เครื่องมือตรวจหาเอสคิวแอลแอนติแพตเทิร์น และการกำหนดกระบวนการรีแฟคทอริงฐานข้อมูล

นางสาวปุณยนุช ขำนิล

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมซอฟต์แวร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2559

SQL Antipatterns Detection Tool and Database Refactoring Process Definition

Miss Poonyanuch Khumnin

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Science Program in Software Engineering
Department of Computer Engineering
Faculty of Engineering
Chulalongkorn University
Academic Year 2016

หัวข้อวิทยานิพนธ์	เครื่องมือตรวจหาเอสคิวแอลแอนติแพตเทิร์น และการกำหนดกระบวนการรีแฟคทอริงฐานข้อมูล
โดย	นางสาวปุณยนุช ขำนิล
สาขาวิชา	วิศวกรรมซอฟต์แวร์
อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก	รองศาสตราจารย์ ดร.ทวิตีย์ เสนีวงศ์ ณ อยุธยา

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

.....คณบดีคณะวิศวกรรมศาสตร์
(รองศาสตราจารย์ ดร.สุพจน์ เตชวรสินสกุล)

คณะกรรมการสอบวิทยานิพนธ์

.....ประธานกรรมการ
(รองศาสตราจารย์ ดร.พรศิริ หมื่นไชยศรี)

.....อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก
(รองศาสตราจารย์ ดร.ทวิตีย์ เสนีวงศ์ ณ อยุธยา)

.....กรรมการ
(รองศาสตราจารย์ ดร.เศรษฐา ปานงาม)

.....กรรมการภายนอกมหาวิทยาลัย
(รองศาสตราจารย์ ดร.กฤดาภัทร สีหารี)

ปุณยนุช ขำนิล : เครื่องมือตรวจหาเอสคิวแอลแอนติแพตเทิร์น และการกำหนดกระบวนการรีแฟคทอริงฐานข้อมูล (SQL Antipatterns Detection Tool and Database Refactoring Process Definition) อ.ที่ปรึกษาวิทยานิพนธ์หลัก: รศ. ดร.ทวิตีย์ เสนีวงศ์ ณ อยุธยา, 128 หน้า.

เอสคิวแอลแอนติแพตเทิร์น เป็นขั้นตอนการทำงานที่ผิดพลาดที่มักพบบ่อย ๆ ในการออกแบบฐานข้อมูลเชิงสัมพันธ์ การใช้เอสคิวแอล และการพัฒนาแอพพลิเคชันฐานข้อมูล ขั้นตอนการทำงานเหล่านี้ทำเพื่อแก้ปัญหาบางอย่าง แต่ในที่สุดก็สามารถนำไปสู่ปัญหาอื่น ๆ ได้ วัตถุประสงค์ของงานวิจัยนี้คือการช่วยผู้ดูแลระบบฐานข้อมูลในการวินิจฉัยหรือตรวจหาเอสคิวแอลแอนติแพตเทิร์น และแนะนำเทคนิคการรีแฟคทอริงฐานข้อมูลเพื่อแก้ปัญหาเอสคิวแอลแอนติแพตเทิร์น โดยเฉพาะอย่างยิ่ง งานวิจัยนี้พยายามที่จะทำให้การตรวจหาเอสคิวแอลแอนติแพตเทิร์นของการออกแบบฐานข้อมูลเชิงตรรกะสามารถทำได้อย่างอัตโนมัติ โดยการพัฒนาเครื่องมือที่ใช้ภาษาทรานแซคเอสคิวแอล เพื่อค้นหาและวิเคราะห์สคีมาฐานข้อมูล เครื่องมือสามารถรายงานเอสคิวแอลแอนติแพตเทิร์นที่อาจเกิดขึ้นและให้คำแนะนำในการรีแฟคทอริงสคีมาของฐานข้อมูลได้ จากการประเมินผลฐานข้อมูลสามแหล่งจากอุตสาหกรรม พบว่าประสิทธิภาพของเครื่องมือนี้เป็นที่น่าพอใจในแง่ของค่าเรียกคืนเอสคิวแอลแอนติแพตเทิร์น แต่เครื่องมือทำนายผิดว่าเกิดแอนติแพตเทิร์นในหลายจุด ซึ่งส่งผลต่อความเที่ยงตรง อย่างไรก็ตาม ผู้วิจัยได้ทำการทดสอบเพื่อเปรียบเทียบประสิทธิภาพของเครื่องมือที่เสนอกับวิธีการตรวจหาเอสคิวแอลแอนติแพตเทิร์นของงานวิจัยอื่น ได้แก่งานวิจัยของ Eessaar พบว่าเครื่องมือที่เสนอมีค่าเรียกคืนและค่าความเที่ยงตรง ส่วนมากดีกว่าวิธีของ Eessaar จากผลการวิจัยสรุปได้ว่าการตรวจสอบเอสคิวแอลแอนติแพตเทิร์น ยังต้องอาศัยการพิจารณาความหมายของข้อมูลอยู่อีกมากและเครื่องมือตรวจสอบควรถูกนำมาใช้ในรูปแบบกึ่งอัตโนมัติมากกว่า กล่าวคือสามารถใช้เครื่องมือในการชี้ตำแหน่งที่เป็นไปได้ที่อาจเกิดแอนติแพตเทิร์นขึ้นในสคีมาฐานข้อมูลจากนั้นจึงใช้การวิเคราะห์เพิ่มเติมจากผู้ดูแลระบบฐานข้อมูล วิธีนี้จะเป็นประโยชน์โดยเฉพาะอย่างยิ่งในบริบทของฐานข้อมูลขนาดใหญ่ที่การตรวจสอบเอสคิวแอลแอนติแพตเทิร์นด้วยมนุษย์เป็นเรื่องยาก

ภาควิชา วิศวกรรมคอมพิวเตอร์ ลายมือชื่อนิสิต .....

สาขาวิชา วิศวกรรมซอฟต์แวร์ ลายมือชื่อ อ.ที่ปรึกษาหลัก .....

ปีการศึกษา 2559

5870948621 : MAJOR SOFTWARE ENGINEERING

KEYWORDS: SQL ANTIPATTERN; DATABASE REFACTORING; TRANSACT- SQL; RELATIONAL DATABASE

POONYANUCH KHUMNIN: SQL Antipatterns Detection Tool and Database Refactoring Process Definition. ADVISOR: ASSOC. PROF. TWITTIE SENIVONGSE, Ph.D., 128 pp.

SQL antipatterns are frequently-made missteps that are commonly found in the design of relational databases, the use of SQL, and the development of database applications. They are intended to solve certain problems but will eventually lead to other problems. The motivation of this research is how to assist database administrators in diagnosing SQL antipatterns and suggest refactoring techniques to solve the antipatterns. Specifically, this research attempts to automate the detection of logical database design antipatterns by developing a tool that uses Transact-SQL language to query and analyze the database schema. The tool reports on potential antipatterns and gives an instruction on how to refactor the database schema. In an evaluation based on three databases from the industry, the performance of the tool is satisfactory in terms of recall of the antipatterns but the tool detects a number of false positives which affect its precision. However, an experiment is further conducted to compare the performance of the proposed tool with that of other SQL antipattern detection research by Eessaar. The result shows that the tool has better recall and precision for most cases of antipattern detection. From this research, it is found that SQL antipatterns detection still largely depends on the semantics of the data and the detection tool should rather be used in a semi-automated manner, i.e it can point out potential problematic locations in the database schema which require further diagnosis by the database administrators. This approach would be useful especially in the context of large databases where manual antipatterns inspection is difficult.

Department: Computer Engineering Student's Signature .....

Field of Study: Software Engineering Advisor's Signature .....

Academic Year: 2016

กิตติกรรมประกาศ

ขอกราบขอบพระคุณ รศ. ดร.ทวิตีย์ เสนีวงศ์ ณ อยุธยา อาจารย์ที่ปรึกษาวิทยานิพนธ์ เป็นอย่างยิ่งที่ได้สละเวลาให้คำปรึกษา คำแนะนำ และแนวทางสำหรับการทำวิทยานิพนธ์ รวมทั้งเป็นผู้ประสานงานให้ความช่วยเหลือแก่นิสิตที่ทำวิทยานิพนธ์ทุกคน

ขอกราบขอบพระคุณ คณะกรรมการคุมสอบวิทยานิพนธ์เป็นอย่างยิ่ง ที่ได้กรุณาแนะนำแนวทาง รวมถึงการตรวจสอบและแก้ไขวิทยานิพนธ์นี้

ขอขอบคุณ คณาจารย์ทุกท่านในภาควิชาวิศวกรรมคอมพิวเตอร์ จุฬาลงกรณ์มหาวิทยาลัย ที่ให้คำแนะนำ ความรู้และแนวทางการทำวิทยานิพนธ์

ขอขอบคุณ เพื่อนๆ หลักสูตรวิศวกรรมซอฟต์แวร์ สำหรับกำลังใจและคำแนะนำในการจัดทำวิทยานิพนธ์

สุดท้ายนี้ ขอกราบขอบพระคุณ บิดา มารดา รวมถึงสมาชิกในครอบครัวที่ให้การสนับสนุนและให้กำลังใจที่ดีเสมอมา

สารบัญ

หน้า

หน้า

หน้า

สารบัญตาราง

สารบัญรูป

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญของปัญหา

ร่องรอยที่ไม่ดี (Bad smell หรือ Code Smell) [1] ในการเขียนโปรแกรมคอมพิวเตอร์หมายถึงโครงสร้างบางอย่างในคำสั่งโปรแกรมที่บ่งชี้ว่ามีการออกแบบที่ขาดหลักการที่ดีหรือมีคุณภาพการออกแบบที่ส่งผลในเชิงลบ ร่องรอยที่ไม่ดีนั้นไม่ทำให้การทำงานของซอฟต์แวร์ผิดพลาด แต่อาจทำให้ทราบถึงความเสี่ยงที่จะเกิดข้อผิดพลาดของระบบในอนาคตได้ ในการแก้ไขร่องรอยที่ไม่ดีจะใช้วิธีการรีแฟคทอริง (Refactoring) ซึ่งเป็นกระบวนการปรับโครงสร้างของซอฟต์แวร์ โดยไม่ทำให้พฤติกรรมการทำงานของซอฟต์แวร์นั้นเปลี่ยนแปลงไป อย่างไรก็ตามเมื่อระบบซอฟต์แวร์ได้ถูกพัฒนาให้มีขนาดใหญ่ขึ้น การค้นหาร่องรอยที่ไม่ดีด้วยมนุษย์เพื่อทำการรีแฟคทอริงนั้นอาจทำได้ยาก จึงทำให้มีระบบอัตโนมัติสำหรับการค้นหาร่องรอยที่ไม่ดีเกิดขึ้นมากมาย [2]

ฐานข้อมูลเป็นส่วนประกอบสำคัญของซอฟต์แวร์ การออกแบบฐานข้อมูลเป็นกระบวนการที่สำคัญ ถ้าการออกแบบฐานข้อมูลขาดหลักการแล้ว จะทำให้ส่งผลกระทบต่อการใช้งาน สมรรถนะและการแก้ไขในภายหลังอาจจะทำได้ยาก ตัวอย่างเช่น การออกแบบตารางเก็บข้อมูลการขายที่มีโครงสร้างเหมือนกันแต่เก็บข้อมูลแยกรายปี โดยกำหนดชื่อตารางตามท้ายด้วยปี พ.ศ. เพื่อให้สามารถเลือกสืบค้นเฉพาะฐานข้อมูลของปีที่ต้องการเท่านั้น แต่เมื่อเวลาผ่านไปหากต้องเพิ่มคอลัมน์ในตารางข้อมูลนี้แล้ว ผู้ดูแลฐานข้อมูลจะพบปัญหาว่าจะต้องแก้ไขโครงสร้างโดยเพิ่มคอลัมน์ทุกตารางให้เหมือนกัน อีกทั้งผู้พัฒนาซอฟต์แวร์ก็ต้องแก้ไขการเรียกใช้ฐานข้อมูลด้วยเช่นกัน จากตัวอย่างปัญหาข้างต้น B. Karwin [3] ได้รวบรวมเอสคิวแอลแอนติแพตเทิร์น (SQL Antipattern) หรือข้อผิดพลาดในการใช้เอสคิวแอลในฐานข้อมูลไว้ และ S. J. Ambler [4] ได้อธิบายการทำรีแฟคทอริงฐานข้อมูลและระบุกระบวนการรีแฟคทอริงไว้โดยละเอียด เอสคิวแอลแอนติแพตเทิร์นจึงเปรียบเหมือนร่องรอยที่ไม่ดี ซึ่งสามารถใช้วิธีรีแฟคทอริงฐานข้อมูลในการแก้ไขเอสคิวแอลแอนติแพตเทิร์นได้เช่นกัน

ผู้วิจัยจึงเกิดคำถามว่า จะสามารถช่วยผู้ดูแลฐานข้อมูลพิจารณาหรือตรวจหาเอสคิวแอลแอนติแพตเทิร์น ก่อนดำเนินการรีแฟคทอริงฐานข้อมูลได้หรือไม่ โดยพัฒนาเป็นเครื่องมือเพื่อช่วยผู้ดูแลฐานข้อมูลในการตรวจหาเอสคิวแอลแอนติแพตเทิร์นและแนะนำกระบวนการรีแฟคทอริงฐานข้อมูล ผู้วิจัยจึงมีแนวคิดในการพัฒนาเครื่องมือตรวจหาเอสคิวแอลแอนติแพตเทิร์นในฐานข้อมูล โดยใช้ภาษาทรานแซก-เอสคิวแอล (Transact-SQL) [5] เพื่อให้ผู้ดูแลฐานข้อมูลนำไปใช้ ผลลัพธ์ที่ได้จะเป็นรายการเอสคิวแอลแอนติแพตเทิร์นที่ตรวจพบในฐานข้อมูล พร้อมทั้งมีการแนะนำกระบวนการรีแฟคทอริงของเอสคิวแอลแอนติแพตเทิร์นนั้น เพื่อช่วยให้ผู้ดูแลฐานข้อมูลพิจารณาและนำไปประยุกต์ใช้ในขั้นตอนการทำรีแฟคทอริงฐานข้อมูลได้ด้วย

1.2 วัตถุประสงค์

1.2.1 กำหนดขั้นตอนวิธีตรวจหาเอสคิวแอลแอนติแพตเทิร์นในฐานข้อมูล

1.2.2 กำหนดกระบวนการรีแฟคทอริงฐานข้อมูลสำหรับเอสคิวแอลแอนติแพตเทิร์นที่ตรวจพบ

1.2.3 พัฒนาเครื่องมือตรวจหาเอสคิวแอลแอนติแพตเทิร์นในฐานข้อมูลตามขั้นตอนวิธีในข้อที่ 1.2.1 และแนะนำกระบวนการรีแฟคทอริงตามกระบวนการที่กำหนดในข้อที่ 1.2.2

1.3 ขอบเขต

1) เครื่องมือสามารถช่วยตรวจหาเอสคิวแอลแอนติแพตเทิร์นในส่วนของการออกแบบฐานข้อมูลเชิงตรรกะ (Logical Database Design) ตามขั้นตอนวิธีตรวจหาที่กำหนด โดยแอนติแพทเทิร์นที่รองรับมี 8 ประเภท ได้แก่
    i. Format Comma-Separated List
    ii. Always Depend on One's Parent
    iii. One Size Fits All
    iv. Leave Out the Constraints
    v. Use a Generic Attribute Table
    vi. Use Dual-Purpose Foreign Key
    vii. Create Multiple Columns
    viii. Clone Table or Columns
2) เครื่องมือสามารถแนะนำกระบวนการรีแฟคทอริงฐานข้อมูลสำหรับเอสคิวแอลแอนติแพตเทิร์นที่ตรวจพบ ตามกระบวนการรีแฟคทอริงที่กำหนด โดยแนะนำเฉพาะ 3 ขั้นตอน ได้แก่ การปรับสกีมา การย้ายข้อมูล และการปรับโปรแกรมที่ใช้เข้าถึง
3) พัฒนาเครื่องมือโดยใช้ทรานแซก-เอสคิวแอล
4) เครื่องมือจะตรวจหาเอสคิวแอลแอนติแพตเทิร์นตามขั้นตอนวิธีตรวจหาซึ่งพิจารณาโครงสร้างของฐานข้อมูลเท่านั้น โดยผู้ดูแลฐานข้อมูลจะเห็นว่าเป็นแอนติแพตเทิร์นหรือไม่ขึ้นอยู่กับการพิจารณาและขอบเขตของธุรกิจของแต่ละฐานข้อมูล
5) เครื่องมือไม่รองรับการจัดการผลกระทบต่อโปรแกรมที่เชื่อมต่ออื่น ๆ จากการทำรีแฟคทอริงตามคำแนะนำ

6) เครื่องมือใช้ได้กับไมโครซอฟท์เอสคิวแอลเซิร์ฟเวอร์เวอร์ชัน 2008 R2 ขึ้นไป
7) เครื่องมือใช้สำหรับผู้ที่มีสิทธิ์เข้าใช้งานฐานข้อมูลที่เป็นเจ้าของฐานข้อมูล
8) เครื่องมือตรวจหาและแนะนำกระบวนการรีแฟคทอริงฐานข้อมูลสำหรับแต่ละเอสคิวแอลแอนติแพตเทิร์นโดยแยกเป็นอิสระจากกัน
9) หากพบเอสคิวแอลแอนติแพตเทิร์นหลายจุด เครื่องมือไม่จัดลำดับการทำรีแฟคทอริงฐานข้อมูลให้
10) เครื่องมือไม่รองรับการเกิดเอสคิวแอลแอนติแพตเทิร์นใหม่หลังจากการดำเนินการรีแฟคทอริง
11) การทดสอบและประเมินผลเครื่องมือจะใช้ฐานข้อมูลในอุตสาหกรรมจริงจำนวนอย่างน้อย 3 ฐานข้อมูล

1.4 ขั้นตอนการดำเนินงาน

1) ศึกษาและปรึกษาหัวข้อ
2) ศึกษางานวิจัยที่เกี่ยวข้อง
3) ออกแบบแนวคิดและการพัฒนา
4) ทดสอบแนวคิดเบื้องต้น
5) ร่างโครงร่างวิทยานิพนธ์
6) แก้ไขโครงร่างวิทยานิพนธ์
7) เตรียมสอบโครงร่างวิทยานิพนธ์
8) สอบโครงร่างวิทยานิพนธ์
9) แก้ไขโครงร่างวิทยานิพนธ์
10) ร่างบทความสำหรับประชุมวิชาการ
11) พัฒนาเครื่องมือ Phase I
12) ทดสอบเครื่องมือ Phase I
13) แก้ไขเครื่องมือตามคำแนะนำ Phase I
14) ร่างบทความสำหรับประชุมวิชาการเพิ่มเติม
15) พัฒนาเครื่องมือ Phase II
16) ทดสอบเครื่องมือ Phase II

17) แก้ไขเครื่องมือตามคำแนะนำ Phase II

18) เผยแพร่บทความวิชาการ

19) พัฒนาเครื่องมือ Phase III

20) ทดสอบเครื่องมือ Phase III

21) แก้ไขเครื่องมือตามคำแนะนำ Phase III

22) ทดลองดำเนินการเพื่อประเมินผล

23) แก้ไขเครื่องมือตามคำแนะนำ

24) ดำเนินการทำเอกสารวิทยานิพนธ์

1.5 ประโยชน์ที่คาดว่าจะได้รับ

ได้ขั้นตอนวิธีตรวจหาเอสคิวแอลแอนติแพตเทิร์น และเชื่อมโยงวิธีการแก้ไขเอสคิวแอลแอนติแพตเทิร์นกับวิธีรีแฟคทอริงฐานข้อมูล โดยพัฒนาเครื่องมือที่ช่วยตรวจหาเอสคิวแอลแอนติแพตเทิร์นและแนะนำกระบวนการรีแฟคทอริงฐานข้อมูล เพื่อช่วยให้ผู้ดูแลฐานข้อมูลนำไปช่วยในการปรับปรุงฐานข้อมูลได้

1.6 ผลงานตีพิมพ์

ส่วนหนึ่งของโครงงานมหาบัณฑิตนี้ได้รับการตอบรับเพื่อตีพิมพ์เป็นบทความวิจัยในหัวข้อเรื่อง “SQL Antipatterns Detection and Database Refactoring Process” โดย Poonyanuch Khumnin and Twittie Senivongse ในงานประชุมวิชาการ 18th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD) ซึ่งจัดขึ้นโดย IEEE Computer Society และ International Association for Computer and Information Science (ACIS) ณ เมืองคานาซาว่า (Kanazawa) ประเทศญี่ปุ่น ระหว่างวันที่ 26 –28 มิถุนายน 2560

บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 ทฤษฎีที่เกี่ยวข้อง

2.1.1 เอสคิวแอลแอนติแพตเทิร์น (SQL Antipatterns) [3]

แอนติแพตเทิร์น (AntiPattern) [6] คือ รูปแบบวิธีที่ต้องการแก้ไขปัญหาบางอย่างที่เกิดขึ้น แต่การแก้ไขปัญหานั้นส่งผลกระทบเป็นรูปแบบเชิงลบ หรือแอนติแพตเทิร์นอาจจะเป็นผลจากการที่ไม่มีความรู้หรือประสบการณ์มากพอในการใช้งาน โดยแอนติแพตเทิร์นจะมีการอธิบายโครงสร้างที่ประกอบไปด้วย รูปแบบที่ทำให้เกิดปัญหา วิธีการที่ทำให้ทราบว่าเกิดแอนติแพตเทิร์น ผลของการมีแอนติแพตเทิร์น และวิธีการแก้ไขแอนติแพตเทิร์น

เอสคิวแอลแอนติแพตเทิร์น (SQL Antipatterns) คือ การอธิบายประสบการณ์การใช้เอสคิวแอล (SQL: Structured Query Language) หรือภาษาคอมพิวเตอร์ที่ใช้ในการเขียนโปรแกรมเพื่อจัดการฐานข้อมูล ซึ่งผิดพลาด B. Karwin ได้รวบรวมเอสคิวแอลแอนติแพตเทิร์นที่พบจากประสบการณ์การทำงานเกี่ยวกับฐานข้อมูล มากกว่า 15 ปี รวมถึงคำถามจากอินเทอร์เน็ตที่มีจำนวน 1,000 กว่าคำถาม โดยสรุปไว้ในหนังสือซึ่งมีรายละเอียดแบ่งเป็น 4 กลุ่ม คือ แอนติแพตเทิร์นของการออกแบบฐานข้อมูลเชิงตรรกะ (Logical Database Design Antipatterns) แอนติแพตเทิร์นของการออกแบบฐานข้อมูลเชิงกายภาพ (Physical Database Design Antipatterns) แอนติแพตเทิร์นของการสืบค้น (Query Antipatterns) และแอนติแพตเทิร์นของการพัฒนาโปรแกรมประยุกต์ (Application Development Antipatterns) ซึ่งในแต่ละกลุ่มจะประกอบไปด้วยแอนติแพตเทิร์นต่าง ๆ และแต่ละแอนติแพตเทิร์นนั้นจะมีรายละเอียด 5 ส่วนประกอบไปด้วย 1) จุดประสงค์ในการใช้งานแล้วทำให้เกิดแอนติแพตเทิร์น (Objective) 2) แอนติแพตเทิร์นนี้ทำให้เกิดปัญหาอย่างไร (Antipattern) 3) วิธีการที่ทำให้ทราบว่าเกิดแอนติแพตเทิร์น (How to Recognize Antipattern) 4) การใช้งานที่เหมาะสมของแอนติแพตเทิร์น (Legitimate Uses of Antipattern) 5) วิธีการแก้ไขแอนติแพตเทิร์น (Solution)

ในงานวิจัยนี้ผู้วิจัยมุ่งเน้นที่กลุ่มแอนติแพตเทิร์นของการออกแบบฐานข้อมูลเชิงตรรกะทั้งหมด 8 ประเภทและอ้างอิงตัวอย่างจากหนังสือของ B. Karwin [3] เพื่อประกอบการอธิบาย โดยสรุปรายละเอียดไว้ดังนี้

1) Format Comma-Separated Lists

การออกแบบที่มีแอนติแพตเทิร์น Format Comma-Separated Lists แสดงในตัวอย่างในรูปที่ 1 กล่าวคือ คอลัมน์ account_id ถูกออกแบบให้เก็บข้อมูลเพื่อเชื่อมโยงไป

ตารางอื่น แต่ข้อมูลเป็นรายการ และมีการออกแบบเครื่องหมายสำหรับคั่นรายการ เช่น เครื่องหมาย , (Comma) โดยการออกแบบที่มีคอลัมน์แบบนี้เรียกได้ว่าเป็นแอนติแพตเทิร์น โดยแสดงตัวอย่างเพิ่มเติมเพื่อให้เห็นข้อมูลของคอลัมน์นี้ในรูปที่ 2 ตัวอย่างค่าของข้อมูลที่ใส่ในคอลัมน์ account_id คือ 12,34 โดยที่มีการใส่เครื่องหมายคั่นระหว่าง account_id ที่มีค่า 12 และ 24 สำหรับสินค้าเดียวกันที่ชื่อว่า 'Visual TurboBuilder' โดยค่า 12 เป็น account_id ที่สามารถเชื่อมไปที่ตาราง account และค่า 24 ก็เชื่อมไปได้เช่นกัน ทั้งนี้เพื่อเก็บข้อมูลว่า 'Visual TurboBuilder' มี account_id คือ 12 และ 24

```
CREATE TABLE Products (
  product_id    SERIAL PRIMARY KEY,
  product_name VARCHAR(1000),
  account_id    VARCHAR(100), -- comma-separated list
);
```

รูปที่ 1 ตัวอย่างเอสคิวแอลแอนติแพตเทิร์น Format Comma-Separated Lists [3]

```
INSERT INTO Products (product_id, product_name, account_id)
VALUES (DEFAULT, 'Visual TurboBuilder', '12,34');
```

รูปที่ 2 ตัวอย่างข้อมูลของการออกแบบเอสคิวแอลแอนติแพตเทิร์น Format Comma-Separated Lists [3]

ตัวอย่างปัญหาเมื่อมีการออกแบบที่ทำให้เกิดแอนติแพตเทิร์นนี้ เช่น

- ยากต่อการสอบถามข้อมูล account_id จากตาราง Products โดยหากต้องการค้นหาข้อมูล account_id ตัวหนึ่งในรายการข้อมูล จะต้องแยกรายการนั้นเพื่อสืบค้นข้อมูล จึงค่อนข้างยาก และถ้าหากเครื่องหมายคั่นเป็นส่วนหนึ่งของค่าข้อมูลแต่ละตัวเอง จะทำให้การแยกรายการข้อมูลอาจเกิดโอกาสผิดพลาดได้
- ยากต่อการตรวจสอบข้อมูลเข้า เช่น จากรูปที่ 3 มีการเพิ่มข้อมูล account_id และใส่ข้อมูลคำว่า “banana” ที่เป็นค่าที่ไม่ถูกต้องเพราะไม่สามารถเชื่อมโยงไปยังตาราง account ได้ แต่ฐานข้อมูลไม่สามารถตรวจสอบความผิดพลาดนี้ได้ เนื่องจากออกแบบการเก็บข้อมูลเป็นรายการนี้ด้วยประเภทข้อมูลเป็นตัวอักษรไว้ จึงยอมรับค่าและใส่เข้าไปในฐานข้อมูลได้

```
INSERT INTO Products (product_id, product_name, account_id) VALUES
(DEFAULT, 'Visual TurboBuilder' , '12,34,banana' );
```

