกตัวเช่น ลาเบลไทป์ เอนทีตีไทป์ แฟคไทป์ และคอนสเตรนทุกตัวจะถูกเก็บอยู่ในตารางเมต้าดาต้า การสร้างไนแอมเมต้าสกีมาทำขึ้นเพื่อออกแบบตารางเมต้าดาต้า ดังนั้นไนแอมเมต้าสกีมาก็คือไนแอมสกีมาที่ใช้สำหรับอธิบายส่วนประกอบของไนแอมสกีมาที่ผู้ใช้งานซอฟต์แวร์ทูลสร้างขึ้น รูปที่4.28 แสดงไนแอมเมต้าสกีมา และรูปที่4.29 แสดงตารางเมต้าดาต้า ที่อยู่ในนอร์มัลฟอร์มที่ 5 (5NF.)ซึ่งได้จากการแปลงเมต้าสกีมารูปที่4.28



รูปที่4.28 ไนแอมเมต้าสกีมา

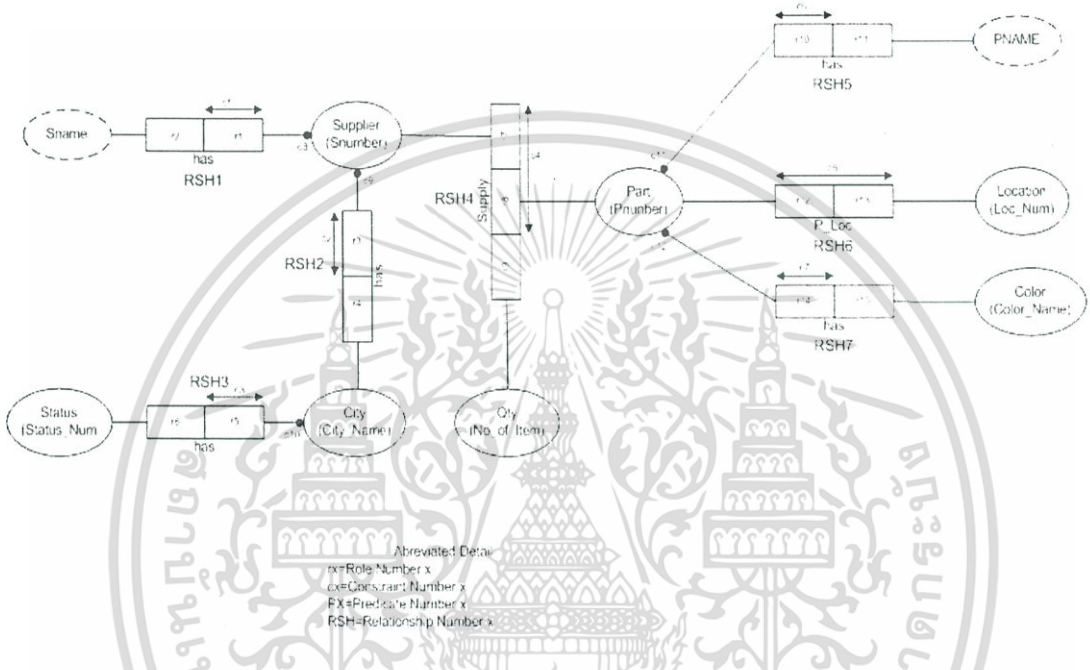


รูปที่4.29 แสดงเมต้าสีกมาของฐานข้อมูลเชิงสัมพันธ์

4.3 ตัวอย่างการแปลงไนแอมสกีมาเป็นเอ็กซ์เอ็มแอลสกีมา

ตัวอย่างการแปลงไนแอมสกีมาเป็นเอ็กซ์เอ็มแอลสกีมาสำหรับงานวิจัยนี้นำเสนอการออกแบบไนแอมสกีมาของ Supplier-Part Database การแปลงไนแอมสกีมาเป็นเอ็กซ์เอ็มแอลสกีมา และการนำเสนอตารางเมต้าดาต้า รวมทั้งตารางเมต้าดาต้าที่ป้อนข้อมูลแล้ว (Populate Meta Table)

4.3.1 การออกแบบไนแอมสกีมาของ Supplier-Part Database แสดงดังรูปที่4.30



รูปที่4.30 ไนแอมสกีมา Supplier-Part Database

4.3.2 การแปลงไนแอมสกีมาเป็นเอ็กซ์เอ็มแอลสกีมา

การแปลงไนแอมสกีมาจากรูปที่ 4.30 เป็นเอ็กซ์เอ็มแอลสกีมา แสดงรายละเอียดดังนี้

4.3.2.1. กำหนดอิลิเมนต์เริ่มต้นของเอ็กซ์เอ็มแอลสกีมา แสดงได้ดังรูปที่ 4.31

```
<?xml version="1.0" encoding="UTF-8"?>
<sp:schema xmlns:sp="http://www.w3.org/2001/XMLSchema">
<!--=======RootElement Definition ======-->
<sp:element name="Supply_Part">  <----- Root Element
    <sp:complexType>
    <sp:sequence>
        <sp:element name="Supplier" type="SupplierType" maxOccurs="unbounded"/>
        <sp:element name="Part" type="PartType" maxOccurs="unbounded"/>
        <sp:element name="City" type="CityType" maxOccurs="unbounded"/>
        <sp:element name="Supply" type="SupplyType" maxOccurs="unbounded"/>
        <sp:element name="P_Loc" type="P_LocType" maxOccurs="unbounded"/>
    </sp:sequence>
    </sp:complexType>
```

รูปที่4.31 อิลิเมนต์เริ่มต้นของเอ็กซ์เอ็มแอลสกีมา

4.3.2.2 แปลงแต่ละลาเบลไทป์เป็น XML Schema Simple type แสดงได้ดังรูปที่ 4.32

```
<!-- =====SimpleType Definition for NIAM Value type======-->
    <sp:simpleType name="SnumberType">
        <sp:restriction base="sp:string">
        <sp:length value="5"/>
        <sp:pattern value="[A-Z]{1}-\d{3}"/>
        </sp:restriction>
    </sp:simpleType>
    <sp:simpleType name="SnameType">
        <sp:restriction base="sp:string">
        <sp:pattern value="[A-Z][a-z]*"/>
        </sp:restriction>
    </sp:simpleType>
    <sp:simpleType name="City_NameType">
        <sp:restriction base="sp:string">
        <sp:pattern value="[A-Z][a-z]*"/>
        </sp:restriction>
    </sp:simpleType>
    <sp:simpleType name="PnumberType">
        <sp:restriction base="sp:string">
        <sp:pattern value="[A-Z]*-\d*"/>
        </sp:restriction>
    </sp:simpleType>
    <sp:simpleType name="PnameType">
        <sp:restriction base="sp:string">
        <sp:pattern value="[A-Z][a-z]*"/>
        </sp:restriction>
    </sp:simpleType>
    <sp:simpleType name="ColorNameType">
        <sp:restriction base="sp:string"/>
    </sp:simpleType>
    <sp:simpleType name="Status_NumType">
        <sp:restriction base="sp:string"/>
    </sp:simpleType>
    <sp:simpleType name="No_of_ItemType">
        <sp:restriction base="sp:integer"/>
    </sp:simpleType>
    <sp:simpleType name="Loc_NumType">
        <sp:restriction base="sp:string"/>
    </sp:simpleType>
```

รูปที่4.32 ซิมเปิลไทป์

4.3.2.3 แปลงไนแอม Entity type และ Main entity type เป็น XML Schema Complex type

```
<!--======ComplexType Definition for NIAM Entity type ======-->
    <sp:complexType name="SupplierType">
    <sp:sequence>
    <sp:element name="Snumber" type="SnumberType"/>
    <sp:element name="Sname" type="SnameType"/>
    <sp:element name="City_Name" type="City_NameType"/>
    </sp:sequence>
    </sp:complexType>
    <sp:complexType name="PartType">
    <sp:sequence>
    <sp:element name="Pnumber" type="PnumberType"/>
    <sp:element name="Pname" type="PnameType"/>
    <sp:element name="ColorName" type="ColorNameType"/>
    </sp:sequence>
    </sp:complexType>
    <sp:complexType name="CityType">
    <sp:sequence>
    <sp:element name="City_Name" type="City_NameType"/>
    <sp:element name="Status_Num" type="Status_NumType"/>
    </sp:sequence>
    </sp:complexType>
```

รูปที่4.33 คอมเพลกซ์ไทป์สำหรับไนแอมแอนติตี้ไทป์

4.3.2.4 แปลงไนแอมไบนารี Many-to-Many relationship เป็นXML Schema Complex type

```
<!--======ComplexType Definition for NIAM Binary Relationship ========-->
    <sp:complexType name="P_LocType">
    <sp:sequence>
    <sp:element name="Pnumber" type="PnumberType"/>
    <sp:element name="Loc_Num" type="Loc_NumType"/>
    </sp:sequence>
    </sp:complexType>
```

รูปที่4.34 คอมเพลกซ์ไทป์สำหรับไนแอมไบนารีรีเลชันชิป

4.3.2.5 แปลงไนแอม ternary fact type เป็น XML Schema Complex type

```
<!--======ComplexType Definition for NIAM Ternary Relationship form Many-to-
Many ========-->
    <sp:complexType name="SupplyType">
    <sp:sequence>
    <sp:element name="Snumber" type="SnumberType"/>
    <sp:element name="Pnumber" type="PnumberType"/>
    <sp:element name="No_of_Item" type="No_of_ItemType"/>
    </sp:sequence>
    </sp:complexType>
```

รูปที่4.35 คอมเพลกซ์ไทป์สำหรับไนแอมเทอร์นารีรีเลชันชิป

4.3.2.6 The unique identifier ของแต่ละ Main entity type แปลงเป็น XML Schema key

```
<!--====Keys  Constrainst =====-->
    <sp:key name="SnumberKey">
        <sp:selector xpath="Supplier"/>
        <sp:field xpath="./Snumber"/>
    </sp:key>
    <sp:key name="PnumberKey">
        <sp:selector xpath="Part"/>
        <sp:field xpath="./Pnumber"/>
    </sp:key>
    <sp:key name="City_NameKey">
        <sp:selector xpath="City"/>
        <sp:field xpath="./City_Name"/>
    </sp:key>
```

รูปที่4.36 คีย์คอนสเตรนสำหรับไนแอมยูนิคไอเด็นทิไฟน์

4.3.2.7 The uniqueness constraint ของไบนารี one-to-one, many-to-many fact type และ n-ary fact type แปลงเป็น XML Schema unique constraint

```
<!--====Unique  Constrainst =====-->
<sp:unique name="SupplyKey">
        <sp:selector xpath="Supply"/>
        <sp:field xpath="./Snumber"/>
        <sp:field xpath="./Pnumber"/>
    </sp:unique>
    <sp:unique name="P_LocKey">
        <sp:selector xpath="P_Loc"/>
        <sp:field xpath="./Pnumber"/>
        <sp:field xpath="./Loc_Num"/>
    </sp:unique>
```

รูปที่4.37 ยูนิคคอนสเตรนสำหรับไนแอมยูนิคเนสคอนสเตรน

4.3.2.8 แปลง many-to-many, n-ary fact type ระหว่าง Main entity type เป็น XML Schema key reference

```
<!--==========Keys Reference ==========-->
    <sp:keyref name="P_Loc_PartRef" refer="PnumberKey">
        <sp:selector xpath="P_Loc"/>
        <sp:field xpath="./Pnumber"/>
    </sp:keyref>
    <sp:keyref name="Supply_SupplierRef1" refer="SnumberKey">
        <sp:selector xpath="Supply"/>
        <sp:field xpath="./Snumber"/>
    </sp:keyref>
    <sp:keyref name="Supply_SupplierRef2" refer="PnumberKey">
        <sp:selector xpath="Supply"/>
        <sp:field xpath="./Pnumber"/>
    </sp:keyref>
    <sp:keyref name="Supplier_CityRef" refer="City_NameKey">
        <sp:selector xpath="Supplier"/>
        <sp:field xpath="./City_Name"/>
    </sp:keyref>
</sp:element>
</sp:schema>
```

รูปที่4.38 คีย์รีเฟอร์เรนซ์สำหรับไนแอมไบนารีและเอ็นนารีรีเลชันชิป

4.3.3 การนำเสนอรายละเอียดของตารางเมต้าดาต้า

ตารางเมต้าดาต้าที่กล่าวถึงในหัวข้อนี้ เป็นรีเลชันแนลสกีมา หรือสกีมาของฐานข้อมูลเชิงสัมพันธ์ที่ได้มาจากการแปลงไนแอมเมต้าสกีมา เนื่องจากการจัดเก็บรายละเอียดของข้อมูลไนแอมสกีมาที่ถูกสร้างขึ้นจะจัดเก็บลงฐานข้อมูลเชิงสัมพันธ์ รายละเอียดการนำเสนอในส่วนของตารางเมต้าดาต้าสำหรับตัวอย่างนี้จะนำเสนอ 2 ส่วนคือการนำเสนอรีเลชันแนลสกีมาของตารางเมต้าดาต้าที่ได้จากการแปลงไนแอมเมต้าสกีมา และ การนำเสนอตารางเมต้าดาต้าที่ถูกป้อนข้อมูลแล้ว

จากตัวอย่างการสร้างไนแอมสกีมาที่กล่าวมาในหัวข้อ 4.3.1 รูปที่ 4.30 แสดงรายละเอียดได้ดังนี้

4.3.3.1 การนำเสนอรีเลชันแนลสกีมาของตารางเมต้าดาต้าที่ได้จากการแปลงไนแอมเมต้าสกีมา

แสดงได้ดังรูปที่ 4.29

4.3.3.2 การนำเสนอตารางเมต้าดาต้า

ตารางเมต้าดาต้าถูกสร้างขึ้นเพื่อจัดเก็บข้อมูลรายละเอียดของไนแอมสกีมาในขณะที่ผู้ใช้โปรแกรมทำการสร้างไนแอมสกีมา โปรแกรมจะทำการจัดเก็บข้อมูลลงตารางเมต้าดาต้ารายละเอียดของตารางเมต้าดาต้าทั้งหมดแสดงได้ดังรูปที่ 4.39 ,4.40 ,4.41 , 4.42 และ รูปที่ 4.43

EntityName	UniqueID
City	City_Name
Color	Color_Name
Location	Loc_Num
Part	Pnumber
Qty	No_of_Item
Status	Status_Num
Supplier	Snumber

รูปที่4.39 ตารางเมต้าดาต้าของ ENTITY

ConstNr	ConstCode
c1	UI
c10	MR
c11	MR
c12	MR
c2	UI
c3	UI
c4	UI
c5	UI
c6	UI
c7	UI
c8	MR
c9	MR

รูปที่4.40 ตารางเมต้าดาต้าของ CONSTRAINT

RelationshipNr	NOR	KOC	AOR
RSH1	has	MN	Binary
RSH2	has	MN	Binary
RSH3	has	MN	Binary
RSH4	Supply	MM	N_ary
RSH5	has	MN	Binary
RSH6	P_Loc	MM	Binary
RSH7	has	MN	Binary

รูปที่4.41 ตารางเมต้าดาต้าของ RELATIONSHIP

RoleNr	EntityName	LabelName	RelationshipN
r1	Supplier		RSH1
r10	Part		RSH5
r11		Pname	RSH5
r12	Part		RSH6
r13	Location		RSH6
r14	Part		RSH7
r15	Color		RSH7
r2		Sname	RSH1
r3	Supplier		RSH2
r4	City		RSH2
r5	City		RSH3
r6	Status		RSH3
r7	Supplier		RSH4
r8	Part		RSH4
r9	Qty		RSH4

รูปที่4.42 ตารางเมต้าดาต้าของ ROLE

ConstNr	RoleNr
c1	r1
c10	r5
c11	r10
c12	r14
c2	r3
c3	r5
c4	r7
c4	r8
c5	r10
c6	r12
c6	r13
c7	r14
c8	r1
c9	r3

รูปที่4.43 ตารางเมต้าดาต้าของ SPANS

บทที่ 5
การพัฒนาซอฟต์แวร์ทูล

การพัฒนาไนแอมซอฟแวร์ทูลเป็นการพัฒนาเครื่องมือขึ้นมาจากแนวคิดของงานวิจัยที่ได้นำเสนอในบทที่ 4 (กระบวนการแปลงไนแอมสกีมาเป็นเอ็กซ์เอ็มแอลสกีมา) สำหรับรายละเอียดที่นำเสนอในบทนี้ประกอบด้วย

การใช้ UML (The Unified Modeling Language) [19,20] ในการนำเสนอรายละเอียดของการพัฒนา และการนำเสนอการใช้งานซอฟต์แวร์ทูลที่ได้จากการพัฒนา เพื่อทำการแปลงไนแอมสกีมาเป็นเอ็กซ์เอ็มแอลสกีมา

5.1 การใช้ UML ในการนำเสนอรายละเอียดการพัฒนาซอฟต์แวร์ทูล

การใช้UML สำหรับการนำเสนอรายละเอียดการพัฒนาซอฟต์แวร์ทูลของงานวิจัยนี้เพราะ UML เป็นภาษาที่ใช้อธิบายโมเดลต่าง ๆ เป็นภาษาที่ใช้กราฟฟิคเป็นสัญลักษณ์ และเป็นภาษามาตรฐานสำหรับสร้างแบบพิมพ์เขียว (blueprint) ให้แก่ระบบงาน และกระบวนการพัฒนาซอฟต์แวร์ สามารถใช้ในการสร้างมุมมอง กำหนดรายละเอียด สร้างระบบงาน และจัดทำเอกสารอ้างอิงให้แก่ระบบงานได้

