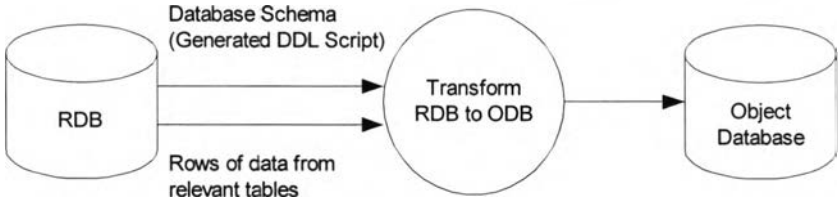
บทที่ 3

การวิเคราะห์และออกแบบระบบแปลงฐานข้อมูลเชิงสัมพันธ์เป็นฐานข้อมูลเชิงวัตถุ

บทนี้จะกล่าวถึงขั้นตอนการวิเคราะห์และออกแบบระบบแปลงฐานข้อมูลเชิงสัมพันธ์เป็นฐานข้อมูลเชิงวัตถุ โดยแยกเป็นหัวข้อหลักๆดังนี้

- แนวคิดในการออกแบบ
- วิเคราะห์องค์ประกอบสคริพท์
- การแปลงแผนแบบเชิงสัมพันธ์เป็นแผนแบบเชิงวัตถุ
- การออกแบบโปรแกรมแปลงฐานข้อมูลเชิงสัมพันธ์เป็นฐานข้อมูลเชิงวัตถุ

3.1 แนวคิดในการออกแบบ



รูปที่ 3.1 Context Diagram แสดงการแปลงฐานข้อมูลเชิงสัมพันธ์เป็นฐานข้อมูลเชิงวัตถุ

รูปที่ 3.1 แสดงแนวคิดในการออกแบบระบบแปลงฐานข้อมูลเชิงสัมพันธ์เป็นฐานข้อมูลเชิงวัตถุ

ข้อมูลที่ใช้เป็นข้อมูลเข้าในโพรเซส Transform RDB to ODB ประกอบด้วย 2 ส่วนคือ

- สคริพท์ (Script) สำหรับการสร้างตารางข้อมูลบนระบบฐานข้อมูลเชิงสัมพันธ์ สคริพท์เหล่านี้ได้มาจากการสั่งให้ RDBMS สร้างให้โดยตรงจากฐานข้อมูลที่มีอยู่แล้วในระบบจัดการฐานข้อมูลเชิงสัมพันธ์
- ข้อมูลในแต่ละตารางของฐานข้อมูลเชิงสัมพันธ์

3.2 องค์ประกอบสคริพท์

สคริพท์ที่ RDBMS สร้างให้ประกอบด้วยชุดคำสั่งสำหรับการสร้างและแก้ไขโครงสร้างฐานข้อมูลเชิงสัมพันธ์ ประกอบด้วยคำสั่งหลักๆดังนี้

- CREATE TABLE : เป็นคำสั่งที่ใช้ในการกำหนดโครงสร้างของตารางมีรูปแบบดังนี้

```
CREATE TABLE <table_name>
( <column name> <data_type> [<size>] [[constraint <constraint_name> constraint_type]
[,<column_name> <data type< [<size>], ....]);
```

- ALTER TABLE : เป็นคำสั่งที่ใช้สำหรับเปลี่ยนโครงสร้างของตารางมีรูปแบบดังนี้

```
ALTER TABLE <table_name>
ADD (<column_name> <data_type> [<size>] [[constraint <constraint_name>
constraint_type]);
```

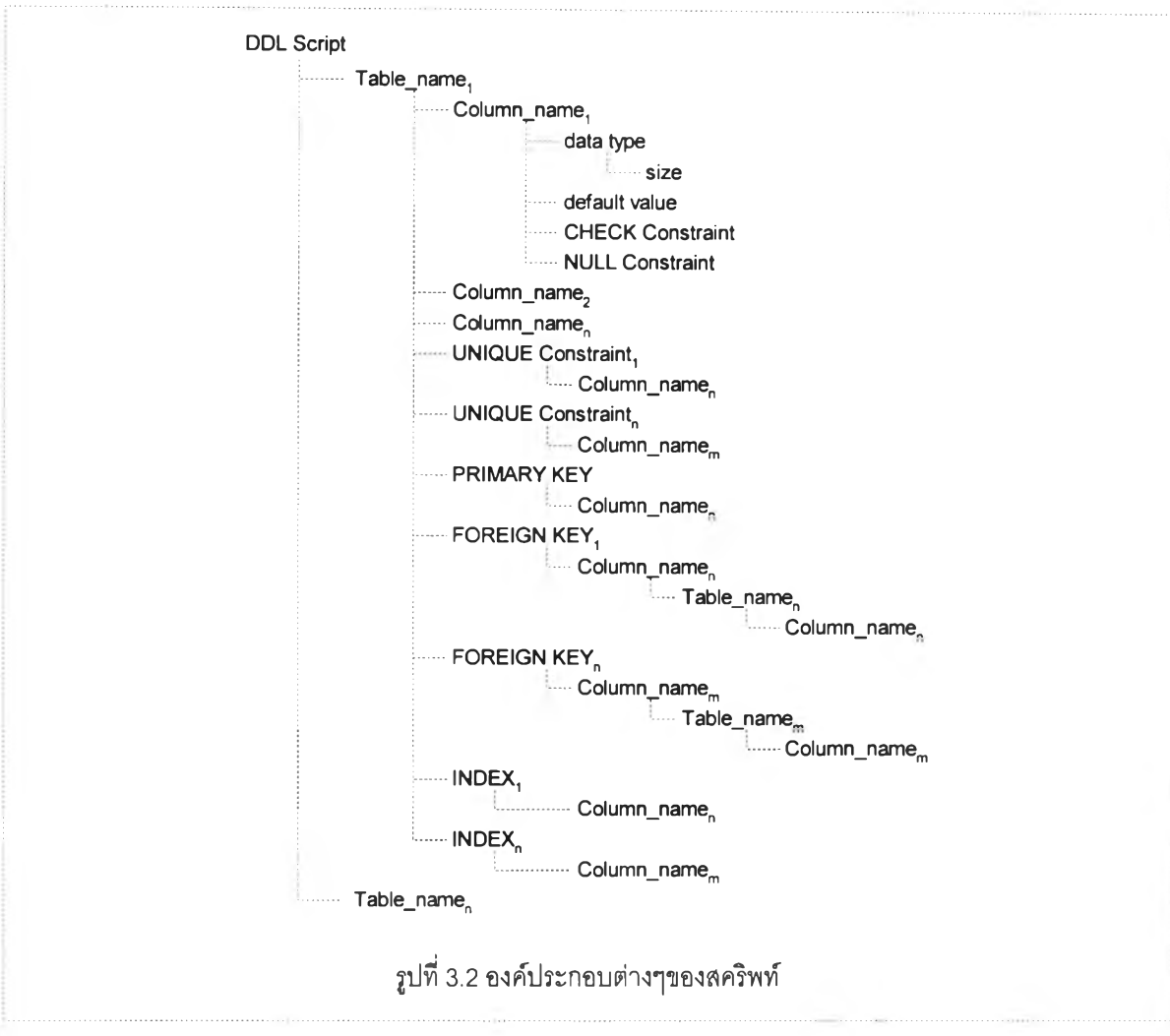
หรือ

```
ALTER TABLE <table_name>
 ADD CONSTRAINT <constraint_name> constraint_type (column_name);
```

- CREATE INDEX : เป็นคำสั่งที่ใช้ในการสร้างดรรชนีให้กับตารางมีรูปแบบดังนี้

```
CREATE INDEX <index_name>
  ON <table_name> (<column_name> [,<column_name>]...);
```

- จากการวิเคราะห์สคริพท์ สามารถนำมาสร้างต้นไม้แสดงรายละเอียดความสัมพันธ์ของตารางดังรูปที่ 3.2



รูปที่ 3.2 องค์ประกอบต่างๆของสคริพท์

รูปที่ 3.2 แสดงองค์ประกอบต่างๆของตารางที่ได้จากการวิเคราะห์สคริพท์ ทำให้ทราบว่าแต่ละตารางประกอบด้วยส่วนประกอบหลักๆดังนี้

ก.) คอลัมน์ ประกอบด้วยส่วนประกอบย่อยดังนี้

- แบบชนิดข้อมูล (Data Type) คือประเภทข้อมูลสำหรับข้อมูลซึ่งเก็บในคอลัมน์นั้นๆ
- ค่าโดยปริยาย (Default Value) คือค่าซึ่งถูกกำหนดให้โดยอัตโนมัติเมื่อคอลัมน์นั้นไม่ได้รับการป้อนค่าให้โดยตรง
- ข้อบังคับการตรวจสอบ (Check Constraint) คือข้อกำหนดการตรวจสอบข้อมูลสำหรับคอลัมน์นั้น
- ข้อบังคับว่าง (Null Constraint) คือข้อกำหนดระบุว่าข้อมูลในคอลัมน์นั้นมีค่าเป็นว่างได้หรือไม่

ข.) คีย์หลัก (Primary Key) คือคีย์ซึ่งแสดงว่ารายการข้อมูลรายการนั้น มีรายการเดียวในตารางนั้นๆ

ค.) ข้อบังคับการมีอยู่อันเดียว (Unique Constraint) คือการกำหนดให้ข้อมูลในคอลัมน์ใด คอลัมน์หนึ่งมีค่าไม่ซ้ำกัน