รูปที่ 3 ตัวอย่างข้อมูลที่ผิดจากการออกแบบที่มีเอสคิวแอลแอนติแพตเทิร์น Format Comma-Separated Lists [3]

2) Always Depend on One's Parent

การออกแบบที่มีแอนติแพตเทิร์น Always Depend on One's Parent แสดงในตัวอย่างในรูปที่ 4 กล่าวคือ ตาราง Comments มีการเก็บข้อมูลในลักษณะเป็นต้นไม้ไว้ในตารางโดยคอลัมน์ comment_id เป็นข้อมูลที่จะขึ้นต่อหรืออ้างอิงไปยังข้อมูลแถวอื่นที่เป็นบรรพบุรุษคือคอลัมน์ชื่อว่า parent_id โดยการออกแบบที่มีคอลัมน์แบบนี้เรียกได้ว่าเป็นแอนติแพตเทิร์น และจากตารางที่ 1 จะแสดงตัวอย่างข้อมูลที่ขึ้นต่อกัน เช่น comment_id ที่มีค่า 2 และ 4 จะขึ้นกับ parent_id ที่มีค่า 1

```
CREATE TABLE Comments (
  comment_id   SERIAL PRIMARY KEY,-- hierarchical data
  parent_id    BIGINT UNSIGNED, --parent
  comment_date DATETIME NOT NULL,
  comment      TEXT NOT NULL,
  FOREIGN KEY (parent_id) REFERENCES Comments(comment_id), );
```

รูปที่ 4 ตัวอย่างเอสคิวแอลแอนติแพตเทิร์น Always Depend on One's Parent [3]

ตารางที่ 1 ตัวอย่างข้อมูลของ Always Depend on One's Parent [3]

comment_ID	parent_ID
1	NULL
2	*1*
3	2
4	*1*
5	4
6	4

ตัวอย่างปัญหาเมื่อมีการออกแบบที่ทำให้เกิดแอนติแพตเทิร์นนี้ เช่น

- ยากต่อการลบข้อมูล เช่น ในการลบข้อมูลบางกิ่งของต้นไม้ จะต้องทำการสอบถามก่อนว่ามีข้อมูลแถวใดบ้างที่อยู่ภายใต้กิ่งนั้น โดยแสดงตัวอย่างเอสคิวแอลการค้นหากิ่งทั้งหมด ก่อนที่จะทำการลบช้อมูลในรูปที่ 5 โดยอ้างอิงข้อมูลจากตารางที่ 1

```
SELECT comment_id FROM Comments WHERE parent_id = 4; -- returns 5 and 6
SELECT comment_id FROM Comments WHERE parent_id = 5; -- returns none

DELETE FROM Comments WHERE comment_id IN ( 5, 6 );
DELETE FROM Comments WHERE comment_id = 4;
```

รูปที่ 5 ตัวอย่างปัญหาการใช้งานเมื่อมีเอสคิวแอลแอนติแพตเทิร์น Always Depend on One's Parent [3]

3) One Size Fits All

การออกแบบที่มีแอนติแพตเทิร์น One Size Fits All แสดงในตัวอย่างในรูปที่ 6 กล่าวคือ ตาราง BugsProducts มีการออกแบบให้มีข้อมูล id เพิ่มขึ้นมาเพื่อใช้เป็นค่าเอกลักษณ์และกำหนดให้เป็นกุญแจหลัก (Primary key) แต่ตารางนี้ยังมีข้อมูล bug_id ซึ่งสามารถใช้เป็นค่าเอกลักษณได้อยู่แล้วแต่ไม่ได้กำหนดเป็นกุญแจหลัก และจากรูปที่ 7 ตัวอย่างค่าของข้อมูลที่ใส่ในคอลัมน์ id คือ 1 และกำหนดให้เป็นกุญแจหลักแต่มี bug_id คือ 'VIS-078' ที่สามารถเป็นกุญแจหลักได้โดยธรรมชาติของข้อมูลเนื่องจากมีลักษณะเป็นเอกลักษณ์อยู่แล้วแต่ไม่ได้กำหนดให้เป็นกุญแจหลัก

```
CREATE TABLE BugsProducts (
id SERIAL PRIMARY KEY,
bug_id BIGINT UNSIGNED NOT NULL,
product_id BIGINT UNSIGNED NOT NULL,
FOREIGN KEY (bug_id) REFERENCES Bugs(bug_id),
FOREIGN KEY (product_id) REFERENCES Products(product_id)
);
```

รูปที่ 6 ตัวอย่างเอสคิวแอลแอนติแพตเทิร์น One Size Fits All [3]

```
INSERT INTO BugsProducts (id,bug_id, description,.)
VALUES (1,'VIS-078','crashes on save',..);
```

รูปที่ 7 ตัวอย่างข้อมูลของการออกแบบเอสคิวแอลแอนติแพตเทิร์น One Size Fits All [3]

ตัวอย่างปัญหาเมื่อมีการออกแบบที่ทำให้เกิดแอนติแพตเทิร์นนี้ เช่น

- ยากต่อการตรวจสอบการใส่ข้อมูลเพิ่มเพราะอาจจะทำให้เพิ่มข้อมูลที่ต้องการให้เป็นเอกลักษณ์ซ้ำกันได้ เช่น จากตารางที่มีคอลัมน์เอกลักษณ์ id ซึ่งกำหนดเป็นกุญแจหลัก และคอลัมน์ bug_id ซึ่งไม่ได้กำหนดให้เป็นกุญแจหลัก แต่เมื่อมีการเพิ่มข้อมูลใหม่ คอลัมน์ id จะถูกตรวจสอบไม่ให้กำหนดค่าซ้ำกัน ในขณะที่คอลัมน์ bug_id ไม่ถูกตรวจสอบและจะมีค่าซ้ำได้ ซึ่งการใช้งานจริงนั้นไม่ต้องการให้ซ้ำ จึงทำให้อาจเกิดข้อผิดพลาดได้ ตามรูปที่ 8

```
INSERT INTO BugsProducts (bug_id, product_id)
VALUES (1234, 1), (1234, 1), (1234, 1); -- duplicates are permitted
```

รูปที่ 8 ตัวอย่างการเพิ่มข้อมูลลงในการออกแบบที่มีเอสคิวแอลแอนติแพตเทิร์น One Size Fits All [3]

4) Leave Out the Constraints

การออกแบบที่มีแอนติแพตเทิร์น Leave Out the Constraints แสดงในตัวอย่างในรูปที่ 9 กล่าวคือ มีการออกแบบตาราง Bugs โดยไม่กำหนดค่าคอนสเตรนต์ เช่น Foreign Key Constraint ทั้งๆที่ควรกำหนดให้คอลัมน์ status เป็น Foreign Key ที่จะเชื่อมโยงไปยังตาราง Bugstatus ได้ โดยการออกแบบที่มีคอลัมน์แบบนี้เรียกได้ว่าเป็นแอนติแพตเทิร์น

```
CREATE TABLE Bugs (
reported_by BIGINT UNSIGNED NOT NULL,
status VARCHAR(20) NOT NULL DEFAULT 'NEW' ); -- Reference Table BugStatus
```

รูปที่ 9 ตัวอย่างการเพิ่มข้อมูลลงในการออกแบบที่มีเอสคิวแอลแอนติแพตเทิร์น Leave Out the Constraints [3]

ตัวอย่างปัญหาเมื่อมีการออกแบบที่ทำให้เกิดแอนติแพตเทิร์นนี้ เช่น

- ยากต่อการเพิ่มข้อมูลและตรวจสอบข้อมูลที่ผิดพลาด เช่น ในการเพิ่มและตรวจสอบข้อมูลซึ่งไม่มีการกำหนด Foreign Key ไว้ จะทำให้ต้องทำการสอบถามเพื่อดูข้อมูลที่สัมพันธ์กันว่ามีอยู่ในฐานข้อมูลแล้วหรือไม่ เช่น หากต้องการที่จะแก้ไขตาราง Bugs ก็จะต้องทำการสอบถามดังรูปที่ 10 ทุกครั้งเพื่อแสดงข้อมูลที่เชื่อมโยงกันอยู่ระหว่างตาราง Bugs และ BugStatus

```
SELECT b.bug_id, b.status
FROM Bugs b LEFT OUTER JOIN BugStatus s
ON (b.status = s.status)
WHERE s.status IS NULL;
```

รูปที่ 10 ตัวอย่างการสอบถามข้อมูลที่มีการออกแบบเอสคิวแอลแอนติแพตเทิร์น Leave Out the Constraints [3]

5) Use a Generic Attribute Table

การออกแบบที่มีแอนติแพตเทิร์น Use a Generic Attribute Table แสดงในตัวอย่างในรูปที่ 11 กล่าวคือ มีการออกแบบตาราง IssueAttributes ให้มีคอลัมน์เพื่อเก็บชื่อแอตทริบิวต์ และค่าของแอตทริบิวต์ไว้อย่างละคอลัมน์ โดยการออกแบบที่มีคอลัมน์แบบนี้เรียกได้ว่าเป็นแอนติแพตเทิร์น ตัวอย่างการเก็บข้อมูลที่ออกแบบด้วยแอนติแพตเทิร์นนี้แสดงไว้ในรูปที่ 12 ซึ่งจะพบว่าข้อมูลคอลัมน์ attr_name จะเก็บชื่อแอตทริบิวต์ เช่น

'date_reported', ‘status', ‘description’ และ attr_value จะเก็บค่าของแอตทริบิวต์ตามชื่อแอตทริบิวต์ข้างต้น

```
CREATE TABLE IssueAttributes (
issue_id BIGINT UNSIGNED NOT NULL,
attr_name VARCHAR(100) NOT NULL,
attr_value VARCHAR(100),
PRIMARY KEY (issue_id, attr_name),
FOREIGN KEY (issue_id) REFERENCES Issues(issue_id) );
```

รูปที่ 11 ตัวอย่างเอสคิวแอลแอนติแพตเทิร์น Use a Generic Attribute Table [3]

```
INSERT INTO IssueAttributes (issue_id, attr_name, attr_value)
VALUES
        (1234, 'product' , '1' ),
        (1234, 'date_reported' , '2009-06-01' ),
        (1234, 'status' , 'NEW' ),
        (1234, 'description' , 'Saving does not work' ),
        (1234, 'reported_by' , 'Bill' ),
        (1234, 'version_affected' , '1.0' ),
        (1234, 'severity' , 'loss of functionality' ),
        (1234, 'priority' , 'high' );
```

รูปที่ 12 ตัวอย่างข้อมูลของการออกแบบเอสคิวแอลแอนติแพตเทิร์น Use a Generic Attribute Table [3]

ตัวอย่างปัญหาเมื่อมีการออกแบบที่ทำให้เกิดแอนติแพตเทิร์นนี้ เช่น

- ยากต่อการตรวจสอบข้อมูล จากรูปที่ 13 หากต้องการเก็บแอตทริบิวต์ 'date_reported' และค่าของแอตทริบิวต์นั้นควรจะเป็นข้อมูลวันที่ แต่ใส่ข้อมูลผิดเข้ามาเป็นคำว่า ‘banana’ ซึ่งควรต้องมีการแจ้งเตือนและต้องไม่สามารถใส่ข้อมูล

ได้ แต่เมื่อมีแอนติแพตเทิร์นนี้ จะสามารถใส่ข้อมูลผิดลงในฐานข้อมูลได้ ทำให้การตรวจสอบข้อมูลนำเข้าตามประเภทที่ต้องการทำได้ยาก เพราะออกแบบประเภทข้อมูลเป็นข้อความสำหรับเก็บค่าของแอตทริบิวต์ทั้งหมด

```
INSERT INTO IssueAttributes (issue_id, attr_name, attr_value)
VALUES (1234, 'date_reported' , 'banana' ); -- Not an error!
```

รูปที่ 13 ตัวอย่างการเพิ่มข้อมูลลงในการออกแบบที่มีเอสคิวแอลแอนติแพตเทิร์น Use a Generic Attribute Table [3]

6) Use Dual-Purpose Foreign Key

การออกแบบที่มีแอนติแพตเทิร์น Use Dual-Purpose Foreign Key แสดงในตัวอย่างในรูปที่ 14 กล่าวคือ มีการออกแบบตาราง Comments ที่มีคอลัมน์ issue_type เพื่อระบุการเชื่อมโยงไปยังตารางอื่นได้มากกว่าหนึ่งตารางคือ ตาราง "Bugs" หรือ "FeatureRequests" และจะทำให้คอลัมน์ issue_id เป็น id ที่มีสองความหมายโดยขึ้นกับว่า issue_type นั้นระบุการเชื่อมโยงไปยังตารางใด โดยการออกแบบที่มีคอลัมน์แบบนี้เรียกได้ว่าเป็นแอนติแพตเทิร์น

```
CREATE TABLE Comments (
comment_id SERIAL PRIMARY KEY,
issue_type VARCHAR(20), -- "Bugs" or "FeatureRequests"
issue_id BIGINT UNSIGNED NOT NULL,
author BIGINT UNSIGNED NOT NULL,
comment_date DATETIME,
comment TEXT,
FOREIGN KEY (author) REFERENCES Accounts(account_id));
```

รูปที่ 14 ตัวอย่างข้อมูลของการออกแบบเอสคิวแอลแอนติแพตเทิร์น Use Dual-Purpose Foreign Key [3]

ตัวอย่างปัญหาเมื่อมีการออกแบบที่ทำให้เกิดแอนติแพตเทิร์นนี้ เช่น

- ยากต่อการสอบถามข้อมูล เช่น เมื่อมี issue_id ที่สามารถเชื่อมโยงข้อมูลไปได้สองตาราง จึงต้องทำการสอบถามข้อมูลนั้นด้วยการใส่เงื่อนไขเป็นชื่อตารางเสมอเพื่อเชื่อมโยงไปดังรูปที่ 15

```
SELECT *
FROM Comments AS c
LEFT OUTER JOIN Bugs AS b
ON (b.issue_id = c.issue_id AND c.issue_type = 'Bugs' )
LEFT OUTER JOIN FeatureRequests AS f
ON (f.issue_id = c.issue_id AND c.issue_type = 'FeatureRequests' );
```

รูปที่ 15 ตัวอย่างการสอบถามข้อมูลทีมีเอสคิวแอลแอนติแพตเทิร์น Use Dual-Purpose Foreign Key [3]

7) Create Multiple Columns

การออกแบบที่มีแอนติแพตเทิร์น Create Multiple Columns แสดงในตัวอย่างในรูปที่ 16 กล่าวคือ มีการออกแบบตาราง Bugs ให้เก็บข้อมูล tag ที่มีลักษณะเดียวกันโดยแยกเป็นหลายคอลัมน์ การออกแบบที่มีคอลัมน์แบบนี้เรียกได้ว่าเป็นแอนติแพตเทิร์น

```
CREATE TABLE Bugs (
bug_id SERIAL PRIMARY KEY,
description VARCHAR(1000),
tag1 VARCHAR(20),
tag2 VARCHAR(20),
tag3 VARCHAR(20));
```

รูปที่ 16 ตัวอย่างข้อมูลของการออกแบบเอสคิวแอลแอนติแพตเทิร์น Create Multiple Columns [3]

ตัวอย่างปัญหาเมื่อมีการออกแบบที่ทำให้เกิดแอนติแพตเทิร์นนี้ เช่น

- ยากต่อการสอบถามข้อมูล เช่น หากต้องการสอบถามข้อมูลเกี่ยวกับ tag ทั้งหมด จะต้องใส่เงื่อนไขเพื่อหาให้ครบทุกคอลัมน์ โดยแสดงดังรูปที่ 17

```
SELECT * FROM Bugs
WHERE tag1 = 'performance'
OR tag2 = 'performance'
OR tag3 = 'performance' ;
```

รูปที่ 17 ตัวอย่างการสืบค้นข้อมูลทีมีเอสคิวแอลแอนติแพตเทิร์น Create Multiple Columns [3]

8) Clone Tables or Columns

การออกแบบที่มีแอนติแพตเทิร์น Clone Tables or Columns แสดงในตัวอย่างในรูปที่ 18 กล่าวคือ มีการออกแบบตาราง Bugs หลายตาราง ซึ่งมีโครงสร้างเหมือนกันเพื่อเก็บข้อมูลในลักษณะเดียวกัน โดยการออกแบบที่มีคอลัมน์แบบนี้เรียกได้ว่าเป็นแอนติแพตเทิร์น

```
CREATE TABLE Bugs_2008 ( . . . );
CREATE TABLE Bugs_2009 ( . . . );
CREATE TABLE Bugs_2010 ( . . . );
```

รูปที่ 18 ตัวอย่างข้อมูลของการออกแบบเอสคิวแอลแอนติแพตเทิร์น Clone Tables or Columns [3]

ตัวอย่างปัญหาเมื่อมีการออกแบบที่ทำให้เกิดแอนติแพตเทิร์นนี้ เช่น

- ยากต่อการแก้ไขตาราง เช่น หากต้องการแก้ไขโครงสร้าง จะจำเป็นต้องแก้ไขให้ครบทุกตารางที่สร้างไว้เหมือนกัน

ตารางที่ 2 แสดงตัวอย่างของรายละเอียดตามโครงสร้างของแอนติแพตเทิร์นซึ่งกำหนดโดย B. Karwin [3] โดยเป็นตัวอย่างของแอนติแพตเทิร์น Clone Tables or Columns

ตารางที่ 2 ข้อมูลแอนติแพตเทิร์น : Clone Tables or Columns

รายการตามโครงสร้าง	รายละเอียดตามโครงสร้าง
จุดประสงค์ในการใช้งานแล้วทำให้เกิดแอนติแพตเทิร์น (Objective)	ต้องการเพิ่มสมรรถนะของข้อมูลจึงแยกปีหรือเวอร์ชันของตารางไว้ เพื่อดึงข้อมูลมาใช้เฉพาะปีหรือเวอร์ชันที่ต้องการ เพื่อให้ไม่ต้องสืบค้นไปยังข้อมูลทั้งหมด

แอนติแพตเทิร์นนี้ทำให้เกิดปัญหาอย่างไร (Antipattern)	• ตัวอย่างเอสคิวแอลการสร้างตารางที่มีโครงสร้างเหมือนกัน
	`CREATE TABLE Bugs_2008 ( . . . );`
	`CREATE TABLE Bugs_2009 ( . . . );`
	`CREATE TABLE Bugs_2010 ( . . . );`
	• ยากต่อการแก้ไขตารางเช่น หากต้องการแก้ไขโครงสร้าง จะจำเป็นต้องแก้ไขให้ครบทุกตารางที่สร้างไว้เหมือนกัน
	• ยากต่อการสอบถามข้อมูล เช่น หากต้องการข้อมูลทั้งหมดก็ต้องสอบถามโดยใช้เงื่อนไขที่เชื่อมทุกตาราง

วิธีการที่ทำให้ทราบว่าเกิดแอนติแพตเทิร์น (How to Recognize Antipattern)	เมื่อมีคำพูดเหล่านี้เกิดขึ้น จะเป็นสัญญาณบอกถึงแอนติแพตเทิร์นนี้ • เราต้องสร้างตารางแยกตามเงื่อนไขอะไร ปี หรือเวอร์ชัน • เราต้องสร้างตารางมากที่สุดกี่ตารางถึงจะพอต่อการใช้งาน • เราพบว่าระบบล่มเนื่องจากไม่มีตารางตามปีที่ระบุเพราะมีการสอบถามข้อมูลในปีที่ไม่ได้สร้างตารางไว้ • เราจะค้นหาข้อมูลในตารางทั้งหมดในครั้งเดียวอย่างไร เมื่อทุกตารางมีคอลัมน์เหมือนกัน

ตารางที่ 2 ข้อมูลแอนติแพตเทิร์น : Clone Tables or Columns (ต่อ)

รายการตามโครงสร้าง	รายละเอียดตามโครงสร้าง
การใช้งานที่เหมาะสมของแอนติแพตเทิร์น (Legitimate Uses of Antipattern)	ใช้สำหรับแยกข้อมูลเก่าออกจากข้อมูลปัจจุบันโดยนำไปเก็บไว้ที่อื่น การวิเคราะห์ข้อมูลเก่ายังสามารถทำได้ด้วยโครงสร้างตารางแบบเดิม และทำให้การสอบถามข้อมูลปัจจุบันมีสมรรถนะที่ดีกว่าการเก็บข้อมูลเก่าและปัจจุบันรวมกัน

วิธีการแก้ไขแอนติแพตเทิร์น (Solution)	สร้างตารางเป็นตารางเดียว และเพิ่มคอลัมน์ที่ระบุลักษณะ เช่น ปี เวอร์ชัน โดยตัวอย่างเป็นเอสคิวแอล คือ
	`CREATE TABLE ProjectHistory (`
	`  project_id  BIGINT,`
	`  year  SMALLINT,`
	`  bugs_fixed  INT,`
	`  PRIMARY KEY (project_id, year),`
	`  FOREIGN KEY (project_id) REFERENCES Projects(project_id));`

2.1.2 การรีแฟคทอริงฐานข้อมูล (Database Refactoring)

M. Fowler [1] กล่าวถึงการรีแฟคทอริงในโค้ดไว้ว่า คือ กระบวนการเปลี่ยนแปลงระบบของซอฟต์แวร์ที่เป็นการแก้ไขโดยไม่เปลี่ยนพฤติกรรมของการทำงานแต่เป็นการทำให้โครงสร้างภายในของซอฟต์แวร์ดีขึ้น ในทำนองเดียวกัน S. J. Ambler [4] นิยามการรีแฟคทอริงฐานข้อมูลไว้ว่าเป็นการเปลี่ยนแปลงสกีมาของฐานข้อมูลเพื่อทำให้มีการออกแบบที่ดีขึ้น โดยยังคงพฤติกรรมและความหมายของข้อมูลอยู่ตามเดิม โดยสามารถทำการรีแฟคทอริงได้ทั้งโครงสร้าง เช่น ตาราง (Table) วิว (View) และในการทำงานเชิงฟังก์ชัน เช่น ทริกเกอร์ (Trigger) สตอร์โปรซีเจอร์ (Stored Procedure) การรีแฟคทอริงฐานข้อมูลนั้นไม่ใช่แค่การดำเนินการกับฐานข้อมูล แต่ยังรวมระบบที่เชื่อมต่อด้วย เช่น แอปพลิเคชัน ซึ่งทำให้การรีแฟคทอริงฐานข้อมูลนั้นมีความแตกต่างจากโค้ด เพราะต้องจัดเตรียมการทั้งในฐานข้อมูลและระบบที่ติดต่ออย่างระมัดระวัง โดยกลุ่มของการรีแฟคทอริงแบ่งเป็น 5 กลุ่ม คือ

1) Structural Refactorings คือ การเปลี่ยนแปลงโครงสร้างตารางของสกีมาในฐานข้อมูล
2) Data Quality Refactorings คือ การเปลี่ยนแปลงเพื่อปรับปรุงหรือทำให้เกิดความต้องกันของค่าข้อมูล

3) Referential Integrity Refactorings คือ การเปลี่ยนแปลงเพื่อให้แน่ใจว่าการอ้างอิงไปยังตารางอื่นมีความถูกต้องสอดคล้อง
4) Architectural Refactorings คือ การเปลี่ยนแปลงเพื่อปรับปรุงการเชื่อมต่อของซอฟต์แวร์ภายนอกกับฐานข้อมูล
5) Method Refactorings คือ การเปลี่ยนแปลงเพื่อปรับปรุงฟังก์ชัน ทริกเกอร์ สตอร์โปรซีเจอร์

ผู้วิจัยจะพิจารณาเฉพาะกลุ่มการรีแฟคทอริงกลุ่ม 1) - 3) โดยแสดงในตารางที่ 3

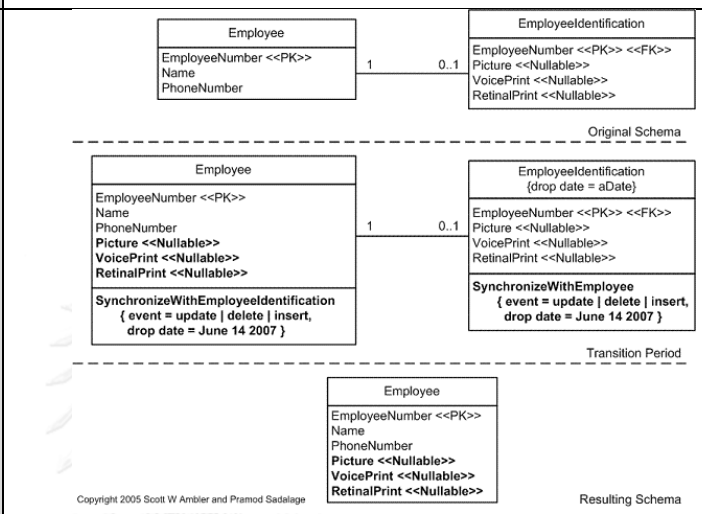
ตารางที่ 3 รายการรีแฟคทอริงฐานข้อมูล

กลุ่ม	วิธีรีแฟคทอริงฐานข้อมูล
Structural Refactorings (จำนวน 17 วิธี)	Drop Column, Drop Table, Drop View, Introduce Calculated Column, Introduce Surrogate Key, Merge Columns, Merge Tables, Move Column, Rename Column, Rename Table, Rename View, Replace LOB With Table, Replace Column, Replace One-to-Many With Associative Table, Replace Surrogate Key with Natural Key, Split Column, Split Table
Data Quality Refactorings (จำนวน 13 วิธี)	Add Lookup Table, Apply Standard Codes, Apply Standard Type, Consolidate Key Strategy, Drop Column Constraint Drop Default Value, Drop Non-Nullable Constraint, Introduce Column Constraint, Introduce Common Format, Introduce Default Value, Make Column Non-Nullable, Move Data, Replace Type Code With Property Flags
Referential Integrity Refactorings (จำนวน 7 วิธี)	Add Foreign Key Constraint, Add Trigger for Calculated Column, Drop Foreign Key Constraint, Introduce Cascading Delete, Introduce Hard Delete, Introduce Soft Delete, Introduce Trigger for History

จากตารางข้างต้นจะมีการนิยามกระบวนการในแต่ละการรีแฟคทอริงฐานข้อมูลแต่ละตัวไว้ 3 ขั้นตอน คือ การปรับสกีมา (Schema Update Mechanics) การย้ายข้อมูล (Data Migration