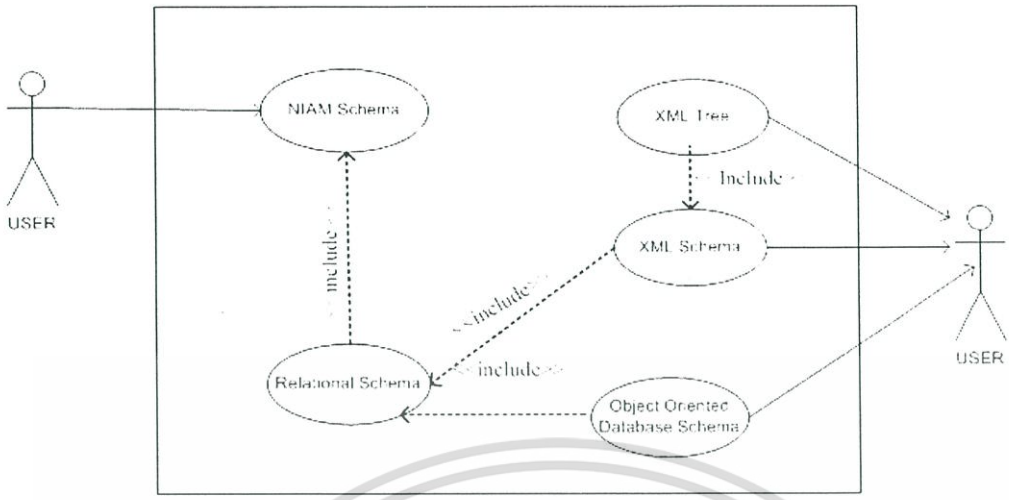
สำหรับงานวิจัยนี้ใช้ UML อธิบายระบบงานซึ่งประกอบด้วย ไดอะแกรมในส่วนของพฤติกรรม (Behavioral Diagram) อันประกอบด้วย ยูสเคสไดอะแกรม (Use Case Diagram) ซีแควนซ์ไดอะแกรม (Sequence Diagram) และ ไดอะแกรมในส่วนของโครงสร้าง (Structural Diagrams) อันประกอบด้วย คอมโพเน็นไดอะแกรม (Component Diagram)

5.1.1 ยูเอ็มแอลในส่วนของพฤติกรรม ประกอบด้วย

5.1.1.1 ยูสเคสไดอะแกรม

ยูสเคสไดอะแกรม ประกอบด้วยองค์ประกอบ 2 ส่วน คือ เอ็กเตอร์(actor) และ ยูสเคส เมื่อ ยูสเคส จะแสดงถึงขอบเขตที่ระบบสนใจ และเอ็กเตอร์ คือสิ่งที่อยู่นอกระบบ เป็นผู้ให้ข้อมูลกับระบบ และเป็นผู้รับผลลัพธ์จากระบบ

ยูสเคสไออะแกรมของระบบ NIAM SOFTWARE TOOL ประกอบด้วย 1 เอ็กเตอร์คือ เอ็กเตอร์ผู้เข้าใช้ระบบงาน และประกอบด้วยยูสเคส 5 ยูสเคส ได้แก่ ยูสเคส NIAM Schema ยูสเคส XML Tree ยูสเคส Relational Schema ยูสเคส XML Schema และ ยูสเคส Object Oriented Schema รายละเอียดแสดงในรูปที่ 5.1



รูปที่ 5.1 Use Case Diagram ของ NIAM SOFTWARE TOOL

จากรูปที่ 5.1 อธิบายการทำงานของระบบได้ว่าเมื่อเอ็กเตอร์ผู้ใช้เข้าใช้งานระบบ เอ็กเตอร์ผู้ใช้จะเข้าสู่ยูสเคส NIAM Schema เพื่อทำการสร้างไนแอมสกีมา

ยูสเคส Relational Schema เป็นยูสเคสที่ทำการแปลงข้อมูลจากไนแอมสกีมามาจัดเก็บในฐานข้อมูลที่ออกแบบไว้ซึ่งการทำงานของยูสเคส Relational Schema จะต้องInclude ยูสเคส NIAM Schema

ยูสเคส XML Schema เป็นยูสเคสที่ทำการแปลงข้อมูลจากไนแอมสกีมาที่ถูกจัดเก็บไว้ในฐานข้อมูล (Meta Data) เป็นเอ็กซ์เอ็มแอลสกีมา ซึ่งการทำงานของยูสเคสนี้จะต้อง Include ยูสเคส Relational Schema

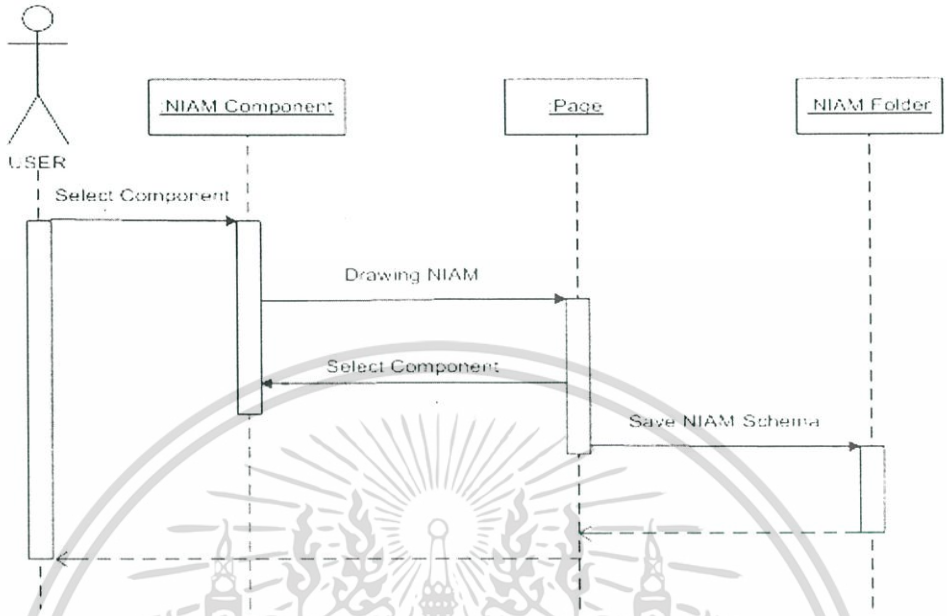
ยูสเคส Object Oriented Database Schema เป็นยูสเคสที่ทำการแปลงข้อมูลจากไนแอมสกีมาที่ถูกจัดเก็บไว้ในฐานข้อมูล (Meta Data) เป็น Object Oriented Database Schema ซึ่งการทำงานของยูสเคสนี้จะต้อง Include ยูสเคส Relational Schema

ยูสเคส XML Tree เป็นยูสเคสที่ทำการแปลงเอ็กซ์เอ็มแอลสกีมาเป็นเอ็กซ์เอ็มแอลทรี ซึ่งการทำงานของยูสเคสนี้จะต้อง Include ยูสเคส XML Schema

5.1.1.2 ซีแควนซ์ไดอะแกรม

ซีแควนซ์ไดอะแกรมแสดงการทำงานระหว่างออบเจ็กต์ต่าง ๆ เมื่อเกิดการส่งข่าวสาร (Message) ระหว่างออบเจ็กต์ในแต่ละเหตุการณ์ โดยมีทิศทางของลูกศรเป็นตัวบ่งบอกถึงทิศ

ทางการส่งข่าวสารระหว่างออบเจ็กต์ การส่งข่าวสารระหว่างออบเจ็กต์ในซีเควนซ์ไดอะแกรมจะมีลำดับการส่งข่าวสารจากซ้ายไปปขวา



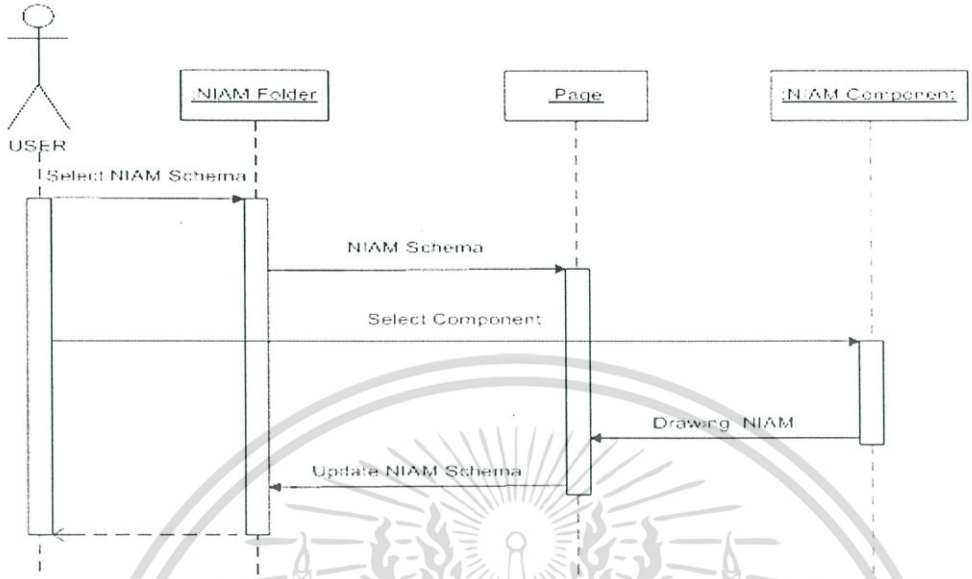
รูปที่ 5.2 Sequence Diagram ในการสร้าง NIAM Schema

รูปที่ 5.2 แสดงซีเควนซ์ไดอะแกรมในการสร้างไนแอมสกีมาของระบบงาน ออบเจ็กต์ของซีแควนซ์ไดอะแกรมประกอบด้วย NIAM Component ,Page , และ NIAM Folder

เมื่อNIAM Component เป็นออบเจ็กต์ที่ให้ผู้ใช้เข้าทำการเลือกเครื่องมือหรือองค์ประกอบของไนแอมที่จะนำมาสร้างไนแอมสกีมา Page เป็นออบเจ็กต์ที่ให้ผู้ใช้ทำการสร้างไนแอมสกีมาที่ได้ทำการเลือกมาจาก NIAM Component และ NIAM Folder เป็นออบเจ็กต์ที่ให้ผู้ใช้ทำการจัดเก็บไนแอมสกีมาที่ได้ทำการสร้างในออบเจ็กต์ Page

การส่งข่าวสารระหว่างออบเจ็กต์จากรูปที่ 5.2 อธิบายได้ว่าเมื่อเอ็กเตอร์ผู้ใช้ต้องการจะสร้างไนแอมสกีมาเอ็กเตอร์ผู้ใช้จะเข้าไปทำการเลือกเครื่องมือหรือองค์ประกอบจากออบเจ็กต์ NIAM Component จากนั้นจะทำการสร้างองค์ประกอบที่เลือกนั้นที่ออบเจ็กต์ Page เมื่อสร้างองค์ประกอบที่ทำการเลือกนั้นแล้วถ้าเอ็กเตอร์ผู้ใช้ยังต้องการจะสร้างองค์ประกอบในออบเจ็กต์ Page เพิ่มเติมก็จะกลับไปยังออบเจ็กต์ NIAM Component ซึ่งการทำงานระหว่างออบเจ็กต์ Page และออบเจ็กต์ NIAM Component จะทำงานสลับกันไปมาจนกระทั่งเอ็กเตอร์ผู้ใช้ทำการสร้างองค์ประกอบนั้นครบจึงจะส่งข้อความการจัดเก็บต่อไปยังออบเจ็กต์ NIAM Folder เพื่อที่จะทำการจัดเก็บไนแอมสกีมาที่ได้ทำการสร้างในออบเจ็กต์ Page

การทำงานของซีเควนซ์ไดอะแกรมของรูปที่ 5.2 จะสิ้นสุดเมื่อเมื่อมีการส่งข้อความย้อนกลับซึ่งจะใช้สัญลักษณ์ลูกศรเส้นประจากออบเจ็กต์ NIAM Folder กลับมายังเอ็กเตอร์ผู้ใช้

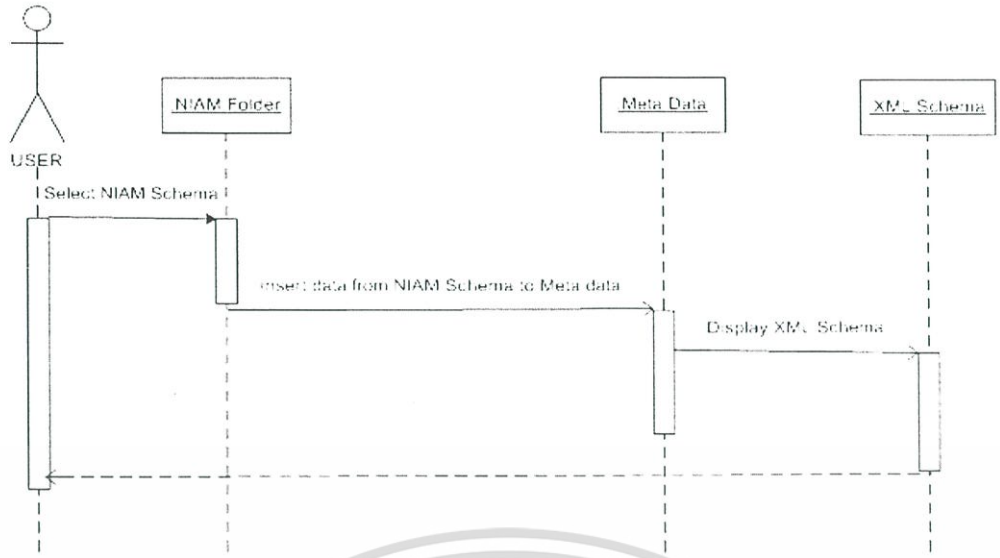


รูปที่ 5.3 Sequence Diagram ในการแก้ไข NIAM Schema

รูปที่5.3 แสดงซีเควนซ์ไดอะแกรมในการแก้ไขไนแอมสกีมาของระบบงาน ออบเจ็กต์ของซีเควนซ์ไดอะแกรมประกอบด้วย NIAM Component ,Page , และ NIAM Folder ออบเจ็กต์แต่ละตัวมีความหมายเช่นเดียวกับคำอธิบายรูปที่ 5.2

การส่งข่าวสารระหว่างออบเจ็กต์จากรูปที่ 5.3 อธิบายได้ว่าเมื่อเอ็กเตอร์ผู้ใช้ต้องการจะแก้ไขไนแอมสกีมาเอ็กเตอร์ผู้ใช้จะเข้าไปทำการเลือกไนแอมสกีมาจากออบเจ็กต์ NIAM Folder ไนแอมสกีมาที่ถูกเลือกจะมาแสดงในออบเจ็กต์เพจ(Object Page) จากนั้นเอ็กเตอร์ผู้ใช้จะทำการเลือกองค์ประกอบจากออบเจ็กต์ NIAM Component เพื่อทำการแก้ไขไนแอมสกีมาที่แสดงอยู่บนออบเจ็กต์ Page เมื่อแก้ไขไนแอมสกีมาจากองค์ประกอบที่ทำการเลือกนั้นแล้วถ้าเอ็กเตอร์ผู้ใช้ยังต้องการจะแก้ไขไนแอมสกีมาในออบเจ็กต์เพจ เพิ่มเติมก็จะกลับไปยังออบเจ็กต์ NIAM Component ซึ่งการทำงานระหว่างออบเจ็กต์เพจ และออบเจ็กต์ NIAM Component จะทำงานสลับกันไปมาจนกระทั่งเอ็กเตอร์ผู้ใช้ทำการแก้ไขไนแอมสกีมาเสร็จสิ้นจึงจะส่งข้อความการแก้ไขไปยังออบเจ็กต์ NIAM Folder เพื่อที่จะทำการจัดเก็บไนแอมสกีมาที่ได้ทำการแก้ไขในออบเจ็กต์เพจ

การทำงานของซีเควนซ์ไดอะแกรมของรูปที่ 5.3 จะสิ้นสุดเมื่อเมื่อมีการส่งข้อความย้อนกลับซึ่งจะใช้สัญลักษณ์ลูกศรเส้นประจากออบเจ็กต์ Page กลับมายังเอ็กเตอร์ผู้ใช้



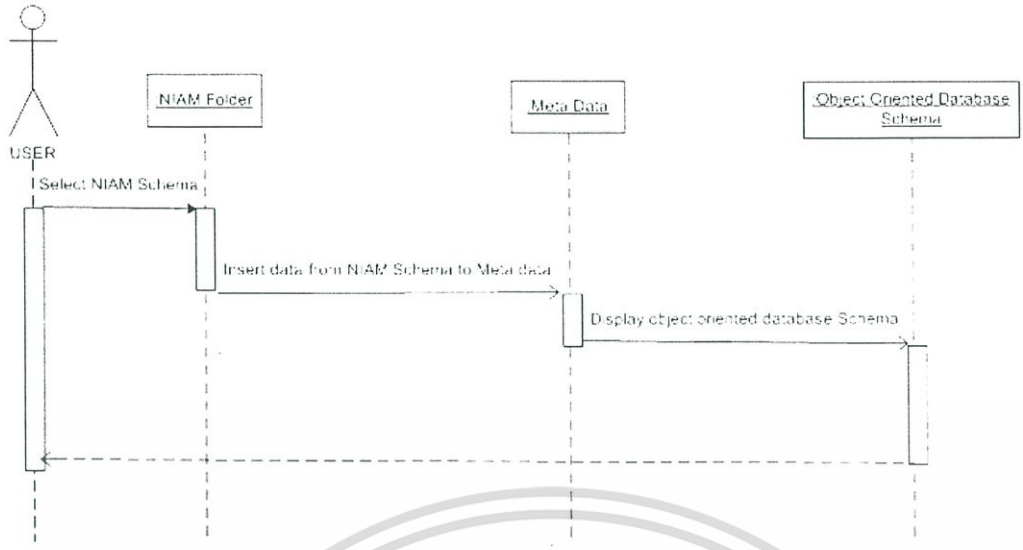
รูปที่ 5.4 Sequence Diagram ในการแปลง NIAM Schema เป็น XML Schema