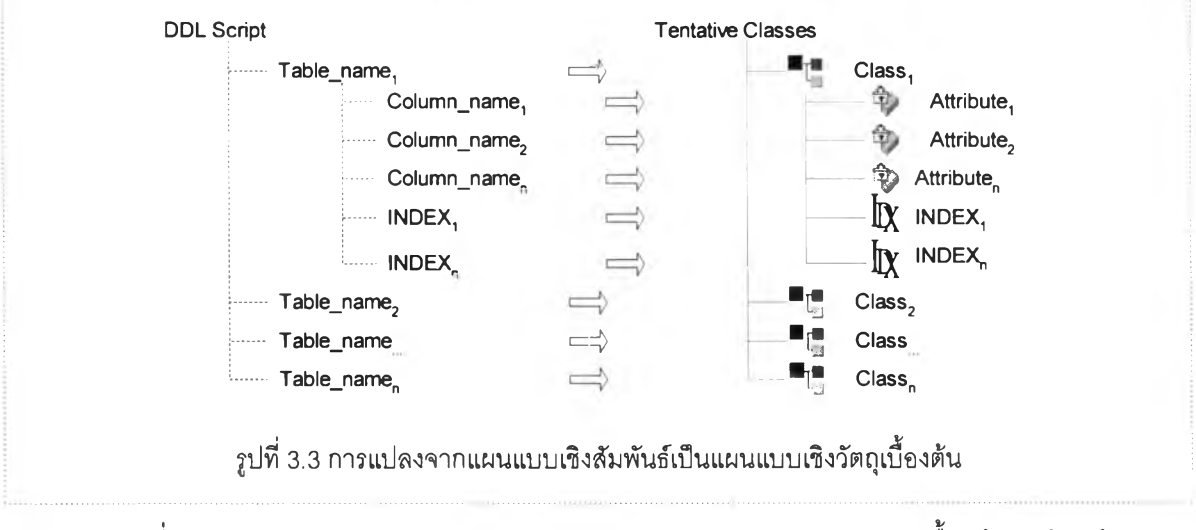
ง.) คีย์นอก (Foreign Key) คือคีย์ซึ่งบอกว่าค่าในคอลัมน์นั้นอ้างอิงไปยังคีย์หลัก ของตารางอื่น

จ.) ดรรชนี (Index) คือการกำหนดการจัดเรียงข้อมูลในตาราง เพื่อช่วยให้การค้นหาข้อมูลในตารางเร็วขึ้น

คีย์นอกของแต่ละตารางจะบอกว่า ตารางนั้นมีความสัมพันธ์กับตารางอื่นโดยแสดงความสัมพันธ์ผ่านคอลัมน์ที่ประกอบเป็นคีย์นอก

จากการวิเคราะห์สคริพท์ที่ได้ สิ่งที่ต้องทำเบื้องต้นสำหรับการออกแบบระบบแปลงฐานข้อมูลเชิงสัมพันธ์เป็นฐานข้อมูลเชิงวัตถุคือ

- สร้างโพรเซสสำหรับการกระจายสคริพท์ เพื่อให้ได้รายละเอียดที่จำเป็นดังรูปที่ 3.2 ซึ่งจะใช้สำหรับโพรเซสอื่นต่อไป
- สร้างโพรเซสสำหรับการแปลงจากแผนแบบเชิงสัมพันธ์เป็นแผนแบบเชิงวัตถุเบื้องต้น โดยอาศัยผลลัพธ์ที่ได้จากโพรเซสการกระจายสคริพท์ การแปลงในขั้นตอนนี้เป็นการแปลงเบื้องต้น โดยกำหนดให้ 1 ตารางเป็น 1 คลาส 1 คอลัมน์เป็น 1 แอททริบิวท์ นอกจากนี้ยังแปลงดรรชนีในแผนแบบเชิงสัมพันธ์เป็นดรรชนีในแผนแบบเชิงวัตถุ



รูปที่ 3.3 การแปลงจากแผนแบบเชิงสัมพันธ์เป็นแผนแบบเชิงวัตถุเบื้องต้น

รูปที่ 3.3 แสดงการแปลงจากแผนแบบเชิงสัมพันธ์เป็นแผนแบบเชิงวัตถุเบื้องต้น โดยกำหนดให้ 1 ตารางเป็น 1 คลาส แต่ละคอลัมน์ในตารางแปลงเป็นแอททริบิวท์ในคลาส และแปลงดรรชนีเป็นดรรชนีในแผนแบบเชิงวัตถุ

3.3 การแปลงแผนแบบเชิงสัมพันธ์เป็นแผนแบบเชิงวัตถุ

ผลจากการแปลงแผนแบบเชิงสัมพันธ์ เป็นแผนแบบเชิงวัตถุเบื้องต้น โดยอาศัยแนวทางการแปลงแผนแบบเชิงวัตถุเป็นแผนแบบเชิงสัมพันธ์ สามารถนำมาเป็นแนวทางในการปรับแผนแบบเชิงวัตถุเบื้องต้นได้ โดยแยกเป็น 2 หัวข้อคือการทำให้มีลักษณะทั่วไป (Generalization) และความเกี่ยวพัน (Association)

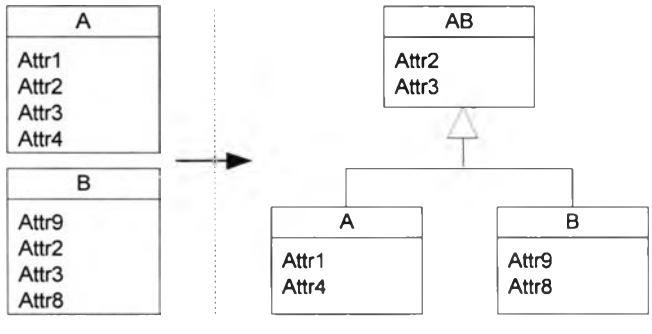
3.3.1 การทำให้มีลักษณะทั่วไป (Generalization)

การค้นหาการสืบทอดคุณสมบัติของคลาส มีหลักในการพิจารณาเป็น 4 กรณีดังนี้

- คลาสซึ่งมีแอททริบิวท์ซ้ำกัน (Replicated Attributes)
- การขึ้นแก่กันทั้งหมด (Inclusion Dependency)
- กรณีคลาส 2 คลาสมีคีย์นอกซึ่งเป็นคีย์หลักชี้ไปยังคลาสที่ 3

3.3.1.1 คลาสซึ่งมีแอททริบิวท์ซ้ำกัน (Replicated Attributes) [2]

ค้นหาคลาสที่มีแอททริบิวท์ซ้ำกันหลายๆแอททริบิวท์ ถ้าพบแสดงว่าคลาสเหล่านี้ เกิดจากการผลักแอททริบิวท์คลาสพ่อลงเป็นแอททริบิวท์คลาสลูก การทำให้มีลักษณะทั่วไปสำหรับคลาสเหล่านี้คือ สร้างคลาสใหม่จากแอททริบิวท์ที่ซ้ำกันและลบแอททริบิวท์ที่ซ้ำกันในคลาสลูก

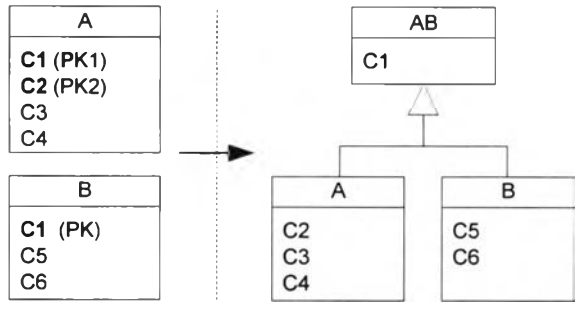


รูปที่ 3.4 การทำให้มีลักษณะทั่วไปสำหรับคลาสที่มีแอททริบิวท์ซ้ำกัน

รูปที่ 3.4 การทำให้มีลักษณะทั่วไปของคลาส A และคลาส B โดยที่ทั้งคลาส A และคลาส B ต่างก็มีแอททริบิวท์ Attr2 และ Attr3 ทำให้สามารถสร้างคลาสใหม่ที่มี Attr2 และ Attr3 เป็นแอททริบิวท์

3.3.1.2 **การขึ้นแก่กันทั้งหมด** (Inclusion Dependency)

เมื่อคีย์หลักของคลาสหนึ่งเป็นซับเซ็ตของคีย์หลักของอีกคลาสหนึ่ง กรณีนี้สามารถทำให้มีลักษณะทั่วไป โดยการสร้างคลาสใหม่เป็นคลาสพ่อของ 2 คลาสนี้ และคลาสใหม่ที่ได้ประกอบด้วยแอททริบิวท์ที่เป็นคีย์หลักซึ่งเป็นซับเซ็ตของคีย์หลักของอีกคลาส