Mechanics) และการปรับโปรแกรมที่ใช้เข้าถึง (Access Program Update Mechanics) โดยยกตัวอย่างแสดงให้เห็นรายละเอียดโดยสรุปของวิธีการรีแฟคทอริงฐานข้อมูล 1 วิธีคือ Merge Tables ตามตารางที่ 4

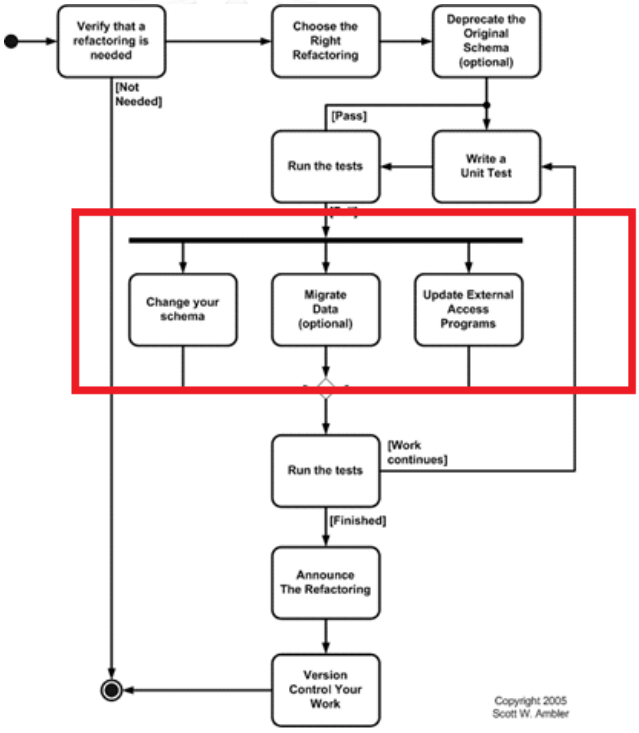
ตารางที่ 4 รายละเอียดวิธีรีแฟคทอริงฐานข้อมูล : Merge Tables ใน 3 ขั้นตอน

การรีแฟคทอริงฐานข้อมูล	Merge Tables
ตารางตัวอย่าง	
การปรับสกีมา (Schema Update Mechanics)	1) พิจารณาตารางสองตารางที่ต้องการรวมกันจากโครงสร้างตัวอย่าง ตาราง Employee และ ตาราง EmployeeIdentification ซึ่งจะรวมข้อมูลเข้าด้วยกัน หากมีคอลัมน์ที่ต่างกันต้องดำเนินการเพิ่มคอลัมน์ (Add Column) ในตารางที่ต้องการให้เป็นตารางรวมคือ Employee ซึ่งจากโครงสร้างตัวอย่าง ทำให้ต้องเพิ่มคอลัมน์ Picture, VoicePrint, RetinalPrint ให้ตาราง Employee 2) สร้างทริกเกอร์เพื่อให้ Employee ทราบถึงการเปลี่ยนแปลง คือ หากตาราง EmployeeIdentification มีการเปลี่ยนแปลง ตาราง Employee ต้องทราบถึงการเปลี่ยนแปลง
การย้ายข้อมูล (Data Migration Mechanics)	สำเนา (Copy) ข้อมูลจากตารางต้นฉบับ (Original Table) ซึ่งก็คือ EmployeeIdentification ไปยังตารางรวม (Merge table) ซึ่งก็คือ Employee

ตารางที่ 4 รายละเอียดวิธีรีแฟคทอริงฐานข้อมูล : Merge Tables ใน 3 ขั้นตอน (ต่อ)

การปรับโปรแกรมที่ใช้เข้าถึง (Access Program Update Mechanics)	1) ปรับโปรแกรมที่มีการเข้าถึงข้อมูลในสองตารางเดิมเหลือเพียงการเข้าถึงตารางรวมเพียงตารางเดียว 2) ตรวจสอบโปรแกรมที่เข้าถึงตารางรวม ว่าหากมีคอลัมน์ใหม่ในตารางรวมเกิดขึ้น คอลัมน์ใหม่นั้นจำเป็นต้องใช้หรือไม่ เช่น หากโปรแกรมเข้าถึงข้อมูล Employee เฉพาะในคอลัมน์ที่มีอยู่เดิม โดยไม่ใช้คอลัมน์ใหม่ที่เพิ่มขึ้นมา ควรรีแฟคทอริงคอลัมน์ใหม่ด้วยวิธี Introduce Default Value ด้วย

โดย 3 ขั้นตอนนี้เป็นส่วนสำคัญของกระบวนการรีแฟคทอริงฐานข้อมูลทั้งหมดตามรูปที่ 19 ที่จะต้องดำเนินการในเครื่องทดสอบก่อน



รูปที่ 19 กระบวนการรีแฟคทอริงฐานข้อมูล [4]

จากรูปที่ 19 กระบวนการรีแฟคทอริงฐานข้อมูลจะประเมินว่าจำเป็นที่ต้องทำรีแฟคทอริงฐานข้อมูลและเลือกรีแฟคทอริงที่เหมาะสม หลังจากนั้นอาจจะดำเนินการเลิกใช้สกีมาต้นฉบับด้วยถ้าจำเป็น และเขียนการทดสอบ ดำเนินการทดสอบ หากไม่ผ่านจะดำเนินการแก้ไขสกีมา ย้ายข้อมูล (ถ้าจำเป็น) ปรับโปรแกรมที่ใช้เข้าถึง ดำเนินการทดสอบซ้ำจนเสร็จสิ้น ประกาศสิ่งที่เปลี่ยนแปลง และสุดท้ายทำการควบคุมเวอร์ชัน

ทั้งนี้ S. J. Ambler ได้กล่าวถึง ร่องรอยที่ไม่ดีในฐานข้อมูล (Database Smell) ไว้เช่นกัน โดยกล่าวไว้เพียงกว้าง ๆ ทั้งหมด 7 แบบคือ Multi-Purpose Column, Multi-purpose Table, Redundant Data, Tables with Many Columns, Tables with Many Rows, "Smart" Columns และ Fear of Change แต่ไม่มีรายละเอียดแบบที่ B. Karwin นิยามไว้

2.1.3 ทรานแซกเอสคิวแอล (Transact-SQL หรือ T-SQL) [5]

ทรานแซก-เอสคิวแอลหรือที-เอสคิวแอล เป็นส่วนต่อขยาย (Extension) ของเอสคิวแอล สำหรับใช้ติดต่อกับฐานข้อมูลเชิงสัมพันธ์ในไมโครซอฟต์เอสคิวแอลเซิร์ฟเวอร์ (Microsoft SQL Server) ซึ่งเป็นโปรแกรมประยุกต์สำหรับจัดการฐานข้อมูลเชิงสัมพันธ์ที่พัฒนาโดยบริษัทไมโครซอฟต์ ทรานแซก-เอสคิวแอลใช้สำหรับสื่อสารกับอินสแตนซ์ของเอสคิวแอลเซิร์ฟเวอร์ โดยสามารถนำเข้าข้อมูลมาเพื่อประมวลผล และประกาศตัวแปรได้ สร้างคำสั่ง สร้างฟังก์ชันสำหรับการประมวลผล และแสดงผลข้อมูลได้ สามารถทำให้จัดการการสืบค้นที่ซับซ้อนได้โดยไม่ต้องเขียนหลายเงื่อนไข และสั่งประมวลผลข้อมูลได้ตามที่ต้องการ ทำให้ผู้ดูแลฐานข้อมูลสามารถเขียนโปรแกรมเพื่อจัดการข้อมูลได้อีกด้วย [7]

2.1.4 การประเมินประสิทธิภาพจากเมตริกซ์ความสับสน (Confusion Matrix)

การประเมินประสิทธิภาพผลลัพธ์จากเครื่องมือตรวจหาแอนติแพตเทิร์นในงานวิจัยนี้จะเปรียบเทียบกับคำตอบที่ตรวจหาด้วยมือ โดยผลที่ได้อยู่ในรูปของเมตริกซ์ความสับสนดังตารางที่ 5

ตารางที่ 5 ค่าเมตริกซ์ความสับสน

เมตริกซ์ความสับสน		ค่าจากโปรแกรม	
		จริง	เท็จ
ค่าจริง	จริง	True Positive (TP)	False Positive (FP)
	เท็จ	False Negative (FN)	True Negative (TN)

True Positive (TP) คือ สิ่งที่โปรแกรมทำนายว่าจริง (เป็นแอนติแพตเทิร์น) และคำตอบบอกว่าจริง

True Negative (TN) คือ สิ่งที่โปรแกรมทำนายว่าไม่จริง (ไม่เป็นแอนติแพตเทิร์น) และคำตอบบอกว่าไม่จริง

False Positive (FP) คือ สิ่งที่โปรแกรมทำนายว่าจริง (เป็นแอนติแพตเทิร์น) แต่คำตอบบอกว่าไม่จริง

False Negative (FN) คือ สิ่งที่โปรแกรมทำนายว่าไม่จริง (ไม่เป็นแอนติแพตเทิร์น) แต่คำตอบบอกว่าจริง

จากข้อมูลในตารางนำไปคำนวณค่าความเที่ยงตรง (Precision) ค่าเรียกคืน (Recall) และค่าประสิทธิภาพโดยรวมของค่าความเที่ยงตรงและค่าเรียกคืน หรือ เอฟ-เมเชอร์ (F-measure) ตามสมการ ***(1) (2)*** และ ***(3)***

$$Precision = TP/TP+FP \quad (1)$$

$$Recall = TP/TP+FN \quad (2)$$

$$F\text{-}measure = 2x\ Recall\ x\ Precision\ /\ (Recall + Precision) \quad (3)$$

2.2 งานวิจัยที่เกี่ยวข้อง

การทบทวนงานวิจัยที่เกี่ยวข้อง แบ่งเป็น 2 กลุ่มคืองานวิจัยเกี่ยวกับการตรวจหาร่องรอยที่ไม่ดีของโค้ดและการรีแฟคทอริงของโค้ด กับงานวิจัยที่เกี่ยวข้องกับการตรวจหาร่องรอยที่ไม่ดีของฐานข้อมูลและทำรีแฟคทอริงฐานข้อมูล

2.2.1 งานวิจัยเกี่ยวกับการตรวจหาร่องรอยที่ไม่ดีของโค้ดและการรีแฟคทอริงของโค้ด

งานวิจัย [8] ได้ทำการทดลองนำเครื่องมือในการหาร่องรอยที่ไม่ดีในโค้ดมา 8 เครื่องมือ แล้วทำการวิเคราะห์ความแตกต่างของเครื่องมือทั้งหมดดังรูปที่ 20 จากรูปทำให้ทราบได้ว่าเครื่องมือแต่ละชนิดจะมีประเภทในการทำงานต่างกันคือ ประเภทที่เป็นโปรแกรมแยกออกมา (Standalone) และประเภทที่เป็นปลั๊กอินของอิคลิปส์ (Eclipse Plugin) เครื่องมือแต่ละเครื่องมือสามารถใช้ในภาษาที่แตกต่างกันแต่ส่วนใหญ่เป็นจาวา มีเพียงเครื่องมือ JDeodorant ที่มีรีแฟคทอริงด้วย เครื่องมือแต่ละชนิดจะสามารถเชื่อมไปที่จุดที่เป็นร่องรอยที่ไม่ดีในโค้ดได้แบบอัตโนมัติและไม่อัตโนมัติ อีกทั้งยังทำการวิเคราะห์ผลของเครื่องมือแยกตามร่องรอยที่ไม่ดีแต่ละตัว เพราะแต่ละเครื่องมือสามารถตรวจหาได้บางตัวเท่านั้น ซึ่งทำให้เห็นถึงศักยภาพของเครื่องมือแยกตามการตรวจหาร่องรอยที่ไม่ดีแต่ละตัว

Tool	Version	Type	Analyzed languages	Refactoring	Link to code
Checkstyle	5.4.1 2011	Eclipse Plugin, Standalone	Java	No	Yes
DECOR	1.0 2009	Standalone	Java	No	No
iPlasma	6.1 2009	Standalone	C++, Java	No	No
inFusion	7.2.11 2010	Standalone	C, C++, Java	No	No
JDeodorant	4.0.4 2010	Eclipse Plugin	Java	Yes	Yes
PMD	4.2.5 2009	Eclipse Plugin, Standalone	Java	No	Yes
Stench Blossom	1.0.4 2009	Eclipse Plugin	Java	No	Yes

Legend:
Analyzed languages: languages that the tool is able to analyze;
Refactoring: whether the tool provides automatic refactoring or not;
Link to code: whether the tool provides the location in the code of the detected smells.

รูปที่ 20 ผลการเปรียบเทียบเครื่องมือการตรวจหาร่องรอยที่ไม่ดี [6]

งานวิจัย [2] ได้ดำเนินการคล้ายกันกับ [8] แต่ทำการทดลองโดยวัดค่าความเที่ยงตรง (Precision) ค่าเรียกคืน (Recall) และคิดเป็นเปอร์เซ็นต์ เทียบกับการตรวจหาโดยผู้เชี่ยวชาญ

จากงานวิจัยทั้ง 2 งานทำให้ผู้วิจัยนำแนวคิดการทำเครื่องมือและประเมินผลการทำงานของเครื่องมือแยกตามประเภทของเอสคิวแอลแอนติแพตเทิร์น โดยใช้ค่าความเที่ยงตรงและค่าระลึกร่วมกับค่าเอฟ-เมเชอร์ด้วย

จากงานวิจัยที่ทำการสำรวจความคิดเห็นเกี่ยวกับร่องรอยที่ไม่ดีนั้น [9] ได้ผลว่า นอกจากเรื่องการตรวจหาร่องรอยที่ไม่ดีแล้ว เครื่องมือควรสามารถทำการแนะนำการรีแฟคทอริงด้วย ในงานวิจัยที่ทำการสำรวจข้อมูลใน Stack Overflow เกี่ยวกับเครื่องมือรีแฟคทอริง [10] ผู้พัฒนาซอฟต์แวร์ยังขาดความรู้เกี่ยวกับรีแฟคทอริง และต้องการให้เครื่องมือเสนอแค็ตตาล็อกแนะนำการรีแฟคทอริงด้วย ซึ่งตามรูปที่ 20 มีเพียงเครื่องมือ JDeodorant ที่สามารถทำรีแฟคทอริงได้ด้วย มีงานวิจัยที่เสนอเครื่องมือตรวจหาร่องรอยที่ไม่ดีและสามารถทำรีแฟคทอริงได้ด้วย ได้แก่ เครื่องมือชื่อ JCodeCanine [11] เป็นปลั๊กอินอิคลิปส์ที่ตรวจหาร่องรอยที่ไม่ดีและมีการแนะนำวิธีรีแฟคทอริง โดยวิธีประเมินเครื่องมือในส่วนของร่องรอยที่ไม่ดีนั้นใช้การวัดเปอร์เซ็นต์ความแม่นยำ (Accuracy) และประเมินแยกตามแต่ละร่องรอยที่ไม่ดีดังเช่นงานวิจัย [8] และ [2] ส่วนการแนะนำการรีแฟคทอริงนั้นได้ประเมินด้วยจำนวนการแนะนำรีแฟคทอริงทั้งหมด จำนวนรีแฟคทอริงที่เมื่อทำแล้วไม่กระทบต่อการคอมไพล์โปรแกรม และจำนวนรีแฟคทอริงที่เมื่อทำแล้วไม่กระทบต่อการคอมไพล์โปรแกรมและยังคงพฤติกรรมตามความหมายเดิมอยู่

จากงานวิจัยที่ทำการสำรวจความคิดเห็นต่อผู้ใช้งานเครื่องมือ ทำให้เห็นถึงความต้องการที่จะให้เครื่องมือสามารถตรวจหาร่องรอยที่ไม่ดีและสามารถแนะนำวิธีรีแฟคทอริงได้ด้วย จึงเป็นแนวทาง

ให้ผู้วิจัยทำการพัฒนาเครื่องมือที่สามารถตรวจหาเอสคิวแอลแอนติแพตเทิร์น พร้อมทั้งสามารถแนะนำการทำรีแฟคทอริงฐานข้อมูลด้วย ในทำนองเดียวกับเครื่องมือ JCodeCanine

2.2.2 งานวิจัยที่เกี่ยวข้องกับการตรวจหาร่องรอยที่ไม่ดีของฐานข้อมูลและทำรีแฟคทอริงฐานข้อมูล

งานวิจัย [12] จะค้นหาฐานข้อมูลโดยการสอบถาม (Query) ข้อมูลตาราง โดยใช้เครื่องมือโพสต์เกรส เอสคิวแอล (PostgreSQL) โดยขอบเขตการหาคือจะตรวจประเภทเอสคิวแอลแอนติแพตเทิร์นของการออกแบบฐานข้อมูลเชิงตรรกะทั้งหมด 8 ประเภทและเอสคิวแอลแอนติแพตเทิร์นของการออกแบบฐานข้อมูลเชิงกายภาพ 2 ประเภท ตามนิยามของ [3] โดยผลลัพธ์ของการสอบถามจะนำเสนอเป็นบริเวณที่น่าสงสัย และวิเคราะห์เอสคิวแอลแอนติแพตเทิร์นแต่ละประเภทโดยรายงานความเป็นไปได้ของการเกิด False Negative (FN) และ False Positive (FP)

งานวิจัย [13] ได้ดำเนินการเปรียบเทียบงานวิจัยต่าง ๆ ที่เกี่ยวข้องกับการทำรีแฟคทอริงฐานข้อมูล ดังรูปที่ 21 ซึ่งส่วนใหญ่มีเครื่องมือสำหรับการรีแฟคทอริง แต่การแนะนำการรีแฟคทอริงยังใช้ตามทฤษฎีของ S.J. Ambler [4]

	Ambler	Domingues	D'Souza	Boehm	Chang	Curino	Liquibase
Automation	No	No	No	No	Limited	No	No
Generic databases	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Concurrency control	Yes	Yes	No	No	No	No	No
Number of refactoring per process	Limited	Limited	Limited	Limited	Limited	No	No
Data Replication	Yes	Yes	No	No	No	No	No
Tool	No	Database Evolution Manager	No	Squash	No	Prisma	Liquibase
Language and Structure	No	No	JAVA Metadata	XML	No	XML	JAVA XML
Improving Database design	Yes	Yes	No	Yes	Yes	Yes	No
Improving query's performance	No	No	Yes	Yes	Yes	No	No
Changes' history	No	No	No	No	No	Yes	Yes
Workflow	Yes	No	No	No	No	Yes	No
Refactoring Suggestion	Yes	No	No	No	No	No	No

รูปที่ 21 งานวิจัยที่เกี่ยวข้องกับการรีแฟคทอริงฐานข้อมูล [13]

หลังจากนั้นงานวิจัย [14] ได้ทำการรีแฟคทอริงเพื่อสร้างมาตรฐานด้านการตั้งชื่อให้ฐานข้อมูลที่มีอยู่หลายแห่ง ให้มีชื่อตาราง ชื่อฟังก์ชัน เป็นมาตรฐานเดียวกัน โดยดำเนินการตรวจหาโดยใช้สคริปต์และแก้ไขด้วยวิธีการรีแฟคทอริงฐานข้อมูล งานวิจัยอีกกลุ่มหนึ่งทำการรีแฟคทอริงฐานข้อมูลและเน้นในมุมมองของซอฟต์แวร์ด้วย เช่น งานวิจัย [15] ได้พัฒนาเครื่องมือที่ช่วยเมื่อมีการ

เปลี่ยนแปลงซอฟต์แวร์และต้องทำการรีแฟคทอริงฐานข้อมูล จะสามารถบอกการเปลี่ยนแปลงของสกีมาที่ได้รับผลกระทบได้ และงานวิจัย [16] ได้นำการรีแฟคทอริงฐานข้อมูลและทดสอบแบบซ้ำมาประยุกต์ใช้ในแอปพลิเคชันมือถือ แล้วทำการทดลองรีแฟคทอริงโค้ด รีแฟคทอริงฐานข้อมูล และวัดคุณภาพก่อนและหลังทำรีแฟคทอริง

จากงานวิจัยที่กล่าวมาข้างต้น ทำให้ทราบถึงสถานะงานวิจัยที่เกี่ยวข้องกับการตรวจหาร่องรอยที่ไม่ดีและการรีแฟคทอริงฐานข้อมูล และ ทำให้เกิดแนวคิดในการปรับปรุงคือ ผนวกรวมการตรวจหาเอสคิวแอลแอนติแพตเทิร์นและการทำรีแฟคทอริงเข้าด้วยกัน โดยจะดำเนินการในทำนองเดียวกับงานวิจัย [12] และ [14] ข้างต้น คือตรวจหาแอนติแพตเทิร์นด้วยวิธีการเขียนสคริปต์เอสคิวแอล แต่จะใช้ขั้นตอนวิธีในการตรวจหาที่ต่างไปจากในงานวิจัย [12] และใช้ทฤษฎีการรีแฟคทอริงฐานข้อมูลเพื่อแก้ไข และเพิ่มการประเมินผลค่าเรียกคืนและค่าความเที่ยงตรง นอกจากนี้ผู้วิจัยจะพัฒนาเครื่องมืออัตโนมัติสำหรับการตรวจสอบ เสนอคำแนะนำในการปรับโครงสร้างและประเมินผลค่าการเรียกคืนและความเที่ยงตรงของประสิทธิภาพในการตรวจหาเอสคิวแอลแอนติแพตเทิร์นแต่ละแอนติแพตเทิร์นเทียบกับของ [12]

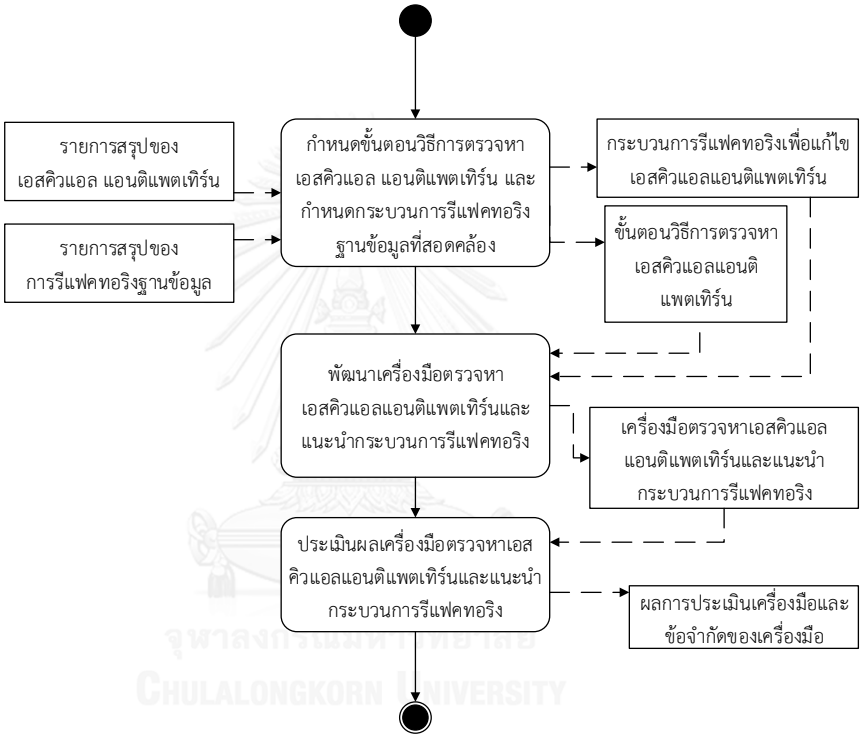
โดยจากทฤษฎีและงานวิจัยทั้ง 2 กลุ่มที่กล่าวมาข้างต้นสามารถสรุปความแตกต่างและเปรียบเทียบกับงานของผู้วิจัย ได้ดังตารางที่ 6

ตารางที่ 6 สรุปทฤษฎีและงานวิจัยที่เกี่ยวข้อง

ทฤษฎีและงานวิจัย	บริบท (Context)		การนิยาม (Definition)		เครื่องมือ (Tool)	
	โค้ด	ฐานข้อมูล	ร่องรอยที่ไม่ดีหรือแอนติแพตเทิร์น	รีแฟคทอริง	ร่องรอยที่ไม่ดีหรือแอนติแพตเทิร์น	รีแฟคทอริง
Karwin, 2010 [3]		✓	✓			
Ambler, 2009 [4]		✓	✓	✓		
Fontana et al.,2012 [8]	✓		✓		✓	✓
Fernandes et al ., 2016 [2]	✓		✓		✓	
Nongpong and Boyland ,2012 [11]	✓		✓	✓	✓	✓
Eassaar,2015 [12]		✓	✓		✓	
Domingues et al., 2014 [13]		✓		✓		
Vial, 2015 [1 4]		✓		✓		✓
Schink 2013 [15]		✓		✓		
Samue et al., 2012 [16]		✓		✓		
งานวิจัย		✓	✓	✓	✓	✓

บทที่ 3
ขั้นตอนวิธีการตรวจหาเอสคิวแอลแอนติแพตเทิร์น และกำหนดกระบวนการรีแฟคทอริงฐานข้อมูล

ผู้วิจัยมีแนวคิดที่จะเสนอเครื่องมือเพื่อช่วยผู้ดูแลฐานข้อมูลในการตรวจหาเอสคิวแอลแอนติแพตเทิร์นของฐานข้อมูล เพื่อให้พิจารณาว่าต้องการรีแฟคทอริงฐานข้อมูลหรือไม่ โดยจะมีการแนะนำกระบวนการรีแฟคทอริงฐานข้อมูลเพื่อแก้ไขแอนติแพตเทิร์นนั้น ซึ่งการดำเนินการเพื่อให้ได้เครื่องมือดังกล่าวนั้นแสดงภาพรวมในรูปที่ 22



รูปที่ 22 ภาพรวมของแนวคิดการทำเครื่องมือตรวจหาเอสคิวแอลแอนติแพตเทิร์นและแนะนำกระบวนการรีแฟคทอริงฐานข้อมูล