รูปที่ 5.4 แสดงซีเควนซ์ไดอะแกรมในการแปลงไนแอมสกีมาเป็นเอ็กซ์เอ็มแอลสกีมาของระบบงาน ออบเจ็กต์ของซีเควนซ์ไดอะแกรมประกอบด้วย NIAM Folder , Meta Data และ XML Schema

เมื่อNIAM Folder เป็นออบเจ็กต์ที่ทำการเก็บไฟล์ไนแอมสกีมาที่ถูกสร้างขึ้น Meta Data คือออบเจ็กต์ที่ทำหน้าที่เก็บข้อมูลที่ได้จากไนแอมสกีมา และ ออบเจ็ก XML Schema คือออบเจ็กต์การแปลงไนแอมสกีมาอยู่ในรูปของของเอ็กซ์เอ็มแอลสกีมาที่ถูกต้องตามกฎของ W3C

การส่งข่าวสารระหว่างออบเจ็กต์จากรูปที่ 5.4 อธิบายได้ว่าเมื่อเอ็กเตอร์ผู้ใช้ต้องการแปลงไนแอมสีมาที่ถูกสร้างขึ้นและเก็บไว้ในออบเจ็กต์ NIAM Folder เป็นเอ็กซ์เอ็มแอลสกีมาที่ถูกต้องตามกฎของ W3C เอ็กเตอร์ผู้ใช้จะทำการเลือกไนแอมสกีมจากออบเจ็กต์ NIAM Folder จากนั้นไนแอมสกีมาที่เลือกจะถูกแปลงเป็นข้อมูลที่เพื่อทำการจัดเก็บในฐานข้อมูลที่สร้างไว้ที่ออบเจ็ก Meta Data จากออบเจ็กต์ Meta Data ข้อมูลที่ถูกจัดเก็บในฐานข้อมูลจะถูกแปลงไปเป็นเอ็กซ์เอ็มแอลสกีมาที่ถูกต้องตามกฎ W3C

การทำงานของซีเควนซ์ไดอะแกรมของรูปที่ 5.4 จะสิ้นสุดเมื่อเมื่อมีการส่งข้อความย้อนกลับซึ่งจะใช้สัญลักษณ์ลูกศรเส้นประจากออบเจ็กต์ XML Schema กลับมายังเอ็กเตอร์ผู้ใช้



รูปที่ 5.5 Sequence Diagram ในการแปลง NIAM Schema เป็น Object Oriented Database Schema

รูปที่ 5.5 แสดงซีแควนซ์ไดอะแกรมในการแปลงไนแอมสกีมาเป็นออบเจ็กต์ออเร็นเต็ดดาต้าเบสสกีมาของระบบงาน ออบเจ็กต์ของซีแควนซ์ไดอะแกรมประกอบด้วย NIAM Folder , Meta Data และ Object Oriented Database Schema

เมื่อNIAM Folder เป็นออบเจ็กต์ที่ทำการเก็บไฟล์ไนแอมสกีมาที่ถูกสร้างขึ้น Meta Data คือออบเจ็กต์ที่ทำหน้าที่เก็บข้อมูลที่ได้จากไนแอมสกีมา และ ออบเจ็ก XML Schema คือ ออบเจ็กต์การแปลงไนแอมสกีมาอยู่ในรูปของของเอ็กซ์เอ็มแอลสกีมาที่ถูกต้องตามกฎของ W3C

การส่งข่าวสารระหว่างออบเจ็กต์จากรูปที่ 5.5 อธิบายได้ว่าเมื่อเอ็กเตอร์ผู้ใช้ต้องการแปลงไนแอมสีมาที่ถูกสร้างขึ้นและเก็บไว้ในออบเจ็กต์ NIAM Folder เป็นเป็นออบเจ็กต์ออเร็นเต็ดดาต้าเบสสกี เอ็กเตอร์ผู้ใช้จะทำการเลือกไนแอมสกีมจากออบเจ็กต์ NIAM Folder จากนั้นไนแอมสกีมาที่เลือกจะถูกแปลงเป็นข้อมูลเพื่อทำการจัดเก็บในฐานข้อมูลที่สร้างไว้ที่ออบเจ็ก Meta Data จากออบเจ็กต์ Meta Data ข้อมูลที่ถูกจัดเก็บในฐานข้อมูลจะถูกแปลงไปเป็นเป็นออบเจ็กต์ออเร็นเต็ดดาต้าเบสสกี

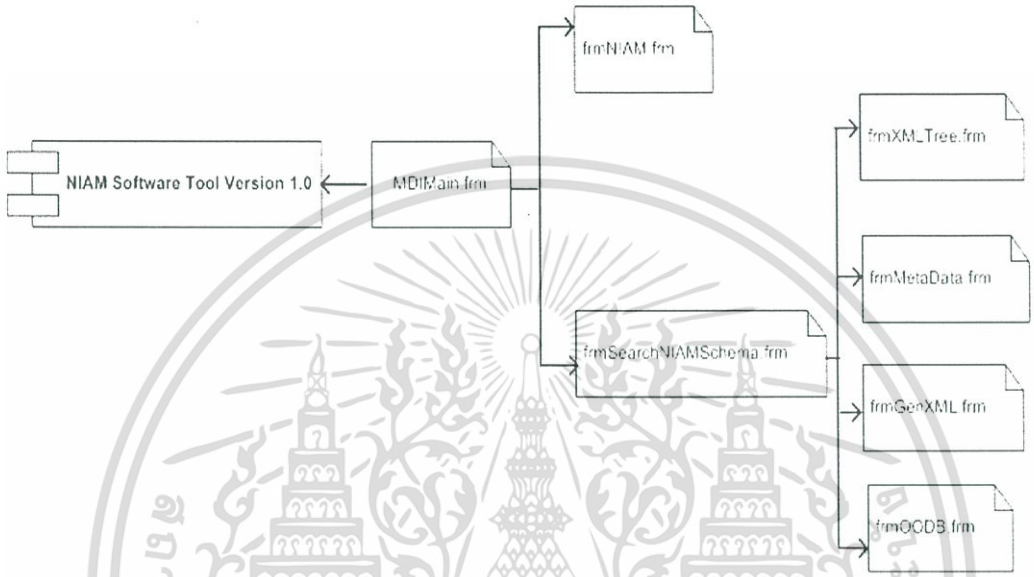
การทำงานของซีเควนซ์ไดอะแกรมของรูปที่ 5.4 จะสิ้นสุดเมื่อเมื่อมีการส่งข้อความย้อนกลับซึ่งจะใช้สัญลักษณ์ลูกศรเส้นประจากออบเจ็กต์ Object Oriented Database Schema กลับมายังเอ็กเตอร์ผู้ใช้

5.1.2 ยูเอ็มแอลในส่วนของโครงสร้าง

5.1.2.1 คอมโพเน็นไดอะแกรม

คอมโพเน็นไดอะแกรมที่นำมาอธิบาย NIAM Software Tool ที่นำเสนอในงานวิจัยนี้ ประกอบด้วย Presentation Logic Syste , Database SubSystem และ Working Logic System

1.Presentation Logic System เป็นส่วนที่แสดงให้เห็นถึง User Interface และ Out put ของระบบ Presentation Logic Systemของระบบงานแสดงได้ดังรูปที่ 5.5

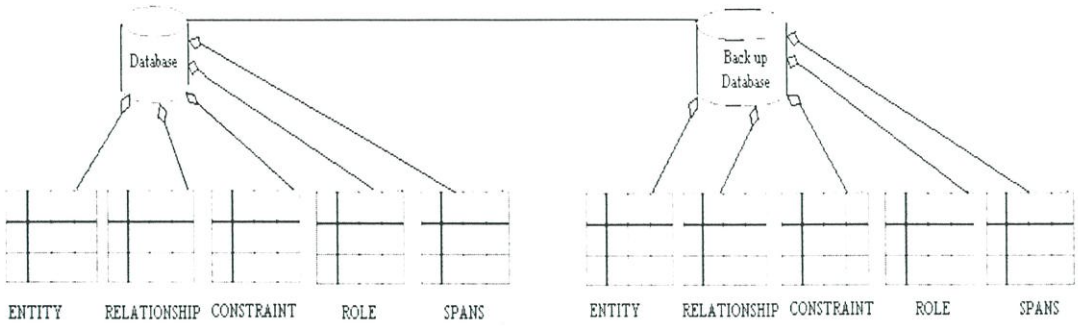


รูปที่ 5.6 Component Diagram ในส่วนของ Presentation Logic System

รูปที่ 5.6 แสดงถึง User Interface ของระบบงาน NIAM Software Tool Version 1.0 ที่ประกอบด้วยหน้าจอเมนูหลัก (MDIMain.frm) ซึ่งในหน้าจอเมนูหลักจะประกอบด้วยการทำงาน 2 ส่วนได้แก่ หน้าจอสำหรับสร้างไนแอมสกีมมา (frmNIAM Schema) และ หน้าจอสำหรับใช้ในการค้นหาไนแอมสกี (frmSearchNIAMSchema.frm) มาที่ได้ทำการสร้างไว้แล้ว

หน้าจอการค้นหาไนแอมสกีมาประกอบด้วยหน้าจอการทำงาน 4 หน้าจอ ได้แก่ หน้าจอการแปลงไนแอมสกีมาเป็นเอ็กซ์เอ็มแอลทรี (frmXMTree.frm) หน้าจอการจัดเก็บข้อมูลจากไนแอมสกีมาลงสู่ฐานข้อมูล (frmMetaData.frm) หน้าจอการแปลงไนแอมสกีมาเป็นเอ็กซ์เอ็มแอลสกีมา (frmGenXML.frm) และหน้าจอการแปลงไนแอมสกีมาเป็นObject Oriented Schema (frmOODB.frm)

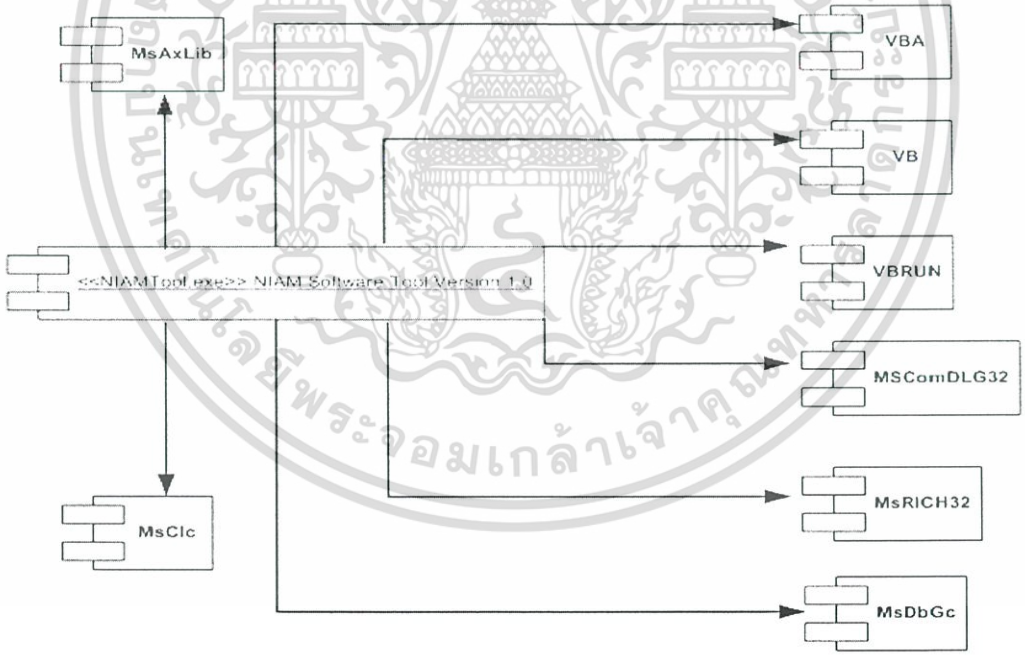
2. Database SubSystem ของระบบ ของระบบงานแสดงได้ดังรูปที่ 5.6



รูปที่ 5.7 Component Diagram ในส่วนของ Database subsystem

รูปที่ 5.7 แสดงคอมโพเนนไดอะเกรมในส่วนของ Database Subsystem ของระบบงานที่แสดงให้เห็นถึงฐานที่จะใช้ในการจัดเก็บข้อมูล (Database) และฐานข้อมูลที่มีไว้สำหรับสำรองข้อมูล (Back Up Database) ซึ่งประกอบด้วยโครงสร้างในการจัดเก็บข้อมูลได้แก่ เทเบิล ENTITY , RELATIONASHIP, CONSTRAINT, ROLE และ SPANS

3. Working Logic System ของระบบ ของระบบงานแสดงได้ดังรูปที่ 5.8

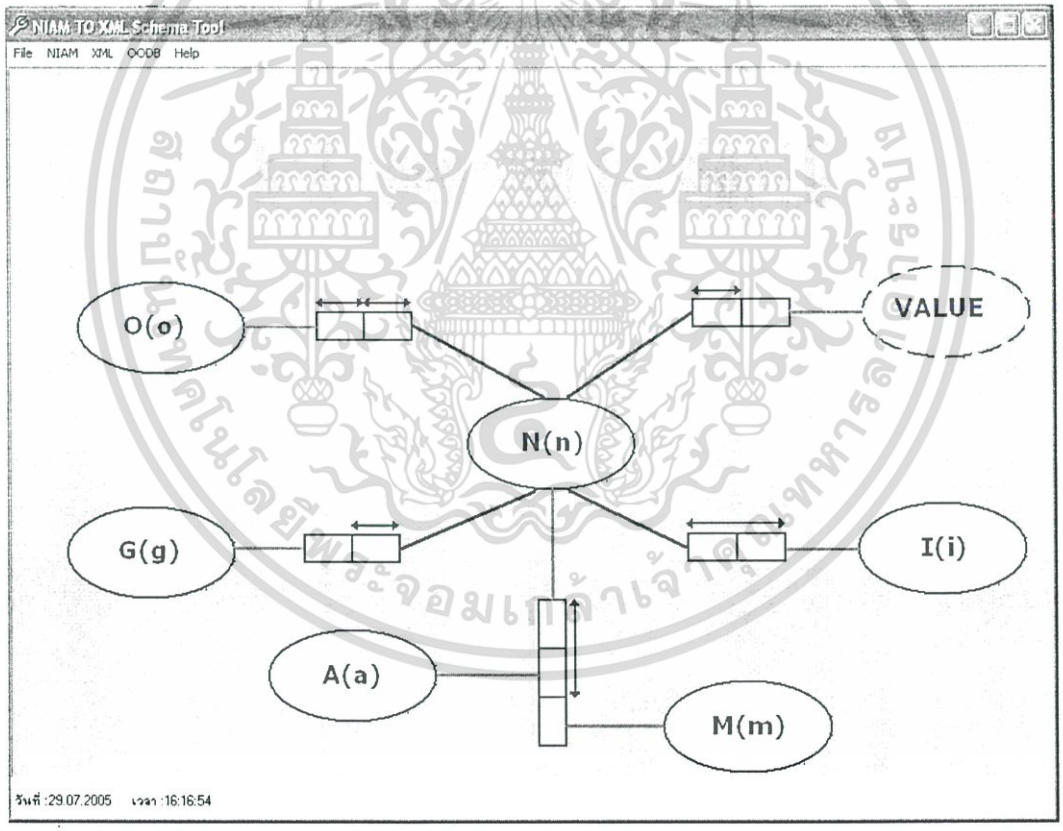


รูปที่ 5.8 Component Diagram ในส่วนของ Working Logic System

รูปที่ 5.8 แสดงคอมโพเนนไดอะเกรมในส่วนของ Working Logic System ซึ่งประกอบด้วยรายละเอียดดังนี้ เมื่อ VBA หมายถึง Visual Basic Version 6.0 for Application ,VB หมายถึง Visual Basic Objects and procedures Version 6.0 , VB RUN หมายถึงVisual Basic Runtime objects and procedures , Std OLE หมายถึง Stdole Version 2.2 (OLE Automation) , MsClc หมายถึง Microsoft windows common control 6.0 , MsDbGc หมายถึง Microsoft Data Bound Grid 6.0, MsAXLib หมายถึง Microsoft Active X Data Objects2.0 Library และ RICH32 หมายถึง Microsoft Rich Textbox Control 6.0