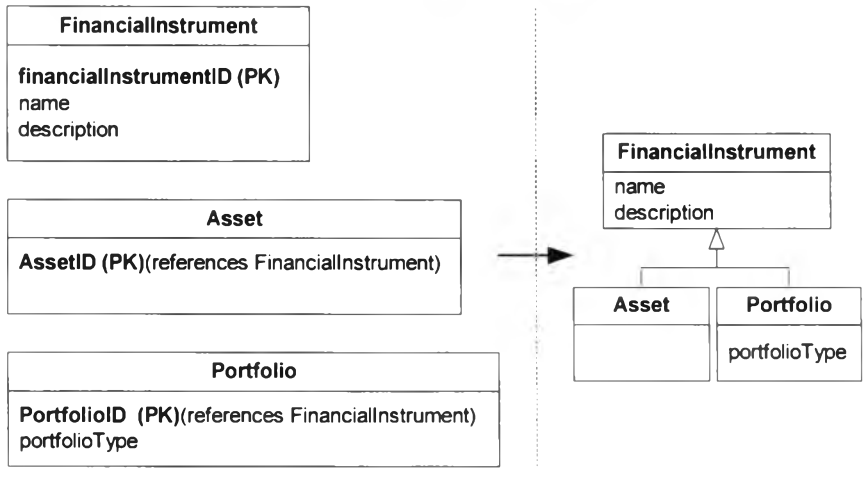


รูปที่ 3.5 การทำให้มีลักษณะทั่วไปสำหรับคลาสที่มีคีย์หลักเป็นซับเซ็ตกัน

รูปที่ 3.5 คลาส A มีแอททริบิวท์ C1 และ C2 ประกอบกันเป็นคีย์หลัก คลาส B มีแอททริบิวท์ C1 เป็นคีย์หลัก ทำให้มีลักษณะทั่วไปโดยการสร้างคลาส AB เป็นคลาสพ่อและคลาส A กับคลาส B เป็นคลาสลูกโดยที่คลาส ABมี C1 เป็นแอททริบิวท์

3.3.1.3 **คลาส 2 คลาสมีคีย์นอกซึ่งเป็นคีย์หลักด้วยชี้ไปยังคลาสที่** 3

คลาส 2 คลาสมีความสัมพันธ์กับคลาสที่ 3 แบบ ONE-TO-ONE โดยที่คีย์หลักเป็นคีย์นอกชี้ไปยังคลาสที่ 3 ความสัมพันธ์ระหว่างคลาสเหล่านี้ เป็นความสัมพันธ์แบบลักษณะทั่วไป โดยที่คลาสที่ 3 เป็นคลาสพ่อ คลาสที่ 1 และคลาสที่ 2 เป็นคลาสลูก



รูปที่ 3.6 การทำให้มีลักษณะทั่วไปสำหรับคลาส 2 คลาสซึ่งมีคีย์หลักเป็นคีย์นอกชี้ไปยังคลาสที่ 3

รูปที่ 3.6 คลาส Asset และคลาส Portfolio ต่างก็มีคีย์หลักเป็นคีย์นอกชี้ไปยังคลาส FinancialInstrument เมื่อผ่านการทำให้มีลักษณะทั่วไปแล้วจะได้คลาส FinancialInstrument เป็นคลาสพ่อและคลาส Asset กับคลาส Portfolio เป็นคลาสลูกดังรูปด้านขวามือ

3.3.2 ความเกี่ยวพันระหว่างคลาส (Association)

ความเกี่ยวพันระหว่างตารางเกิดขึ้นเมื่อตารางต่างๆมีความสัมพันธ์กันผ่านคีย์นอก จำแนกความสัมพันธ์เป็น 3 ประเภทคือ ONE-TO-ONE ONE-TO-MANY และ MANY-TO-MANY

ความสัมพันธ์แบบ ONE-TO-ONE และ ONE-TO-MANY ทราบโดยการวิเคราะห์ข้อมูลจริงในฐานข้อมูล โดยการใช้คำสั่ง Select Statement ช่วยหาจำนวนรายการในตารางข้อมูลเหล่านั้น เช่นสมมติตาราง T1 มีคีย์นอกชี้ไปยังตาราง T2 ถ้าทราบว่า 1 รายการจากตาราง T2 มีจำนวนรายการในตาราง T1 มากกว่า 1 รายการ และรายการ 1 รายการของตาราง T1 มีจำนวนรายการในตาราง T2 ไม่มากกว่า 1 รายการ สรุปได้ว่าความสัมพันธ์ระหว่างตาราง T2 กับตาราง T1 เป็นความสัมพันธ์แบบ ONE-TO-MANY

ถ้า 1 รายการในตาราง T2 มีเพียง 1 รายการในตาราง T1 เท่านั้นและ 1 รายการในตาราง T1 มีเพียง 1 รายการในตาราง T2 แล้วสรุปได้ว่าความสัมพันธ์ระหว่าง T2 กับ T1 เป็นแบบ ONE-TO-ONE

ดังนั้นคลาส 2 คลาสที่เกิดจากตารางที่มีความสัมพันธ์กันผ่านคีย์นอกสามารถสรุปได้ว่ามีความเกี่ยวพันกัน ซึ่งเป็นความเกี่ยวพันแบบ ONE-TO-ONE หรือ ONE-TO-MANY ขึ้นกับผลการวิเคราะห์ข้อมูลในฐานข้อมูลที่เกี่ยวข้อง

อย่างไรก็ตาม ถ้าคลาสตั้งแต่ 2 คลาสขึ้นไปมีคีย์นอก (ซึ่งต่างก็เป็นคีย์หลักของคลาสนั้นๆ) ชี้ไปยังคลาสเดียวกันและความสัมพันธ์ระหว่างคลาสเหล่านั้นกับคลาสที่ถูกอ้างอิงเป็น ONE-TO-ONE แล้วคลาสเหล่านี้กับคลาสที่ถูกอ้างอิงมีการสืบทอดคุณสมบัติกัน โดยที่คลาสที่ถูกอ้างอิงเป็นคลาสพ่อและคลาสที่อ้างอิงเป็นคลาสลูก

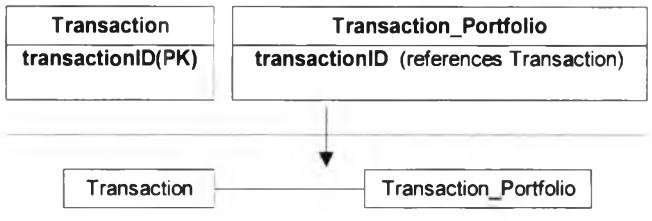
การค้นหาความเกี่ยวพัน (Association) ระหว่างคลาสสามารถจำแนกเป็น 4 กรณีดังนี้

3.3.2.1 **ความเกี่ยวพันระหว่างคลาสแบบที่หนึ่ง**

คลาสที่มีคีย์นอกชี้ไปยังคลาสอื่นโดยที่คีย์นอกนี้ไม่ได้เป็นคีย์หลัก หรือเป็นส่วนใดส่วนหนึ่งของคีย์หลักแล้ว คลาส 2 คลาสนี้มีความเกี่ยวพันระหว่างกัน โดยที่เป็นความสัมพันธ์แบบ ONE-TO-MANY หรือ ONE-TO-ONE ขึ้นกับการวิเคราะห์ข้อมูลจากตารางที่เกี่ยวข้อง

3.3.2.2 **ความเกี่ยวพันระหว่างคลาสแบบที่สอง**

คลาสที่มีคีย์หลักซึ่งเป็นคีย์นอกด้วย แปลงเป็นความเกี่ยวพันระหว่างคลาส เมื่อคลาสที่ถูกอ้างอิงไม่มีคลาสอื่นที่มีความสัมพันธ์ลักษณะเช่นเดียวกันกับคลาสนี้

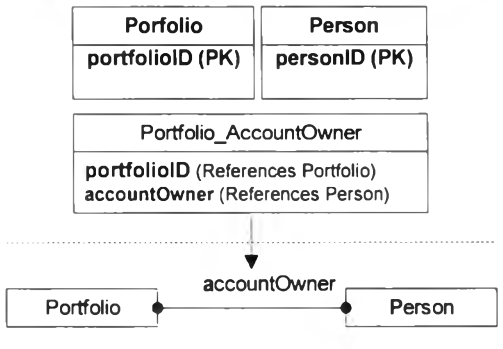


รูปที่ 3.7 การหาความเกี่ยวพันระหว่างคลาสแบบที่สอง

รูปที่ 3.7 คลาส Transaction_Portfolio มี transactionID เป็นคีย์หลักและ transactionID เป็นคีย์นอกชี้ไปยังคลาส Transaction คลาส Transaction มีความเกี่ยวพันกับคลาส Transaction_Portfolio แบบ ONE-TO-ONE

3.3.2.3 **ความเกี่ยวพันระหว่างคลาสแบบที่สาม**

คลาสซึ่งมีคีย์นอก 2 ตัวประกอบเป็นคีย์หลัก และคลาสนี้มีแอททริบิวท์ที่ประกอบเป็นคีย์หลักเท่านั้น ความสัมพันธ์ระหว่างคลาสที่ถูกอ้างอิงโดยคีย์นอก 2 ตัวนี้เป็นความสัมพันธ์แบบ MANY-TO-MANY

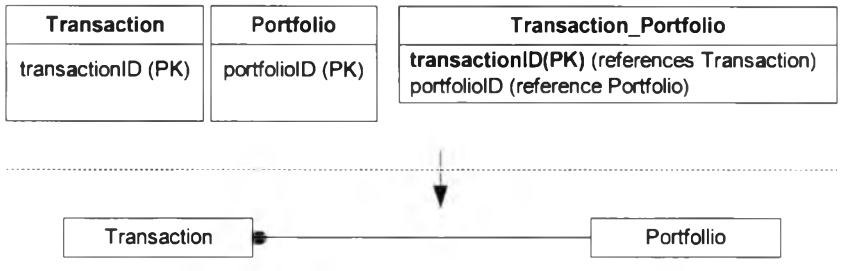


รูปที่ 3.8 การหาความเกี่ยวพันระหว่างคลาสแบบที่สาม

จากรูปที่ 3.8 คลาส Portfolio_AccountOwner มีคีย์หลักซึ่งประกอบด้วยคีย์นอก 2 ตัวชี้ไปยังคลาส Portfolio และคลาส Person จะได้ความสัมพันธ์ระหว่าง Portfolio กับ Person เป็นความเกี่ยวพันแบบ MANY-TO-MANY และเนื่องจากคลาส Portfolio_AccountOwner เป็นคลาสซึ่งประกอบด้วยคีย์นอกเท่านั้นสามารถตัดทิ้งได้ โดยไม่ทำให้ความถูกต้องของแผนแบบเสียไป