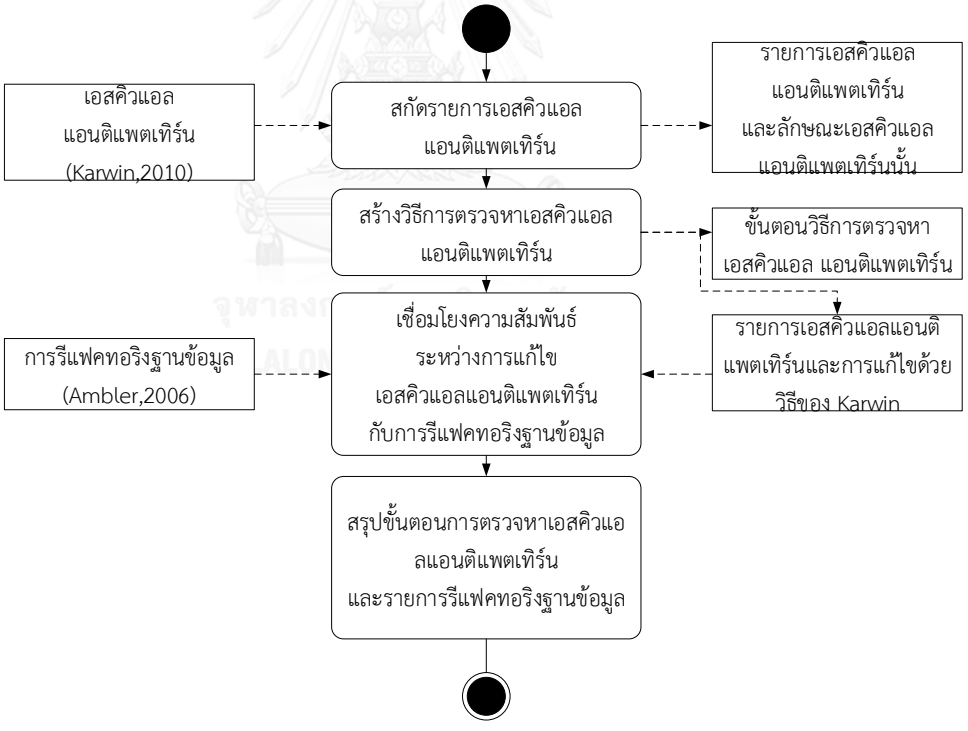
จากรูปที่ 22 มีการดำเนินการทั้งหมด 3 ขั้นตอน โดยขั้นตอนแรกจะกล่าวในบทที่ 3 และขั้นตอนที่ 2 กล่าวในบทที่ 4 และขั้นตอนสุดท้ายในบทที่ 5

3.1 กำหนดขั้นตอนวิธีการตรวจหาเอสคิวแอลแอนติแพตเทิร์นและกำหนดกระบวนการรีแฟคทอริงฐานข้อมูล

ขั้นตอนวิธีในการตรวจหานั้น จะหลีกเลี่ยงการเข้าถึงโครงสร้างหรือข้อมูลจากทั้งฐานข้อมูล แต่ควรกำหนดขอบเขตเฉพาะส่วนที่มีแนวโน้มว่าจะเป็นเอสคิวแอลแอนติแพตเทิร์น โดยการกำหนดเงื่อนไขบางอย่างที่สนใจ แล้วดำเนินการตรวจหาเอสคิวแอลแอนติแพตเทิร์นในส่วนนั้น ซึ่งผู้วิจัยจะ

นำเสนอในรูปภาพการตรวจหาของแต่ละแอนติแพตเทิร์นโดยแสดงคำว่า “Potential Zone” เนื่องจากว่าการตรวจหาเอสคิวแอลแอนติแพตเทิร์นในลักษณะต่าง ๆ นั้นเป็นการตรวจสอบจากข้อมูลภายในตารางหรือโครงสร้างของตาราง ถ้าหากเราเข้าถึงส่วนนั้นทั้งหมดภายในฐานข้อมูลเพื่อนำมาตรวจสอบ จะทำให้การค้นหาใช้เวลานาน

หลังจากนั้นจะศึกษาวิธีการแก้ไขเอสคิวแอลแอนติแพตเทิร์น และวิธีการรีแฟคทอริงแบบต่าง ๆ เพื่อมาเชื่อมโยงกันแล้วกำหนดกระบวนการรีแฟคทอริง ซึ่งประกอบด้วยขั้นตอนการใช้วิธีรีแฟคทอริงแบบต่าง ๆ โดยขั้นตอนทั้งหมดดังกล่าวนั้นแสดงภาพรวมในรูปที่ 23 และแสดงรายละเอียดในแต่ละแอนติแพตเทิร์นในหัวข้อ 3.1.1-3.1.8 โดยแต่ละแอนติแพตเทิร์นจะประกอบไปด้วยหัวข้อหลัก 2 เรื่อง คือ 1) การกำหนดขั้นตอนวิธีการตรวจหาเอสคิวแอลแอนติแพตเทิร์น ซึ่งเป็นวิธีที่ผู้วิจัยออกแบบสำหรับเป็นแนวทางก่อนที่จะนำไปพัฒนาด้วยภาษาทรานแซกเอสคิวแอล 2) การกำหนดกระบวนการรีแฟคทอริงฐานข้อมูล ผู้วิจัยจะนำวิธีการแก้ไขเอสคิวแอลแอนติแพตเทิร์นของ B. Karwin [6] ที่แนะนำวิธีการแก้ไขแต่ละแอนติแพตเทิร์นไว้มาวิเคราะห์ว่าจะเลือกใช้วิธีการรีแฟคทอริงฐานข้อมูลของ S. J. Ambler [4] วิธีใดมาใช้จึงจะได้ผลตรงตามวิธีการแก้ไขนั้น ๆ



รูปที่ 23 ภาพรวมการกำหนดขั้นตอนวิธีการตรวจหาเอสคิวแอลแอนติแพตเทิร์น และกำหนดกระบวนการรีแฟคทอริงฐานข้อมูล

สำหรับเอสคิวแอลแอนติแพตเทิร์นทั้ง 8 ประเภทที่กล่าวไว้ในหัวข้อที่ 2.1.1 มีขั้นตอนวิธีการตรวจหาและกระบวนการรีแฟคทอริงดังรายละเอียดต่อไปนี้

3.1.1 Format Comma-Separated Lists

จากรายละเอียดของ Format Comma-Separated Lists ในหัวข้อที่ 2.1.1 ข้อย่อย 1) นั้น ได้ยกตัวอย่างตามตัวอย่างด้านล่างนี้อีกครั้งเพื่อประกอบการอธิบายขั้นตอนวิธีการตรวจหาและกระบวนการรีแฟคทอริง โดยมีรายละเอียดดังนี้

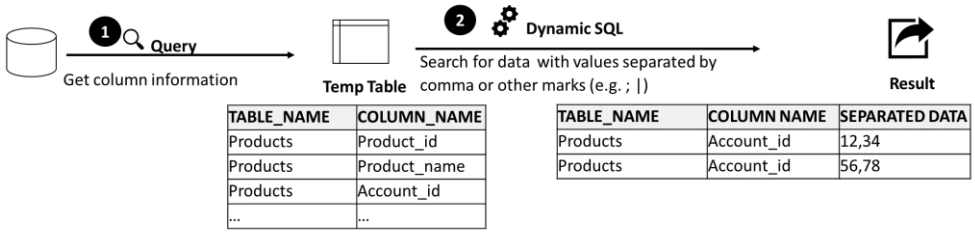
ตัวอย่าง

```
CREATE TABLE Products (
  product_id    SERIAL PRIMARY KEY,
  product_name VARCHAR(1000),
  account_id    VARCHAR(100), -- comma-separated list
        );
INSERT INTO Products (product_id, product_name, account_id)
VALUES (DEFAULT, 'Visual TurboBuilder', '12,34');
```

1) การกำหนดขั้นตอนวิธีการตรวจหาเอสคิวแอลแอนติแพตเทิร์น

ขั้นตอนวิธีการตรวจหาเอสคิวแอลแอนติแพตเทิร์น Format Comma-Separated Lists แสดงไว้ในรูปที่ 24 ขั้นแรก (1) ทำการกำหนดส่วนของการค้นหาโดยการสอบถาม (Query) ข้อมูลคอลัมน์และตารางในฐานข้อมูลและเก็บลงตารางชั่วคราว (Temp Table) ไว้โดยใช้คำสั่ง INFORMATION_SCHEMA.COLUMNS

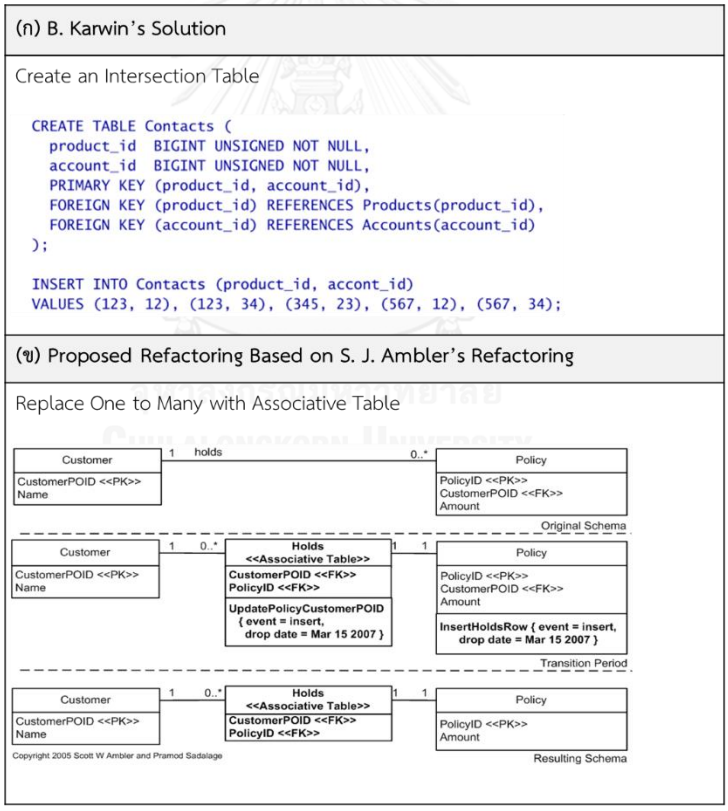
ขั้นต่อมา (2) เริ่มตรวจหาเอสคิวแอลแอนติแพตเทิร์น โดยการสร้างไดนามิคเอสคิวแอล (Dynamic SQL ซึ่งเป็นการเขียนการสอบถาม ลักษณะ String และใช้ Exec () ที่สามารถใส่ Parameter ได้) สำหรับวนลูปเพื่อตรวจหาข้อมูลตารางและคอลัมน์ทั้งหมด ที่มีเครื่องหมายคั่นอยู่ เช่น , ; | โดยเครื่องหมายคั่นนั้น ผู้วิจัยได้ออกแบบให้ผู้ใช้เครื่องมือสามารถกำหนดได้



รูปที่ 24 ขั้นตอนวิธีการตรวจหาเอสคิวแอลแอนติแพตเทิร์น Format Comma-Separated Lists

2) การกำหนดกระบวนการรีแฟคทอริงฐานข้อมูล

สำหรับตัวอย่างข้างต้นแนวทางการแก้ไขเอสคิวแอลแอนติแพตเทิร์นนี้ของ B. Karwin เสนอให้ทำการสร้างตาราง Contacts ซึ่งมีความสัมพันธ์แบบ many-to-many กับตาราง Products และ Accounts (This new table Contacts implements a many-to-many relationship between Products and Accounts) โดยแสดงในรูปที่ 25 (ก) คือต้องมีการสร้าง Intersection Table โดยจะมีการสร้างตาราง Contacts เพื่อเก็บข้อมูลที่เชื่อมกัน เป็นตารางใหม่ ซึ่งกระบวนการรีแฟคทอริงฐานข้อมูลที่สามารถนำไปใช้กับเอสคิวแอลแอนติแพตเทิร์นนี้ ได้แก่ Replace One to Many with Associative Table. โดยจากรูปที่ 25 (ข) จะเห็นว่าจะมีการเก็บค่าเชื่อมกันระหว่าง 2 ตารางคือ Customer กับ Policy ด้วยการสร้าง Associative Table ชื่อตารางว่า Holds ซึ่งมีลักษณะเดียวกันกับการสร้างตาราง Contacts ในรูปที่ 25 (ก)



รูปที่ 25 การแก้ไขแอนติแพตเทิร์น Format Comma-Separated Lists (ก) แนวทางจาก B. Karwin (ข) ข้อเสนอของผู้วิจัย

3.1.2 Always Depend on One's Parent

จากรายละเอียดของ Always Depend on One's Parent ในหัวข้อที่ 2.1.1 ข้อย่อย 2) นั้น ได้ยกตัวอย่างตามตัวอย่างด้านล่างนี้อีกครั้งเพื่อประกอบการอธิบายขั้นตอนวิธีการตรวจหาและกระบวนการรีแฟคทอริง โดยมีรายละเอียดดังนี้

ตัวอย่าง

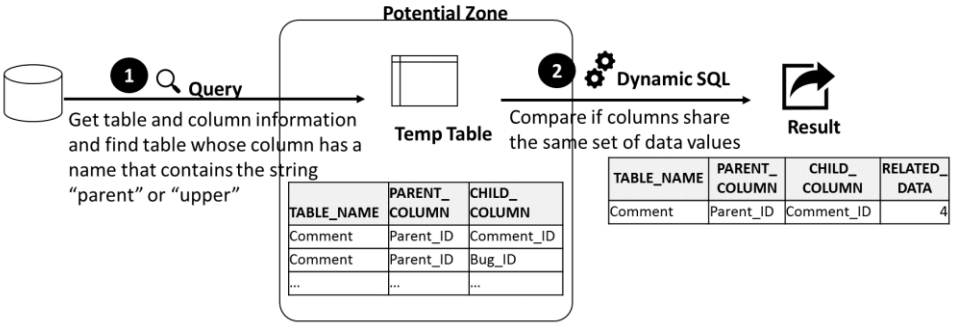
```
CREATE TABLE Comments (
  comment_id   SERIAL PRIMARY KEY,-- hierarchical data
  parent_id    BIGINT UNSIGNED, --parent
  comment_date DATETIME NOT NULL,
  comment      TEXT NOT NULL,
  FOREIGN KEY (parent_id) REFERENCES Comments(comment_id), );
```

comment_ID	parent_ID
1	NULL
2	*1*
3	2
4	*1*
5	4
6	4

1) การกำหนดขั้นตอนวิธีการตรวจหาเอสคิวแอลแอนติแพตเทิร์น

ขั้นตอนวิธีการตรวจหาเอสคิวแอลแอนติแพตเทิร์น Always Depend on One's Parent แสดงไว้ในรูปที่ 26 ขั้นแรก (1) ทำการกำหนดส่วนค้นหาโดยเริ่มจากสร้างตารางชั่วคราวสำหรับเก็บข้อมูลตารางที่มีชื่อที่สามารถสันนิษฐานได้ว่าในตารางนั้นมีการออกแบบเพื่อเก็บข้อมูลที่มีความสัมพันธ์แบบแม่ลูก โดยการเลือกตารางที่มีชื่อคอลัมน์ประกอบไปด้วยคำว่า "PARENT" หรือ "UPPER" ซึ่งการสันนิษฐานนี้เพื่อกำหนดการค้นหาให้แคบลงดังที่กล่าวไปแล้วข้างต้น หากต้องการเพิ่มหรือลดคำที่แสดงความสัมพันธ์แบบแม่ลูกเครื่องมืออนุญาตให้ทำการแก้ไขได้ จากนั้นตรวจสอบเพิ่มเติมว่าในตารางนั้น มีคอลัมน์อื่น ๆ ที่สามารถเป็นคอลัมน์ลูกได้อีกหรือไม่ โดยการตรวจสอบจะมีการหาว่า ข้อมูลของคอลัมน์จะต้องมีประเภทของข้อมูล (Data Type) เดียวกันด้วย

ขั้นต่อมา (2) เริ่มตรวจหาเอสคิวแอลแอนติแพตเทิร์น โดยการนำข้อมูลในตารางชั่วคราวมาตรวจสอบว่ามีความสัมพันธ์แม่ลูกกันหรือไม่ โดยการสร้างไดนามิคเอสคิวแอลวนลูปเพื่อนำตารางคอลัมน์แม่และคอลัมที่คาดว่าจะเป็นลูก มาตรวจข้อมูลว่ามีข้อมูลที่ซ้ำกันหรือไม่ ยิ่งไปกว่านั้นเพื่อช่วยสนับสนุนผู้ใช้งาน เครื่องมือยังสามารถแสดงจำนวนข้อมูลในคอลัมน์แม่ที่คอลัมน์ลูกไปอ้างอิงซ้ำกัน เพื่อให้พิจารณาได้เพิ่มเติม ด้วยสมมติฐานที่ว่าถ้ามีการอ้างอิงข้อมูลซ้ำกันจำนวนมาก ก็อาจจะมีการเก็บข้อมูลเป็นโครงสร้างแม่ลูกกัน



รูปที่ 26 ขั้นตอนวิธีการตรวจหาเอสคิวแอลแอนติแพตเทิร์น Always Depend on One's Parent

2) การกำหนดกระบวนการรีแฟคทอริงฐานข้อมูล

สำหรับตัวอย่างข้างต้นแนวทางการแก้ไขเอสคิวแอลแอนติแพตเทิร์นของ B. Karwin เสนอให้ทำการสร้างคอลัมน์ใหม่ในตารางเพื่อเก็บเส้นทาง (Path) ซึ่งแสดงความสัมพันธ์แม่ลูกระหว่างคอลัมน์ในตาราง (In the Comments table, instead of the parent_id column, define a column called path) โดยแสดงในรูปที่ 27 (ก) จะมีการสร้างคอลัมน์ใหม่เพื่อเก็บเส้นทาง กระบวนการรีแฟคทอริงฐานข้อมูลที่สามารถนำไปใช้กับเอสคิวแอลแอนติแพตเทิร์นนี้ ได้แก่ (1) Introduce New Column ดังตัวอย่างในรูปที่ 27 (ข) เพื่อทำการสร้างคอลัมน์ใหม่สำหรับเก็บเส้นทางจากนั้นใช้ (2) Drop Column เพื่อลบคอลัมน์แม่ออกไป

(ก) B. Karwin's Solution

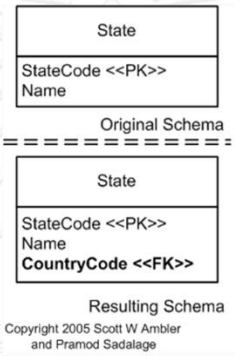
Path Enumeration

```
CREATE TABLE Comments (
  comment_id   SERIAL PRIMARY KEY,
  path         VARCHAR(1000),
  bug_id       BIGINT UNSIGNED NOT NULL,
  author       BIGINT UNSIGNED NOT NULL,
  comment_date DATETIME NOT NULL,
  comment      TEXT NOT NULL,
  FOREIGN KEY (bug_id) REFERENCES Bugs(bug_id),
  FOREIGN KEY (author) REFERENCES Accounts(account_id)
);
```

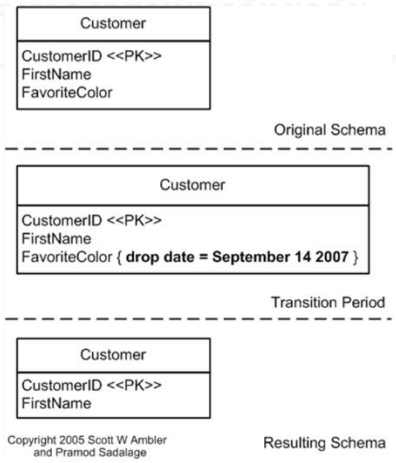
comment_id	path	author	comment
1	1/	Fran	What's the cause of this bug?
2	1/2/	Ollie	I think it's a null pointer.
3	1/2/3/	Fran	No, I checked for that.
4	1/4/	Kukla	We need to check for invalid input.
5	1/4/5/	Ollie	Yes, that's a bug.
6	1/4/6/	Fran	Yes, please add a check.
7	1/4/6/7/	Kukla	That fixed it.

(ข) Proposed Refactoring Based on S. J. Ambler's Refactoring

(1) Introduce new column



(2) Drop Column



รูปที่ 27 การแก้ไขแอนติแพตเทิร์น Always Depend on One's Parent (ก) แนวทางจาก B. Karwin (ข) ข้อเสนอของผู้วิจัย

3.1.3 One Size Fits All

จากรายละเอียดของ One Size Fits All ในหัวข้อที่ 2.1.1 ข้อย่อย 3) นั้น ได้ยกตัวอย่างตามตัวอย่างด้านล่างนี้อีกครั้งเพื่อประกอบการอธิบายขั้นตอนวิธีการตรวจหาและกระบวนการรีแฟคทอริง โดยมีรายละเอียดดังนี้

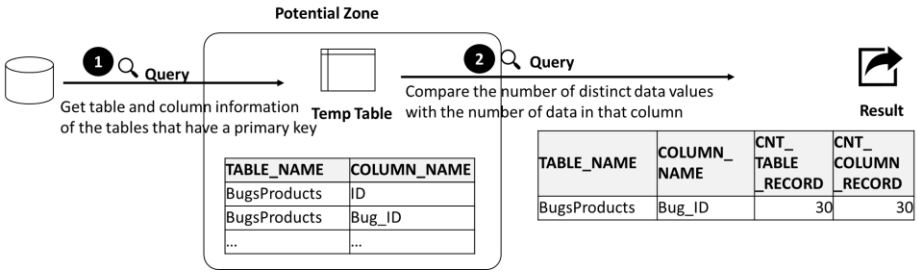
ตัวอย่าง

```
CREATE TABLE BugsProducts (
id SERIAL PRIMARY KEY,
bug_id BIGINT UNSIGNED NOT NULL,
product_id BIGINT UNSIGNED NOT NULL,
FOREIGN KEY (bug_id) REFERENCES Bugs(bug_id),
FOREIGN KEY (product_id) REFERENCES Products(product_id)
);
INSERT INTO BugsProducts (id,bug_id, description,.)
 VALUES (1,'VIS-078','crashes on save',..);
```

1) การกำหนดขั้นตอนวิธีการตรวจหาเอสคิวแอลแอนติแพตเทิร์น

ขั้นตอนวิธีการตรวจหาเอสคิวแอลแอนติแพตเทิร์น One Size Fits All แสดงไว้ในรูปที่ 28 ขั้นแรก (1) ทำการกำหนดส่วนค้นหา เริ่มจากการหาตารางที่มีการใช้คีย์หลัก และมีการกำหนดคอนสเตรนต์ยูนีค (UNIQUE) ให้กับคอลัมน์ก่อน จากนั้นนำมาเทียบกับคอลัมน์ทั้งหมดในตารางเพื่อหาคอลัมน์ที่ไม่ได้เป็นคีย์หลักและไม่ได้กำหนดคอนสเตรนต์ยูนีคไว้ แล้วบันทึกลงในตารางชั่วคราว

ขั้นต่อมา (2) เริ่มตรวจหาเอสคิวแอลแอนติแพตเทิร์น โดยการสันนิษฐานว่าจะมีคอลัมน์อื่นที่ยูนีคพอจะเป็นไอดี (ID) หรือคีย์หลักได้ โดยจะใช้วิธีนับจำนวนข้อมูลที่ไม่ซ้ำกันที่อยู่ในคอลัมน์นั้น เทียบกับจำนวนข้อมูลของคอลัมน์นั้นทั้งหมดซึ่งถ้ามีจำนวนเท่ากัน น่าจะแสดงว่าคอลัมน์มีความยูนีคและน่าจะเป็นคีย์หลักได้



รูปที่ 28 ขั้นตอนวิธีการตรวจหาเอสคิวแอลแอนติแพตเทิร์น One Size Fits All

2) การกำหนดกระบวนการรีแฟคทอริงฐานข้อมูล

สำหรับตัวอย่างนี้แนวทางการแก้ไขเอสคิวแอลแอนติแพตเทิร์นนี้ของ B. Karwin เสนอให้ใช้คอลัมน์ที่ยูนีคและไม่มีค่าเป็นค่าว่างเป็นคีย์หลักแทนคอลัมน์ ID (If your table contains an attribute that's guaranteed to be unique, is non-null, and can serve to identify the row, don't feel obligated to add a pseudo key solely for the sake of tradition.) โดยแสดงในรูปที่ 29 (ก) คือมีการสร้างตารางที่กำหนดคอลัมน์ Bug_id แทนการใช้ ID ซึ่งกระบวนการรีแฟคทอริงฐานข้อมูลที่สามารถนำไปใช้กับเอสคิวแอลแอนติแพตเทิร์นนี้ ได้แก่ (1) Replace Surrogate Key With Natural Key ดังรูปที่ 29 (ข) โดยที่ Natural Key คือคีย์ที่เกิดจากข้อมูลที่สามารถเป็นคีย์ได้โดยธรรมชาติข้อมูล ซึ่ง State Code จะเทียบได้กับ Bug_id ในตัวอย่าง โดยทำการแทนที่คอลัมน์เดิมซึ่งเรียกว่าเป็น Surrogate Key คือคีย์ที่เราสร้างไว้ ได้แก่ StatePOID ซึ่งเทียบได้กับ ID ในตัวอย่าง จากนั้น (2) ใช้วิธีการรีแฟคทอริง Drop Column เพื่อลบคอลัมน์ ID

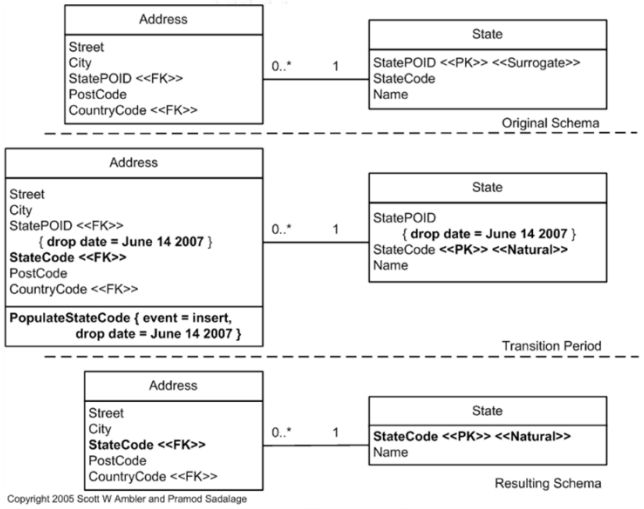
(ก) B. Karwin's Solution

Embrace Natural Keys and Compound Keys

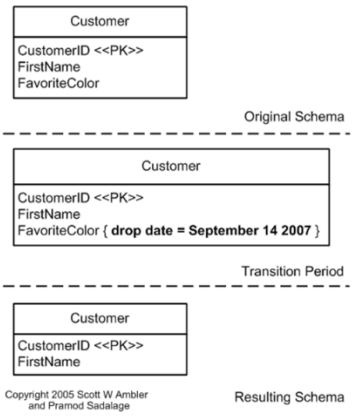
```
CREATE TABLE BugsProducts (
  bug_id      BIGINT UNSIGNED NOT NULL,
  product_id  BIGINT UNSIGNED NOT NULL,
  PRIMARY KEY (bug_id, product_id),
  FOREIGN KEY (bug_id) REFERENCES Bugs(bug_id),
  FOREIGN KEY (product_id) REFERENCES Products(product_id)
);

INSERT INTO BugsProducts (bug_id, product_id)
  VALUES (1234, 1), (1234, 2), (1234, 3);


INSERT INTO BugsProducts (bug_id, product_id)
  VALUES (1234, 1); -- error: duplicate entry
```

(ข) Proposed Refactoring Based on S. J. Ambler's Refactoring

(1) Replace Surrogate Key With Natural Key



(2) Drop Column



รูปที่ 29 การแก้ไขแอนติแพตเทิร์น One Size Fits All (ก) แนวทางจาก B. Karwin (ข) ข้อเสนอของผู้วิจัย