5.2 การใช้งานซอฟแวร์ทูล

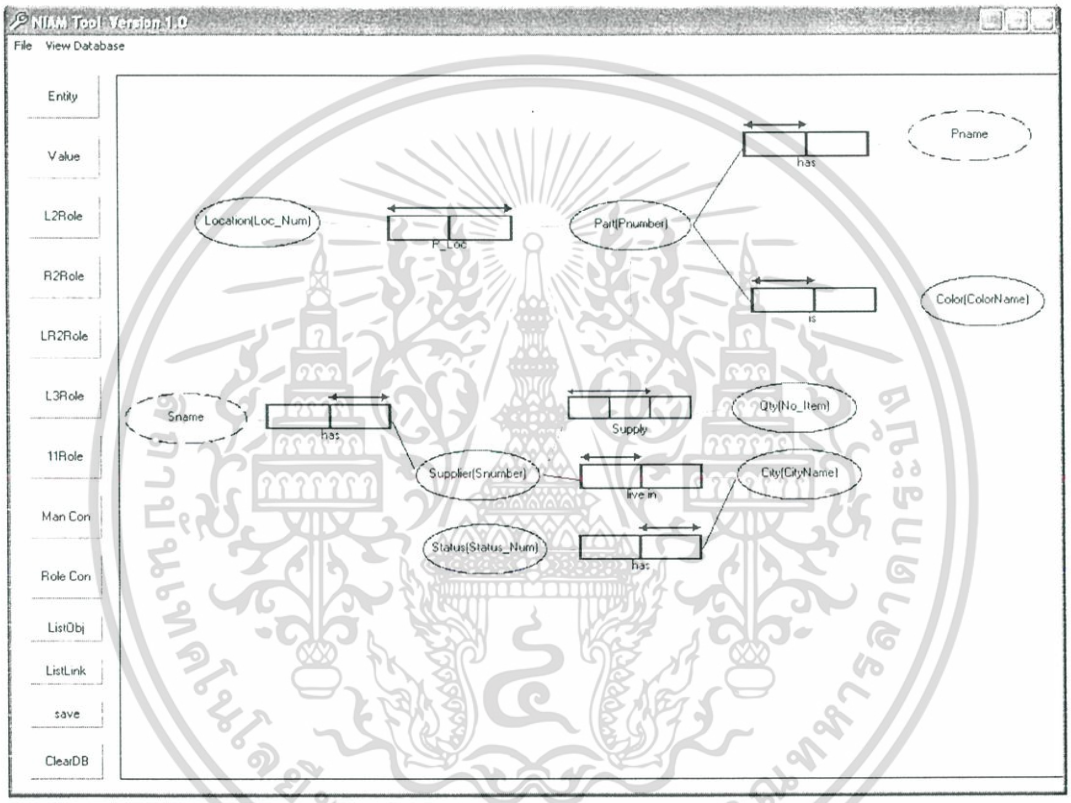
NIAM Software Tool Version 1.0 เป็นโปรแกรมที่พัฒนาขึ้นมาจากแนวคิดการแปลงไนแอมสกีมาเป็นเอ็กซ์เอ็มแอลสกีมาที่ผู้วิจัยได้นำเสนอในบทที่ 4 กระบวนการออกแบบ NIAM Software Tool Version 1.0 นำเสนอในหัวข้อ 5.1 การใช้งาน NIAM Software Tool Version 1.0 ประกอบด้วยรายละเอียดดังต่อไปนี้



รูปที่ 5.9 หน้าจอหลักของ NIAM Software Tool Version 1.0

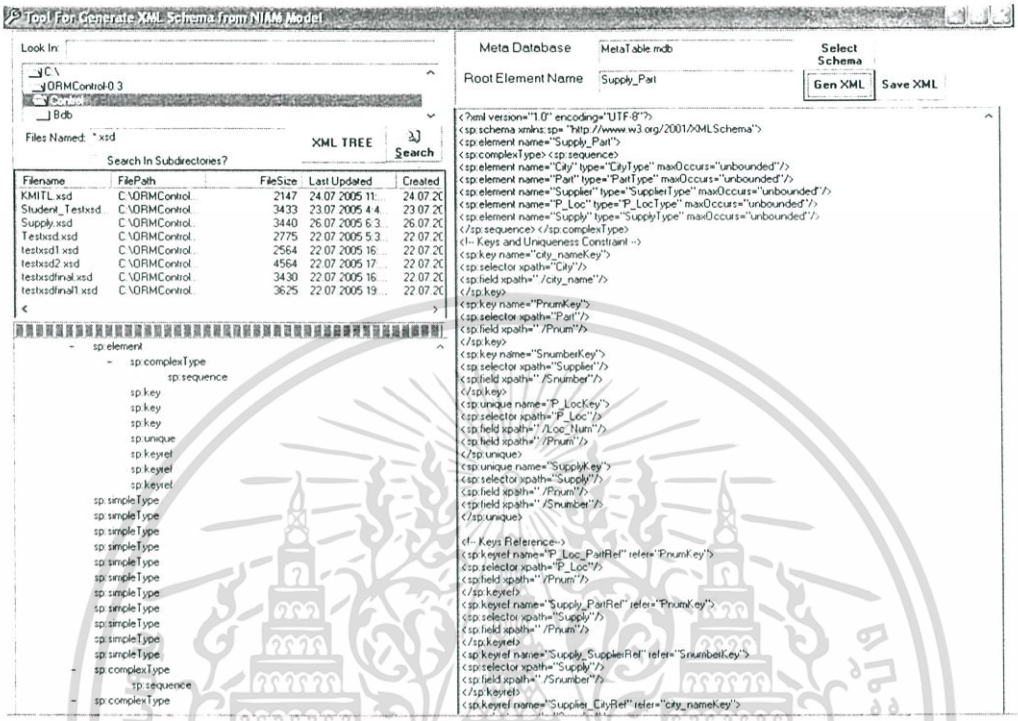
รูปที่ 5.9 หน้าจอหลักของ NIAM Software Tool Version 1.0 ประกอบด้วยรายะเอียดการทำงานดังนี้

1. เมนู File ประกอบด้วยฟังก์ชันการทำงาน 2 ฟังก์ชัน คือ เมนู Print และมนู Exit
2. เมนู NIAM เป็นเมนูของการสร้างและการแก้ไขไนแอมสกีมาประกอบด้วย ฟังก์ชันการทำงาน 3 ฟังก์ชันคือ เมนู Create NIAM เป็นเมนูสำหรับการสร้างไนแอมสกีมา เมนู Save เป็นเมนูสำหรับจัดเก็บไนแอมสกีมาที่ผู้ใช้สร้างขึ้น และเมนู Open เป็นเมนูสำหรับการเรียกไนแอมสกีมาที่ต้องการมาทำการแก้ไข



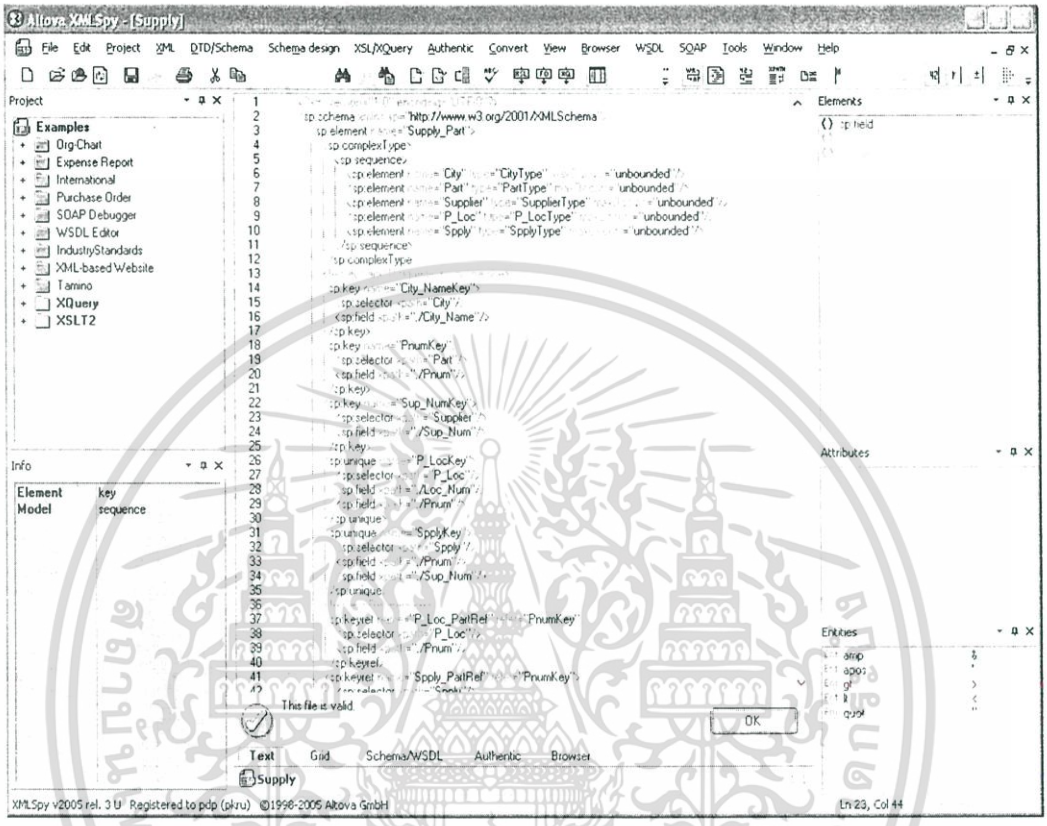
รูปที่ 5.10 หน้าจอการสร้างและแก้ไข NIAM Schema

3. เมนู Gen XML Schema เป็นเมนูสำหรับให้ผู้ใช้ทำการสร้างเอ็กซ์เอ็มแอลสกีมาโดยการเลือกฐานข้อมูลที่จัดเก็บข้อมูลของไนแอมสีมาที่ผู้ใช้สร้างขึ้น(เมต้าดาต้า) หลังจากนั้นจึงทำการกำหนด Root Element แล้วคลิกปุ่ม Gen XML



รูปที่ 5.11 หน้าจอแสดง XML Schema ที่ได้จากการแปลงไนแอมสกีมา

4. เมนู Create XML Document เป็นเมนูสำหรับให้ผู้ใช้ทำการตรวจเช็คความถูกต้องของเอ็กซ์เอ็มแอลสกีมาที่สร้างเสร็จแล้วโดยจะทำการตรวจเช็คกับซอฟต์แวร์ทูลที่นิยมใช้ในปัจจุบัน งานวิจัยนี้ได้นำเอา XML Spy มาใช้ในการอ้างอิง เพื่อให้มั่นใจได้ว่าสกีมาที่สร้างขึ้นถูกต้องและสามารถใช้งานร่วมกับซอฟต์แวร์ทูลที่นิยมใช้ได้



รูปที่ 5.12 หน้าจอแสดงการตรวจเช็คความถูกต้องของ XML Schema ที่ได้จากการแปลงในแอมสกีมา

บทที่ 6
การแปลงเอกสารเอ็กซ์เอ็มแอลเป็นไนแอมสกีมา

การแปลงเอกสารเอ็กซ์เอ็มแอลเป็นไนแอมสกีมา(Reverse Engineering from XML Document to NIAM Schema) มีจุดประสงค์เพื่อทำการนอร์มัลไลเซชันเอกสารเอ็กซ์เอ็มแอลที่ไม่ได้ผ่านขบวนการออกแบบที่ถูกต้อง ให้เป็นเอกสารเอ็กซ์เอ็มแอลที่มีสกีมาที่ถูกต้อง และกำจัดปัญหาเรื่องความซ้ำซ้อนของข้อมูลเพื่อไม่เกิดปัญหาเมื่อทำการแลกเปลี่ยนข้อมูลระหว่าฐานข้อมูล สามารถสรุปประโยชน์ของการทำนอร์มัลไลเซชันเอกสารเอ็กซ์เอ็มแอลได้ดังนี้

1. เพื่อทำการสร้างเอ็กซ์เอ็มแอลสกีมาให้กับเอกสารเอ็กซ์เอ็มแอลที่ไม่ได้ผ่านขบวนการออกแบบอย่างถูกต้อง หรือเพื่อช่วยปรับปรุงเอกสารเอ็กซ์เอ็มแอลที่ด้อยประสิทธิภาพให้มีประสิทธิภาพที่ครบถ้วนสมบูรณ์ เพื่อที่จะแลกเปลี่ยนข้อมูลระหว่างฐานข้อมูล

2.เพื่อกำจัดปัญหาเรื่องความซ้ำซ้อนของข้อมูลเมื่อนำเอกสารเอ็กซ์เอ็มแอลไปใช้ในการแลกเปลี่ยนข้อมูลระหว่างฐานข้อมูล

3.เพื่อเป็นการปรับเปลี่ยนเอ็กซ์เอ็มแอลสกีมา และเอกสารเอ็กซ์เอ็มแอลที่สื่อสารกันระหว่างองค์กรให้เป็นสกีมาที่เป็นรูปแบบเดียวกัน

6.1 อัลกอริทึมในการแปลงเอกสารเอ็กซ์เอ็มแอลเป็นไนแอมสกีมา

ขั้นตอนในการแปลงเอกสารเอ็กซ์เอ็มแอลเป็นไนแอมสกีมามีทั้งหมด 7 ขั้นตอนดังนี้

1 ทำการแปลงเอกสารเอ็กซ์เอ็มแอลเป็นที่อยู่ในรูปแบบลำดับชั้น(Hierarchical XML Document) เป็นเอกสารเอ็กซ์เอ็มแอลแบบ Flat XML

2 นำข้อมูลจากเอกสารเอ็กซ์เอ็มแอลในขั้นตอนที่ 1 มาทำการหาการขึ้นต่อกันของแอทริบิวต์ (Function Dependency :FD)

3 ทำการแปลงแต่ละ FD ในขั้นตอนที่ 2 เป็นไนแอมแฟคไทป์ โดยกำหนดให้แต่ละ FD เท่ากับหนึ่งแฟคไทป์

4 ทำการเพิ่ม Uniqueness Constraint ของแต่ละแฟคไทป์(อาจพิจารณาจาก Key , Keyref ของเอ็กซ์เอ็มแอลสกีมาของเดิม)

5 ทำการเพิ่ม Mandatory Constraint ให้กับแต่ละแฟคไทป์(อาจพิจารณาจาก min , max Occurent Constraint ของเอ็กซ์เอ็มแอลสกีมาเดิม)

6 ทำการตรวจเช็ค และเพิ่ม Constraint อื่นๆ เพื่อให้ได้ไนแอมสกีมาที่ถูกต้อง

7 ทำการรวมแต่ละ Fact type ที่ได้จากขั้นตอนที่ผ่านมาเพื่อให้ได้ไนแอมสกีมาที่สมบูรณ์

6.2 ตัวอย่างการแปลงเอกสารเอ็กซ์เอ็มแอลเป็นไนแอมสกีมา

เอกสารเอ็กซ์เอ็มแอลที่แสดงดังรูปที่ 6.1 เป็นเอกสารเอ็กซ์เอ็มแอลแบบลำดับชั้น(Hierachical XML)

```
<?xml version="1.0" encoding="UTF-8"?>
<orders>
        <order>
                <cust_no>C001</cust_no>
                <cust_name>พฤกล จันทร์ดาว</cust_name>
                <cust_zone>
                        <city>อยุธยา</city>
                        <zone_id>001</zone_id>
                </cust_zone>
                <product>
                        <product_id>P001</product_id>
                        <product_name>PUMP</product_name>
                        <price>1500 Baht</price>
                        <quantity>3</quantity>
                </product>
        </order>
                <order>
                <cust_no>C001</cust_no>
                <cust_name>พฤกล จันทร์ดาว</cust_name>
                <cust_zone>
                        <city>อยุธยา</city>
                        <zone_id>001</zone_id>
                </cust_zone>
                <product>
                        <product_id>P003</product_id>
                        <product_name>BOLT</product_name>
                        <price>5 Baht</price>
                        <quantity>120</quantity>
                </product>
        </order>
                <order>
                <cust_no>C002</cust_no>
                <cust_name>นที สิงห์ทอง</cust_name>
                <cust_zone>
                        <city>สิงห์บุรี</city>
                        <zone_id>002</zone_id>
                </cust_zone>
                <product>
                        <product_id>P004</product_id>
                        <product_name>Breaker-07</product_name>
                        <price>300</price>
                        <quantity>10</quantity>
                </product>
        </order>
                <order>
                <cust_no>C003</cust_no>
                <cust_name>สุวิทย์ ศรีทอง</cust_name>
                <cust_zone>
                        <city>สิงห์บุรี</city>
                        <zone_id>002</zone_id></zone_id>
                </cust_zone>
                <product>
                        <product_id>P001</product_id>
                        <product_name>PUMP</product_name>
                        <price>1500 Baht</price>
                        <quantity>5</quantity>
                </product>
        </order>
</orders>
```

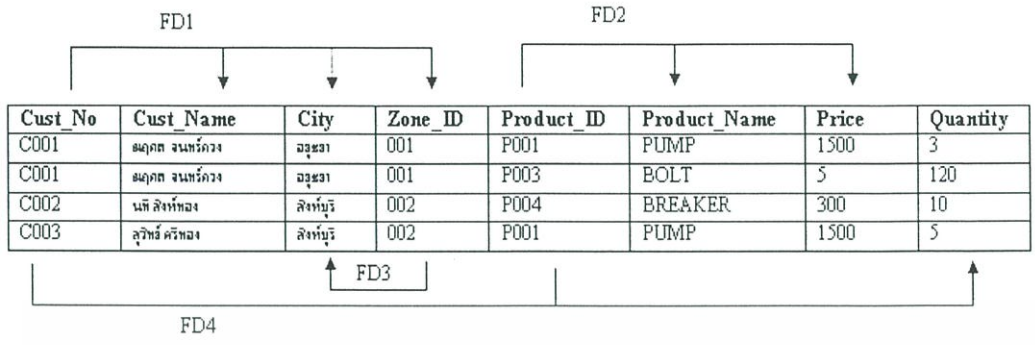
รูปที่ 6.1 แสดง Hierarchical XML

1 ทำการแปลงเอกสารเอ็กซ์เอ็มแอลเป็นที่อยู่ในรูปแบบลำดับชั้น(Hierarchical XML Document) เป็นเอกสารเอ็กซ์เอ็มแอลแบบ Flat XML

```
<?xml version="1.0" encoding="UTF-8"?>
<orders>
        <order>
                <cust_no>C001</cust_no>
                <cust_name>พฤกล จันทร์ดาง</cust_name>
                <city>อยุธยา</city>
                <zone_id>001</zone_id>
                <product_id>P001</product_id>
                <product_name>PUMP</product_name>
                <price>1500 Baht</price>
                <quantity>3</quantity>

        </order>
                <order>
                <cust_no>C001</cust_no>
                <cust_name>พฤกล จันทร์ดาง</cust_name>
                <city>อยุธยา</city>
                <zone_id>001</zone_id>
                <product_id>P003</product_id>
                <product_name>BOLT</product_name>
                <price>5 Baht</price>
                <quantity>120</quantity>

        </order>
                <order>
                <cust_no>C002</cust_no>
                <cust_name>นที กิงก์ทอง</cust_name>
                <city>ลิงก์บุรี</city>
                <zone_id>002</zone_id>
                <product_id>P004</product_id>
                <product_name>Breaker-07</product_name>
                <price>300</price>
                <quantity>10</quantity>

        </order>
                <order>
                <cust_no>C003</cust_no>
                <cust_name>ทวิทย์ ศรีทอง</cust_name>
                <city>ลิงก์บุรี</city>
                <zone_id>002</zone_id>
                <product_id>P001</product_id>
                <product_name>PUMP</product_name>
                <price>1500 Baht</price>
                <quantity>5</quantity>

        </order>
</orders>
```