3.3.2.4 **ความเกี่ยวพันระหว่างคลาสแบบที่สี่**

คลาสซึ่งประกอบด้วยคีย์นอก 2 ตัวโดยที่คีย์นอกตัวใดตัวหนึ่งเป็นคีย์หลักด้วย และคลาสนี้ประกอบด้วยแอททริบิวท์ที่ประกอบเป็นคีย์นอกเท่านั้น ความสัมพันธ์ระหว่างคลาสที่ถูกอ้างอิงจะเป็นแบบ ONE-TO-MANY โดยที่คลาสที่ถูกอ้างอิงผ่านคีย์หลักจะเป็นความสัมพันธ์ฝั่ง MANY

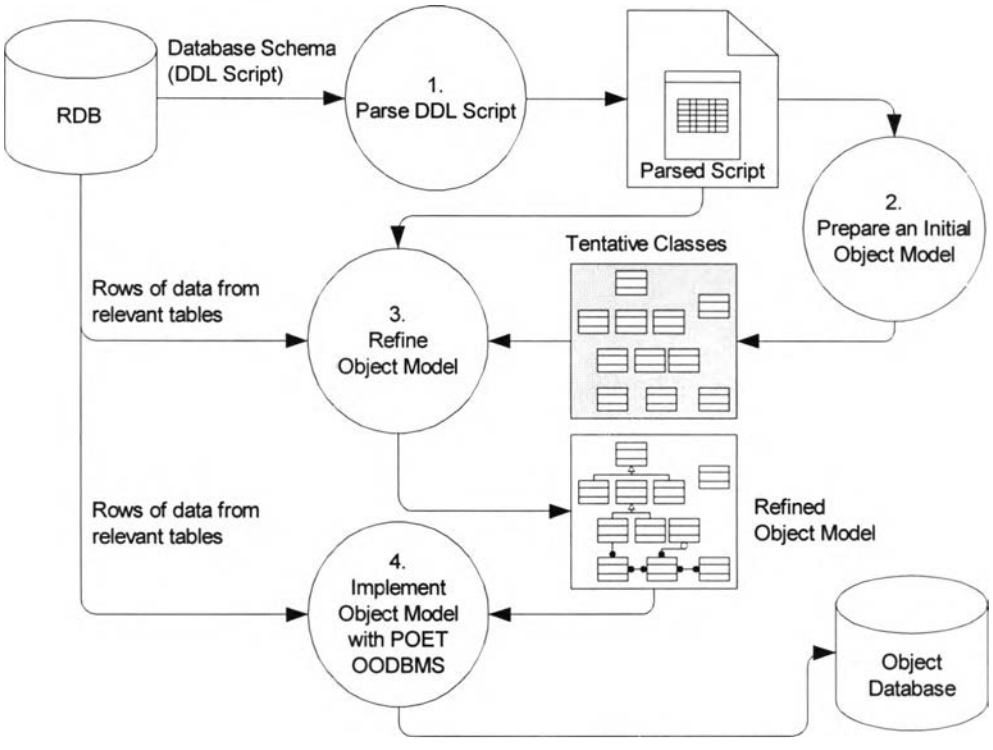


รูปที่ 3.9 การหาความเกี่ยวพันระหว่างคลาสแบบที่สี่

จากรูปที่ 3.9 คลาส Transaction_Portfolio ประกอบด้วยคีย์นอก 2 ตัว คีย์นอกตัวแรกชี้ไปยังคลาส Transaction และ คีย์นอกตัวที่ 2 ชี้ไปยังคลาส Portfolio โดยที่คีย์นอกตัวแรกเป็นคีย์หลักด้วย ถ้าความสัมพันธ์ระหว่าง Portfolio กับ Transaction_portfolio เป็น ONE-TO-MANY แล้วความสัมพันธ์ระหว่าง Portfolio กับ Transaction เป็นความสัมพันธ์แบบ ONE-TO-MANY

3.4 การออกแบบโปรแกรมแปลงฐานข้อมูลเชิงสัมพันธ์เป็นฐานข้อมูลเชิงวัตถุ

จากการศึกษาและวิเคราะห์วิธีการในการแปลงแผนแบบเชิงสัมพันธ์เป็นแผนแบบเชิงวัตถุ โปรแกรมสำหรับแปลงฐานข้อมูลเชิงสัมพันธ์เป็นฐานข้อมูลเชิงวัตถุจะประกอบด้วยโพรเซสหลัก 4 โพรเซสดังแสดงในรูปที่ 3.10



รูปที่ 3.10 ภาพรวมของโปรแกรมแปลงฐานข้อมูลเชิงสัมพันธ์เป็นฐานข้อมูลเชิงวัตถุ

รูปที่ 3.10 แสดงโพรเซสหลักของโปรแกรมแปลงฐานข้อมูลเชิงสัมพันธ์เป็นฐานข้อมูลเชิงวัตถุ ประกอบด้วยโพรเซสหลักๆ 4 โพรเซสดังนี้

- Parse DDL Script
- Prepare an Initial Object Model
- Refine Object Model
- Implement Object Model with POET OODBMS

3.4.1 **กระจายสคริพท์** (Parse DDL Script)

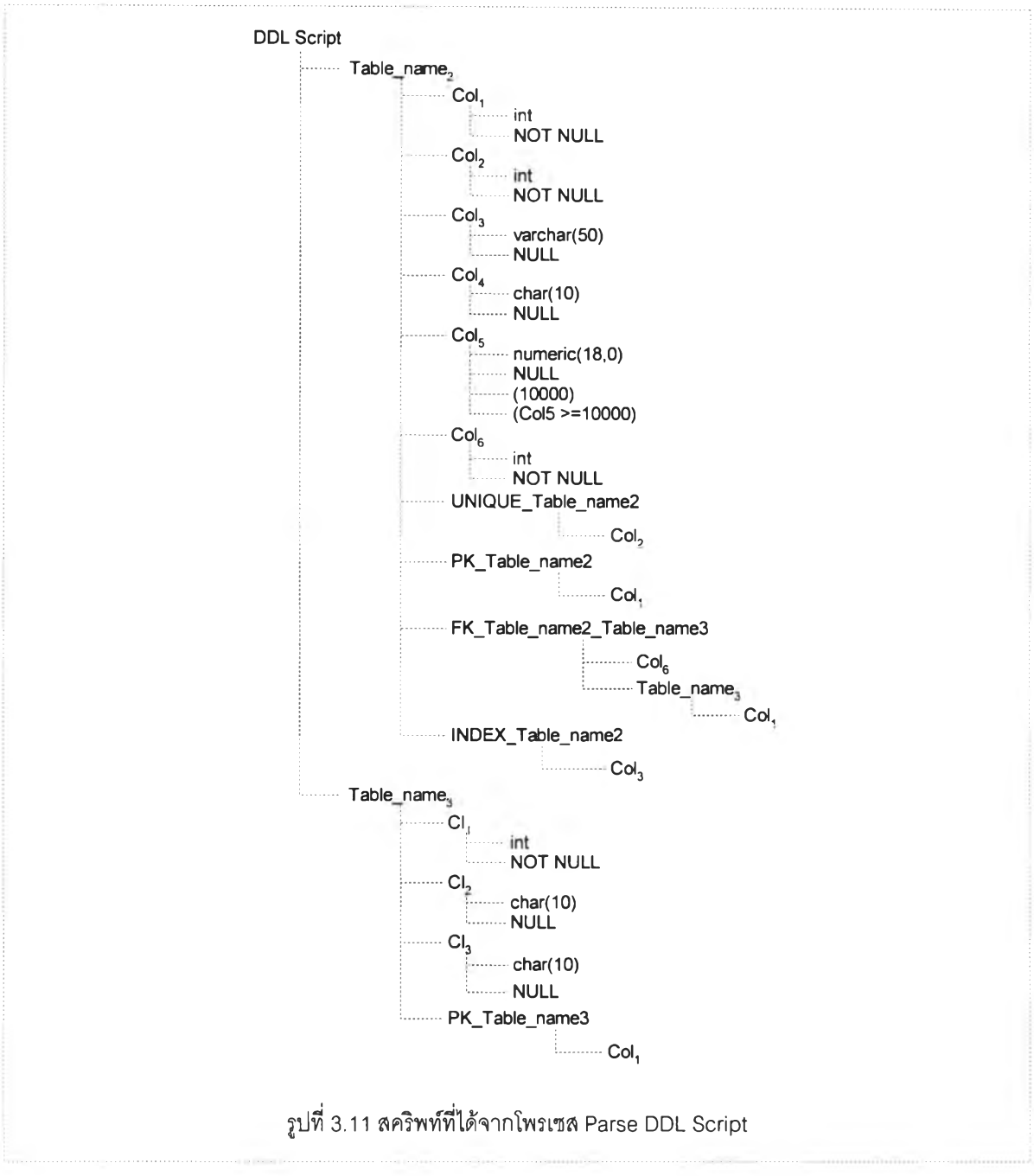
การทำงานของโพรเซสนี้เป็นการกระจายสคริพท์ เพื่อจำแนกรายการที่จำเป็นสำหรับเป็นข้อมูลเข้าในการเตรียมแผนแบบข้อมูลเชิงวัตถุเบื้องต้น

```
CREATE TABLE [dbo].[Table_name2] (
        [Col1] [int] NOT NULL ,
        [Col2] [int] NOT NULL ,
        [Col3] [varchar] (50) NULL ,
        [Col4] [char] (10) NULL ,
        [Col5] [numeric](18, 0) NULL ,
        [Col6] [int] NOT NULL
) ON [PRIMARY]
CREATE TABLE [dbo].[Table_name3] (
        [Cl1] [int] NOT NULL ,
        [Cl2] [char] (10) NULL ,
        [Cl3] [char] (10) NULL
) ON [PRIMARY]
ALTER TABLE [dbo].[Table_name2] WITH NOCHECK ADD
        CONSTRAINT [DF_Table_name2_Col5] DEFAULT (10000) FOR [Col5],
        CONSTRAINT [PK_Table_name2] PRIMARY KEY  NONCLUSTERED
        (
                [Col1]
        )  ON [PRIMARY] ,
        CONSTRAINT [UNIQUE_Table_name2] UNIQUE  NONCLUSTERED
        (
                [Col2]
        )  ON [PRIMARY] ,
        CONSTRAINT [CK_Table_name2] CHECK ([Col5] >= 10000)
ALTER TABLE [dbo].[Table_name3] WITH NOCHECK ADD
        CONSTRAINT [PK_Table_name3] PRIMARY KEY  NONCLUSTERED
        (
                [Cl1]
        )  ON [PRIMARY]
 CREATE  INDEX [INDEX_Table_name2] ON [dbo].[Table_name2]([Col3]) ON [PRIMARY]
ALTER TABLE [dbo].[Table_name2] ADD
        CONSTRAINT [FK_Table_name2_Table_name3] FOREIGN KEY
        (
                [Col6]
        ) REFERENCES [dbo].[Table_name3] (
                [Cl1]
        )
```