3.1.4 Leave Out the Constraints

จากรายละเอียดของ Leave Out the Constraints ในหัวข้อที่ 2.1.1 ข้อย่อย 4) นั้น ได้ยกตัวอย่างตามตัวอย่างด้านล่างนี้อีกครั้งเพื่อประกอบการอธิบายขั้นตอนวิธีการตรวจหาและกระบวนการรีแฟคทอริง โดยมีรายละเอียดดังนี้

ตัวอย่าง

```
CREATE TABLE Bugs (
reported_by BIGINT UNSIGNED NOT NULL,
status VARCHAR(20) NOT NULL DEFAULT 'NEW' ); -- Reference Table BugStatus
```

1) การกำหนดขั้นตอนวิธีการตรวจหาเอสคิวแอลแอนติแพตเทิร์น

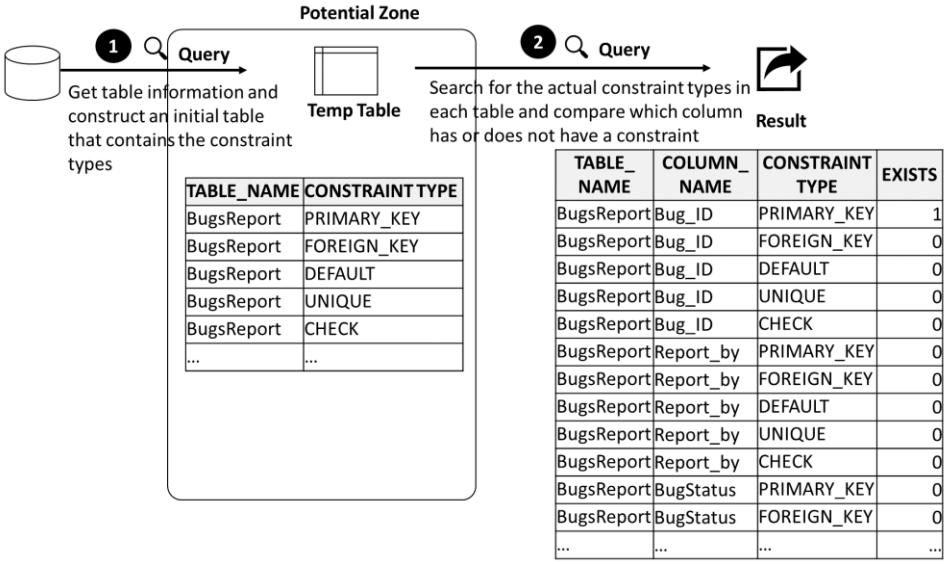
แบ่งการค้นหาเป็นสองวิธี โดยวิธีแรกคือ รายงานการสร้างคอนสเตรนต์ต่าง ๆ ที่มีอยู่แล้วในฐานข้อมูลเพื่อแสดงให้ผู้ดูแลฐานข้อมูลพิจารณาในการสร้างคอนสเตรนต์เพิ่มเติมได้เอง เนื่องจากการตรวจหาของงานวิจัยนี้ ไม่สามารถระบุได้ว่าควรสร้างคอนสเตรนต์แบบใดให้กับข้อมูลใดบ้าง เนื่องจากต้องใช้องค์ความรู้ในโดเมนของฐานข้อมูลนั้นในการพิจารณา ผู้วิจัยจึงตรวจสอบเอสคิวแอลแอนติแพตเทิร์นนี้ ด้วยวิธีการตรวจสอบคอนสเตรนต์ของแต่ละตารางแทน โดยแสดงผลให้เห็นว่าหากคอลัมน์ใดมีประเภทคอนสเตรนต์กำหนดไว้จะแสดงเป็นค่า 1 ถ้าไม่มีแสดงเป็นค่า 0 ขั้นตอนวิธีการตรวจหาเอสคิวแอลแอนติแพตเทิร์น Leave Out the Constraints แบบวิธีแรกแสดงไว้ในรูปที่ 30

ขั้นแรก (1) ทำการกำหนดส่วนค้นหา โดยเริ่มจากการสร้างข้อมูลตั้งต้นเพื่อกำหนดการตรวจสอบคอนสเตรนต์ ดังต่อไปนี้ คือ

- PRIMARY_KEY_CONSTRAINT
- UNIQUE_CONSTRAINT
- FOREIGN_KEY_CONSTRAINT
- CHECK_CONSTRAINT
- DEFAULT_CONSTRAINT

และใช้คำสั่ง INFORMATION_SCHEMA.TABLES เพื่อให้ได้ข้อมูลตาราง ขั้นต่อมา (2) เริ่มหาเอสคิวแอลแอนติแพตเทิร์น โดยการนำข้อมูลตั้งต้นมาเปรียบเทียบกับคอนสเตรนต์ที่ถูกสร้างภายในระบบ โดยการใช้ SYS.OBJECT

ส่วนวิธีที่สองนั้นจะใช้ขั้นตอนการตรวจหาจากหัวข้อที่ 3.1.3 มาตรวจสอบคอลัมน์ที่มีความยูนีคและสามารถเป็นคีย์หลักได้ แต่ไม่ได้กำหนดคอนสเตรนต์ PRIMARY_KEY_CONSTRAINT และ UNIQUE_CONSTRAINT ไว้เพิ่มเติมด้วย



รูปที่ 30 ขั้นตอนวิธีการตรวจหาเอสคิวแอลแอนติแพตเทิร์น Leave Out the Constraints

2) การกำหนดกระบวนการรีแฟคทอริงฐานข้อมูล

สำหรับตัวอย่างข้างต้นการแก้ไขเอสคิวแอลแอนติแพตเทิร์นนี้ของ B. Karwin เสนอให้ระบุคอนสเตรนต์ที่เหมาะสมให้กับคอลัมน์ ตัวอย่างเช่น ในกรณีของการระบุ FOREIGN_KEY_CONSTRAINT ได้มีการเสนอให้ “Your database design by using foreign key constraints to enforce referential integrity. Instead of searching for and correcting data integrity mistakes, you can prevent these mistakes from entering your database in the first place” ดังตัวอย่างในรูปที่ 31 (ก) ซึ่งกระบวนการรีแฟคทอริงฐานข้อมูลที่สามารถนำไปใช้กับเอสคิวแอลแอนติแพตเทิร์นนี้ ได้แก่ Make Column Non-Nullable, Add Foreign Key Constraint, Introduce New Column Constraint, Introduce Default Value ดังแสดงในรูปที่ 31 (ข) โดยที่

- การระบุ PRIMARY_KEY_CONSTRAINT ใช้การรีแฟคทองริงแบบ Make Column Non-Nullable ก่อนการกำหนดคอนสเตรนต์
- การระบุ UNIQUE_CONSTRAINT ใช้การรีแฟคทองริงแบบ Make Column Non-Nullable ก่อนการกำหนดคอนสเตรนต์

- การระบุ FOREIGN_KEY_CONSTRAINT ใช้การรีแฟคทองริงแบบ Add Foreign Key Constraint
- การระบุ CHECK_CONSTRAINT ใช้การรีแฟคทองริงแบบ Introduce New Column Constraint
- การระบุ DEFAULT_CONSTRAINT ใช้การรีแฟคทองริงแบบ Introduce Default Value

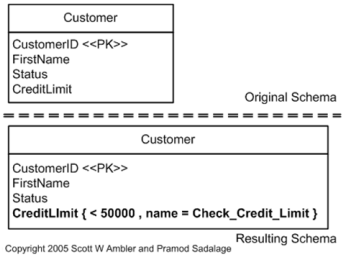
(ก) B. Karwin's Solution

Declare Constraints

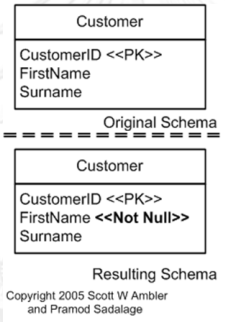
```
CREATE TABLE Bugs (
  -- . . .
  reported_by       BIGINT UNSIGNED NOT NULL,
  status            VARCHAR(20) NOT NULL DEFAULT 'NEW',
  FOREIGN KEY (reported_by) REFERENCES Accounts(account_id),
  FOREIGN KEY (status) REFERENCES BugStatus(status)
);
```

(ข) Proposed Refactoring Based on S. J. Ambler's Refactoring

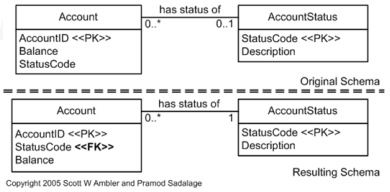
- Introduce New Column Constraint



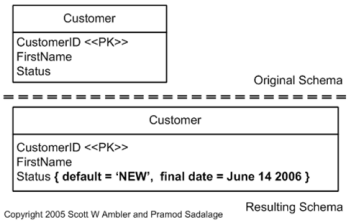
- Make Column Non-Nullable



- Add Foreign Key Constraint



- Introduce Default Value



รูปที่ 31 การแก้ไขแอนติแพตเทิร์น Leave Out the Constraints (ก) แนวทางจาก B. Karwin (ข) ข้อเสนอของผู้วิจัย

3.1.5 Use a Generic Attribute Table

จากรายละเอียดของ Use a Generic Attribute Table ในหัวข้อที่ 2.1.1 ข้อย่อย 5) นั้น ได้ยกตัวอย่างตามตัวอย่างด้านล่างนี้อีกครั้งเพื่อประกอบการอธิบายขั้นตอนวิธีการตรวจหาและกระบวนการรีแฟคทอริง โดยมีรายละเอียดดังนี้

ตัวอย่าง

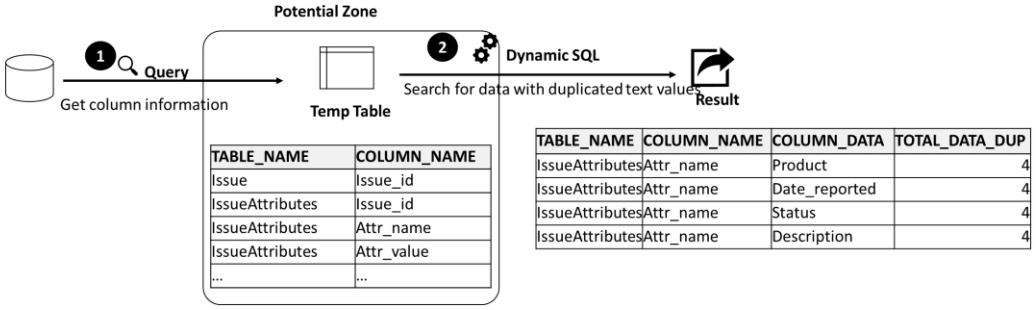
```
CREATE TABLE IssueAttributes (
issue_id BIGINT UNSIGNED NOT NULL,
attr_name VARCHAR(100) NOT NULL,
attr_value VARCHAR(100),
PRIMARY KEY (issue_id, attr_name),
FOREIGN KEY (issue_id) REFERENCES Issues(issue_id) );

INSERT INTO IssueAttributes (issue_id, attr_name, attr_value)
VALUES
        (1234, 'product' , '1' ),
        (1234, 'date_reported' , '2009-06-01' ),
        (1234, 'status' , 'NEW' ),
        (1234, 'description' , 'Saving does not work' ),
        (1234, 'reported_by' , 'Bill' ),
        (1234, 'version_affected' , '1.0' ),
        (1234, 'severity' , 'loss of functionality' ),
        (1234, 'priority' , 'high' );
```

1) การกำหนดขั้นตอนวิธีการตรวจหาเอสคิวแอลแอนติแพตเทิร์น

ขั้นตอนวิธีการตรวจหาเอสคิวแอลแอนติแพตเทิร์น Use a Generic Attribute Table แสดงไว้ในรูปที่ 32 ขั้นแรก (1) ทำการกำหนดส่วนของการค้นหาโดยการสอบถาม (Query) ข้อมูลคอลัมน์และตารางในฐานข้อมูลและเก็บลงตารางชั่วคราว (Temp Table) ไว้โดยใช้คำสั่ง INFORMATION_SCHEMA.COLUMNS โดยสันนิษฐานว่าข้อมูลของคอลัมน์นั้น

จะต้องเป็นข้อมูลที่ไม่ใช่ประเภทตัวเลข ทศนิยม วันที่ เนื่องจากต้องการตรวจหาค่าข้อมูลในคอลัมน์ซึ่งมีโอกาสเป็นแอตทริบิวต์ ดังนั้นคอลัมน์จึงต้องเป็นประเภทตัวอักษร

ขั้นต่อมา (2) เริ่มตรวจหาเอสคิวแอลแอนติแพตเทิร์น โดยการสร้างไดนามิคเอสคิวแอล (Dynamic SQL) สำหรับวนลูปเพื่อตรวจหาข้อมูลในตารางและคอลัมน์ทั้งหมด ที่มีค่าข้อมูลซ้ำกัน และแสดงข้อมูลจำนวนการซ้ำ เพื่อให้ผู้ใช้สามารถทราบชื่อคอลัมน์ที่อาจจะมีการเก็บข้อมูลแอตทริบิวท์นั้นอยู่ โดยแสดง



รูปที่ 32 ขั้นตอนวิธีการตรวจหาเอสคิวแอลแอนติแพตเทิร์น Use a Generic Attribute Table

2) การกำหนดกระบวนการรีแฟคทอริงฐานข้อมูล

สำหรับตัวอย่างข้างต้นแนวทางการแก้ไขเอสคิวแอลแอนติแพตเทิร์นนี้ของ B. Karwin เสนอให้มีการสร้างตารางใหม่และนำค่าข้อมูลที่ซ้ำกันและเป็นแอตทริบิวท์มาสร้างเป็นคอลัมน์ต่างๆ ในตารางใหม่นั้น (store all related types in one table, with distinct columns for every attribute that exists in any type.) โดยแสดงในรูป 33 (ก) คือดำเนินการสร้างตารางใหม่ สำหรับเก็บข้อมูลแอตทริบิวต์มาเป็นคอลัมน์แทน ซึ่งกระบวนการรีแฟคทอริงฐานข้อมูลที่สามารถนำไปใช้กับเอสคิวแอลแอนติแพตเทิร์นนี้ ได้แก่ (1) Introduce New table โดยแสดงในรูปที่ 33 (ข) สำหรับใช้สร้างตาราง Issue จากนั้น (2) ใช้ Drop Table เพื่อลบตาราง IssueAttributes ออกไป

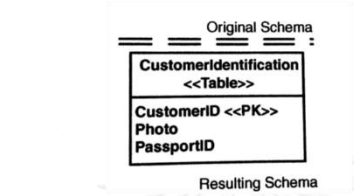
(ก) B. Karwin's Solution

Class Table Inheritance

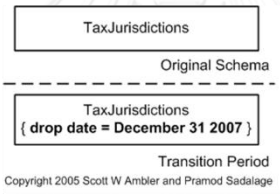
```
CREATE TABLE Issues (
  issue_id         SERIAL PRIMARY KEY,
  reported_by      BIGINT UNSIGNED NOT NULL,
  product_id       BIGINT UNSIGNED,
  priority         VARCHAR(20),
  version_resolved VARCHAR(20),
  status           VARCHAR(20),
  issue_type       VARCHAR(10),  -- BUG or FEATURE
  severity         VARCHAR(20),  -- only for bugs
  version_affected VARCHAR(20),  -- only for bugs
  sponsor          VARCHAR(50),  -- only for feature requests
  FOREIGN KEY (reported_by) REFERENCES Accounts(account_id)
  FOREIGN KEY (product_id) REFERENCES Products(product_id)
);
```

(ข) Proposed Refactoring Based on S. J. Ambler's Refactoring

(1) Introduce New table



(2) Drop Table



รูปที่ 33 การแก้ไขแอนติแพตเทิร์น Use a Generic Attribute Table (ก) แนวทางจาก B. Karwin (ข) ข้อเสนอของผู้วิจัย

3.1.6 Use Dual-Purpose Foreign Key

จากรายละเอียดของ Use Dual-Purpose Foreign Key ในหัวข้อที่ 2.1.1 ข้อย่อย 6) นั้น ได้ยกตัวอย่างตามตัวอย่างด้านล่างนี้อีกครั้งเพื่อประกอบการอธิบายขั้นตอนวิธีการตรวจหาและกระบวนการรีแฟคทอริง โดยมีรายละเอียดดังนี้

ตัวอย่าง

CREATE TABLE Comments (

comment_id SERIAL PRIMARY KEY,

issue_type VARCHAR(20), -- "Bugs" or "FeatureRequests"

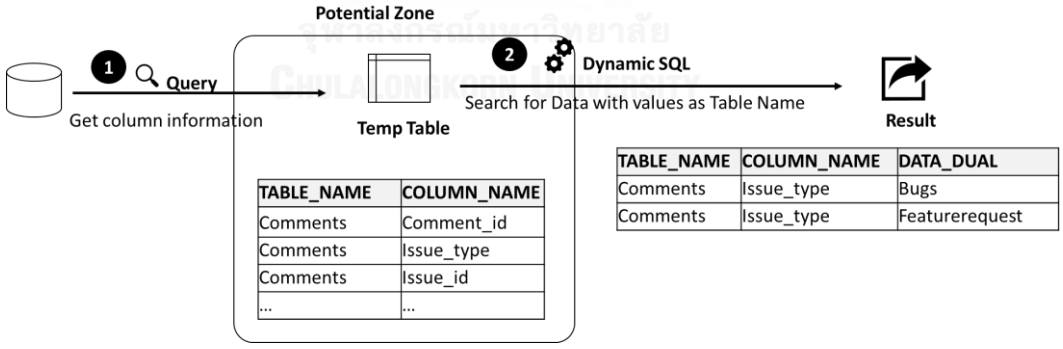
issue_id BIGINT UNSIGNED NOT NULL,

```
author BIGINT UNSIGNED NOT NULL,
comment_date DATETIME,
comment TEXT,
FOREIGN KEY (author) REFERENCES Accounts(account_id));
```

1) การกำหนดขั้นตอนวิธีการตรวจหาเอสคิวแอลแอนติแพตเทิร์น

ขั้นตอนวิธีการตรวจหาเอสคิวแอลแอนติแพตเทิร์น Use Dual-Purpose Foreign Key แสดงไว้ในรูปที่ 34 ขั้นแรก (1) ทำการกำหนดส่วนของการค้นหาโดยการสอบถาม (Query) ข้อมูลคอลัมน์และตารางในฐานข้อมูลและเก็บลงตารางชั่วคราว (Temp Table) ไว้ โดยใช้คำสั่ง INFORMATION_SCHEMA.COLUMNS โดยสันนิษฐานว่าข้อมูลของคอลัมน์นั้นจะต้องเป็นข้อมูลที่ไม่ใช่ประเภทตัวเลข ทศนิยม วันที่ เนื่องจากต้องการตรวจหาค่าข้อมูลในคอลัมน์ซึ่งมีโอกาสเป็นชื่อตาราง ดังนั้นคอลัมน์จึงต้องเป็นประเภทตัวอักษร

ขั้นต่อมา (2) เริ่มตรวจหาเอสคิวแอลแอนติแพตเทิร์น โดยการสร้างไดนามิคเอสคิวแอล (Dynamic SQL) สำหรับวนลูปเพื่อตรวจหาข้อมูลตารางและคอลัมน์ทั้งหมด ที่มีข้อมูลชื่อตารางอยู่ เพื่อสันนิษฐานว่าคอลัมน์นั้นจะเก็บชื่อตารางเพื่อใช้เชื่อมโยงไปยังข้อมูลในตารางนั้น



รูปที่ 34 ขั้นตอนวิธีการตรวจหาเอสคิวแอลแอนติแพตเทิร์น Use Dual-Purpose Foreign Key

2) การกำหนดกระบวนการรีแฟคทอริงฐานข้อมูล

สำหรับตัวอย่างข้างต้นแนวทางการแก้ไขเอสคิวแอลแอนติแพตเทิร์นนี้ของ B. Karwin เสนอให้สร้างตาราง BugsComments และ FeaturesComments แยกออกมาจากตาราง Comments ซึ่งเป็นตารางแม่และสร้าง Foreign key เพื่อเชื่อมกลับไปยังตาราง Comments (Create a separate intersection table for each parent table, and in

each intersection table include a foreign key, as well as a foreign key to the respective parent table.) ดังรูป 35 (ก) ซึ่งกระบวนการรีแฟคทอริงฐานข้อมูลที่สามารภนำไปใช้กับเอสคิวเอสคิวแอลแอนติแพตเทิร์นนี้ ได้แก่ (1) Split Table ดังรูปที่ 35 (ข) เพื่อแยกตารางใหม่ออกมาและ (2) ดำเนินการ Add Foreign Key Constraint เพื่อเชื่อมโยง Foreign Key กลับไปยังตารางแม่ จากนั้น (3) Drop Column ในตารางแม่ซึ่งเก็บชื่อตาราง (คือ issue_type) ออกไป

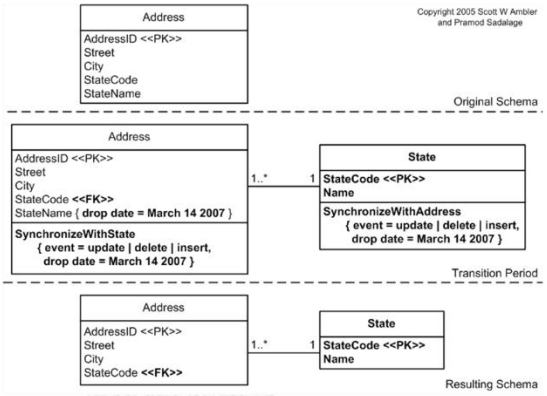
(ก) B. Karwin's Solution

Creating Intersection Tables

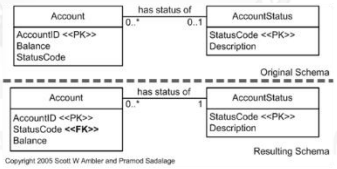
```
CREATE TABLE BugsComments (
  issue_id    BIGINT UNSIGNED NOT NULL,
  comment_id  BIGINT UNSIGNED NOT NULL,
  PRIMARY KEY (issue_id, comment_id),
  FOREIGN KEY (issue_id) REFERENCES Bugs(issue_id),
  FOREIGN KEY (comment_id) REFERENCES Comments(comment_id)
);

CREATE TABLE FeaturesComments (
  issue_id    BIGINT UNSIGNED NOT NULL,
  comment_id  BIGINT UNSIGNED NOT NULL,
  PRIMARY KEY (issue_id, comment_id),
  FOREIGN KEY (issue_id) REFERENCES FeatureRequests(issue_id),
  FOREIGN KEY (comment_id) REFERENCES Comments(comment_id)
);
```

(ข) Proposed Refactoring Based on S. J. Ambler's Refactoring

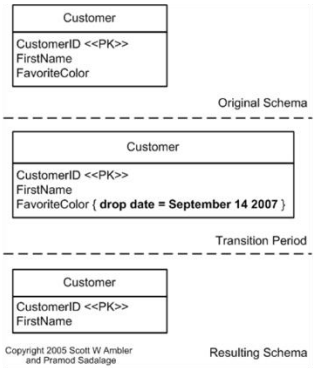
(1) Split Table



(2) Add Foreign Key Constraint



(3) Drop Column



รูปที่ 35 การแก้ไขแอนติแพตเทิร์น Use Dual-Purpose Foreign Key (ก) แนวทางจาก B. Karwin (ข) ข้อเสนอของผู้วิจัย

3.1.7 Create Multiple Columns

จากรายละเอียดของ Create Multiple Columns ในหัวข้อที่ 2.1.1 ข้อย่อย 7) นั้น ได้ยกตัวอย่างตามตัวอย่างด้านล่างนี้อีกครั้งเพื่อประกอบการอธิบายขั้นตอนวิธีการตรวจหาและกระบวนการรีแฟคทอริง โดยมีรายละเอียดดังนี้

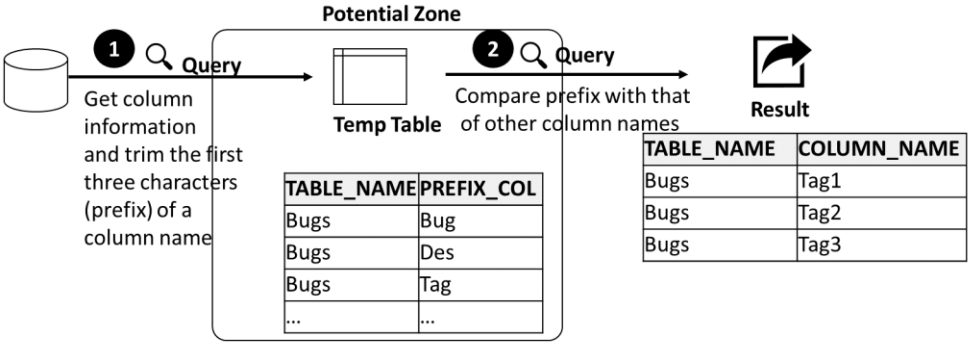
ตัวอย่าง

```
CREATE TABLE Bugs (
bug_id SERIAL PRIMARY KEY,
description VARCHAR(1000),
tag1 VARCHAR(20),
tag2 VARCHAR(20),
tag3 VARCHAR(20));
```

1) การกำหนดขั้นตอนวิธีการตรวจหาเอสคิวแอลแอนติแพตเทิร์น

ขั้นตอนวิธีการตรวจหาเอสคิวแอลแอนติแพตเทิร์น Create Multiple Columns แสดงไว้ในรูปที่ 36 ขั้นแรก (1) ทำการกำหนดส่วนค้นหา เริ่มจากการหาชื่อคอลัมน์ด้วยคำสั่ง INFORMATION_SCHEMA.COLUMNS และนำชื่อแต่ละคอลัมน์มาตัดแค่ 3 ตัวอักษรแรกเพื่อมาเทียบกัน โดยสันนิษฐานว่าคอลัมน์ที่มี 3 ตัวอักษรแรกเหมือนกันจะมีโอกาสเป็นคอลัมน์ที่เก็บค่าข้อมูลหลาย ๆ ค่า ของแอตทริบิวต์เดียวกัน (Multivalued Attribute) ทั้งนี้เครื่องมืออนุญาตให้สามารถกำหนดจำนวนตัวอักษรแรกที่ใช้เปรียบเทียบกันได้