รูปที่ 6.2 แสดง Flat XML ที่แปลงมาจากรูปที่ 5.1

2. นำข้อมูลจากเอกสารเอ็กซ์เอ็มแอลในขั้นตอนที่1 มาทำการหา Function Dependency(FD)



Cust_No	Cust_Name	City	Zone_ID	Product_ID	Product_Name	Price	Quantity
C001	สมคิด จนหร่วง	อยุธยา	001	P001	PUMP	1500	3
C001	สมคิด จนหร่วง	อยุธยา	001	P003	BOLT	5	120
C002	นพี ลิงห์ทอง	ลิงห์บุรี	002	P004	BREAKER	300	10
C003	สุวิทย์ ศรีทอง	ลิงห์บุรี	002	P001	PUMP	1500	5

รูปที่ 6.3 แสดง Functional Dependency (FD) จาก XML Document รูปที่ 6.2

จากรูปที่ 6.3 เขียนอยู่ในรูปของFunctional Dependency ได้ดังนี้

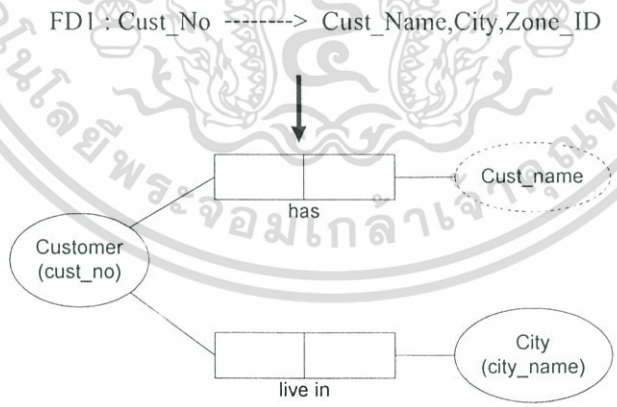
FD1 : Cust_No --------> Cust_Name,City,Zone_ID

FD2 : Produc_ID --------> Product_Name,Proice

FD3 : Zone_ID --------> City

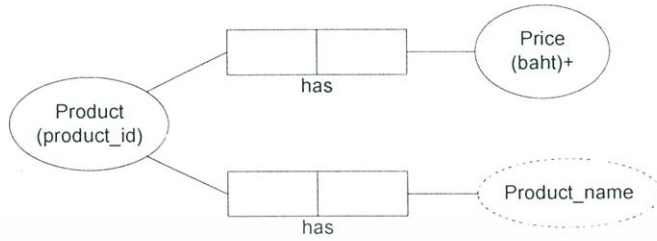
FD4 : Cust_NO,Product_ID --------> Quantity

3. ทำการแปลงแต่ละ FD ในขั้นตอนที่2 มาวาดไนแอมแฟคไทป์ โดยกำหนดให้แต่ละ FD เท่ากับหนึ่งแฟคไทป์



รูปที่ 6.4 แสดงไนแอมแฟคไทป์ของ FD1

FD2 : Produc_ID --------> Product_Name,Proice

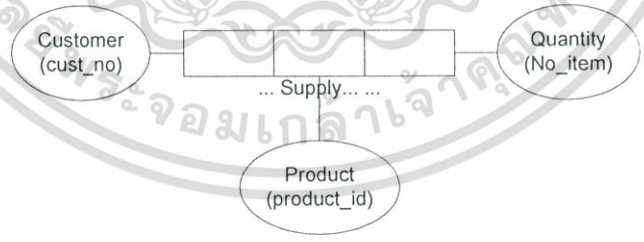


รูปที่ 6.5 แสดงไนแอมแฟคไทป์ของ FD2

FD3 : Zone_ID --------> City

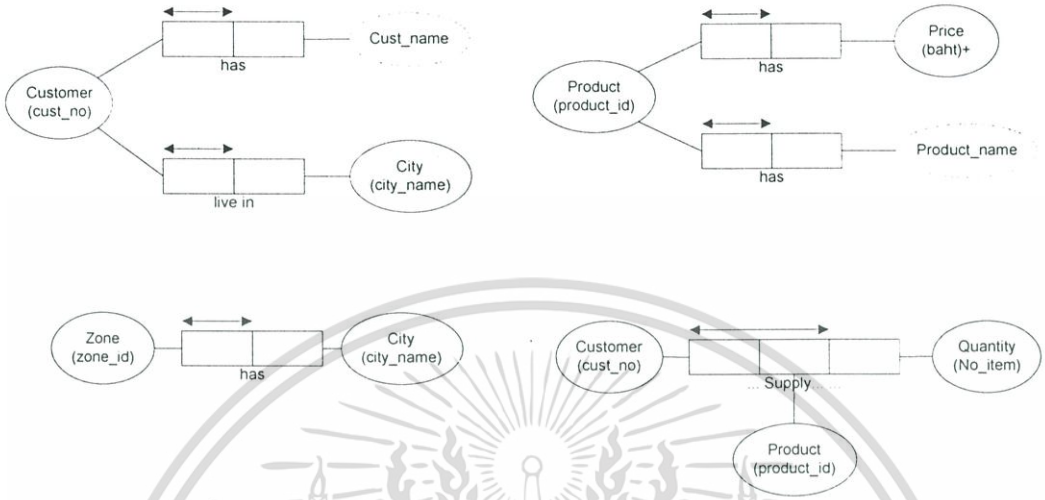
รูปที่ 6.6 แสดงไนแอมแฟคไทป์ของ FD3

FD4 : Cust_NO,Product_ID --------> Quantity



รูปที่ 6.7 แสดงไนแอมแฟคไทป์ของ FD4

4. ทำการเพิ่ม Uniqueness Constraint ของแต่ละแฟคไทป์(อาจพิจารณาจาก Key , Keyref ของเอ็กซ์เอ็มแอลสกีมาของเดิม)



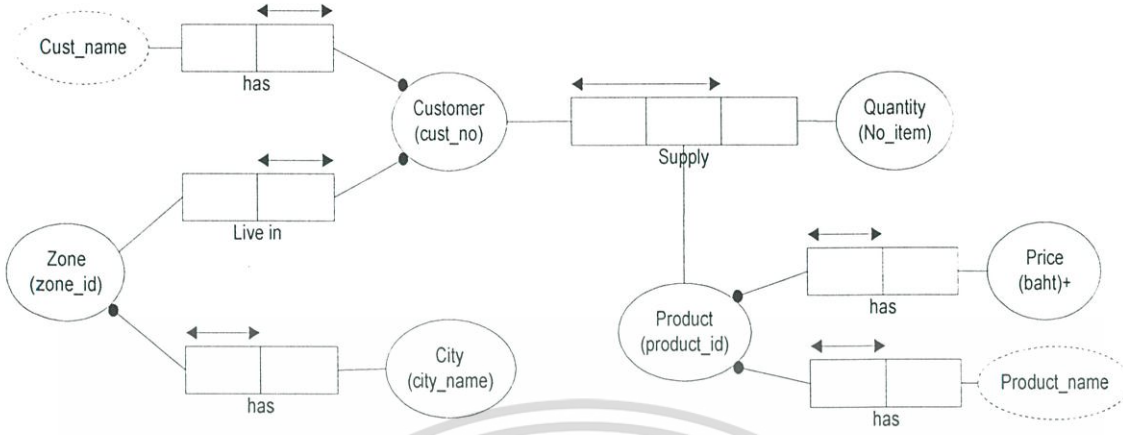
รูปที่ 6.8 แสดงการเพิ่ม Uniqueness Constraint ของแต่ละ Fact Type

5. ทำการเพิ่ม Mandatory Constraint ให้กับแต่ละแฟคไทป์(อาจพิจารณาจาก min , max Occurent Constraint ของเอ็กซ์เอ็มแอลสกีมาเดิม)

รูปที่ 6.9 แสดงการเพิ่ม Mandatory Constraint ของแต่ละ Fact Type

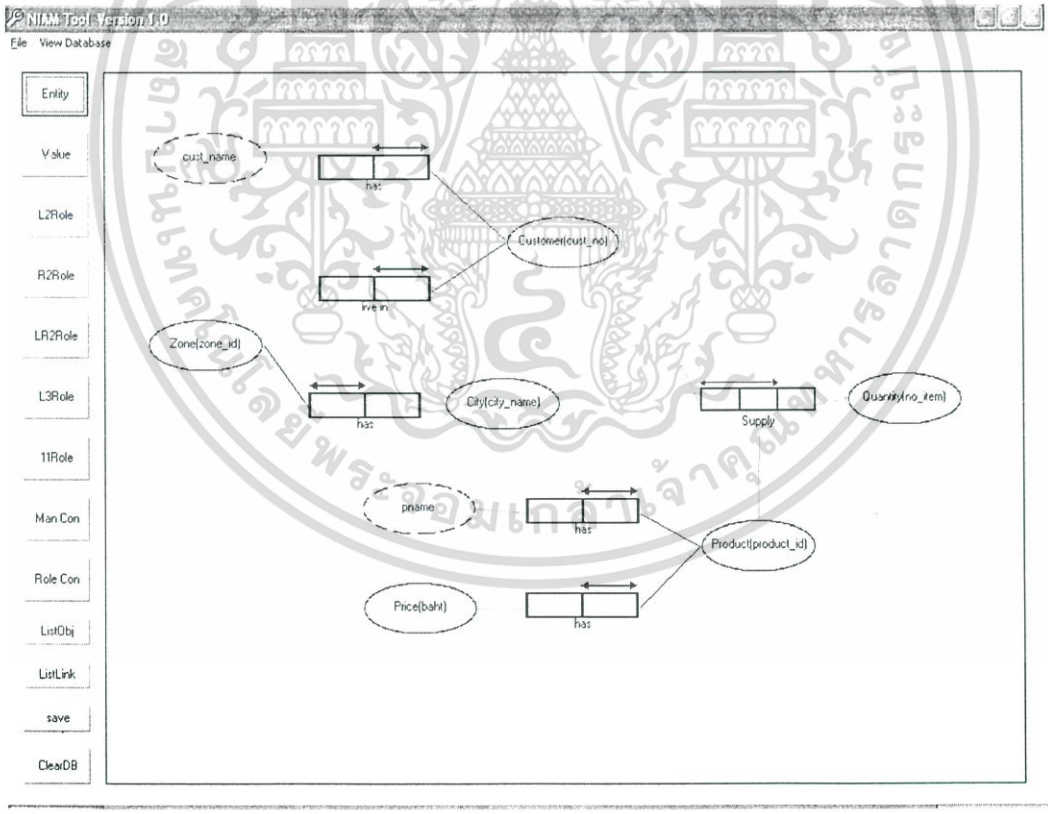
6. ทำการตรวจเช็ค และเพิ่ม Constraint อื่นๆ เพื่อให้ได้ในแอมสกีมาที่ถูกต้อง

7. ทำการรวมแต่ละ Fact type ที่ได้จากขั้นตอนที่ผ่านมาเพื่อให้ได้ไนแอมสกีมาที่สมบูรณ์



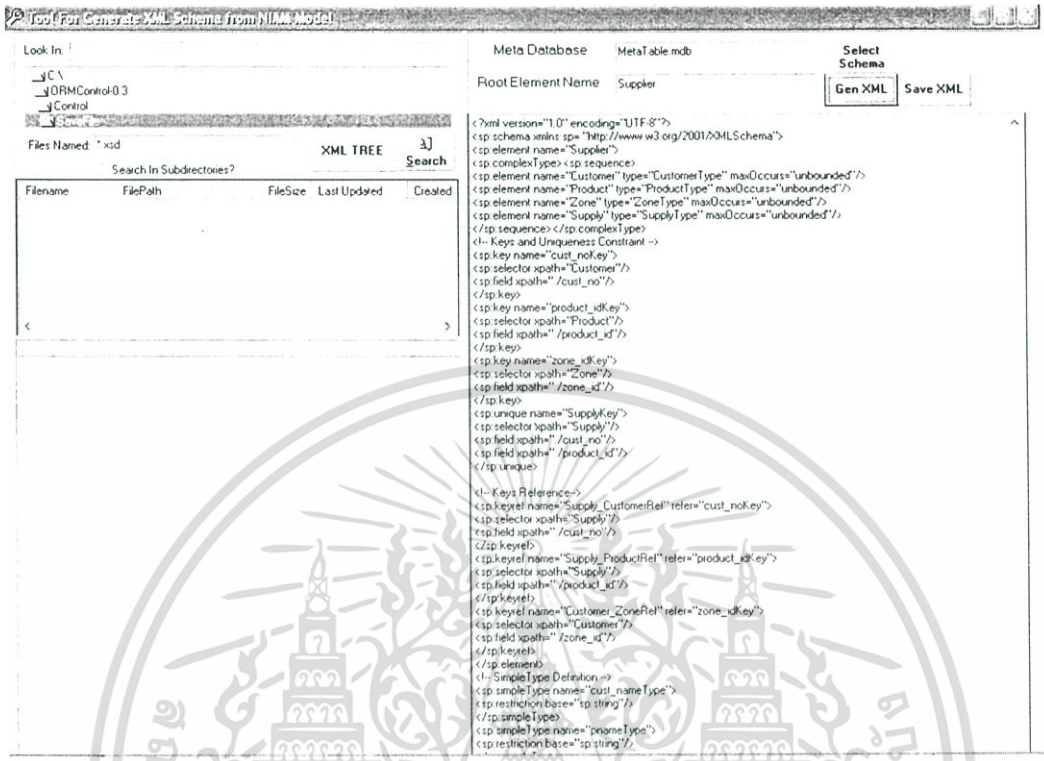
รูปที่ 6.10 แสดงไนแอมสกีมาที่ได้จากการรวมของแต่ละ Fact Type ทั้งหมด

เมื่อนำไนแอมสกีมาที่สมบูรณ์ไปสร้างในซอฟต์แวร์โปรแกรมที่ออกแบบในบทที่ 5 แสดงดังรูปที่ 6.11



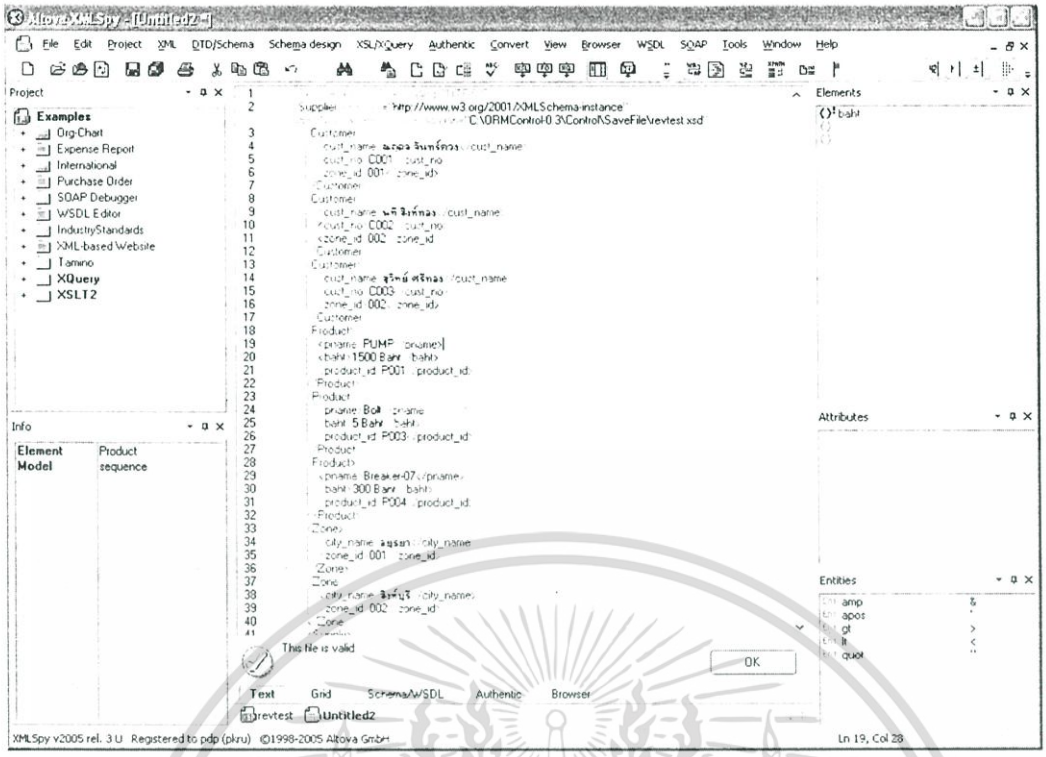
รูปที่ 6.11 แสดงไนแอมสกีมาที่ผู้ใช้สร้างในซอฟต์แวร์โปรแกรม