ตารางที่ 3.1 ตัวอย่างสคริพท์สำหรับสร้างตารางในฐานข้อมูลเชิงสัมพันธ์

ตารางที่ 3.1 ตัวอย่างสคริพท์ซึ่งเป็นข้อมูลเข้าสำหรับโพรเซสกระจายสคริพท์ สคริพท์นี้ได้จากการใช้โปรแกรม Enterprise Manager ของ MS SQL Server ช่วยสร้างเก็บในรูปของแฟ้มข้อความ (Text File)

ผลจากโพรเซสกระจายสคริพท์นี้เรียกว่า Parsed Script



รูปที่ 3.11 สคริพท์ที่ได้จากโพรเซส Parse DDL Script

รูปที่ 3.11 แสดงความสัมพันธ์ระหว่างรายการต่างๆใน Parsed Script หรือสคริพท์ที่กระจายได้ โดยที่การกระจายสคริพท์จะกระจายเพื่อให้ได้องค์ประกอบสคริพท์ซึ่งได้จากการวิเคราะห์สคริพท์ในหัวข้อ 3.2 องค์ประกอบสคริพท์

3.4.2 **เตรียมแผนแบบเชิงวัตถุเบื้องต้น** (Prepare an Initial Object Model)

โพรเซสเตรียมแผนแบบเชิงวัตถุเบื้องต้น จะใช้สคริพท์ที่ผ่านการกระจายแล้วจากโพรเซส Parse DDL Script เป็นข้อมูลเข้า โดยจะมีการแปลงตารางเป็นคลาส คอลัมน์เป็นแอททริบิวท์ และดรรชนีเป็นดรรชนีในแผนแบบเชิงวัตถุ

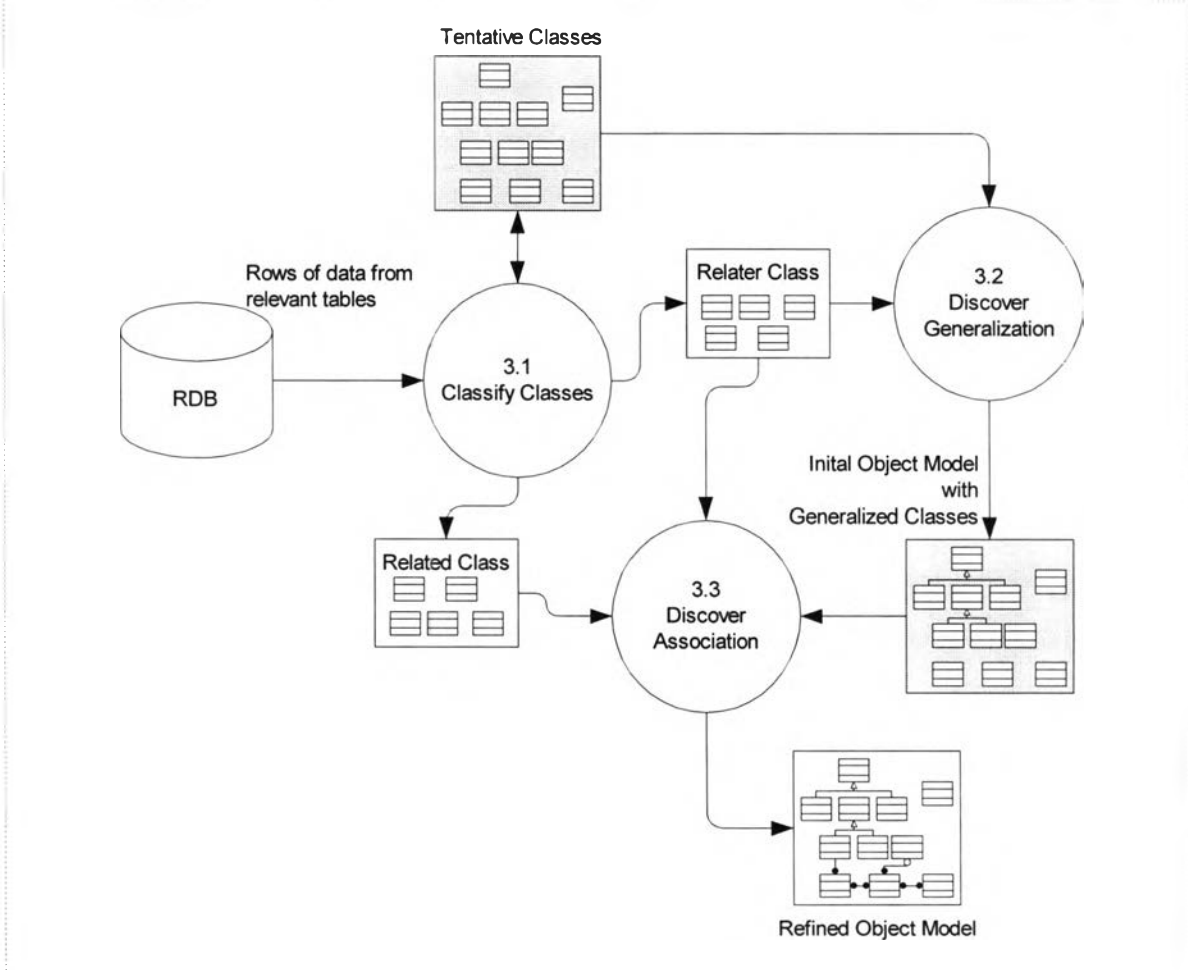
กำหนดให้ชื่อตารางเป็นชื่อคลาส ชื่อคอลัมน์ในตารางเป็นชื่อแอททริบิวท์ ประเภทข้อมูลในคอลัมน์เป็นประเภทข้อมูลในแอททริบิวท์ ขนาดข้อมูลในคอลัมน์เป็นขนาดข้อมูลในแอททริบิวท์

ผลจากโพรเซส Prepare an Initial Object Model เรียกว่า Initial Object Model หรือแผนแบบเชิงวัตถุเบื้องต้น คลาสต่างๆในแผนแบบเชิงวัตถุเบื้องต้นเรียกว่า Tentative Classes

3.4.3 ปรับแผนแบบเชิงวัตถุ (Refine Object Model)

โพรเซสปรับแผนแบบเชิงวัตถุ เป็นโพรเซสทำหน้าที่ปรับแผนแบบเชิงวัตถุเบื้องต้นเป็นแผนแบบเชิงวัตถุที่พร้อมสำหรับการนำไปสร้างฐานข้อมูลเชิงวัตถุต่อไป

โพรเซสนี้ประกอบด้วยโพรเซสย่อย 3 โพรเซส ดังแสดงในรูปที่ 3.12



รูปที่ 3.12 แสดงการปรับแผนแบบเชิงวัตถุเบื้องต้นเป็นแผนแบบเชิงวัตถุที่สมบูรณ์

รูปที่ 3.12 แสดงการปรับแผนแบบเชิงวัตถุเบื้องต้นเป็นแผนแบบเชิงวัตถุที่สมบูรณ์ ประกอบด้วย 3 โพรเซสย่อยคือ

- Classify Classes
- Discover Generalization
- Discover Association

3.4.3.1 **จำแนกคลาส** (Classify Classes)

โพรเซสนี้ทำหน้าที่จำแนกคลาส Tentative Classes เป็น 2 กลุ่มคือ Relater Classes และ Related Classes

- Relater Classes เป็นคลาสที่ถูกแปลงจากตารางที่มีคีย์นอกชี้ไปยังตารางอื่น
- Related Classes เป็นคลาสที่ถูกแปลงจากตารางที่ถูกอ้างอิงโดยตารางที่ถูกแปลงเป็น Relater Classes

โพรเซสนี้อาศัยข้อมูลเข้าในการทำงาน 2 ส่วนคือ Tentative Classes และรายการข้อมูลในตารางที่เกี่ยวข้องจากฐานข้อมูลเชิงสัมพันธ์ ผลของโพรเซสนี้คือ Relater Classes และ Related Classes รวมทั้งความสัมพันธ์ระหว่าง Related Classes กับ Relater Classes โดยที่ความสัมพันธ์ในที่นี้คือ ONE-TO-ONE และ ONE-TO-MANY