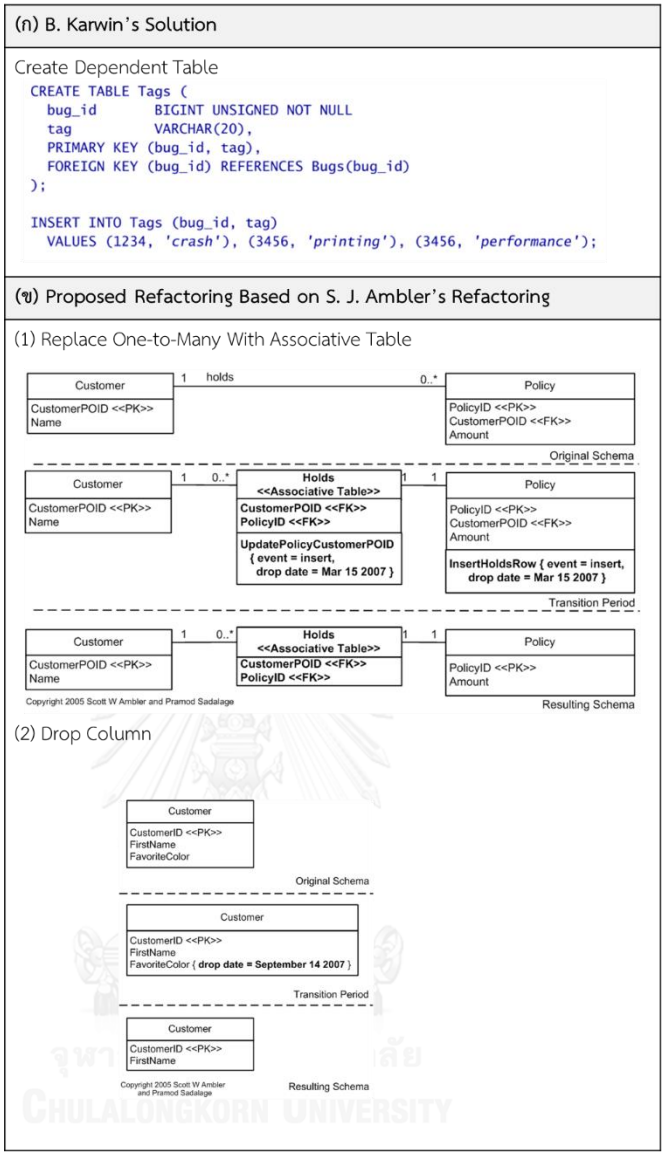
ขั้นต่อมา (2) เริ่มตรวจหา เอสคิวแอลแอนติแพตเทิร์น โดยการนำข้อมูล 3 ตัวอักษรแรกของแต่ละคอลัมน์ในตารางเดียวกันมาเปรียบเทียบกัน และเพิ่มเงื่อนไขการตรวจหาด้วยว่าตัวอักษรสุดท้ายจะเป็นตัวเลข



รูปที่ 36 ขั้นตอนวิธีการตรวจหาเอสคิวแอลแอนติแพตเทิร์น Create Multiple Columns

2) การกำหนดกระบวนการรีแฟคทอริงฐานข้อมูล

สำหรับตัวอย่างข้างต้นแนวทางการแก้ไขเอสคิวแอลแอนติแพตเทิร์นนี้ของ B. Karwin กล่าวว่า เสนอให้สร้างตาราง Dependent เพิ่มขึ้นมา เพื่อเก็บค่าหลาย ๆ ค่าของแอตทริบิวต์เดียวกัน แล้วสร้าง Foreign key เพื่อเชื่อมโยงกลับไปยังตารางแม่ (“Create a dependent table with one column for the multivalued attribute. Store the multiple values in multiple rows instead of multiple columns. Also, define a foreign key in the dependent table to associate the values to its parent row”) ดังรูปที่ 37 (ก) โดยสร้างตาราง Tags เป็นตาราง Dependent วิธีนี้คล้ายกับการสร้างตาราง Associative ดังนั้นกระบวนการรีแฟคทอริงฐานข้อมูลที่สามารถนำไปใช้กับเอสคิวแอลแอนติแพตเทิร์นนี้ ได้แก่ (1) Replace One to many With Associative Table ดังรูปที่ 37 (ข) ซึ่งตาราง Holds นั้นเทียบได้กับตาราง Tags จากนั้น (2) ทำการ Drop Column ในตารางแม่ ซึ่งเก็บค่าข้อมูลหลายค่าของแอตทริบิวต์ออกไป ได้แก่ คอลัมน์ Tag1, Tag2, Tag3



รูปที่ 37 การแก้ไขแอนติแพตเทิร์น Create Multiple Columns (ก) แนวทางจาก B. Karwin (ข) ข้อเสนอของผู้วิจัย

3.1.8 Clone Tables or Columns

จากรายละเอียดของ Clone Table or Columns ในหัวข้อที่ 2.1.1 ข้อย่อย 8) นั้น ได้ยกตัวอย่างตามตัวอย่างด้านล่างนี้อีกครั้งเพื่อประกอบการอธิบายขั้นตอนวิธีการตรวจหาและกระบวนการรีแฟคทอริง โดยมีรายละเอียดดังนี้

ตัวอย่าง

CREATE TABLE *Bugs_2008* (. . .);

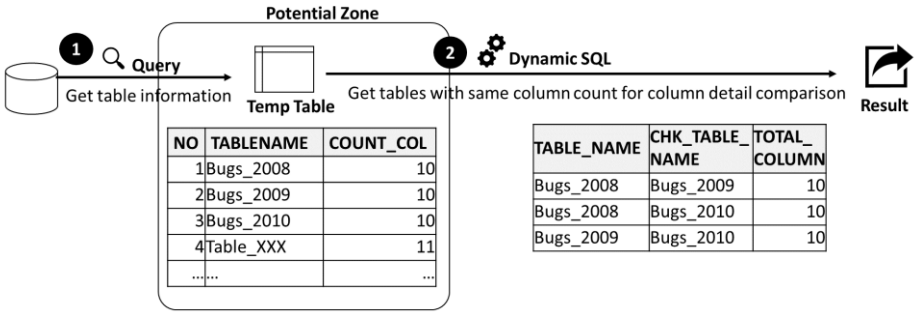
CREATE TABLE *Bugs_2009* (. . .);

CREATE TABLE *Bugs_2010* (. . .);

1) การกำหนดขั้นตอนวิธีการตรวจหาเอสคิวแอลแอนติแพตเทิร์น

ขั้นตอนวิธีการตรวจหาเอสคิวแอลแอนติแพตเทิร์น Clone Table or Columns แสดงไว้ในรูปที่ 38 ขั้นแรก (1) ทำการกำหนดส่วนค้นหา เริ่มจากการสร้างตารางชั่วคราวโดยใส่ข้อมูล ชื่อคอลัมน์ทั้งหมดและนับจำนวนคอลัมน์ในแต่ละตาราง เพื่อจัดกลุ่มการเปรียบเทียบตารางที่มีจำนวนคอลัมน์เท่ากัน

ขั้นต่อมา (2) เริ่มตรวจหาเอสคิวแอลแอนติแพตเทิร์น โดยดำเนินการเปรียบเทียบข้อมูลโดยใช้เทคนิคไดนามิคเอสคิวแอล เพื่อวนลูปเปรียบเทียบตาราง ที่มีจำนวนคอลัมน์เท่ากันเพื่อหาตารางที่มีรายละเอียดแบบเดียวกัน โดยพิจารณารายละเอียดของคอลัมน์ เช่น ประเภทของข้อมูล ความยาว รูปแบบ ซึ่งได้จากการใช้ INFORMATION_SCHEMA.COLUMNS



รูปที่ 38 ขั้นตอนวิธีการตรวจหาเอสคิวแอลแอนติแพตเทิร์น Clone Table or Columns

2) การกำหนดกระบวนการรีแฟคทอริงฐานข้อมูล

สำหรับตัวอย่างข้างต้นแนวทางการแก้ไขเอสคิวแอลแอนติแพตเทิร์นนี้ของ B. Karwin เสนอให้สร้างตาราง Dependent เพิ่มขึ้นมา (The remedy for Metadata Tribbles columns/table is to create a dependent table.) ด้วยรูปที่ 39 (ก) โดยเริ่มจากการนำข้อมูลทุกตารางมารวมกันในตารางที่ต้องการ และสร้างคอลัมน์เพิ่มเพื่อเก็บปี หรือเวอร์ชัน ของข้อมูล ก่อนที่จะลบตารางที่ไม่ใช้ออกจากฐานข้อมูล ซึ่งกระบวนการรีแฟคทอริงฐานข้อมูลที่สามารถนำไปใช้กับเอสคิวแอลแอนติแพตเทิร์นนี้คือ (1) Merge Table ดังรูปที่ 39 (ข) จากนั้น (2) Introduce New Column และ (3) Drop Table

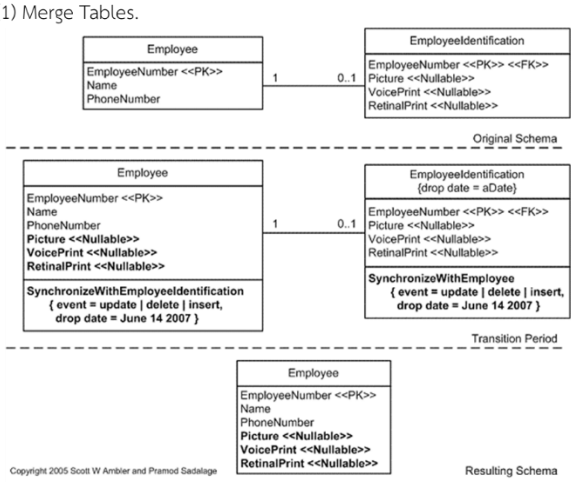
(ก) B. Karwin's Solution

Create a dependent table.

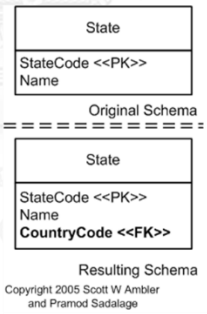
```
CREATE TABLE ProjectHistory (
  project_id  BIGINT,
  year        SMALLINT,
  bugs_fixed  INT,
  PRIMARY KEY (project_id, year),
  FOREIGN KEY (project_id) REFERENCES Projects(project_id)
);
```

(ข) Proposed Refactoring Based on S. J. Ambler's Refactoring

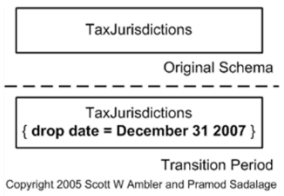
(1) Merge Tables.



(2) Introduce New Column



(3) Drop Tables.



รูปที่ 39 การแก้ไขแอนติแพตเทิร์น Clone Tables or Columns (ก) แนวทางจาก B. Karwin (ข) ข้อเสนอของผู้วิจัย

จากขั้นตอนวิธีทั้งหมดที่กล่าวมา สามารถสรุปรายการวิธีรีแฟคทอริงสำหรับแต่ละเอสคิวแอลแอนติแพตเทิร์นได้ดังตารางที่ 7

ตารางที่ 7 ผลลัพธ์รายการเอสคิวแอลแอนติแพตเทิร์นและการแก้ไขด้วยวิธีรีแฟคทอริงฐานข้อมูล

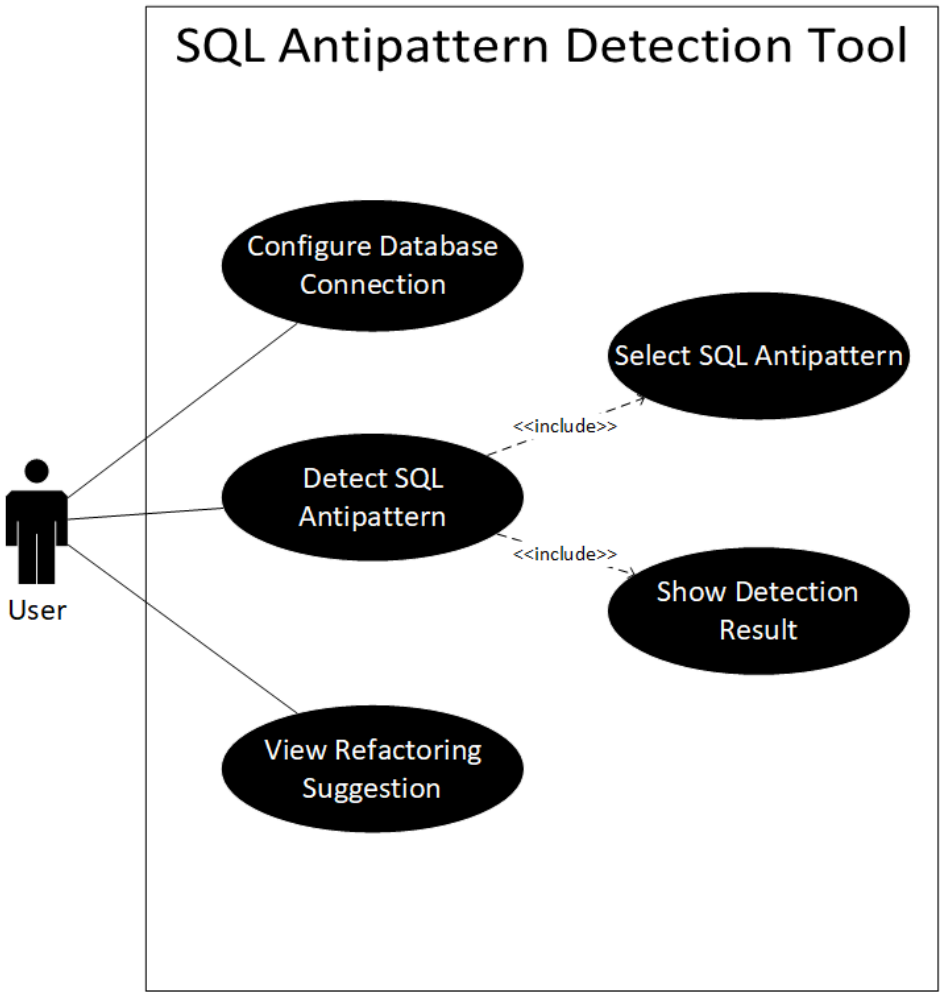
เอสคิวแอลแอนติแพตเทิร์น	รายการวิธีรีแฟคทอริงฐานข้อมูล
Format Comma-Separated Lists	Replace One to Many with Associative Table.
Always Depend on One's Parent	1) Introduce New Column 2) Drop Column
One Size Fits All	1) Replace Surrogate Key with Natural Key 2) Drop Column
Leave Out the Constraints	• Introduce New Column Constraint • Make column non-nullable • Add foreign key Constraint • Introduce Default Value
Use a Generic Attribute Table	1) Introduce New table 2) Drop Table
Use Dual-Purpose Foreign Key	1) Split Table 2) Add foreign key Constraint 3) Drop Column
Create Multiple Columns	1) Replace One to many With Associative Table 2) Drop Column
Clone Tables or Columns	1) Merge Table 2) Intorduce New Column 3) Drop Table

บทที่ 4
การพัฒนาเครื่องมือตรวจหาเอสคิวแอลแอนติแพตเทิร์นและแนะนำกระบวนการรีแฟคทอริง

การพัฒนาเครื่องมือจะพัฒนาโดยใช้ภาษาทรานแซก-เอสคิวแอล ตามรายการขั้นตอนวิธีที่ออกแบบไว้ในบทที่ 3 และจะได้เครื่องมือตรวจหาเอสคิวแอลแอนติแพตเทิร์นและนำการกำหนดกระบวนการรีแฟคทอริงมาแสดงผลการแนะนำด้วย

4.1 การออกแบบหน้าที่การทำงานของเครื่องมือ

ฟังก์ชันการทำงานของเครื่องมือแสดงดังแผนภาพยูสเคสรูปที่ 40 และอธิบายไว้ในตารางที่ 8



รูปที่ 40 แผนภาพยูสเคส

ตารางที่ 8 รายละเอียดหน้าที่การทำงานของเครื่องมือ

ยูสเคส	รายละเอียดหน้าที่การทำงานของยูสเคส
Configure Database Connection	ตั้งค่าในการเชื่อมต่อกับฐานข้อมูล
Detect SQL Antipattern	ตรวจหาเอสคิวแอลแอนติแพตเทิร์น โดยจะมียูสเคสที่ถูกเรียกทั้งหมด 2 ตัว คือ 1) Select SQL Antipattern เลือกเอสคิวแอลแอนติแพตเทิร์นที่ต้องการตรวจหา 2) Show Detection Result แสดงผลข้อมูลการตรวจหา
View Refactoring Suggestion	แสดงรายการแนะนำรีแฟคทอริง

4.2 การออกแบบสถาปัตยกรรมของเครื่องมือ

สถาปัตยกรรมของเครื่องมือแบ่งออกเป็น 3 ส่วน ได้แก่ ไคลเอนต์ เซิร์ฟเวอร์ และ ฐานข้อมูล ดังรูปที่ 41 สามารถอธิบายในแต่ละส่วน ได้ดังนี้

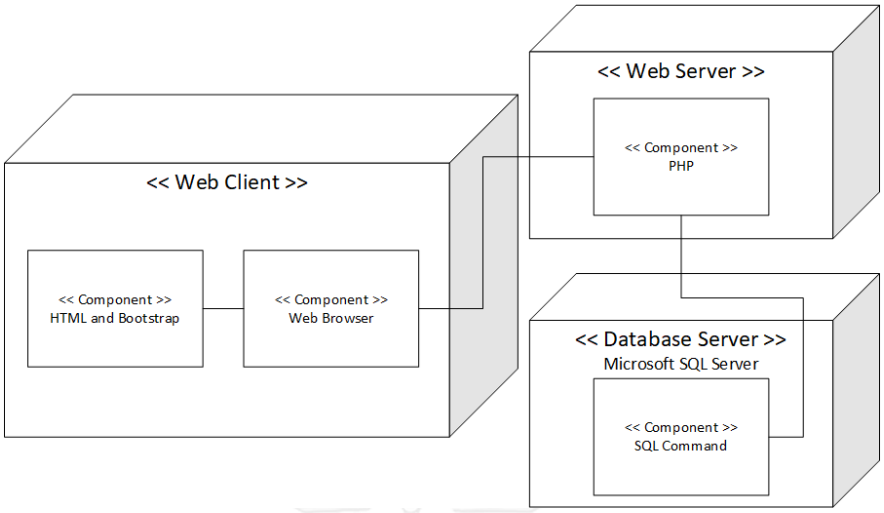
1. ไคลเอนต์ เนื่องจากเครื่องมือต้องสามารถใช้งานได้จากกลุ่มผู้ใช้งาน จึงจำเป็นต้องพัฒนาเครื่องมือในรูปแบบเว็บแอปพลิเคชันให้สามารถรองรับได้ด้วยเบราว์เซอร์ที่เป็นมาตรฐาน โดยพัฒนาด้วยเอชทีเอ็มแอล เป็นหลัก รวมไปถึงต้องมีหน้าจอแสดงผลที่รองรับความละเอียดและขนาดหน้าจอที่หลากหลาย จึงนำเฟรมเวิร์กบูตสแตรป (Bootstrap) มาใช้พัฒนาส่วนต่อประสานเพื่อรองรับการใช้งานแบบตอบโต้ (Responsive) ดังนั้นเครื่องที่ออกแบบมานี้สามารถรองรับหน้าจอได้หลายขนาด

2. เซิร์ฟเวอร์ การประมวลผลเพื่อตรวจจับแอนติแพตเทิร์นและแนะนำการรีแฟคทอริง ฐานข้อมูลนั้นจะเกิดขึ้นในส่วนนี้ทั้งหมด โดยถูกพัฒนาด้วยภาษาพีเอชพีเพื่อติดต่อกับ ไคลเอนต์ และ ฐานข้อมูล ได้ ซึง เซิร์ฟเวอร์ เรียกใช้ เอสคิวแอลคอมมานด์ ต่าง ๆ ภายใน ฐานข้อมูล เพื่อแสดงผลแก่ผู้ใช้งาน

3. ฐานข้อมูล เป็นส่วนประกอบหลักของเครื่องมือ โดยฐานข้อมูล นี้คือส่วนของการเก็บข้อมูลเพื่อให้เครื่องมือทำการค้นหาแอนติแพตเทิร์น ประกอบไปด้วยส่วนของ เอสคิวแอลคอมมานด์ เช่น ทรานแซก-เอสคิวแอล

ทรานแซก-เอสคิวแอลหรือที-เอสคิวแอล นั้น เป็นส่วนต่อขยาย (Extension) ของเอสคิวแอล สำหรับใช้ติดต่อกับฐานข้อมูลเชิงสัมพันธ์ในไมโครซอฟท์เอสคิวแอลเซิร์ฟเวอร์ (Microsoft SQL Server) ซึ่งเป็นโปรแกรมประยุกต์สำหรับจัดการฐานข้อมูลเชิงสัมพันธ์ที่พัฒนาโดยบริษัทไมโครซอฟท์ทรานแซก-เอสคิวแอลใช้สำหรับสื่อสารกับอินสแตนซ์ของเอสคิวแอลเซิร์ฟเวอร์ โดย

สามารถนำเข้าข้อมูลมาเพื่อประมวลผล และประกาศตัวแปรได้ สร้างคำสั่ง สร้างฟังก์ชันสำหรับการประมวลผล และแสดงผลข้อมูลได้



รูปที่ 41 การออกแบบสถาปัตยกรรมของเครื่องมือ

4.3 การพัฒนาเครื่องมือ

4.3.1 ฮาร์ดแวร์ที่ใช้ในการพัฒนาเครื่องมือ

1) เครื่องคอมพิวเตอร์ MacBook Air

- หน่วยประมวลผล 1.6GHz dual-core Intel Core i5
- หน่วยความจำ 4 GB
- ฮาร์ดดิสก์ ความจุ 128 GB
- จอภาพ 11 นิ้ว

2) เครื่องคอมพิวเตอร์ Notebook Toshiba

- หน่วยประมวลผล 3.2 GHz dual-core Intel Core i5
- หน่วยความจำ 8 GB
- ฮาร์ดดิสก์ ความจุ 1 TB
- จอภาพ 14 นิ้ว

4.3.2 ซอฟต์แวร์ที่ใช้ในการพัฒนาเครื่องมือ

1) ระบบปฏิบัติการ

- ระบบปฏิบัติการไมโครซอฟท์วินโดวส์ 10 (Microsoft Windows 10)

2) เครื่องมือที่ใช้ในการพัฒนา

- เอสคิวแอล เซิร์ฟเวอร์ แมเนจเมนต์ สตูดิโอ (SQL server management studio)
- พีจีแอดมิน (PG admin)
- แบรคเก็ตส์ (Brackets)

3) เครื่องมือที่ใช้ในการออกแบบพัฒนาส่วนต่อประสานผู้ใช้

- ไฟร์ฟอกซ์ (Firefox)
- เพนซิล (Pencil)

4) เครื่องมือสนับสนุนการพัฒนา

- แวมป์ เซิร์ฟเวอร์ (Wamp server)
- ไมโครซอฟท์ เอสคิวแอล เซิร์ฟเวอร์ (Microsoft SQL server)
- โพสต์เกรส เอสคิวแอล (PostgreSQL)

4.4 ขั้นตอนการทำงานของเครื่องมือ

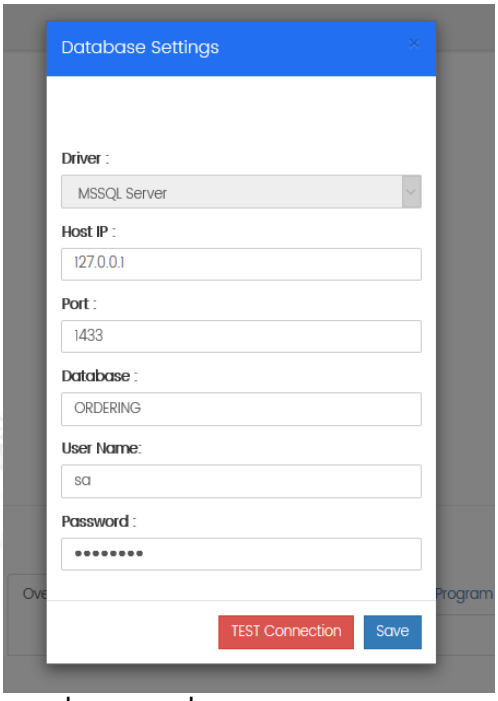
เครื่องมือจะมีขั้นตอนการทำงานหลัก ดังนี้

1) **ข้อมูลนำเข้า** คือ ฐานข้อมูลที่ต้องการตรวจหาเอสคิวแอลแอนติแพตเทิร์น และผู้ดูแลฐานข้อมูลจะป้อนข้อมูลนำเข้าให้กับเครื่องมือในกรณีที่เอสคิวแอลแอนติแพตเทิร์นต้องใช้ข้อมูลนำเข้าเพื่อช่วยในการตรวจหา เช่น ระบุเครื่องหมายคั่นที่ต้องการให้ตรวจหาสำหรับกรณีแอนติแพตเทิร์น Comma-Separated Lists ระบุคำสำคัญ (Keyword) สำหรับตรวจหาแอนติแพตเทิร์น Always Depend on One's Parent
2) เครื่องมือจะทำการตรวจหาแอนติแพตเทิร์นโดยผู้วิจัยนำขั้นตอนวิธีจากบทที่ 3 ในข้อย่อย 1) ของหัวข้อที่ 3.1.1 – 3.1.8 มาพัฒนาโดยใช้ทรานแซก-เอสคิวแอล ซึ่งการทำงานของทรานแซก-เอสคิวแอลนั้น จะสามารถนำไปกระทำการ (Execute) เพื่อค้นหาแอนติแพตเทิร์นในฐานข้อมูลได้ รายละเอียดของทรานแซก-เอสคิวแอลของขั้นตอนวิธีอยู่อยู่ในภาคผนวก ก.
3) **ข้อมูลออก** คือ (1) ผลลัพธ์การตรวจหาเอสคิวแอลแอนติแพตเทิร์น ที่จะระบุรายละเอียดของการตรวจหา เช่น การตรวจหาเอสคิวแอลแอนติแพตเทิร์น Clone Table or

Column ถ้าตรวจพบเอสคิวแอลแอนติแพตเทิร์นนี้ 2 จุด จะแสดงข้อความตรวจพบ 2 รายการ และ (2) การแนะนำกระบวนการรีแฟคทอริงฐานข้อมูลสำหรับแอนติแพตเทิร์นที่ตรวจพบ จากบทที่ 3 ในข้อย่อย 2) ของหัวข้อที่ 3.1.1 – 3.1.8