หลังจากที่ผู้ใช้สร้างไนแอมสกีมา และบันทึกแล้วจึงทำการสร้างเอ็กซ์เอ็มแอลสกีมาอย่างอัตโนมัติจากซอฟต์แวร์โปรแกรม เอ็กซ์เอ็มแอลสกีมาที่ได้เป็นเอ็กซ์เอ็มแอลสกีมาที่กำจัดความซ้ำซ้อนของข้อมูล



รูปที่ 6.12 แสดงเอ็กซ์เอ็มแอลสกีมาที่สร้างจากซอฟต์แวร์โปรแกรม

เมื่อสร้างเอกสารเอ็กซ์เอ็มแอลที่อ้างอิงกับสกีมาที่ได้ในรูปที่ 6.12 จะทำให้ได้เอกสารเอ็กซ์เอ็มแอลแบบ Flat XML แสดงดังรูป 6.13



รูปที่ 6.13 แสดงเอกสารเอ็กซ์เอ็มแอล(Flat XML)ที่อ้างอิงกับเอ็กซ์เอ็มแอลสกีมารูปที่ 6.12

บทที่ 7
สรุปผลการวิจัยและข้อเสนอแนะ

7.1 สรุปผลการวิจัย

งานวิจัยฉบับนี้ได้นำเสนออัลกอริทึมในการแปลงไนแอมสกีมาเป็นเอ็กซ์เอ็มแอลสกีมา และสร้างซอฟต์แวร์ทูลที่ทำงานตามอัลกอริทึมที่ได้ศึกษามาเพื่อเป็นการแสดงให้เห็นถึงการนำไปประยุกต์ใช้งานจริง การนำไนแอมสกีมามาใช้เป็นกรอบแนวคิดในการสร้างเอ็กซ์เอ็มแอลสกีมานั้นถือว่าเป็นการออกแบบเอ็กซ์เอ็มแอลสกีมาในระดับแนวคิดก่อนที่จะเป็นเอ็กซ์เอ็มแอลสกีมาที่จะนำไปใช้งานจริง ซึ่งเป็นแนวคิดที่เกิดขึ้นใหม่จากงานวิจัยนี้โดยยังไม่มีซอฟต์แวร์ทูลตัวใดในปัจจุบันกล่าวถึงมาก่อน ซอฟต์แวร์ทูลที่มีอยู่ในปัจจุบันนี้มีเพียงการนำเอากราฟฟิคที่นำเสนอรูปแบบของโครงสร้างข้อมูลแบบต้นไม้(Tree Based) มาใช้ในการสร้างเอ็กซ์เอ็มแอลสกีมาเพื่อความสะดวกในการสร้างที่ดีกว่าการเขียนเป็นข้อความ(Text Based) เท่านั้น ดังนั้นการนำไนแอมสกีมามาใช้ในการกำหนดกรอบแนวคิดในการสร้างเอ็กซ์เอ็มแอลสกีมาจึงมีข้อดีกว่าการใช้กราฟฟิกแสดงโครงสร้างแบบต้นไม้ในการสร้างเอ็กซ์เอ็มแอลสกีมาหลายประการ เช่น

- ไนแอมสกีมานำเสนอกฎบังคับความถูกต้องของฐานข้อมูลได้ง่ายกว่า
- การนำเสนอกราฟฟิกในรูปแบบโครงสร้างต้นไม้มีกฎบังคับความถูกต้องน้อยกว่าไนแอม
- การตรวจสอบความถูกต้องทำได้ง่ายกว่า โดยการทดลองป้อนข้อมูลลงในไนแอมสกีมาได้โดยตรง
- ไนแอมสกีมาเหมาะที่จะนำมาใช้บรรยายความรู้ของผู้เชี่ยวชาญมากกว่า เนื่องจากมีรากฐานมาจากโครงสร้างของภาษาธรรมชาติ
- การนำไนแอมสกีมามาใช้เป็นตัวกำหนดกรอบแนวคิดทำให้สามารถขจัดความซ้ำซ้อนของข้อมูลได้

การแปลงไนแอมสกีมาเป็นเอ็กซ์เอ็มแอลสกีมาสามารถกระทำได้หลายวิธีแต่ละวิธีก็มีจุดประสงค์แตกต่างกัน สำหรับงานวิจัยนี้มีจุดประสงค์หลักเพื่อขจัดความซ้ำซ้อนของข้อมูล และนำเสนอกฎบังคับความถูกต้องของฐานข้อมูลได้อย่างครบถ้วน ทั้งนี้เพื่อความถูกต้องเมื่อต้องการนำเอ็กซ์เอ็มแอลสกีมาไปใช้ในการกำหนดโครงสร้างของเอกสารเอ็กซ์เอ็มแอลที่แลกเปลี่ยนข้อมูลระหว่างฐานข้อมูล แต่อย่างไรก็ตามการแปลงไนแอมสกีมาเป็นเอ็กซ์เอ็มแอลสกีมายังไม่สามารถกระทำได้อย่างครบถ้วนสมบูรณ์เนื่องจากมีคุณสมบัติบางประการที่ไม่สามารถเข้ากันได้ของทั้งไนแอมสกีมาและเอ็กซ์เอ็มแอลสกีมา กล่าวคือ

- เอ็กซ์เอ็มแอลสกีมาไม่สนับสนุน Exclusion constraints และ Subset constraints ที่มีอยู่ในคุณสมบัติของไนแอมสกีมา

- เอ็กซ์เอ็มแอลสกีมาไม่สนับสนุน Disjunctive mandatory constraints ที่มีในคุณสมบัติของไนแอมสกีมา นอกจากนี้แล้วเอ็กซ์เอ็มแอลสกีมายังไม่สนับสนุนกฎบังคับความถูกต้องของข้อมูลอีกหลายประการที่มีในคุณสมบัติของไนแอมสกีมาเช่น Frequency constraints และ Ring constraints เป็นต้น
- เอ็กซ์เอ็มแอลสกีมาไม่สนับสนุนการสืบทอดแบบMultiple inheritance แต่ไนแอมสกีมาสนับสนุนการสืบทอดได้ทั้งแบบ Single inheritance และ Multiple inheritance
- ไนแอมสกีมาไม่สนับสนุนคุณสมบัติบางประการที่มีในเอ็กซ์เอ็มแอลสกีมา ได้แก่ การสร้างรูปแบบของข้อมูล(Patterns) ตามที่ผู้ใช้ต้องการ การจัดเรียงลำดับของข้อมูล และการเก็บข้อมูลที่เป็นข้อความปนกับแท็กได้ แสดงดังรูปที่ 7.1

```
<paragraph> The Toyota Car <model> ALTIS </model>
design in <year> 2000 </year> in industrial garden
<lacation> Ayutthaya Thailand </location> <paragraph>
```

รูปที่ 7.1 แสดงข้อความที่ปนกับแท็กของเอ็กซ์เอ็มแอลสกีมา

จากรูปที่ 7.1 แสดงการเก็บมูลของเอกสารเอ็กซ์เอ็มแอลที่มีข้อความรวมอยู่กับแท็ก ส่วนที่ขีดเส้นใต้แสดงส่วนที่เป็นข้อความ เป็นคุณสมบัติที่ไนแอมสกีมาไม่สนับสนุนในกรณีที่เป็นไนแอมสกีมาจะสามารถแสดงได้เฉพาะส่วนที่เป็นแท็กและข้อมูลในแท็กเท่านั้น

7.2 ปัญหาและอุปสรรค

ปัญหาและอุปสรรคของการทำวิจัยนี้สามารถแบ่งออกเป็นสองส่วนคือ ส่วนของการออกแบบอัลกอริทึมในการแปลงไนแอมสกีมาเป็นเอ็กซ์เอ็มแอลสกีมา และส่วนของการสร้างซอฟต์แวร์ทูล

ปัญหาและอุปสรรคของการออกแบบอัลกอริทึมในการแปลงไนแอมสกีมาเป็นเอ็กซ์เอ็มแอลสกีมาคือการรวบรวมคุณสมบัติต่างๆของสกีมาทั้งสองแล้วนำมาเทียบเคียงกันเพื่อหาอัลกอริทึมในการแปลงที่ถูกต้อง และครบถ้วน ซึ่งทั้งไนแอมสกีมาและเอ็กซ์เอ็มแอลสกีมาต่างก็มีรูปแบบที่แตกต่างกัน และมีกฎบังคับความถูกต้องที่ต่างกัน ทำให้ไม่สามารถแปลงจากไนแอมสกีมาเป็นเอ็กซ์เอ็มแอลสกีมาได้ทุกคุณสมบัติ

ปัญหาและอุปสรรคของการสร้างซอฟต์แวร์ทูลคือการแปลงจากไฟล์รูปภาพเป็นไฟล์ข้อความและการสร้างตัวอ่านไฟล์ข้อความที่ได้เพื่อที่จะนำข้อความในแต่ละออบเจ็กต์ของไนแอมสกีมา มาป้อนลงฐานข้อมูลเชิงสัมพันธ์สำหรับสร้างตารางเมต้า(Meta Table)

7.3 แนวทางการพัฒนาต่อ

สำหรับแนวทางในการพัฒนาต่อจากงานวิจัยนี้มีได้หลายแนวทางเช่น การพัฒนาซอฟต์แวร์ทูลให้มีความสมบูรณ์มากขึ้น การสร้างอัลกอริทึมสำหรับแปลงจากเอ็กซ์เอ็มแอลสกีมาเป็นไนแอมสกีมา(XML Schema to NIMA Schema)

การพัฒนาซอฟต์แวร์ทูลให้มีความสมบูรณ์มากขึ้นโดยสามารถสร้างเอกสารเอ็กซ์เอ็มแอลที่สามารถอ้างอิงกับสกีมาที่สร้างขึ้นมาพร้อมทั้งสามารถตรวจเช็คตวามถูกต้องตรงกันระหว่างข้อมูลในเอกสารเอ็กซ์เอ็มแอลและข้อกำหนดที่สร้างขึ้นในเอ็กซ์เอ็มแอลสกีมา(Validate) เพื่อรับประกันความถูกต้องของเอกสารเอ็กซ์เอ็มแอลและอำนวยความสะดวกให้กับผู้ใช้งาน

การสร้างอัลกอริทึมสำหรับแปลงจากเอ็กซ์เอ็มแอลสกีมาเป็นไนแอมสกีมา(XML Schema to NIMA Schema)ซึ่งเป็นการกระทำย้อนกลับขบวนการของงานวิจัยนี้เพื่อสามารถนำเอ็กซ์เอ็มแอลสกีมาที่เป็นไฟล์ข้อความแปลงกลับไปเป็นไฟล์รูปภาพของไนแอมสกีมา เพื่อความสะดวกในการตรวจสอบความถูกต้องครบถ้วน และง่ายต่อการแก้ไขปรับปรุงเพราะเป็นการแปลงจากระดับลอจิก (Logical Level) ไปสู่ระดับบนหรือระดับแนวคิด (Conceptual Level)

เอกสารอ้างอิง

[1] Erik T.Ray., Learning XML: O'reilly ;2001

[2] Halpin, T. "Business Rule and Object Roles Modeling" Database Programming & Design, October 1996, vol. 9, no. 10, pp. 66-72

[3] Van Griethuysen, J. J.(ed) (1982), "Concepts and Terminology for the Conceptual Schema and the information base". ISO Report TC97/SC5/WG3.

[4] Nijssen, G.M.& Halpin, T.A. Conceptual Schema and Relational Database Design: Prentic Hall; 1989.

[5] C. M. R. Leung, G.M. Nijssen, "From a NIAM Conceptual Schema into the optimal SQL Relational Database Schema". Australian Computer Journal Volume19(2), 69-75 (1987)

[6] Halpin, T. *Information Modeling and Relational Database*, Morgan Kuafmann, April 2001

[7] Mani M.,Lee D.,Richard R., "Semantic Data Modeling using XML Schemas". In Proceedings 20th Intl. Conf. on Conceptual Modeling (ER), 2001.

[8] Routledge N., Bird L.,Goodchild A., "UML and XML Schema". [Online]. Available :http://titanium.dstc.edu.au/papers/adc2002.pdf

[9] Lee D.,Mani M.,Chiu F.,W.Chu W, "Net&CoT:Translating Relational Schemas to XML Schemas using Semantic Constraints". CIKM'02 November 4-9,2002 Copyright 2002 ACM.

[10] XMLSPY5 Enterprise Edition(Version 5 rel.3) Copy right, 1998-2002 Altova GmbH. [Online]. Available :http://www.altova.com/products_ide.html

[11] Chen, P.P., "The Entity Relationship Model", ACM Transctions on Database Systems, Vol 1, 1976.

[12] Halpin, T., "Object Role-Modeling: an Overview".[Online]. Available : http://www.orm.net/pdf/ORMwhitePaper.pdf

[13] XML Schema Part 0: Primer Second Edition W3C Recommendation 28 October 2004 [Online]. Available : http://www.w3.org/TR/xmlschema-0/

[14] XML Schema Part 1: Structures Second Edition W3C Recommendation 28 October 2004 [Online]. Available : http://www.w3.org/TR/xmlschema-1/

[15] XML Schema Part 2: Datatypes Second Edition W3C Recommendation 28 October 2004 [Online]. Available : http://www.w3.org/TR/xmlschema-2/

[16] W3 School [Online]. Available: http://www.w3schools.com/

[17] C.J. Date, An Introduction to Database Systems, Seventh Edition, Addison-Wesley Publishing Company, 2000.

[18] Elmasri,Ramez and Navathe,Shamkant B., Fundamental of Database Systems. Redwood City,CA :The Benjamin/Cummings Publising Company,Inc.1989.

[19] Booch G.,Rumbaugh J.,Jacobson I., The Unified Modeling Language User Guide :Addison – Wesley 1999

[20]Larman C., Applying UML and patterns :an introduction to object-oriented analysis and design : Prentice Hall, 1998.

[21] Bernus, P., Mertins, K. & Schmidt, G. Handbook on Architectures of Information Systems :Springer-Verlag,Berlin,1998.

ภาคผนวก

ผลงานวิจัยที่ได้รับการตีพิมพ์

1. Narudol Chankuang, Suphamit Chittayasothorn. "A Software Tool for Object and XML Schemas Generation". Communications, Computers and signal Processing, 2003. PACRIM. 2003 IEEE Pacific Rim Conference on Volume 2, 28-30 Aug. 2003 Page(s):675 - 678 vol.2

2.Narudol Chankuang, Suphamit Chittayasothorn. "An Object and XML Database Schemas Design Tool ". Information Technology: Coding and Computing, 2004. Proceedings. ITCC 2004. International Conference on Volume 2, 5-7 April 2004 Page(s):421 - 427 Vol.2

IEEE

2003 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM '03)

August 28-30, 2003 Victoria, B.C., Canada

Conference Timetable UPDATED

Session Details UPDATED

Status of submitted papers UPDATED

Conference Registration .doc .pdf

Arrival in and Staying in Victoria

Transportation from Downtown to Conference

Internet Cafe (e-mail access): August 29 at Engineering Lab Wing ELW A317, 10:00 a.m. -3:00 p.m.

Credit Card Payment

Parking Information

A Software Tool for Object and XML Schemas Generation

Narudol Chankuang, Suphamit Chittayasothorn
Department of Computer Engineering Faculty of Engineering, King Mongkut's Institute of Technology, Ladkrabang
Bangkok, Thailand 10520.
Email: s4061635@kmitl.ac.th ,suphamit@kmitl.ac.th

Abstract – **Nowadays relational database schemas are designed by using well-known database design techniques such as the Entity Relationship Model and the normalization process. The result schemas can be guaranteed to have minimum redundancies if the fifth normal form (5NF) is achieved. However, in more recent database schema design such as the design of object database schema, the concern about minimum redundancies does not seem to be an important issue. Functional dependencies may still appear in an object class of the class diagram thus introduce update anomalies. This paper presents the use of NIAM, a well-established conceptual schema model, as a conceptual model for the design of object databases. A transformation from a NIAM to an OODB schema with minimum redundancy is presented. The conceptual schema can also be transformed into an XML schema. This is a good approach to XML schema design since NIAM gives a conceptual framework for the design. A transformation from a NIAM schema to an XML schema is presented. A software tool for object and XML schema generation has been developed.**

I. Introduction

Existing software tools for object database schema generations are commonly based on the Entity Relationship Model [1] or the UML class diagrams [2]. These diagrams show relationships between entity types or object classes but do not show relationships between attributes. This means that even basic dependencies such as functional dependencies (FD) between attributes cannot be represented. Redundancies caused by FDs between a part of the key attributes and non-key attributes may occur. Also, a very common type of redundancy caused by FDs between non-key attributes may occur. The result object schemas therefore still have update, insertion and deletion anomalies and are not easy to work with. This research project presents a software tool for object database design based on a well-defined NIAM conceptual schema model. The input to this tool is a conceptual schema for the user's application and the output is an object database schema that guarantees not to have redundancies and free from insert, delete and update anomalies.

Another potential application of the NIAM conceptual schema model is its usage as a conceptual framework for XML schema designers. Nowadays, the XML (Extensible Markup Language) [3,4,5] becomes a popular data transfer language between data sources. An XML schema defines the structure of XML documents. It is human readable but not very easy to understand especially for complex schemas. Software tools provide graphical editing facilities are available commercially[6,7,8]. These editors use a tree structure to represent the XML schemas without any conceptual frameworks. Our software tool also presents the use of the NIAM conceptual schema model as a conceptual framework for an XML schema and generates an XML database schema from an input NIAM conceptual schema.

II. Niam

NIAM is a fact-oriented conceptual schema model. Main concepts in NIAM include entity types, label types (or value types), fact types and reference types (or bridge type). Similar to other semantic data model, an entity in NIAM is an object of interest. Entities which have the same properties form entity types. However, there is no concept of attribute. Instead, there are label types which are types of label or values associated with each entity type. Fig. 1. shows an entity type STUDENT and its associated label types SNUMBER and SNAME. The distinction between object (entity) and value (label) is very clear.