3.4.3.2 **ค้นหาการมีลักษณะทั่วไป** (Discover Generalization)

การค้นหาการมีลักษณะทั่วไป จำแนกเป็น 3 กรณีคือ

- Class with Replicated Attributes
- Classes with Inclusion Dependencies
- คลาสมากกว่า 1 คลาสมี Foreign Key ซึ่งเป็น Primary Key ด้วย ชี้ไปยังคลาสเดียวกัน

โพรเซสนี้จะค้นหาการสืบทอดคุณสมบัติระหว่างคลาส ผลลัพธ์จะได้คลาสเพิ่มขึ้นสำหรับ 2 กรณีแรก โดยจะมีการกำหนดว่าคลาสใดถูกสืบทอดคุณสมบัติโดยคลาสใด และคลาสใดสืบทอดคุณสมบัติมาจากคลาสอะไร

3.4.3.3 **ค้นหาความเกี่ยวพันระหว่างคลาส** (Discover Association)

ความเกี่ยวพันกัน (Association) เกิดขึ้นเมื่อคลาสต่างๆมีความสัมพันธ์กันผ่านคีย์นอก โดยจะจำแนกความสัมพันธ์เป็น 3 ประเภทคือ

- ONE-TO-ONE
- ONE-TO-MANY

- MANY-TO-MANY

โพรเซสนี้จะค้นหาความเกี่ยวพันกันระหว่างคลาสโดยอาศัย Relater Classes, Related Classes และ Tentative Classes ซึ่งผ่านโพรเซสค้นหาการมีลักษณะทั่วไปแล้วเป็นข้อมูลเข้าในการประมวลผล

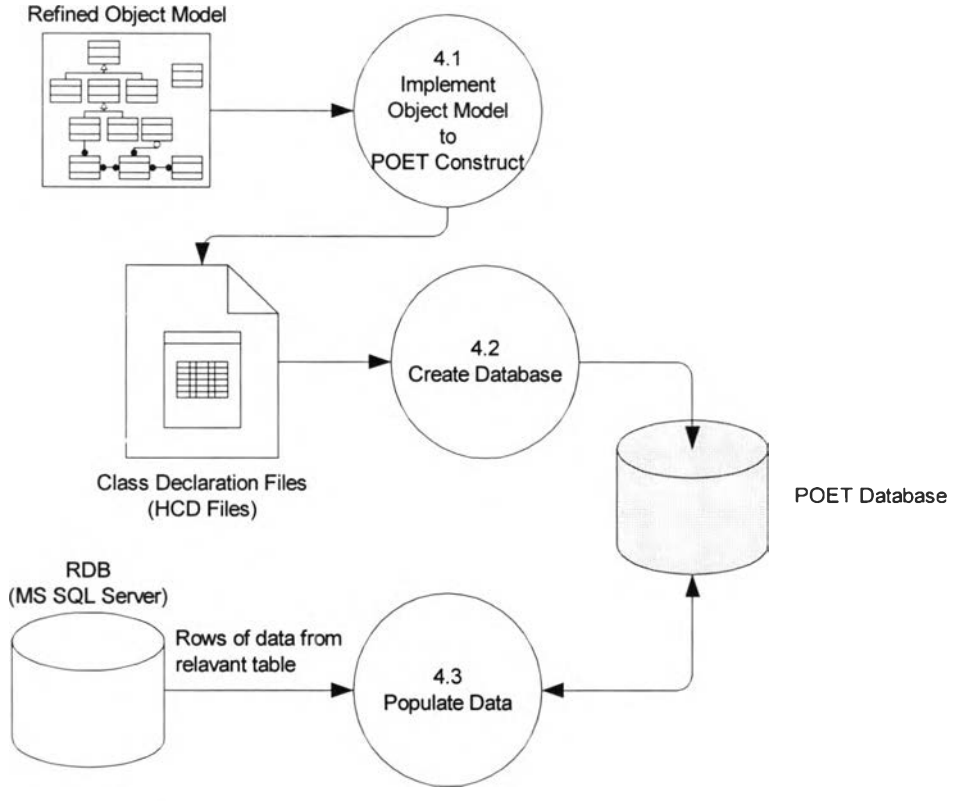
การค้นหาความเกี่ยวพันระหว่างคลาส พิจารณาจาก Relater Classes ที่ได้จากโพรเซสจำแนกคลาสเป็นหลัก จำแนกเป็น 4 กรณีคือ

1. Relater Class มีคีย์นอกชี้ไปยังตารางอื่นโดยที่คีย์นอกไม่เป็นคีย์หลัก หรือคีย์นอกไม่เป็นส่วนใดส่วนหนึ่งของคีย์หลัก
2. Relater Class เพียง 1 คลาสที่มีคีย์นอกซึ่งเป็นคีย์หลักด้วย ชี้ไปยัง Related Class
3. Relater Class ที่มีคีย์หลักซึ่งประกอบด้วยคีย์นอก 2 ตัว
4. Relater Class ซึ่งมีคีย์นอก2 ตัวโดยที่คีย์นอกตัวใดตัวหนึ่งเป็น คีย์หลักของ Relater Class ด้วย

ผลของโพรเซสความเกี่ยวพันระหว่างคลาส จะได้แผนแบบเชิงวัตถุที่สมบูรณ์มากขึ้นเรียกว่า Refined Object Model

3.4.4 **การใช้งานแผนแบบเชิงวัตถุบน** POET OODBMS (Implement Object Model with POET OODBMS)

เป็นโพรเซสสำหรับนำแผนแบบเชิงวัตถุติดตั้งบนระบบจัดการฐานข้อมูลเชิงวัตถุ POET



รูปที่ 3.13 แสดงโพรเซส Implement Object Model with POET OODBMS

รูปที่ 3.13 แสดงโพรเซสการติดตั้งใช้งานแผนแบบเชิงวัตถุบนระบบจัดการฐานข้อมูลเชิงวัตถุ POET ประกอบด้วย 3 โพรเซสย่อยคือ

- Implement Object Model to POET Construct
- Create Database
- Populate Data

3.4.4.1 **กำหนดการใช้งานแผนแบบเชิงวัตถุบนโครงสร้างของ** POET (Implement Object Model to POET Construct)

เป็นโพรเซสสำหรับแปลงแผนแบบเชิงวัตถุให้อยู่ในรูปซึ่งสามารถใช้งานบนระบบจัดการฐานข้อมูลเชิงวัตถุ POET ได้แยกเป็น 5 ส่วน

- Implement Domains
- Implement Classes
- Implement Generalizations
- Implement Associations
- Implement Indexes

ก.) Implement Domains

ในระบบจัดการฐานข้อมูลเชิงวัตถุ POET คลาสที่สามารถจัดเก็บในฐานข้อมูลจะมีประเภทข้อมูล (Data Type) ของ Data Member หรือแอททริบิวท์ที่มีประเภทข้อมูลดังต่อไปนี้

- Basic C++ Types
- POET Basic Types
- ODMG* Basic Types
- Embedded Objects หรือ Structures
- Pointers/Ondemands to Persistent Objects
- Collections

ในแผนแบบเชิงวัตถุ ที่ได้จากโพรเซสปรับแผนแบบเชิงวัตถุ ประเภทข้อมูลยังคงเป็นประเภทข้อมูลของระบบจัดการฐานข้อมูลเชิงสัมพันธ์ ในที่นี้คือระบบจัดการฐานข้อมูล MS SQL Server

---

* ODMG – Object Data Management Group

ANSI SQL Data Types	MS SQL Server Data Types
CHAR(1)	char(1)
CHAR(n)	char(n)
DECIMAL	Decimal
SMALLINT	Smallint
REAL	Real
INTEGER	Int
FLOAT	float
DOUBLE	float
NUMERIC	numeric
VARCHAR	varchar
LONGVARCHAR	varchar
BIT	bit
TINYINT	tinyint
BIGINT	int
BINARY	binary
VARBINARY	varbinary
LONGVARBINARY	varbinary
DATE	datetime
TIME	datetime
TIMESTAMP	timestamp

ตารางที่ 3.2 ANSI SQL Data Types กับ MS SQL Data Types

ประเภทข้อมูลจะยึดถือตาม ANSI SQL Data Types เป็นหลัก ตารางที่ 3.2 แสดงการเปรียบเทียบ ANSI SQL Data Types กับ Microsoft SQL Server Data Types

MS SQL Server Data Types	**POET C++ Data Types**
Bit	PtBool (bool)
int	long
smallint	int
tinyint	short
Decimal	PtString
numeric	PtString
Money	long
smallmoney	long
float	double
real	float
Datetime	PtDateTime
Smalldatetime	PtDate
Timestamp	PtDateTime
Char	PtString
Varchar	PtString
Text	PtString
Binary	PtBlob
Varbinary	PtBlob
Image	PtBlob

ตารางที่ 3.3 MS SQL Data Types กับ POET C++ Data Types

ตารางที่ 3.3 แสดงการเทียบประเภทข้อมูลของ MS SQL Server Data Types กับประเภทข้อมูลของ POET C++ Data Types

ข.) Implement Classes

การแปลงคลาสในแผนแบบเชิงวัตถุเป็น POET C++ Class ประกอบด้วย 2 ส่วน คือ

Declaration

C++ Header files ประกอบด้วยแอททริบิวท์ที่ได้จากแผนแบบเชิงวัตถุที่หามาได้ แอททริบิวท์ที่กำหนด Association ระหว่างคลาส และเม็ดธอด (Method)