4.5 การออกแบบส่วนต่อประสานผู้ใช้งานของเครื่องมือ

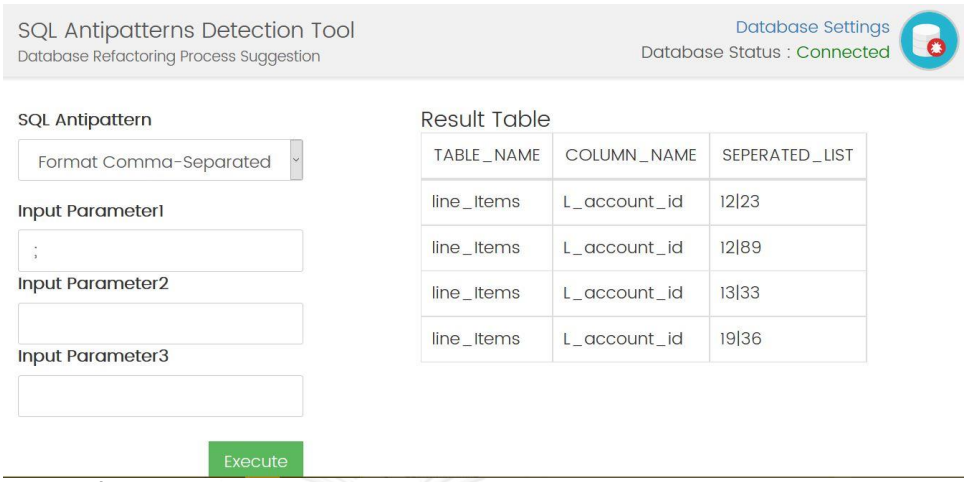
การออกแบบส่วนต่อประสานผู้ใช้งานของเครื่องมือจะแสดงตามลำดับการใช้งานของผู้ใช้ โดยขั้นแรก ผู้ใช้งานจะต้องดำเนินการตั้งค่าการเชื่อมต่อฐานข้อมูลก่อน โดยมีข้อมูลนำเข้า ได้แก่ ประเภทไดรเวอร์ ไอพีแอดเดรส พอร์ท ชื่อฐานข้อมูล ผู้ใช้งานสำหรับการเชื่อมต่อ รหัสผ่านผู้ใช้งาน เมื่อกรอกครบถ้วนแล้ว สามารถกดดำเนินการทดสอบการเชื่อมต่อได้ ดังแสดงในรูปที่ 42



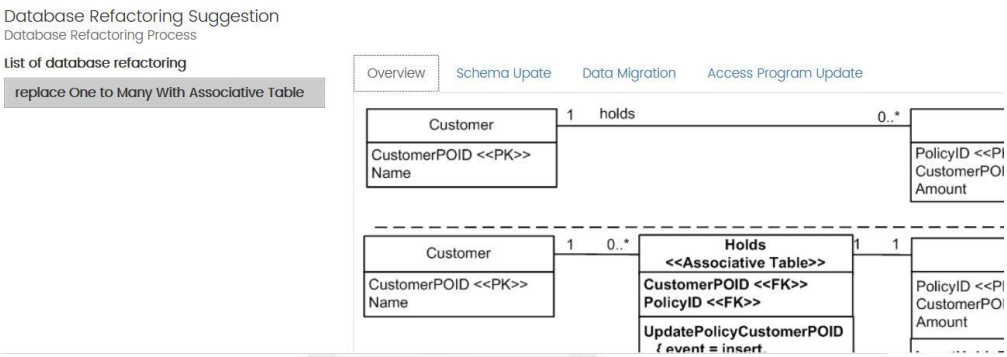
รูปที่ 42 การเชื่อมต่อฐานข้อมูล

เมื่อเชื่อมต่อสำเร็จจะสามารถดำเนินการขั้นถัดไป โดยออกแบบส่วนต่อประสานเครื่องมือให้แสดงเป็น 2 ส่วน คือ ส่วนของด้านบนจะเป็นส่วนของการค้นหาเอสคิวแอลแอนติแพตเทิร์นและการ

แสดงผลลัพธ์การค้นหาเอสคิวแอลแอนติแพตเทิร์นดังแสดงในรูปที่ 43 และ ส่วนของการแสดงคำแนะนำรีแฟคทอริงฐานข้อมูลดังแสดงในรูปที่ 44

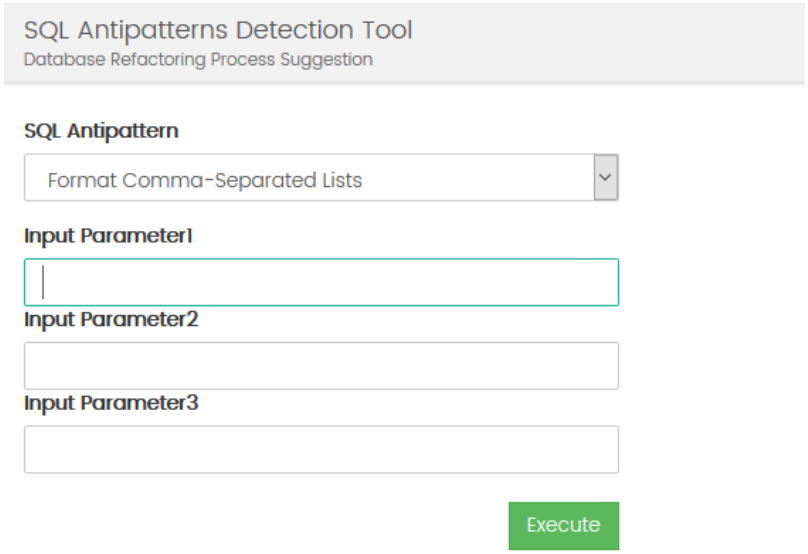


รูปที่ 43 ภาพหน้าจอส่วนของการค้นหาเอสคิวแอลแอนติแพตเทิร์น



รูปที่ 44 ภาพหน้าจอส่วนของการแสดงคำแนะนำรีแฟคทอริงฐานข้อมูล

ทั้งนี้ในส่วนของการค้นหาเอสคิวแอลแอนติแพตเทิร์นนั้น นอกจากผู้ใช้งานจะสามารถเลือกรายการเอสคิวแอลแอนติแพตเทิร์นที่ต้องการค้นหาภายในฐานข้อมูลได้แล้ว ก็ยังสามารถให้ผู้ใช้งานกรอกข้อมูลนำเข้าเพิ่มได้ (เฉพาะบางเอสคิวแอลแอนติแพตเทิร์น) โดยส่วนต่อประสานจะแสดงช่องรับข้อมูลนำเข้าโดยมีคำว่า “Input Parameter” เพื่อให้ผู้ใช้กรอกข้อมูลเพิ่มเติม ดังแสดงในรูปที่ 45



รูปที่ 45 กรณีที่มีข้อมูลนำเข้า

บทที่ 5
การประเมินผลเครื่องมือตรวจหาเอสคิวแอลแอนติแพตเทิร์นและแนะนำกระบวนการรีแฟคทอริง

การประเมินผลจะดำเนินการกับฐานข้อมูลจำลองและฐานข้อมูลที่สำเนามาจากฐานข้อมูลของระบบในอุตสาหกรรมจริง 3 ฐานข้อมูล โดยรายละเอียดโครงสร้างและข้อมูลที่สำเนาจากฐานข้อมูลของระบบในอุตสาหกรรมจริงจะไม่สามารถเผยแพร่ได้ จึงได้แสดงเฉพาะผลการทดลอง ส่วนฐานข้อมูลจำลองได้มีการสร้างให้มีลักษณะคล้ายกับฐานข้อมูลของระบบในอุตสาหกรรมจริงและฐานข้อมูลตาม B. Karwin ซึ่งจะทำให้ได้ผลสรุปการทดลองที่คล้ายฐานข้อมูลจริงเช่นกัน โดยแสดงรายละเอียดฐานข้อมูลในตารางที่ 9

ตารางที่ 9 รายละเอียดฐานข้อมูล

ฐานข้อมูล	ขนาด(กิกะไบต์)	จำนวนตาราง	จำนวนคอลัมน์
ฐานข้อมูลจำลอง	0.04	33	147
BankCallcenter	3.2	513	10409
TeleSale	5.3	487	8860
FoodOrdering	17.1	337	5490

การประเมินผลจะทำใน 2 ลักษณะ คือ การประเมินผลประสิทธิภาพของเครื่องมือโดยเปรียบเทียบกับวิธีการของงานวิจัยของ Eessaar [12] และการประเมินผลประสิทธิภาพของเครื่องมือกับฐานข้อมูลจริงในอุตสาหกรรม

5.1 การประเมินผลการตรวจหาโดยเปรียบเทียบกับ Eessaar โดยใช้ฐานข้อมูลจำลอง

หัวข้อนี้แสดงการประเมินผลประสิทธิภาพในการตรวจหาแอนติแพตเทิร์นด้วยขั้นตอนวิธีที่ผู้วิจัยเสนอกับวิธีการที่เสนอโดย Eessaar [12]

5.1.1 การดำเนินการทดสอบ

การดำเนินการทดสอบมีรายละเอียดดังนี้

1) ใส่รายการเอสคิวแอลแอนติแพตเทิร์นลงในฐานข้อมูลจำลอง และบันทึกรายการคำตอบว่ามีเอสคิวแอลแอนติแพตเทิร์นกี่จุด อยู่ตรงจุดใดบ้าง ให้เครื่องมือเอสคิวแอลแอนติแพตเทิร์นกระทำการ (Execute) การตรวจหาพร้อมทั้งใช้วิธีการของ Eessaar ในการตรวจหา โดยสามารถดูรายละเอียดเพิ่มเติมของรายการทดสอบได้ในภาคผนวก ข

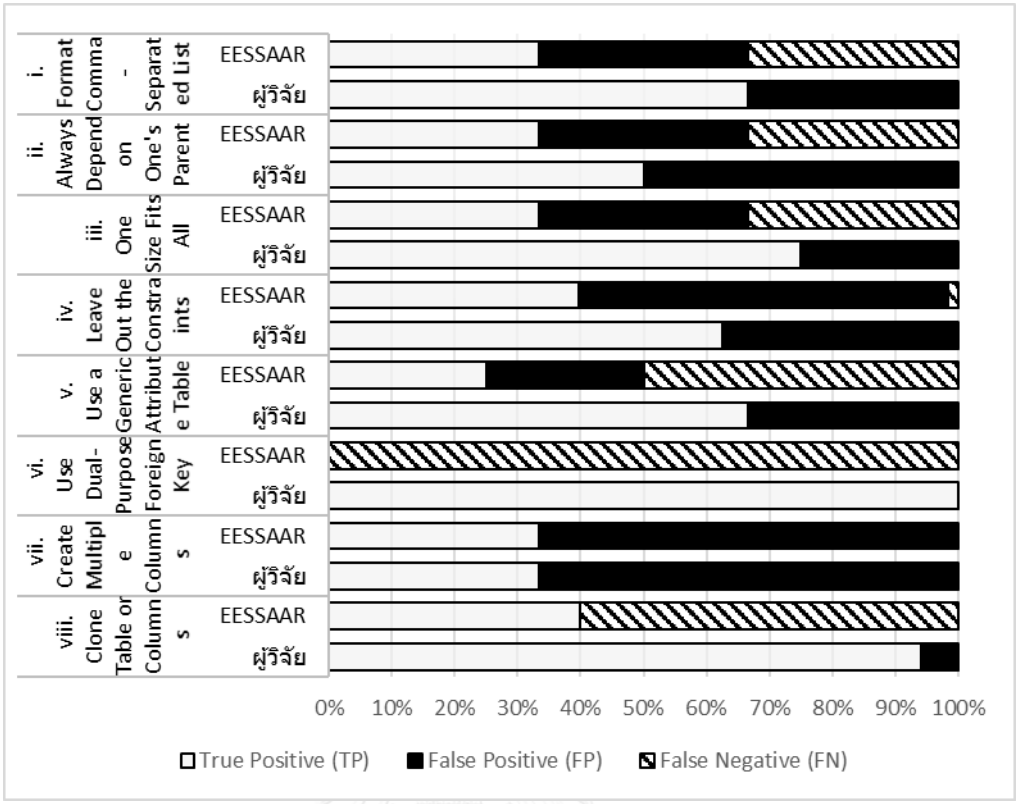
2) บันทึกผลลัพธ์ที่ได้จากเครื่องมือเทียบกับเฉลย โดยแยกตามรายการเอสคิวแอลแอนติแพตเทิร์น แล้วนำผลมาคำนวณค่าประสิทธิภาพการตรวจหา ได้แก่ ค่าความเที่ยงตรง ค่าเรียกคืน และค่าเอฟ-เมเชอร์ ตามสมการ *(1) (2)* และ *(3)* ในหัวข้อที่ 2.1.4
3) วิเคราะห์ผลค่าประสิทธิภาพ

5.1.2 ผลการทดสอบ

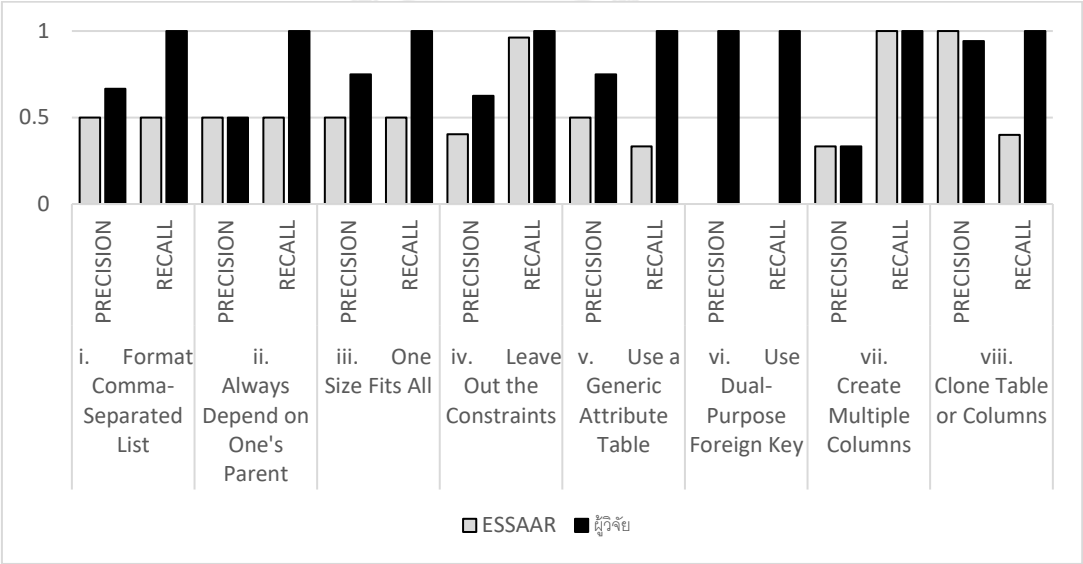
จากการดำเนินการทดสอบได้บันทึกค่า True Positive (TP), False Positive (FP) และ False Negative (FN) พร้อมทั้งค่าความเที่ยงตรง ค่าเรียกคืน และค่าเอฟ-เมเชอร์ของผู้วิจัยกับ Eessaar ไว้ในตารางที่ 10 โดยจะแสดงเปอร์เซ็นต์เปรียบเทียบค่า True Positive (TP) False Positive (FP) และ False Negative (FN) ดังกราฟรูปที่ 46 เปรียบเทียบค่าความเที่ยงตรง ค่าเรียกคืน ดังกราฟในรูปที่ 47 และเปรียบเทียบค่าเอฟ-เมเชอร์ ด้วยกราฟในรูปภาพที่ 48

ตารางที่ 10 รายละเอียดผลการประเมินประสิทธิภาพโดยใช้ฐานข้อมูลจำลอง

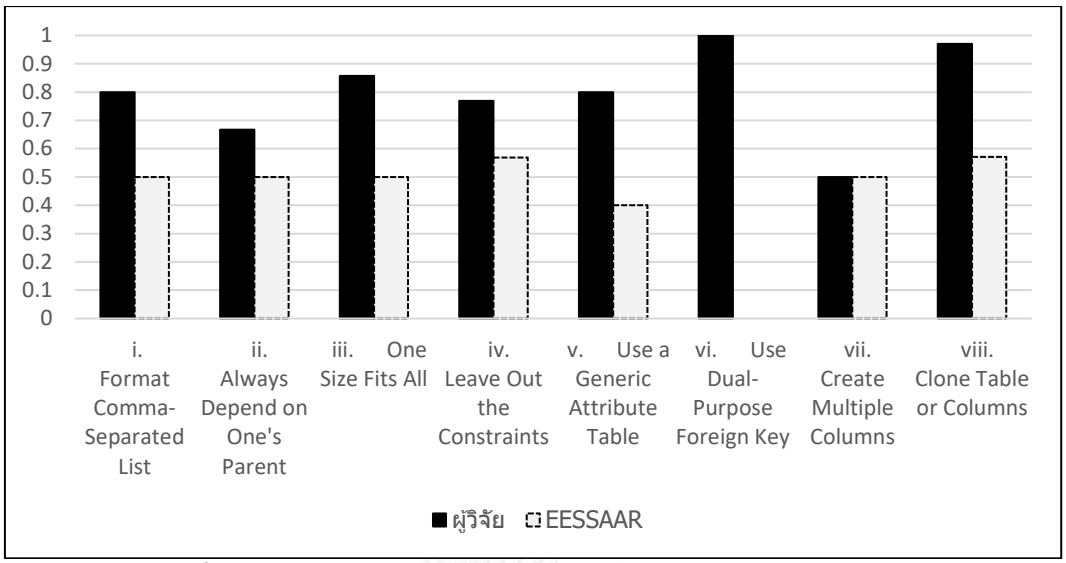
เอสคิวแอลแอนติแพตเทิร์น	ผู้เขียน	TP	FP	FN	ค่าความเที่ยงตรง	ค่าเรียกคืน	ค่าเอฟ-เมเชอร์
Format Comma-Separated List	ผู้วิจัย	2	1	0	0.667	1.000	0.800
	EESSAAR	1	1	1	0.500	0.500	0.500
Always Depend on One's Parent	ผู้วิจัย	1	1	0	0.500	1.000	0.667
	EESSAAR	1	1	1	0.500	0.500	0.500
One Size Fits All	ผู้วิจัย	3	1	0	0.750	1.000	0.857
	EESSAAR	1	1	1	0.500	0.500	0.500
Leave Out the Constraints	ผู้วิจัย	5	3	0	0.625	1.000	0.769
	EESSAAR	25	37	1	0.403	0.962	0.568
Use a Generic Attribute Table	ผู้วิจัย	3	1	0	0.750	1.000	0.857
	EESSAAR	1	1	2	0.500	0.333	0.400
Use Dual-Purpose Foreign Key	ผู้วิจัย	2	0	0	1.000	1.000	1.000
	EESSAAR	0	0	2	0.000	0.000	0.000
Create Multiple Columns	ผู้วิจัย	1	2	0	0.333	1.000	0.500
	EESSAAR	1	2	0	0.333	1.000	0.500
Clone Tables or Columns	ผู้วิจัย	16	1	0	0.941	1.000	0.970
	EESSAAR	8	0	12	1.000	0.400	0.571



รูปที่ 46 ภาพกราฟเปอร์เซ็นต์เปรียบเทียบค่า True Positive (TP) False Positive (FP) และ False Negative (FN) ของผู้วิจัยกับ Eessaar



รูปที่ 47 กราฟเปรียบเทียบค่าความเที่ยงตรงและค่าเรียกคืน ของผู้วิจัยกับ Eessaar



รูปที่ 48 กราฟเปรียบเทียบค่าเอฟ-เมเชอร์ ของผู้วิจัยกับ Eessaar

5.1.3 วิเคราะห์ผลการทดสอบ

จากผลการทดสอบดังตารางที่ 10 ค่าเอฟ-เมเชอร์ของผู้วิจัยส่วนมากมีค่ามากกว่าในแต่ละเอสคิวแอลแอนติแพตเทิร์น โดยอภิปรายผลแยกแต่ละแอนติแพตเทิร์นได้ดังนี้

1) Format Comma-Separated List

ผลที่ได้ของ Eessaar น้อยกว่าของผู้วิจัยเนื่องจาก Eessaar พิจารณาเฉพาะข้อมูลที่เป็นเครื่องหมายคั่นแค่ “,” เท่านั้น และเมื่อมีกรณีทดสอบเป็นเครื่องหมาย “|” ทำให้ตรวจหาไม่พบแต่เครื่องมือของผู้วิจัยสามารถตรวจพบได้ และสามารถที่จะเพิ่มข้อมูลนำเข้าเป็นเครื่องหมายคั่นอื่นใส่เข้าไปได้ แต่การตรวจหาของผู้วิจัยและ Eessaar นั้นยังได้ข้อมูลที่บรรจุเครื่องหมายคั่น แต่ไม่ใช่เอสคิวแอลแอนติแพตเทิร์นนี้ด้วย เช่น ข้อมูลที่อยู่ที่บรรจุเครื่องหมาย “,” ไว้

2) Always Depend on One's Parent

ผลที่ได้ของ Eessaar น้อยกว่าของผู้วิจัย เนื่องจาก Eessaar พิจารณาเฉพาะคอลัมน์แม่ลูกซึ่งเป็น Foreign keys เท่านั้น และเมื่อเพิ่มกรณีทดสอบที่ไม่ได้สร้าง Foreign key ไว้ ทำให้ตรวจหาไม่พบ แต่เครื่องมือของผู้วิจัยสามารถตรวจหาพบ เพราะไม่ได้กำหนดเงื่อนไขเกี่ยวกับ Foreign key ไว้ เนื่องจากมีโอกาสที่จะเกิดเอสคิวแอลแอนติแพตเทิร์นนี้ได้โดยพิจารณาจากข้อมูล แม้จะไม่ได้สร้าง Foreign key ก็ตาม

3) One Size Fits All

ผลที่ได้ของ Eessaar น้อยกว่าของผู้วิจัย เนื่องจาก Eessaar พิจารณาเฉพาะในกรณีที่ตารางมีการใช้ชื่อคอลัมน์ว่า “ID” เป็นคีย์หลัก และเมื่อเพิ่มกรณีทดสอบที่ใช้คำอื่น ทำให้ตรวจหาไม่พบ แต่ในความเป็นจริงแล้ว คอลัมน์ไม่จำเป็นต้องชื่อว่า ID โดยผู้วิจัยไม่ได้กำหนดการหาตารางด้วยชื่อ จึงสามารถตรวจพบได้

4) Leave Out the Constraints

ผลที่ได้ของ Eessaar น้อยกว่าของผู้วิจัย เนื่องจาก Eessaar ค้นหาคอลัมน์ที่อยู่ในคนละตารางแต่มีชื่อคอลัมน์ตรงกันโดยสันนิษฐานว่าควรจะมีการสร้าง Foreign Key ระหว่างกัน แต่ผลที่ได้นั้นไม่ใช่สิ่งที่ถูกต้องตามเฉลย เนื่องจากตารางที่มีชื่อคอลัมน์เหมือนกันไม่จำเป็นว่าจะเป็นการเชื่อมโยงกันเสมอไป ทำให้ผลที่ได้ของ Eessaar ต่ำ แต่ผู้วิจัยจะใช้วิธีการแสดงข้อมูล คอนสเตรนต์ แทนและเพิ่มการหา คอนสเตรนต์ ที่เป็นคีย์หลักและยูนีคเท่านั้น

5) Use a Generic Attribute Table

ผลที่ได้ของ Eessaar น้อยกว่าของผู้วิจัย เนื่องจาก Eessaar ค้นหาจากชื่อตารางที่คาดว่าจะมีการเก็บข้อมูลแอตทริบิวต์เท่านั้น เช่น object และเมื่อเพิ่มกรณีทดสอบที่ใช้คำอื่น ทำให้ตรวจหาไม่เจอ เนื่องจากไม่ตรวจสอบความสัมพันธ์เพิ่มเติม ซึ่งผู้วิจัยที่ใช้การตรวจสอบข้อมูลเพิ่มเติมด้วย

6) Use Dual-Purpose Foreign Key

ผลของ Eessaar นั้นตรวจไม่พบแอนติแพตเทิร์นนี้ เนื่องจากตรวจสอบด้วยการค้นหา ข้อมูล คอนสเตรนต์ ของคอลัมน์ในประเภท Check และหาคู่ของคอลัมน์ที่มีชื่อเหมือนกัน แม้ว่าจะใช้กรณีทดสอบตาม B Karwin แล้วก็ตาม แต่วิธีการของผู้วิจัยสามารถตรวจหาแอนติแพตเทิร์นนี้ได้

7) Create Multiple Columns

ผลของ Eessaar และผู้วิจัยได้เท่ากันเนื่องจากแนวการคิดคล้ายกันคือเปรียบเทียบชื่อคอลัมน์และตรวจสอบข้อมูลตัวเลขที่ลงท้าย แต่ผู้วิจัยได้เพิ่มเติมฟังก์ชันเพื่อให้แตกต่างคือ สามารถแก้ไขจำนวนตัวอักษรที่ใช้เปรียบเทียบความเหมือนของคอลัมน์ได้

8) Clone Table or Columns

ผลที่ได้ของ Eessaar น้อยกว่าของผู้วิจัย เนื่องจาก Eessaar ค้นหาจากชื่อตารางที่ตัดตัวเลขด้านท้ายออกและมีโครงสร้างเหมือนกันมาเทียบ แต่กรณีทดสอบเพิ่มเติมเช่นกรณีที่โครงสร้างตารางเหมือนกันแต่อาจจะมีชื่อที่เป็นตัวอักษรคือ order_a และ order_b ทำให้ขั้นตอนวิธีการตรวจหาของ Essaar ไม่สามารถตรวจพบ

5.2 การประเมินผลการตรวจหาโดยใช้ฐานข้อมูลที่สำเนามาจากฐานข้อมูลของระบบในอุตสาหกรรมจริง

หัวข้อนี้แสดงการประเมินผลประสิทธิภาพในการตรวจหาแอนติแพตเทิร์นด้วยขั้นตอนวิธีที่ผู้วิจัยเสนอโดยใช้ฐานข้อมูลจริง