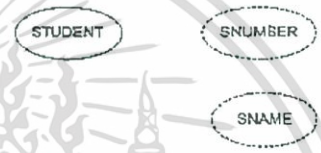


Fig. 1. An Entity Type and its Label Type.

A relationship between an entity type and a label type is called a reference type. Each entity type may have more than one reference type associated with it but there must be at least one one-to-one reference type for identification purpose. A selected one-to-one reference type is called the unique identifier. Fig. 2. shows the entity type STUDENT with its unique identifier S# and a many-to-one reference type with the label type SNAME.

Fig. 2. A Reference Type in NIAM.

A fact type is a relationship between entity types. Fact types must be elementary (cannot be further decomposed) since each of it is based on a deep-structured natural language sentence. Fact types and reference types can be collectively called relationship types. Fig. 3. shows a NIAM conceptual schema of a student database.

0-7803-7978-0/03/$17.00 ©2003 IEEE

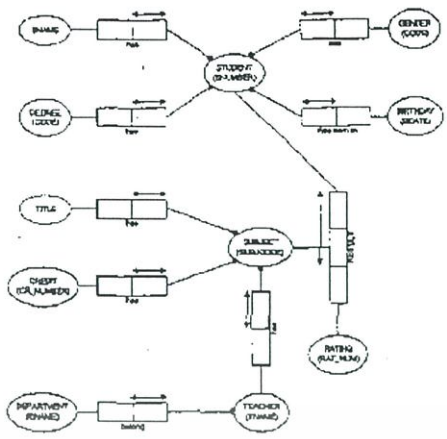


Fig. 3. A Student Conceptual Schema.

III. A Transformation From NIAM Schema to XML Schema

The full transformation steps from a NIAM conceptual schema to object and XML schemas are described in [9]. In this paper the NIAM to OODB transformation steps are summarized as follows:

(1) Each entity type which participated in more than one relationship type is defined to be a main entity type. Each entity type participated in only one relationship type is defined to be a leaf-node entity type.
(2) For each main entity type, create an object class. Many-to-one and one-to-one binary relationship types between leaf-node entity types and the entity type are transformed into simple attributes. Many-to-many binary relationship types are transformed into multivalued attributes (collection types).
(3) Binary fact types between main entity types are transformed into associations between the corresponding object types.
(4) Each n-ary fact type has to be transformed in to a corresponding nested fact type. An object class is created for a nested fact type. Each role involves in a main entity type is transformed into a reference pointer. Each role of a leaf-node entity type is transformed into an attribute.
(5) Entity types and entity subtypes of subtype hierarchies are transformed into corresponding object super classes and object subclasses.

An object database schema shown in Fig. 4. is transformed from the NIAM conceptual schema in Fig.3.

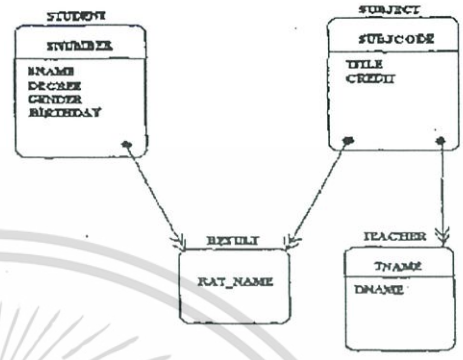


Fig. 4.The Student Object Database

IV. The NIAM Conceptual Meta Schema.

Since inputs from users are NIAM conceptual schemas from users' applications, they need to be kept in the software tool's database meta tables. A conceptual meta schema is required for the design of meta tables. The NIAM conceptual meta schema is a NIAM conceptual schema which describes components of a NIAM conceptual schema. Our NIAM conceptual schema is shown in Fig. 5. The corresponding meta tables in 5NF are shown in Fig. 6.

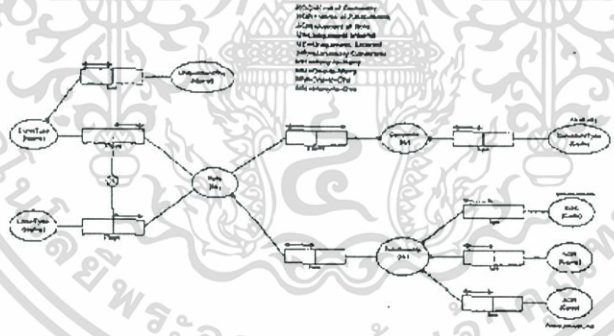


Fig. 5. A NIAM Conceptual Meta Schema

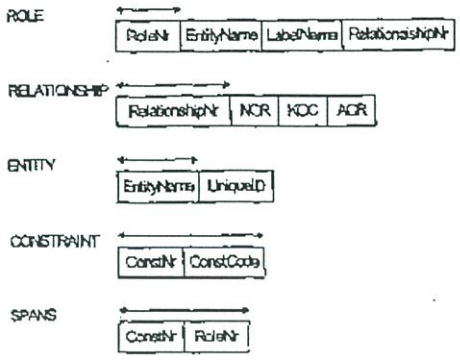


Fig. 6.The Meta Tables

The software tool captures users' conceptual schemas from the screen and populates the details of the conceptual schemas into the meta tables. The details of user's conceptual schema from Fig. 3. are populated in the meta tables as shown in Fig. 7. A display screen of the software tool that shows an output result in XML is shown in Fig. 8.

ROLE : Table

RoleNr	EntityName	LabelName	RelationshipNr
r1		SNAME	RSH1
r10	SUBJECT		RSH5
r11	SUBJECT		RSH6
r12	SUBJECT		RSH6
r13	SUBJECT		RSH6
r14	CREDIT		RSH7
r15	SUBJECT		RSH7
r16	SUBJECT		RSH8
r17	TEACHER		RSH10
r18	DEPARTMENT		RSH9
r19	TEACHER		RSH9
r2	STUDENT		RSH1
r3	STUDENT		RSH2
r4	GENDER		RSH2
r5	DEGREE		RSH3
r6	STUDENT		RSH3
r7	STUDENT		RSH4
r8	BIRTHDAY		RSH4
r9		TITLE	RSH5

RELATIONSHIP : Table

RelationshipNr	NOR	KOC	AOR
RSH1	has	MN	Binary
RSH2	has	MN	Binary
RSH3	has	MN	Binary
RSH4	was born in	MN	Binary
RSH5	has	MN	Binary
RSH6	RESULT	MN	N_ary
RSH7	has	MN	Binary
RSH8	has	MN	Binary
RSH9	belong	MN	Binary

ENTITY : Table

EntityName	UniqueID
BIRTHDAY	BDATE
CREDIT	CR_NUMBER
DEGREE	CODE
DEPARTMENT	DNAME
GENDER	CODE
RATING	RAT_NUM
STUDENT	SNUMBER
SUBJECT	SUBJCODE
TEACHER	TNAME

CONSTRAINT : Table

ConstNr	ConstCode
c1	UI
c10	MR
c11	MR
c12	MR
c13	MR
c14	MR
c15	MR
c16	MR
c17	MR
c2	UI
c3	UI
c4	UI
c5	UI
c6	UI
c7	UI
c8	UI
c9	UI

SPANS : Table

ConstNr	RoleNr
c1	r2
c10	r2
c11	r3
c12	r6
c13	r7
c14	r10
c15	r15
c16	r16
c17	r19
c2	r3
c3	r6
c4	r7
c5	r10
c6	r11
c6	r12
c7	r15
c8	r16
c9	r19

Fig. 7. Populated Meta Tables.

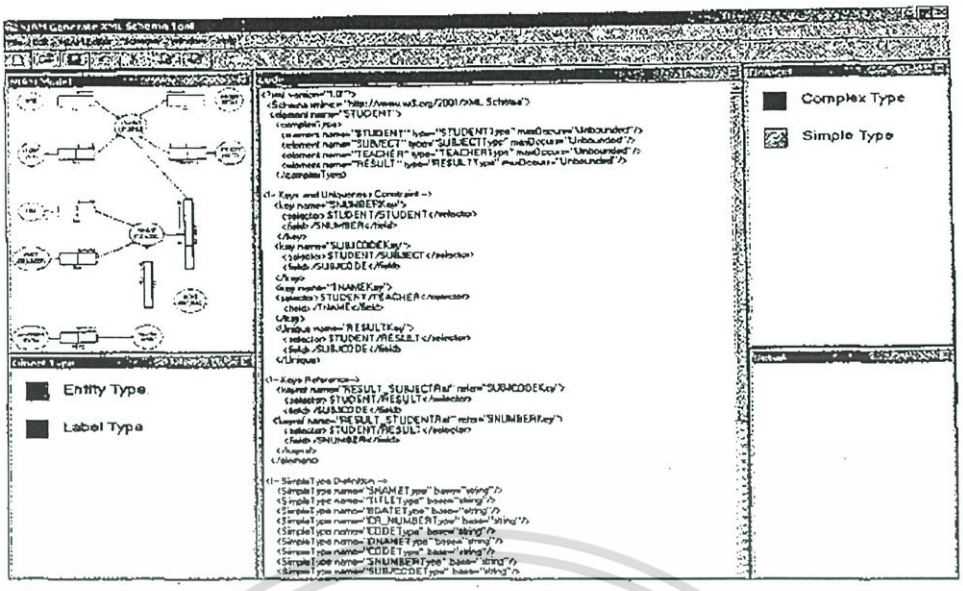


Fig. 8. A Sample Output Screen

V. Conclusion

This research project presents the development of a software tool for object schema generation without redundancies within object classes. The tool also serves as a conceptual framework for an XML editor. Instead of allowing users to directly enter XML schema and population, NIAM is used as a conceptual framework for XML schema generation and the instances can then be entered into the editor based on the generated schema.

VI. References

[1] Chen,P.P. "The Entity-Relationship Model Toward a Unified View of Data." ACM Transaction on Database System 1976;Vol 1 No 1.

[2] Larman C., *Applying UML and patterns: an introduction to object-oriented analysis and design*, Prentice Hall, 1998

[3] W3C. *XML-Schema Part 0:* Primer. Available at:http://www.w3.org/TR/xmlschema-0/

[4] W3C. *XML-Schema Part 1:* Structures. Available at:http://www.w3.org/TR/xmlschema-1/

[5] W3C. *XML-Schema Part 2:* Available at:http://www.w3.org/TR/xmlschema-2/

[6] *XMLSPY3 Enterprise Edition*(Version 5 rel.3) Copy right, 1998-2002 Altova GmbH.

[7] Turbo XML 2.3.1.100/JRE 1.3.1-02b02 Sun Microsoft systems Inc.

[8] Oxygen XML Editor V.2.0.1(Xerces-Java2.5.0) Copyright(C)2002-2003 SyncRO Soft Ltd.

[9] N. Chankuang, "NIAM Conceptual Schema Based Software Tool for Object and XML Schemas Generation," M.S. thesis Dept. Computer Engineering Faculty of Engineering, King Mongkut's Institute of Technology, Ladkrabang to be Submit 2003.

Call for Papers

International Conference on Information Technology: Coding and Computing

ITCC 2004

April 5-7, 2004

The Orleans [email], Las Vegas, Nevada

Sponsored by IEEE Computer Society

NEW: DEADLINE EXTENSION: November 21, 2003

Authors of all submitted papers to this track, are kindly requested to send via email details of the submission (paper ID and title) to the Track Chair (cannataro@unicz.it).

Track on
Modern Web and Grid Systems

An Object and XML Database Schemas Design Tool

Narudol Chankuang, Suphamit Chittayasothorn
Department of Computer Engineering
Faculty of Engineering, King Mongkut's Institute of Technology, Ladkrabang
Bangkok, Thailand 10520
Email: s4061635@kmitl.ac.th , suphamit@kmitl.ac.th

Abstract.

Nowadays relational database schemas are designed by using well-known database design techniques such as the Entity Relationship Model and the normalization process. The result schemas can be guaranteed to have minimum redundancies if the fifth normal form (5NF) is achieved. However, in more recent database schema design such as the design of object database schema, the concern about minimum redundancies does not seem to be an important issue. Functional dependencies may still appear in an object class of the class diagram thus introduce update anomalies. This paper first presents the use of NIAM, a well-established conceptual schema model, as a conceptual model for the design of object databases. A transformation from a NIAM to an OODB schema with minimum redundancy is presented. The conceptual schema can also be transformed into Extensible Markup Language (XML) which is originally a language for document management. However, it now gains popularity in database representation. It is particularly useful as a data format when an application must communicate with another application. This paper also presents the NIAM conceptual schema model as a conceptual design tool for XML schema. A software tool that allows users to create NIAM schemas and generate object and XML schemas is developed.

1. Introduction

A conceptual schema is a representation of a universe of discourse.[1] This is a conceptual framework of an information base system. In principle, a conceptual schema should capture all fact types, rules and integrity constraints. One of the more popular conceptual schema model is NIAM (Nijssen's Information Analysis Methodology) [2]. A strong point about NIAM is that there is a transformation algorithm that transforms a NIAM conceptual schema into the fifth normal form (5NF) relational schemas [3,4]. An extension of NIAM called the Object Role Model is now supported by Microsoft Visio [4].

At present, object databases become common in real life applications. Existing object database design techniques such as the class diagram of the UML [5,6] is a popular design tool for object databases. However, it does not pay attention to details such as functional dependencies between attributes. The result is that an object database designed by this technique may still have redundancies.

To facilitate object database schema design without redundancies, recent extensions to NIAM such as the OONIAM [7] and ENIAM [8] have been proposed. These extensions proposed the separation of the conceptual schema into a main schema and several sub schemas. The extensions ease transformation from the conceptual schemas to object database schemas. However, object database designers who are familiar with the original NIAM and already have NIAM conceptual schemas of their existing information systems prefer to have a transformation direct from the NIAM schemas to object database schemas instead of adopting an extended version of NIAM.

The first part of this paper presents a transformation from the original version of the NIAM conceptual schema to an object database schema.

Another potential application of the NIAM conceptual schema model is its usage as a conceptual framework for XML schema designers. Nowadays, the XML (Extensible Markup Language) [9,10,11] becomes a popular data transfer language between data sources. An XML schema defines the structure of XML documents. It is human readable but not very easy to understand especially for complex schemas.

Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'04)
0-7695-2108-8/04 $ 20.00 © 2004 IEEE

Software tools provide graphical editing facilities are available commercially[12]. These editors use a tree structure to represent the XML schemas without any conceptual frameworks. Most research on XML has neglected the data modeling aspects of XML schemas. Recently, there is an attempt to formalized a core set of features found in various XML schema languages into Xgrammar [13] and transforms an XML schema corresponds with the XGrammar into an ER diagram[14]. However, the ER model does not shows relationships between attributes. Dependencies between attributes may still exists thus cause redundancies in the XML schemas. The second part of this paper presents the use of the NIAM conceptual schema model as a conceptual framework for an XML schema and provides a transformation from NIAM to XML database schema.

The third part of the paper introduces a software tool that allows users to enter NIAM conceptual schemas and generate corresponding object database schemas (at present, only in Cache OODBMS [15] syntax) and XML schemas which guarantee minimum redundancy. The tool also has an XML editor feature that allows users to create XML documents according to the generated XML Schemas.

2. NIAM

NIAM is a fact-oriented conceptual schema model. Main concepts in NIAM include entity types, label types (or value types), fact types and reference types (or bridge type). Similar to other semantic data model, an entity in NIAM is an object of interest. Entities which have the same properties form entity types. However, there is no concept of attribute. Instead, there are label types which are types of label or values associated with each entity type. Fig. 1 shows an entity type Supplier and its associated label types Snumber and Sname. The distinction between object (entity) and value (label) is very clear.

Figure 1. An entity type and its label types.

A relationship between an entity type and a label type is called a reference type. Each entity type may have more than one reference type associated with it but there must be at least one one-to-one reference type for identification purpose. A selected one-to-one reference type is called the unique identifier. Fig. 2 shows the entity type Supplier with its unique identifier Snumber and a many-to-one reference type with the label type Sname.

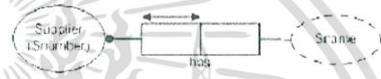


Figure 2. A reference type in NIAM

A fact type is a relationship between entity types. Fact types must be elementary (cannot be further decomposed) since each of it is based on a deep-structured natural language sentence. Fact types and reference types can be collectively called relationship types. Fig. 3 shows a NIAM conceptual schema of the popular Supplier-Part database [16].

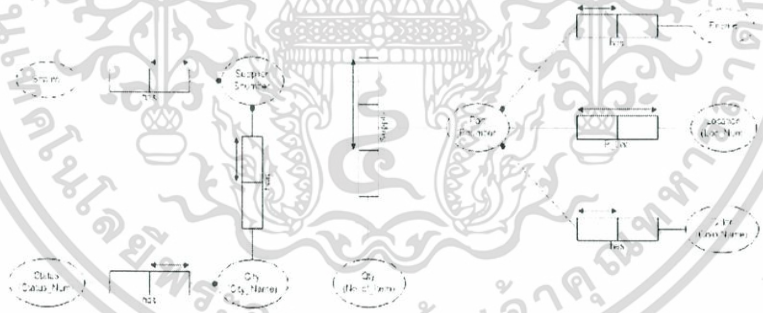


Figure 3. A NIAM conceptual schema of the Supplier-Part database.

Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'04)
0-7695-2108-8/04 $ 20.00 © 2004 IEEE

3. A Transformation from NIAM to Object Database Schema.

Since the concept of entity type in NIAM is slightly different from the concept of entity type in the ER diagram and object class in the UML class diagram, main entity types which are to be transformed into object classes must be identified first. Once, they are identified, fact types and reference types associated with them form attributes of the corresponding object classes. Many-to-one and one-to-one relationship types becomes simple attributes and many-to-many relationship types becomes multivalued collection types. Fact types among the main entity types are then transformed into associations between object classes. The transformation steps are summarized as follows:

(1) Each entity type which participated in more than one relationship type is defined to be a main entity type. Each entity type participated in only one relationship type is defined to be a leaf-node entity type.
(2) For each main entity type, create an object class. Many-to-one and one-to-one binary relationship types between leaf-node entity types and the entity type are transformed into simple attributes. Many-to-many binary relationship types are transformed into multivalued attributes (collection types).
(3) Binary fact types between main entity types are transformed into associations between the corresponding object types.
(4) Each n-ary fact type has to be transformed in to a corresponding nested fact type. An object class is created for a nested fact type. Each role involves in a main entity type is transformed into a reference pointer. Each role of a leaf-node entity type is transformed into an attribute.
(5) Entity types and entity subtypes of subtype hierarchies are transformed into corresponding object super classes and object subclasses.

An object database schema in Cache Object Definition Language shown in Fig. 4 is transformed from the NIAM conceptual schema in Fig.3. Note that the object class City comes out as a separate class. Using other database design methodologies, City and Status could be attributes of the entity type Supplier. Functional dependency between them could cause redundancies in the database.

```
Class SupplyPart.Supplier Extends %Persistent [
ClassType = persistent, ProcedureBlock ]
{
    Property Snumber As %String;
    Property Sname As %String;
    Relationship TheCity As City [ Cardinality = many,
Inverse = TheSupplier ];
    Relationship TheSupply As Supply [ Cardinality =
many, Inverse = TheSupplier ];
}

Class SupplyPart.Part Extends %Persistent [ ClassType =
  persistent, ProcedureBlock ]
{
    Property Pnumber As %String;
    Property Pname As %String;
    Property Color As %String;
    Property Location As %String [ Collection = array ];
    Relationship TheSupply As Supply [ Cardinality =
many, Inverse = ThePart ];
}

Class SupplyPart.City Extends %Persistent [ ClassType =
  persistent, ProcedureBlock ]
{
    Property City As %String;
    Property Status As %String;
    Relationship TheSupplier As Supplier [ Cardinality =
  one, Inverse = TheCity ];
}

Class SupplyPart.Supply Extends %Persistent [
ClassType = persistent, ProcedureBlock ]
{
    Property Qty As %Integer;
    Relationship ThePart As Part [ Cardinality = many,
Inverse = TheSupply ];
    Relationship TheSupplier As Supplier [ Cardinality =
  many, Inverse = TheSuppy ];
}
```

Figure 4. The Supplier-Part Object Database Schema

4. A Transformation from NIAM Schema to XML Schema

Following a similar approach from the above transformation from NIAM to OODB schema, a transformation from a NIAM schema to an XML schema is presented. In practice, a NIAM conceptual schema will be created first. It will then be transformed into a corresponding XML schema. Users who enter data instances direct via an editor now have a well-structured XML schema to follow. This practice is a much better practice than creating XML schema at will without any conceptual frameworks. The transformation steps are as follow.

(1) Specify a root element name of the XML schema.

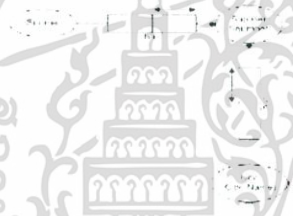
Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'04)
0-7695-2108-8/04 $ 20.00 © 2004 IEEE

(2) Each label type and leaf-node entity type is transformed into an XML simple type.

```
<simpleType name="SnumberType">
  <sp:restriction base="sp:string">
    <sp:length value="5" />
    <sp:pattern value="[A-Z]{1}[0-9]{3}" />
  </sp:restriction>
</sp:simpleType>
<simpleType name="SnameType">
  <sp:restriction base="sp:string">
    <sp:pattern value="[A-Z][a-z]*" />
  </sp:restriction>
</sp:simpleType>
```

Figure 5. An entity type SUPPLIER, its value types and its corresponding XML schema.

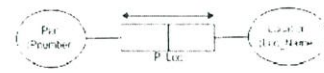
(3) Each main entity type is transformed into an XML complex type. Many-to-one and one-to-one binary relationship types that the main entity type is involved transform into elements of the complex type. Each corresponding element name is either the unique identifier of participating entity types or label types.



```
<complexType name="SupplierType">
  <element name="Snumber" type="SnumberType" />
  <element name="Sname" type="SnameType" />
  <element name="City_Name" type="City_NameType" />
</complexType>
```

Figure 6. A NIAM schema with many-to-one relationships and a corresponding XML schema.

(4) Each binary many-to-many fact type is transformed into an XML complex type.



```
<complexType name="P_LocType">
  <element name="Pnumber" type="PnumberType" />
  <element name="Loc_Num" type="Loc_NumType" />
</complexType>
```

Figure 7. A NIAM schema with many-to-many relationships and a corresponding XML schema

(5) Each n-ary fact type is transformed into an XML complex type.

```
<complexType name="SupplyType">
  <element name="Snumber" type="SnumberType" />
  <element name="Pnumber" type="PnumberType" />
  <element name="No_of_Item" type="No_of_ItemType" />
</complexType>
```

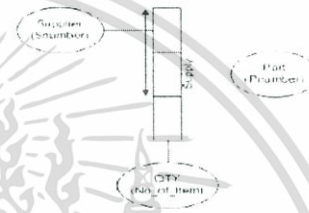


Figure 8. A NIAM schema with a ternary fact type and a corresponding XML schema

(6) The unique identifier of each main entity type is transformed into an XML schema key.

```
<key name="SnumberKey">
  <selector="Supplier Part Supplier" selector />
  <field="Snumber" field />
</key>
```

(7) The uniqueness constraint on roles of a binary many-to-many fact type or an n-ary fact type is transformed into an XML unique constraint.

```
<unique name="SupplyKey">
  <selector="Supplier Part Sppher" selector />
  <field="Snumber" field />
  <field="Pnumber" field />
</unique>
```

Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'04)
0-7695-2108-8/04 $ 20.00 © 2004 IEEE

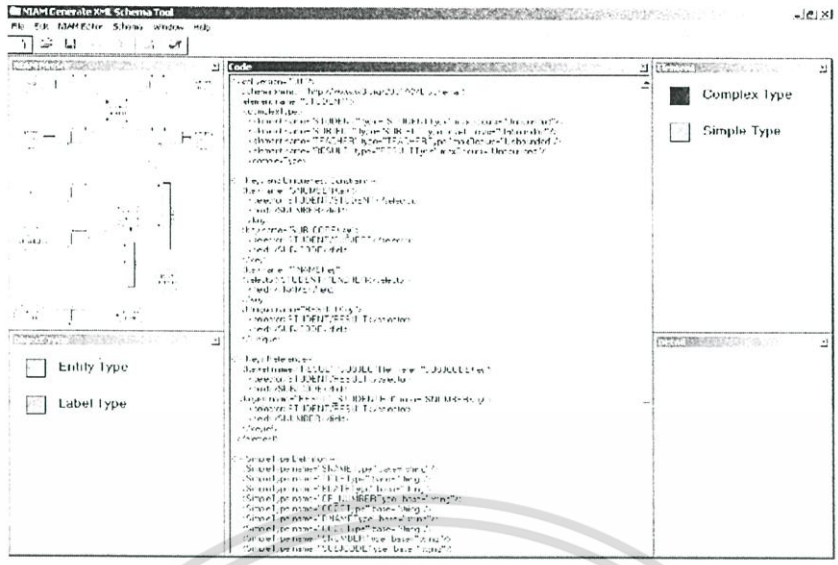


Figure 9. A sample screen of the software tool

(8) Fact types between main entity types form XML schema key references.

```
<keyref name="Supply_SupplierRef1" refer
"SnumberKey">
  <selector>Supplier_Part_Supply</selector>
  <field>./Snumber</field>
</keyref>
<keyref name="Supply_SupplierRef2" refer
"PnumberKey">
  <selector>Supplier_Part_Supply</selector>
  <field>./Pnumber</field>
</keyref>
```

5. A NIAM-based Software Tool

A software tool that allowed the user to create and edit NIAM conceptual schemas was developed. The user may choose to generate 5NF relational database schemas, object database schemas or XML schemas. The generation of object and XML schemas follows the transformations discussed in previous sections. An XML editing screen is provided for users who want to create XML documents according to the generated schemas. An example screen of the tool is shown in Fig. 9.

6. Conclusion

The NIAM conceptual schema model is a useful tool for conceptual database design. The original transformation from a NIAM schema to 5NF relational schema proved to be very useful in real life applications. Our proposed transformation to object and XML schemas will extend its popularity towards modern database applications.

7. References

[1] Van Griethuysen, J. J.(ed) (1982), "Concepts and Terminology for the Conceptual Schema and the information base", ISO Report TC97 SC5/WG3.

[2] Nijssen, G.M.& Halpin, T.A. *Conceptual Schema and Relational Database Design*. Prentice Hall, 1989.

[3] C. M. R. Leung, G.M. Nijssen, "From a NIAM Conceptual Schema into the optimal SQL Relational

Database Schema" Australian Computer Journal Volume19(2), 69-75 (1987)

[4] Halpin, T. *Information Modeling and Relational Database*, Morgan Kuafmann, April 2001

[5] Muller P and Muller A., *Insant UML*, Wrox Press Ltd. 1997

[6] Larman C., *Applying UML and patterns an introduction to object-oriented analysis and design*, Prentice Hall, 1998.

[7] Pasaya, B.; Chittayasothorn, S., "A Temporal Object Oriented Conceptual Schema Model" Proc. PACRIM. 2001 IEEE Pacific Rim Conference, Volume: 2 , 2001,Page(s): 724 -727 vol.2

[8] Puntheeranurak, S ; Chittayasothorn, S., "An Extended NIAM Conceptual Schema for Object Databases" Proc. of the 24th International Conference on Information Technology Interfaces, 2002. ITI2002. Cavtat, Croatia, June 24-27,2002,Page(s):57-61 vlo.1

[9] W3C. *XML Schema Part 0*:Primer.Available at:http://www.w3.org/TR/xmlschema-0

[10] W3C. *XML-Schema Part 1*:Structures. Available at:http://www.w3.org/TR/xmlschema-1

[11] W3C. *XML-Schema Part 2*: Available at: http://www.w3.org/TR/xmlschema-2

[12] *XMLSPY5 Enterprise Edition*(Version 5 rel.3) Copyright, 1998-2002 Altova GmbH.

[13] Mani, M., Lee, D. and Muntz, M., "Semantic Data Modeling using XML Schemas" Proc. Of the 20th International Conference on Conceptual Modeling (ER), Yogohama, Japan, 2001.

[14] Chen, P.P., "The Entity Relationship Model", ACM Transactions on Database Systems, Vol 1, 1976.

[15] Intersystems Corporation, Cache' Post-Relational Database Advanced Management System Guide, 2003.

[16] C.J. *Date, An Introduction to Database Systems*, Seventh Edition, Addison-Wesley Publishing Company, 2000.

8. Appendix: The corresponding XML schema generated from the NIAM conceptual schema of Figure. 3.

```
<?xml version="1.0" encoding="UTF-8"?>
<sp:schema
xmlns:sp="http://www.w3.org/2001/XMLSchema">
<!--                                    -->
<sp:element name="Supply_Part">
    <sp:complexType>
    <sp:sequence>
        <sp:element name="Supplier"
type="SupplierType" maxOccurs="unbounded"/>
        <sp:element name="Part" type="PartType"
maxOccurs="unbounded"/>
        <sp:element name="City" type="CityType"
maxOccurs="unbounded"/>
        <sp:element name="Supply" type="SupplyType"
maxOccurs="unbounded"/>
        <sp:element name="P_Loc" type="P_LocType"
maxOccurs="unbounded"/>
    </sp:sequence>
    </sp:complexType>

<!--                                                    -->
    <sp:key name="SnumberKey">
        <sp:selector xpath="Supplier"/>
        <sp:field xpath="./Snumber"/>
    </sp:key>
    <sp:key name="PnumberKey">
        <sp:selector xpath="Part"/>
        <sp:field xpath="./Pnumber"/>
    </sp:key>
    <sp:key name="City_NameKey">
        <sp:selector xpath="City"/>
        <sp:field xpath="./City_Name"/>
    </sp:key>
    <sp:unique name="SupplyKey">
        <sp:selector xpath="Supply"/>
        <sp:field xpath="./Snumber"/>
        <sp:field xpath="./Pnumber"/>
    </sp:unique>
    <sp:unique name="P_LocKey">
        <sp:selector xpath="P_Loc"/>
        <sp:field xpath="./Pnumber"/>
        <sp:field xpath="./Loc_Num"/>
    </sp:unique>

<!--                                                    -->
    <sp:keyref name="P_Loc_PartRef"
refer="PnumberKey">
        <sp:selector xpath="P_Loc"/>
        <sp:field xpath="./Pnumber"/>
    </sp:keyref>
    <sp:keyref name="Supply_SupplierRef1"
refer="SnumberKey">
        <sp:selector xpath="Supply"/>
        <sp:field xpath="./Snumber"/>
    </sp:keyref>
    <sp:keyref name="Supply_SupplierRef2"
refer="PnumberKey">
        <sp:selector xpath="Supply"/>
        <sp:field xpath="./Pnumber"/>
    </sp:keyref>
    <sp:keyref name="Supplier_CityRef"
refer="City_NameKey">
        <sp:selector xpath="Supplier"/>
        <sp:field xpath="./City_Name"/>
    </sp:keyref>
</sp:element>
<!--                                                    -->
<!--                                                    -->
    <sp:simpleType name="SnumberType">
        <sp:restriction base="sp:string">
```

Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'04)
0-7695-2108-8/04 $ 20.00 © 2004 IEEE

```
        <sp:length value="5"/>
        <sp:pattern value="[A-Z]{1}-\d{3}"/>
        </sp:restriction>
    </sp:simpleType>
    <sp:simpleType name="SnameType">
        <sp:restriction base="sp:string">
        <sp:pattern value="[A-Z][a-z]*"/>
        </sp:restriction>
    </sp:simpleType>
    <sp:simpleType name="City_NameType">
        <sp:restriction base="sp:string">
        <sp:pattern value="[A-Z][a-z]*"/>
        </sp:restriction>
    </sp:simpleType>
    <sp:simpleType name="PnumberType">
        <sp:restriction base="sp:string">
        <sp:pattern value="[A-Z]*-\d*"/>
        </sp:restriction>
    </sp:simpleType>
    <sp:simpleType name="PnameType">
        <sp:restriction base="sp:string">
        <sp:pattern value="[A-Z][a-z]*"/>
        </sp:restriction>
    </sp:simpleType>
    <sp:simpleType name="ColorNameType">
        <sp:restriction base="sp:string"/>
    </sp:simpleType>
    <sp:simpleType name="Status_NumType">
        <sp:restriction base="sp:string"/>
    </sp:simpleType>
    <sp:simpleType name="No_of_ItemType">
        <sp:restriction base="sp:integer"/>
    </sp:simpleType>
    <sp:simpleType name="Loc_NumType">
        <sp:restriction base="sp:string"/>
    </sp:simpleType>

<!--
        -->
    <sp:complexType name="SupplierType">
    <sp:sequence>
    <sp:element name="Snumber"
    type="SnumberType"/>
    <sp:element name="Sname" type="SnameType"/>
    <sp:element name="City_Name"
    type="City_NameType"/>
    </sp:sequence>
    </sp:complexType>

    <sp:complexType name="PartType">
    <sp:sequence>
    <sp:element name="Pnumber"
    type="PnumberType"/>
    <sp:element name="Pname" type="PnameType"/>
    <sp:element name="ColorName"
    type="ColorNameType"/>
    </sp:sequence>
    </sp:complexType>
    <sp:complexType name="CityType">
    <sp:sequence>
    <sp:element name="City_Name"
    type="City_NameType"/>
    <sp:element name="Status_Num"
    type="Status_NumType"/>
    </sp:sequence>
    </sp:complexType>
<!--
                                        -->
    <sp:complexType name="SupplyType">
    <sp:sequence>
    <sp:element name="Snumber"
    type="SnumberType"/>
    <sp:element name="Pnumber"
    type="PnumberType"/>
    <sp:element name="No_of_Item"
    type="No_of_ItemType"/>
    </sp:sequence>
    </sp:complexType>

<!--
                                        -->
    <sp:complexType name="P_LocType">
    <sp:sequence>
    <sp:element name="Pnumber"
    type="PnumberType"/>
    <sp:element name="Loc_Num"
    type="Loc_NumType"/>
    </sp:sequence>
    </sp:complexType>
</sp:schema>
```

Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'04)
0-7695-2108-8/04 $ 20.00 © 2004 IEEE

ประวัติผู้เขียน

ชื่อ-นามสกุล	นายณฤดล จันทร์ควง
วันเดือนปีเกิด	วันที่ 12 มิถุนายน 2515 ที่จังหวัดกำแพงเพชร
ที่อยู่	84/1 หมู่ 7 ต.ธำมรงค์ อ.เมือง จ.กำแพงเพชร
ประวัติการศึกษา	2540 สำเร็จการศึกษาวิศวกรรมศาตรบัณฑิตสาขาวิศวกรรมระบบวัดและควบคุม คณะวิศวกรรมศาสตร์มหาวิทยาลัยเทคโนโลยีมหานคร

ประสบการณ์ทำงาน

พ.ศ. 2538-2542	ผู้บริหารโรงงานบริษัทในเครือซิเมนต์ไทย
พ.ศ. 2547-ปัจจุบัน	ผู้จัดการโรงพิมพ์เครื่องเนชั่น มัลติมิเดีย กรุ๊ป จำกัด(มหาชน)