การแปลง Object Model เป็น POET C++ Class Declaration ประกอบด้วยขั้นตอนต่างๆดังนี้

- กำหนดคำสงวน persistent สำหรับคลาสซึ่งอ็อบเจคท์ของคลาสนั้นต้องเก็บลงฐานข้อมูล

```
persistent class Invoice
{
    ...
};
```

- กำหนดให้ Attributes ที่จะถูกแปลงเป็น Data Members ใน POET C++ Class อยู่ในส่วน private

```
persistent class Invoice
{
private:
   PtString   Name;
   int      numEntries;
            ...
};
```

- กำหนด Copy Constructor และ Assignment Operator

```
Persistent class Invoice
{
private:
   PtString   Name;
   Customer   *pCustomer;
   int      numEntries;

   Invoice& operator = (const Invoice&);
   Invoice( const Invoice& );
            ...
};
```

- กำหนด Constructor ว่างๆและ Virtual Destructor ในส่วนของ public

```
persistent class Invoice
{
private:
   PtString   Name;
         ...

public:
   Invocie();
   virtual ~Invoice();
};
```

- นอกจากนี้ต้องเพิ่มแอททริบิวท์สำหรับแสดงความเกี่ยวพัน(Associations) ระหว่างคลาสต่างๆด้วย

Methods implied by the object model

หมายถึงเม็ดธอดที่เกี่ยวกับการสร้างอ็อบเจคท์ การทำลายอ็อบเจคท์ และเม็ดธอดเกี่ยวกับการเข้าถึงแอททริบิวท์ต่างๆ

ในที่นี้จะไม่กล่าวถึงส่วนนี้เนื่องจากข้อมูลในฐานข้อมูลเชิงสัมพันธ์ไม่มีส่วนที่เกี่ยวพันกับเม็ดธอดและ POET เตรียมไว้ให้แล้วเมื่อไม่ได้กำหนดให้โดยตรง

แฟ้มข้อมูลที่ได้จากขั้นตอน Implement Classes จะมีชื่อตามคลาสและนามสกุลเป็น HCD (ClassName.hcd)

ค.) Implement Generalizations

ในการสืบทอดคุณสมบัติระหว่างคลาสพ่อกับคลาสลูก การแปลงส่วนของคลาสพ่อ ทำเช่นเดียวกับการแปลงคลาสปกติทั่วไปตัวอย่างเช่นคลาส Car และคลาส PrescriptionDrug เป็นคลาสลูกของคลาส Product ฉะนั้นการประกาศคลาส Product กำหนดเป็นดังนี้

```
persistent class Product
{
public:
   PtString name;
   long list_price;

   Product& operator = (const Product&);
   Product( const Product&);

public:
   Product();
   virtual ~Product();
};
```

คลาส Car สืบทอดคุณสมบัติจากคลาส Product ดังนั้นสามารถประกาศคลาสดังนี้

```
persistent class Car : public Product
{
public:
   unsigned short horsePower;
   unsigned short milePerGallon;

   Car& operator = (const Car&);
   Car( const Car&);

public:
   Car();
   virtual ~Car();
};
```

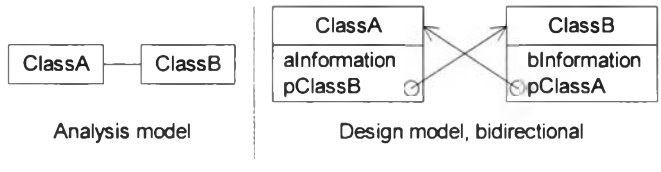
ทำนองเดียวกันคลาส PrescriptionDrug สามารถประกาศคลาสดังนี้

```
persistent class PrescriptionDrug : public Product
{
private:
   PrescriptionDrug& operator = (const PrescriptionDrug&);
   PrescriptionDrug(const PrescriptionDrug&);

   unsigned short count;
   PtDate expirationDate;

public:
   PrescriptionDrug();
   virtual ~PrescriptionDrug();
};
```

ง.) Implement Associations

การกำหนดความเกี่ยวพันระหว่างคลาสใน POET C++ Class Declaration ขึ้นกับประเภทของความสัมพันธ์ว่าเป็นแบบ ONE-TO-ONE ONE-TO-MANY หรือ MANY-TO-MANY

<u>กรณีความเกี่ยวพันแบบ ONE-TO-ONE</u>



รูปที่ 3.14 แสดงรูปแบบการกำหนดความสัมพันธ์แบบ ONE-TO-ONE ระหว่างคลาส

คลาสมีความสัมพันธ์ต่อกันแบบ ONE-TO-ONE เช่นคลาส ClassA กับคลาส ClassB ในรูปที่ 3.14 คลาสทั้งสองจะถูกแปลงเป็น C++ Class Declaration ดังนี้

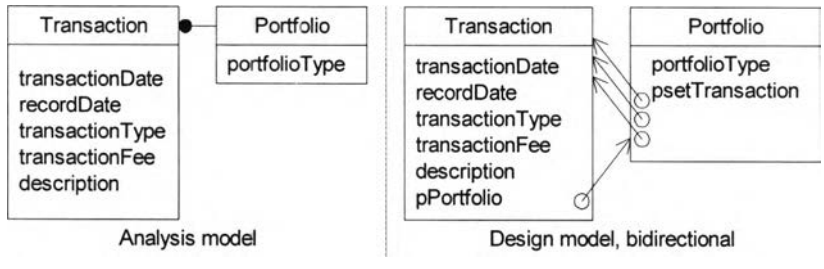
ที่คลาส ClassA กำหนดแอททริบิวท์ประเภทพอยท์เตอร์ชี้ไปยังคลาส ClassB โดยกำหนดชื่อพอยท์เตอร์ขึ้นต้นด้วย p ดังนี้

```
Persistent class ClassB;

persistent class ClassA
{
public:
   ClassB *pClassB;
   ...
};
```

ทำนองเดียวกันกรณีคลาส ClassB ชี้ไปคลาส ClassA ประกาศคลาสดังนี้

```
persistent class ClassA;

persistent class ClassB
{
public:
   ClassA *pClassA;
   ...
};
```

กรณีความเกี่ยวพันแบบ ONE-TO-MANY



รูปที่ 3.15 การกำหนดความสัมพันธ์แบบ ONE-TO-MANY ระหว่างคลาส

กรณีความสัมพันธ์ระหว่างคลาสเป็นแบบ ONE-TO-MANY กำหนดกลุ่มของพอยท์เตอร์ (Collection of Pointers) ในคลาสด้านที่มีความสัมพันธ์ "ONE-TO" และกำหนดพอยท์เตอร์ในคลาสด้านที่มีความสัมพันธ์ "MANY-TO" ชี้ไปยังคลาสฝั่งตรงข้าม

จากรูปที่ 3.15 ด้านซ้ายมือคลาส Portfolio มีความสัมพันธ์กับคลาส Transaction แบบ ONE-TO-MANY ดังนั้นรูปด้านซ้ายมือคลาส Portfolio กำหนดกลุ่ม

พอยท์เตอร์ psetTransaction ชี้ไปยังคลาส Transaction และในคลาส Transaction กำหนดพอยท์เตอร์ชี้กลับมายังคลาส Portfolio

กลุ่มของพอยท์เตอร์ใน POET แบ่งเป็น 3 ประเภทคือ

cset<type> หมายถึง compact set มีการเชื่อมโยงกับอ็อบเจ็คท์อื่นไม่มาก

lset<type> หมายถึง large set มีการเชื่อมโยงกับอ็อบเจ็คท์อื่นมาก

hset<type> หมายถึง huge set มีการเชื่อมโยงกับอ็อบเจ็คท์อื่นมากที่สุด

การเลือกใช้งานขึ้นกับผู้ใช้งาน โดยต้องคำนึงว่าอ็อบเจ็คท์นั้นๆ มีการอ้างอิงกับอ็อบเจ็คท์อื่นมากน้อยเพียงใด วิธีการเลือกที่ปลอดภัยที่สุดคือ เลือกเซ็ตที่ใหญ่พอสมควร ในที่นี้จะเลือกเป็น lset<type>

จากรูปที่ 3.15 กำหนด POET C++ Class Declaration สำหรับคลาส Portfolio ดังนี้

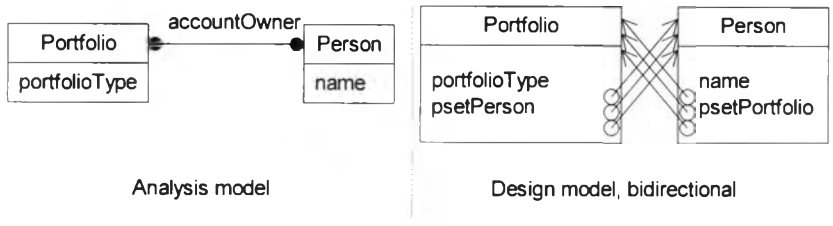
```
persistent class Transaction;

persistent class Portfolio
{
public:
   int portfolioType;
   lset<Transaction *> psetTransaction;
   ...
};
```

และกำหนด POET C++ Class Declaration สำหรับคลาส Transaction ดังนี้

```
persistent class Portfolio;

persistent class Transaction
{
public:
   Portfolio *pPortfolio;
   ...
};
```

กรณีความเกี่ยวพันแบบ MANY-TO-MANY

กรณีคลาสสองคลาสมีความสัมพันธ์ระหว่างกันแบบ MANY-TO-MANY กำหนดกลุ่มของพอยท์เตอร์ชี้ไปยังคลาสฝั่งตรงกันข้าม



รูปที่ 3.16 การกำหนดความสัมพันธ์แบบ MANY-TO-MANY ระหว่างคลาส

จากรูปที่ 3.16 กำหนดให้คลาส Portfolio มีกลุ่มของพอยท์เตอร์ชี้ไปยังคลาส Person และทำนองเดียวกันคลาส Person กำหนดกลุ่มของพอยท์เตอร์ชี้ไปยังคลาส Portfoilio ดังนั้น POET C++ Class Declaration สำหรับคลาสทั้งสองกำหนดดังนี้

```
persistent class Person;

persistent class Portfolio
{
private:
   lset<Person *> psetPerson;
   ...
};
```

และคลาส Person ประกาศคลาสดังนี้

```
Persistent class Portfolio

Persistent class Person
{
private:
   lset<Portfolio *> psetPortfolio;
   ...
};
```

จ.) Implement Indexes

การแปลงดรรชนีสามารถแปลงโดยตรง ประกอบด้วย 2 ส่วนคือส่วนที่หนึ่งกำหนดในส่วน Public ของการประกาศคลาสโดยการใช้คำสงวน useindex ตามด้วยชื่อดรรชนี ส่วนที่สองเป็นส่วนของการกำหนดนิยามดรรชนี โดยการกำหนดว่าดรรชนีนั้นกระทำที่ Data Member ชื่ออะไรดังตัวอย่างต่อไปนี้

```
persistent class Customer;
persistent class Entry;

persistent class Invoice
{
  private:
    PtString name;
    Customer *pCustomer;
    depend lset<Entry *> psetEntry;

    Invoice& operator = (const Invoice&);
    Invoice(const Invoice&);
  public:
    Invoice();
    ~Invoice();
    useindex NameIndex;
};

indexdef NameIndex : Invoice
{
  name;
}
```

3.4.4.2 สร้างฐานข้อมูล (Create Database)

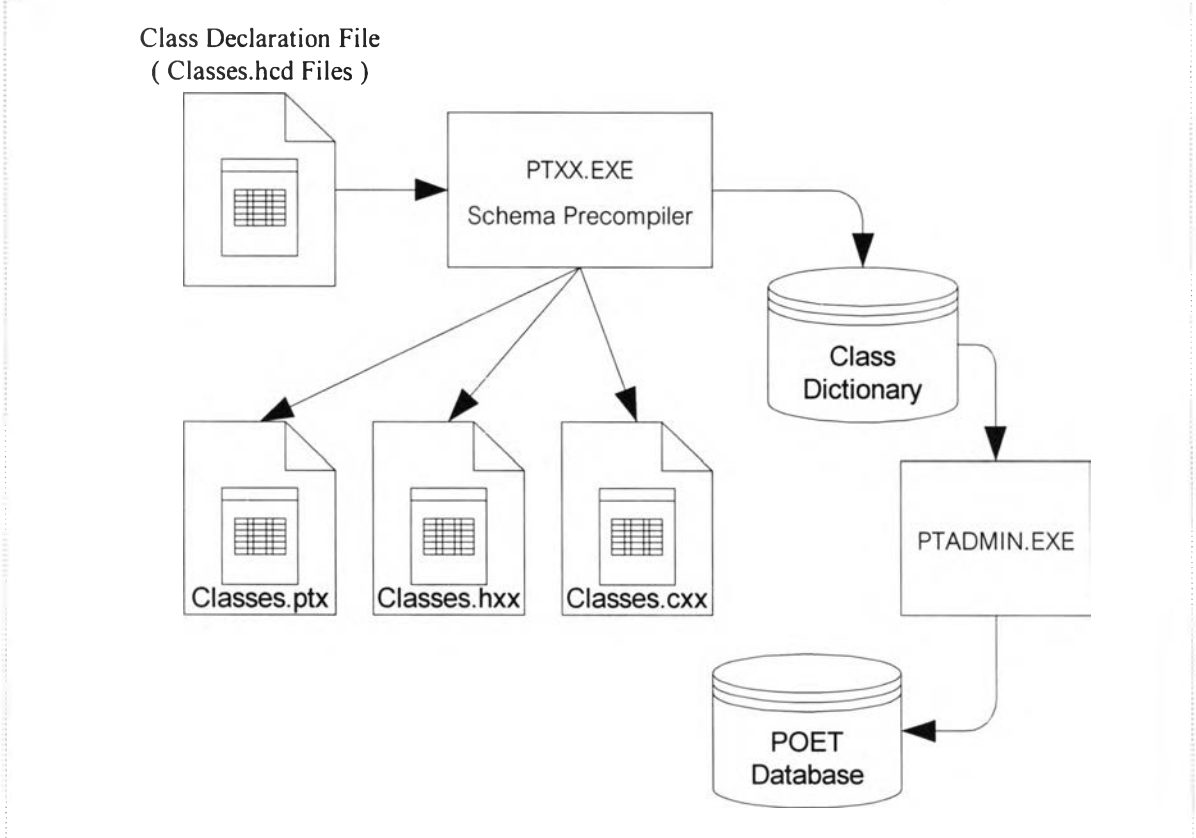
จาก C++ Class Declaration ได้มีการกำหนดคำสงวนและข้อมูลต่างๆเพื่อให้ POET เข้าใจโครงสร้างของคลาสต่างๆรวมทั้งความสัมพันธ์ระหว่างคลาสเหล่านั้น กระบวนการทั้งหมดในโพรเซสกำหนดการใช้งานแผนแบบเชิงวัตถุบนโครงสร้าง POET กำหนดสิ่งเหล่านี้และได้ผลลัพธ์เป็นแฟ้มข้อความการประกาศคลาสสำหรับการสร้างฐานข้อมูลใน POET โดยกำหนดให้ 1 แฟ้มข้อมูลต่อ 1 คลาส กำหนดชื่อแฟ้มตามชื่อคลาสและมีนามสกุลเป็น HCD

จากแฟ้ม HCD ที่ได้ผ่านโปรแกรม PTXX Schema Precompiler ของ POET จะได้ Class Dictionary และแฟ้มต่างๆที่เกี่ยวข้อง

PTXX.EXE ทำหน้าที่ดังนี้

- ตรวจเช็ค Semantic ของ Persistent Class Declarations
- สร้าง Class Dictionary
- สร้าง Class Factory Files (CXX File) เพื่อบอก POET ว่าอ็อบเจคท์จะถูกสร้างและจัดการบนหน่วยความจำอย่างไร จะใช้สำหรับคอมไพล์และลิงค์กับโปรแกรมที่เขียนเพื่อจัดการกับฐานข้อมูลนี้ในภายหลัง
- สร้าง Class Factory Header Files (PTX File)
- สร้าง Class Declaration Header (HXX File) ซึ่งเป็น C++ มาตรฐานแทนแฟ้ม HCD ใช้สำหรับรวมกับ Application Source Files

PTADMIN.EXE ทำหน้าที่สร้างฐานข้อมูลบน POET โดยใช้ Class Dictionary ที่ได้จากขั้นตอนการทำงานของโปรแกรม PTXX Schema Compiler

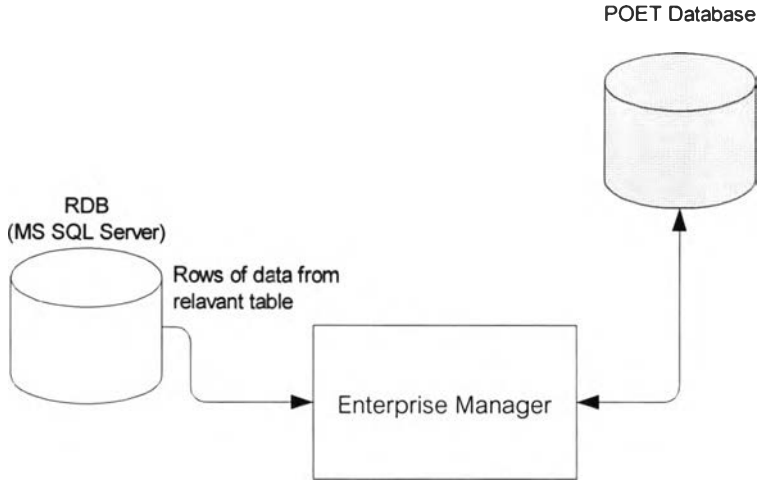


รูปที่ 3.17 การสร้างฐานข้อมูลบนระบบจัดการฐานข้อมูล POET

รูปที่ 3.17 แสดงขั้นตอนการสร้างฐานข้อมูลบน POET โดยใช้ PTXX Precompiler และ PTADMIN ของ POET

3.4.4.3 **นำเข้าข้อมูล** (Populate Objects)

ขั้นตอนนี้ เป็นขั้นตอนการนำเข้าข้อมูลจากฐานข้อมูลบนระบบจัดการฐานข้อมูลเชิงสัมพันธ์ MS SQL Server เก็บบนฐานข้อมูลในระบบจัดการฐานข้อมูลเชิงวัตถุ POET ซึ่งได้จากโพรเซส Create Database



รูปที่ 3.18 การนำข้อมูลจากฐานข้อมูลเชิงสัมพันธ์เข้าเก็บในฐานข้อมูล POET

รูปที่ 3.18 แสดงการนำเข้ารายการข้อมูลแต่ละตารางบนฐานข้อมูลเชิงสัมพันธ์ (MS SQL Server) ลงเก็บในฐานข้อมูลบนระบบจัดการฐานข้อมูล POET โดยใช้โปรแกรม Enterprise Manager ของ Microsoft SQL Server

วิธีการนำเข้ารายการข้อมูลจากฐานข้อมูลบน MS SQL Server ลงบนฐานข้อมูลในระบบจัดการฐานข้อมูล POET แสดงในภาคผนวก ค.