5.2.1 การดำเนินการทดสอบ

การดำเนินการทดสอบมีรายละเอียดดังนี้

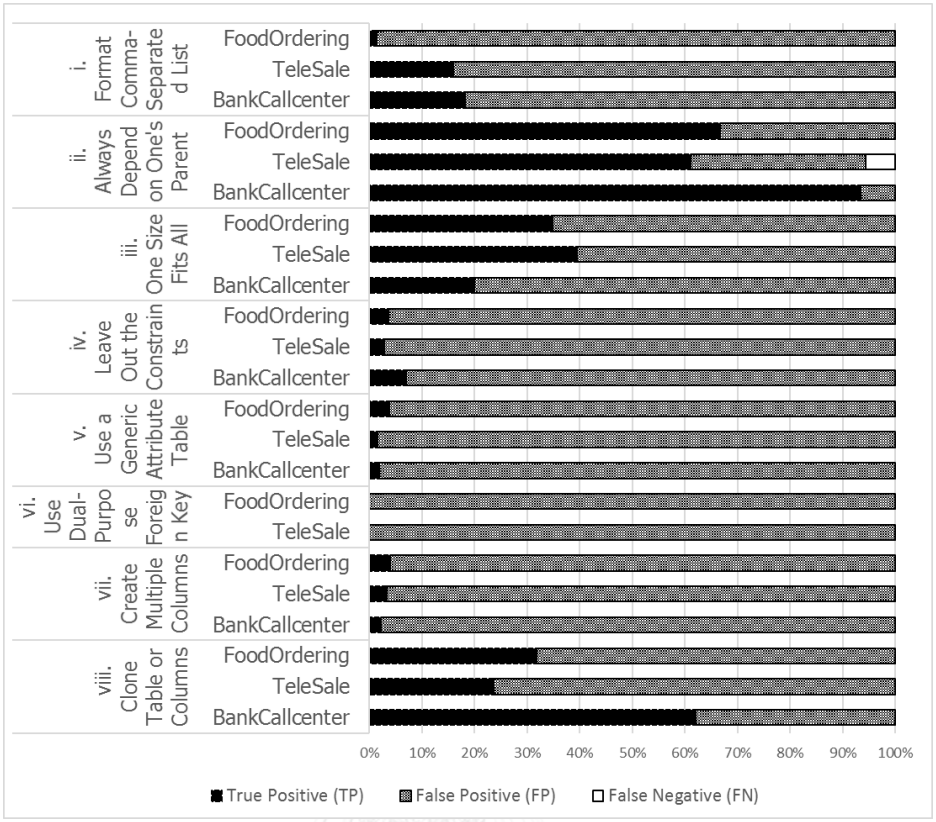
1) ผู้ดูแลฐานข้อมูลที่มีประสบการณ์ในการใช้ฐานข้อมูลของระบบในอุตสาหกรรมดังในตารางที่ 9 เป็นเวลา 5 ปี ทำการตรวจหารายการเอสคิวแอลแอนติแพตเทิร์นด้วยตนเองและบันทึกรายการผลลัพธ์
2) ผู้ดูแลฐานข้อมูลใช้เครื่องมือเอสคิวแอลแอนติแพตเทิร์น
3) บันทึกผลลัพธ์ที่ได้จากเครื่องมือเทียบกับเฉลย โดยแยกตามรายการเอสคิวแอลแอนติแพตเทิร์น แล้วนำผลมาคำนวณค่าประสิทธิภาพการตรวจหา ได้แก่ ค่าความเที่ยงตรง ค่าเรียกคืน และค่าเอฟ-เมเชอร์ ตามสมการ *(1) (2)* และ *(3)* ในหัวข้อที่ 2.1.4
4) วิเคราะห์ผลค่าประสิทธิภาพ

5.2.2 ผลการทดสอบ

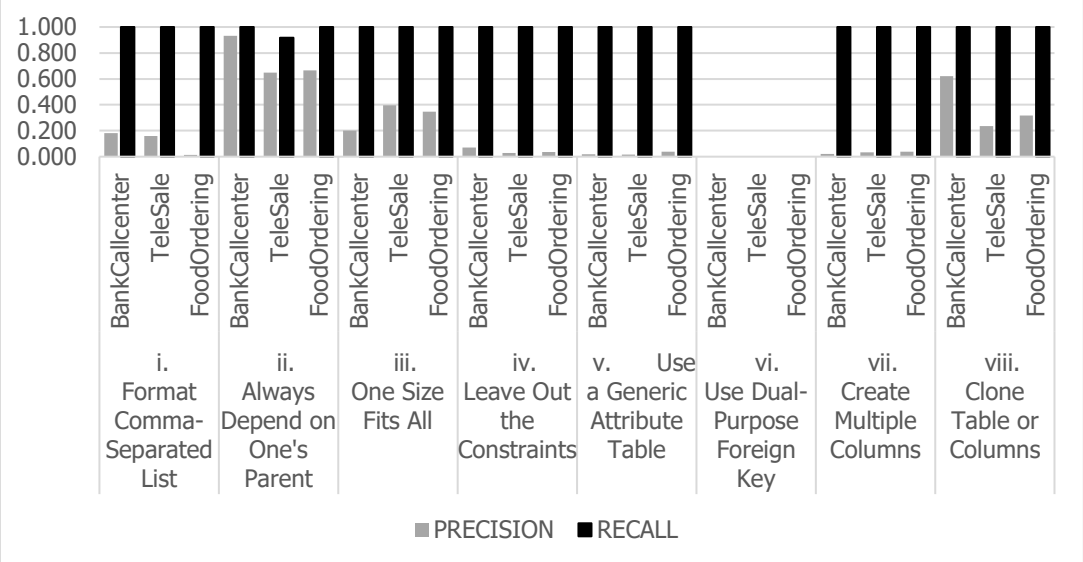
จากการดำเนินการทดสอบได้บันทึกค่า True Positive (TP), False Positive (FP) และ False Negative (FN) พร้อมทั้งค่าความเที่ยงตรง ค่าเรียกคืน และค่าเอฟ-เมเชอร์ ของ 3 ฐานข้อมูลไว้ในตารางที่ 11 โดยจะแสดงภาพกราฟเปอร์เซ็นต์เปรียบเทียบค่า True Positive (TP), False Positive (FP) และ False Negative (FN) ดังกราฟในรูปที่ 49 เปรียบเทียบค่าความเที่ยงตรง ค่าเรียกคืน ดังกราฟในรูปที่ 50 และเปรียบเทียบค่าเอฟ-เมเชอร์ ด้วยกราฟในรูปที่ 51

ตารางที่ 11 รายละเอียดผลการประเมินประสิทธิภาพโดยใช้ฐานข้อมูลจริง

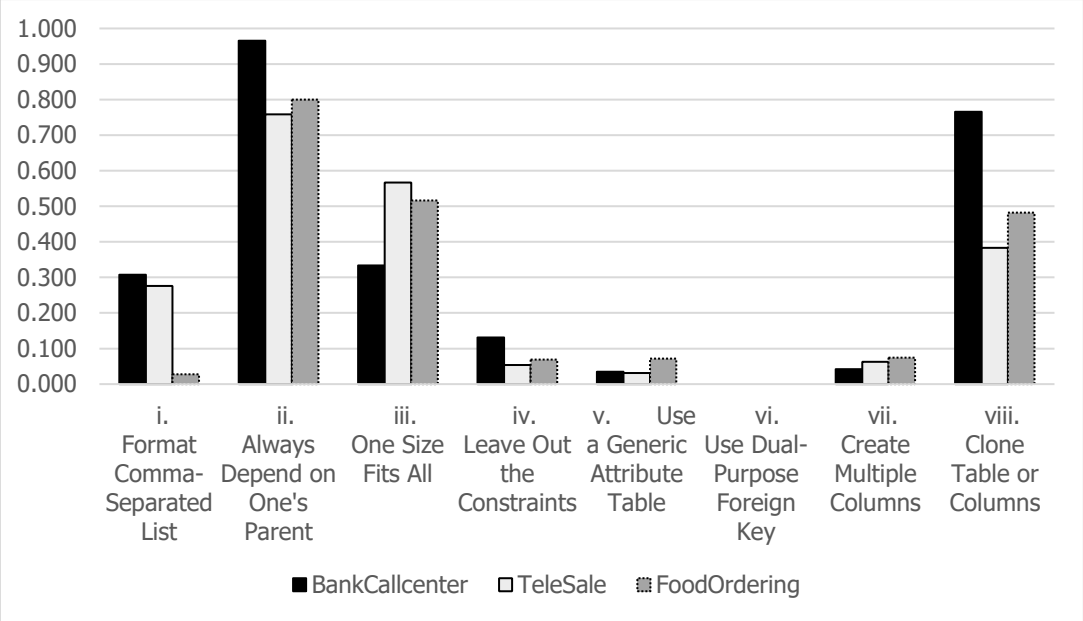
เอสคิวแอลแอนติแพตเทิร์น	ฐานข้อมูล	TP	FP	FN	ค่าความเที่ยงตรง	ค่าเรียกคืน	ค่าเอฟ-เมเชอร์
Format Comma-Separated List	BankCallcenter	4	18	0	0.182	1.000	0.308
	TeleSale	4	21	0	0.160	1.000	0.276
	FoodOrdering	1	70	0	0.014	1.000	0.028
Always Depend on One's Parent	BankCallcenter	14	1	0	0.933	1.000	0.966
	TeleSale	11	6	1	0.647	0.917	0.759
	FoodOrdering	4	2	0	0.667	1.000	0.800
One Size Fits All	BankCallcenter	6	24	0	0.200	1.000	0.333
	TeleSale	17	26	0	0.395	1.000	0.567
	FoodOrdering	8	15	0	0.348	1.000	0.516
Leave Out the Constraints	BankCallcenter	8	106	0	0.070	1.000	0.131
	TeleSale	4	140	0	0.028	1.000	0.054
	FoodOrdering	6	161	0	0.036	1.000	0.069
Use a Generic Attribute Table	BankCallcenter	2	111	0	0.018	1.000	0.035
	TeleSale	2	125	0	0.016	1.000	0.031
	FoodOrdering	3	78	0	0.037	1.000	0.071
Use Dual-Purpose Foreign Key	BankCallcenter	0	26	0	0.000	N/A	N/A
	TeleSale	0	11	0	0.000	N/A	N/A
	FoodOrdering	0	9	0	0.000	N/A	N/A
Create Multiple Columns	BankCallcenter	1	46	0	0.021	1.000	0.042
	TeleSale	1	30	0	0.032	1.000	0.063
	FoodOrdering	1	25	0	0.038	1.000	0.074
Clone Tables or Columns	BankCallcenter	131	80	0	0.621	1.000	0.766
	TeleSale	36	116	0	0.237	1.000	0.383
	FoodOrdering	34	73	0	0.318	1.000	0.482



รูปที่ 49 ภาพกราฟเปอร์เซ็นต์เปรียบเทียบค่า True Positive (TP), False Positive (FP) และ False Negative (FN) ของ 3 ฐานข้อมูล



รูปที่ 50 กราฟเปรียบเทียบค่าความเที่ยงตรงและค่าเรียกคืน ของ 3 ฐานข้อมูล



รูปที่ 51 กราฟเปรียบเทียบค่าเอฟ-เมเซอร์ ของ 3 ฐานข้อมูล

5.2.3 วิเคราะห์ผลการทดสอบ

จากผลการทดสอบในฐานข้อมูลอุตสาหกรรมจริงดังตารางที่ 11 จะได้ค่าความเที่ยงตรงต่ำ และส่งผลต่อเนื่องทำให้ค่าเอฟ-เมเชอร์ต่ำด้วย แม้ว่าค่าเรียกคืนได้ค่าที่สูงและน่าพึงพอใจ แต่โดยภาพรวมทั้ง 3 ฐานข้อมูลนี้จะได้ค่าเอฟ-เมเชอร์ในแต่ละเอสคิวแคลแอนติแพตเทิร์นไล่เลี่ยกัน เนื่องมาจากฐานข้อมูลนี้ผู้ออกแบบฐานข้อมูลเป็นคนเดียวกันและเชื่อมต่อเข้ากับซอฟต์แวร์ที่หน้าที่การทำงานคล้ายกัน

สาเหตุที่ค่าความเที่ยงตรงต่ำนั้น เพราะค่า False Positive (FP) จากการตรวจหามีจำนวนมาก โดยตัวอย่างสาเหตุที่เกิดขึ้นแสดงไว้ในตารางที่ 12

ตารางที่ 12 ตัวอย่างโครงสร้างและข้อมูลของฐานข้อมูลอุตสาหกรรมที่ทำให้ False Positive (FP) มีค่าสูง

<table>
<tr><th>เอสคิวแอลแอนติแพตเทิร์น</th><th>ตัวอย่าง</th></tr>
<tr><td>Format Comma-Separated List</td><td>CREATE TABLE [REPORT_PARAMETER](
[VAR_ID] [varchar](20) NOT NULL,
[VAR_VALUE] [varchar](200) NULL)
<table>
<tr><th>VAR_ID</th><th>VAR_VALUE</th></tr>
<tr><td>1</td><td>|002-Call Center|003-Web |</td></tr>
<tr><td>2</td><td>|001-COMPLAINT|002-SUGGEST|</td></tr>
</table></td></tr>
<tr><td>Always Depend on One’s Parent</td><td>CREATE TABLE [dbo].[INCIDENT](
[INCIDENT_ID] [varchar](20) NOT NULL,
[ORG_NAME] [varchar](200) NULL,
[RECEIVED_BY] [varchar](100) NULL,
[ASSIGNEE] [9] NULL,
[HEAD] [9] NULL,
[UPPERHEAD] [9] NULL)</td></tr>
</table>

ตารางที่ 12 ตัวอย่างโครงสร้างและข้อมูลของฐานข้อมูลอุตสาหกรรมที่ทำให้ False Positive (FP) มีค่าสูง (ต่อ)

<table>
<tr><th>เอสคิวแอลแอนติแพตเทิร์น</th><th>ตัวอย่าง</th></tr>
<tr><td>One Size Fits All และ Leave Out the Constraints (Primary Key, Unique)</td><td>SELECT [REPORT_ID],[DESCRIPTION]
FROM [REPORT]
<table><tr><th>GROUP_ID</th><th>DESCRIPTION</th></tr><tr><td>1</td><td>Service North Report</td></tr><tr><td>2</td><td>Summary Report</td></tr><tr><td>3</td><td>Detail Report</td></tr><tr><td>4</td><td>Executive Report</td></tr><tr><td>5</td><td>Manager Report</td></tr></table></td></tr>
<tr><td>Use a Generic Attribute Table</td><td>SELECT [NAME], [POSITION_NAME]
FROM [EMPLOYEE]
<table><tr><th>NAME</th><th>POSITION_NAME</th></tr><tr><td>สุขใจ</td><td>ผู้จัดการสาขา</td></tr><tr><td>สุกานดา</td><td>ผู้จัดการสาขา</td></tr><tr><td>สุกัญญา</td><td>ผู้จัดการสาขา</td></tr><tr><td>สุขสบาย</td><td>ผู้จัดการสาขา</td></tr></table></td></tr>
<tr><td>Use Dual-Purpose Foreign Key</td><td>SELECT [CAPTION_ID],[CAPTION_DESC]
FROM [MENU_CONFIG]
<table><tr><th>CAPTION_ID</th><th>CAPTION_DESC</th></tr><tr><td>1</td><td>STATUS</td></tr></table>CREATE TABLE STATUS () ;</td></tr>
</table>

ตารางที่ 12 ตัวอย่างโครงสร้างและข้อมูลของฐานข้อมูลอุตสาหกรรมที่ทำให้ False Positive (FP) มีค่าสูง (ต่อ)

เอสคิวแอลแอนติแพตเทิร์น	ตัวอย่าง
Create Multiple Columns	[ADDR_1] [varchar](100) NULL, [ADDR_2] [varchar](100) NULL, [SITEADDR1] [varchar](60) NULL, [SITEADDR2] [varchar](60) NULL, [BILLADDR1] [varchar](60) NULL, [BILLADDR2] [varchar](60) NULL, [PROBLEM_DETAIL1] [varchar](4000) NULL, [PROBLEM_DETAIL2] [varchar](4000) NULL, [PROBLEM_DETAIL3] [varchar](4000) NULL, [PROBLEM_DETAIL4] [varchar](4000) NULL,
Clone table or column	CREATE TABLE [dbo].[CONTACT_CHANNEL]([CONTACT_ID] [varchar](20) NOT NULL, [ADDR_SEQ_NO] [9] NOT NULL, [SEQ_NO] [9] NOT NULL, [CHANNEL_TYPE] [varchar](20) NULL, [CHANNEL_DESC] [varchar](200) NULL, [CREATE_USER_ID] [9] NULL) CREATE TABLE [dbo].[ACCOUNT_CHANNEL]([ACCOUNT_ID] [varchar](20) NOT NULL, [ADDR_SEQ_NO] [9] NOT NULL, [SEQ_NO] [9] NOT NULL, [CHANNEL_TYPE] [varchar](20) NULL, [CHANNEL_DESC] [varchar](200) NULL,, [CREATE_USER_ID] [9] NULL)

จากตารางข้างต้นอภิปรายสาเหตุแยกแต่ละแอนติแพตเทิร์นได้ดังนี้

1) Format Comma-Separated List

มีข้อมูลที่เป็นข้อมูลรายละเอียดเช่น VAR_VALUE ที่เก็บค่าตัวแปรสำหรับนำไปแสดงบนหน้าจอ โดยมีการเก็บรายการไว้ในฐานข้อมูล เพื่อหน้าจอจะตัดข้อความด้วยเครื่องหมายคั่น “|” เช่น 002_Callcenter และ 003_Web เพื่อนำไปแสดงบนหน้าจอแต่ค่าข้อมูลนี้ไม่ได้เป็นข้อมูลที่เชื่อมโยงไปยังตารางอื่น จึงไม่ใช่แอนติแพตเทิร์น แต่เครื่องมือตรวจพบที่จุดนี้

2) Always Depend on One's Parent

ข้อมูลในคอลัมน์ที่ดูเหมือนจะมีความสัมพันธ์ระหว่างคอลัมน์แม่กับคอลัมน์ลูก เช่น HEAD และ UPPERHEAD แต่เมื่อตรวจสอบแล้วข้อมูล UPPERHEAD ไม่ใช่ Parent ของ HEAD ตามลำดับชั้น แต่เป็นเพียงการเก็บข้อมูลว่า HEAD มีเจ้านายคือ UPPER HEAD

3) One Size Fits All และ Leave Out the Constraints (Primary Key, Unique)

มีคอลัมน์ที่ไม่สามารถเป็นคีย์ได้โดยธรรมชาติของข้อมูล แต่เป็นคอลัมน์ที่มีค่าไม่ซ้ำกันเลย เช่น DESCRIPTION เนื่องจากค่าในคอลัมน์เป็นข้อความยาว

4) Use a Generic Attribute Table

มีข้อมูลในคอลัมน์ที่เก็บค่าซ้ำ ๆ กันดังเช่นตัวอย่างตารางเก็บค่า POSITION_NAME ที่ซ้ำ ๆ กัน แต่คำว่าผู้จัดการสาขา ไม่ใช่แอตทริบิวต์

5) Use Dual-Purpose Foreign Key

มีคอลัมน์ที่มีรายละเอียดตรงกับชื่อตารางแต่ไม่ได้เป็นการเชื่อมโยงไปที่ตารางอื่น เช่น ตาราง MENU_CONFIG เก็บค่าคำว่า “STATUS” ซึ่งไปตรงกับตาราง STATUS แต่คำว่า “STATUS” นี้ไม่ได้หมายถึงชื่อตาราง STATUS

6) Create Multiple Columns

มีคอลัมน์ที่ปรากฏเป็นแอตทริบิวต์หลายค่า แต่ในความเป็นจริงเป็นส่วนหนึ่งของแอตทริบิวต์อื่น ๆ เช่น ADDR_1 และ ADDR_2 โดยทั้งคู่เป็นส่วนหนึ่งของแอตทริบิวต์ Address ทั้งหมด

7) Clone Table or Columns

มีตารางที่มีโครงสร้างตารางเหมือนกันเช่น CONTACT_CHANNEL และ ACCOUNT_CHANNEL แต่ไม่ใช่การโคลนนิ่งกัน

โดยจากสาเหตุข้างต้นนั้น หากต้องการให้ค่า False Positive ลดลงเพื่อทำให้ค่าความเที่ยงตรงสูงขึ้น ผู้ออกแบบฐานข้อมูลควรที่จะสามารถระบุข้อมูลองค์ความรู้เกี่ยวกับฐานข้อมูลนั้นเพิ่มเติมให้กับเครื่องมือได้ เพื่อเป็นการเพิ่มข้อมูลนำเข้าสำหรับคัดกรองส่วนที่ไม่น่าจะเป็นแอนติแพตเทิร์นออก และขอบเขตการตรวจหาก็จะสามารถลดการค้นพบข้อมูลที่ทำให้เกิด False Positive ได้ เช่น จากข้อ 6) ถ้าผู้ออกแบบฐานข้อมูลสามารถใส่ข้อมูลนำเข้าเป็น ADDR_1 และ ADDR_2 เพื่อไม่ให้เครื่องมือทำการตรวจหาเพราะทราบอยู่แล้วว่า ADDR_1 และ ADDR_2 ไม่ใช่แอนติแพตเทิร์น แต่เป็นส่วนหนึ่งของแอตทริบิวต์ Address ทั้งหมด อย่างไรก็ตาม หากมีการระบุข้อมูลนำเข้าดังข้างต้น อาจจะส่งผลกระทบให้ค่าเรียกคืนลดลงได้เพราะอาจจะทำให้การตรวจหาไม่ได้ทำการตรวจในทุกส่วนของฐานข้อมูล

นอกจากนี้ จากตารางที่ 11 ยังพบกรณี False Negative ของแอนติแพตเทิร์น Always Depend on One's Parent จำนวน 1 รายการ เนื่องจากในตารางมีคอลัมน์ที่มีความสัมพันธ์แบบแม่ลูกกัน คือ คอลัมน์ CATEGORY_LV2 อ้างอิงคอลัมน์แม่ CATEGORY_LV1 แต่เนื่องจากเครื่องมือค้นหาคอลัมน์ที่มีความสัมพันธ์กันจากชื่อคอลัมน์ที่มีคำว่า "PARENT" หรือ "UPPER" จึงไม่สามารถค้นพบความสัมพันธ์แบบแม่ลูกระหว่าง CATEGORY_LV1 และ CATEGORY_LV2 ได้ ดังนั้นในทำนองเดียวกับกรณีของ False Positive หากผู้ออกแบบฐานข้อมูลทำการระบุคำสำคัญในการใช้ค้นหาความสัมพันธ์แบบแม่ลูกเพิ่มเติม เช่น "LV" ก็จะช่วยลด False Negative ในกรณีนี้ได้

5.3 การประเมินผลการแนะนำกระบวนการรีแฟคทอริง

การประเมินผลว่ากระบวนการรีแฟคทอริงที่แนะนำนั้น เมื่อนำไปใช้ปรับปรุงฐานข้อมูลแล้ว การทำงานเชิงฟังก์ชันในฐานข้อมูล เช่น เอสคิวแอลคอมมานด์ ยังสามารถทำงานได้เหมือนเดิมหรือไม่ โดยจะทดสอบโดยใช้ฐานข้อมูลจำลอง ดังรายละเอียดในภาคผนวก ค โดยมีขั้นตอนดังนี้

1) สร้างกรณีทดสอบโดยใช้ เอสคิวแอลคอมมานด์ ที่ตารางที่ถูกดำเนินการรีแฟคทอริงนั้นเกี่ยวข้องด้วยและบันทึกผลลัพธ์ที่ได้จากการใช้กรณีทดสอบ
2) ทำการรีแฟคทอริงฐานข้อมูล
3) เนื่องจากการรีแฟคทอริงฐานข้อมูล ส่งผลให้โครงสร้างฐานข้อมูลเปลี่ยนจึงทำการสร้างกรณีทดสอบโดยใช้เอสคิวแอลคอมมานด์ชุดใหม่
4) บันทึกผลลัพธ์ที่ได้ จากการใช้กรณีทดสอบชุดใหม่
5) เปรียบเทียบผลลัพธ์จากกรณีทดสอบทั้งสองชุด ผลลัพธ์จากการประเมินแสดงในตารางที่ 13

ตารางที่ 13 ผลกรณีทดสอบของกระบวนการรีแฟคทอริง

เอสคิวแอลแอนติแพตเทิร์น	รายการวิธีรีแฟคทอริงฐานข้อมูล	ผลลัพธ์ก่อนและหลังการดำเนินการรีแฟคทอริง
Format Comma-Separated Lists	Replace One to Many with Associative Table.	ผลลัพธ์ตรงกัน
Always Depend on One's Parent	1) Introduce New Column 2) Drop Column	ผลลัพธ์ตรงกัน
One Size Fits All	1) Replace Surrogate Key with Natural Key 2) Drop Column	ผลลัพธ์ตรงกัน
Leave Out the Constraints	• Introduce New Column Constraint • Make column non-nullable • Add foreign key Constraint • Introduce Default Value	ผลลัพธ์ตรงกัน
Use a Generic Attribute Table	1) Introduce New table 2) Drop Table	ผลลัพธ์ตรงกัน
Use Dual-Purpose Foreign Key	1) Split Table 2) Add foreign key Constraint 3) Drop Column	ผลลัพธ์ตรงกัน
Create Multiple Columns	1) Replace One to many With Associative Table 2) Drop Column	ผลลัพธ์ตรงกัน
Clone Table or Column	1) Merge Table 2) Add Column 3) Drop Table	ผลลัพธ์ตรงกัน

บทที่ 6
สรุปผลการวิจัยและข้อเสนอแนะ

6.1 สรุปผลการดำเนินโครงงาน

งานวิจัยนี้เสนอเครื่องมือที่สามารถช่วยผู้ดูแลระบบฐานข้อมูลในการวิเคราะห์และตรวจหาเอสคิวแอลแอนติแพตเทิร์น 8 ประเภท และแนะนำเทคนิคการรีแฟคทอริงฐานข้อมูลได้ โดยพัฒนาด้วยทรานแซก-เอสคิวแอล อย่างไรก็ตามผู้วิจัยเห็นว่า ผู้ดูแลระบบฐานข้อมูลจำเป็นต้องตรวจสอบสกีมาเพิ่มเติม เพราะขั้นตอนวิธีในการตรวจหาไม่เพียงพอที่จะวิเคราะห์การออกแบบโครงสร้างฐานข้อมูลได้อย่างอัตโนมัติ ซึ่งจากการทดลองกับฐานข้อมูลจริงจะพบ False Positive จำนวนมาก ดังนั้นควรใช้เครื่องมือนี้ในลักษณะกึ่งอัตโนมัติแทน เช่น สามารถใช้ชี้ตำแหน่งที่อาจเป็นปัญหาได้ภายในสกีมาฐานข้อมูลจากนั้นทำการวินิจฉัยเพิ่มเติมโดยผู้ดูแลระบบฐานข้อมูล วิธีนี้จะเป็นประโยชน์โดยเฉพาะอย่างยิ่งในบริบทของฐานข้อมูลขนาดใหญ่ที่การตรวจสอบด้วยมนุษย์เป็นเรื่องยาก อย่างไรก็ตามจากการเปรียบเทียบประสิทธิภาพกับวิธีการตรวจหาแอนติแพตเทิร์นของ Eeassar [12] พบว่าขั้นตอนวิธีของผู้วิจัยมีประสิทธิภาพดีกว่าใน 7 แอนติแพตเทิร์น และมีประสิทธิภาพเท่ากัน 1 แอนติแพตเทิร์น

6.2 ปัญหาและข้อจำกัด

1. การใช้ฐานข้อมูลที่สำเนามาจากฐานข้อมูลจริงจำเป็นต้องใช้ผู้ที่มีความรู้ในการสร้างฐานข้อมูลนั้น มิฉะนั้นการดำเนินการทำผลเฉลยจะทำให้ผิดพลาดได้
2. การใช้ฐานข้อมูลที่สำเนามาจากฐานข้อมูลจริงอาจจะไม่มีตัวอย่างข้อมูลสำหรับเอสคิวแอลแอนติแพตเทิร์นนั้นได้ เช่น จากฐานข้อมูลจริงไม่พบแอนติแพตเทิร์น Use Dual-Purpose Foreign Key
3. การใช้ฐานข้อมูลที่สำเนามาจากฐานข้อมูลจริงจะทำให้เราได้เห็นข้อมูลในมุมมองของฐานข้อมูลในขอบเขตตามธุรกิจนี้เท่านั้น ซึ่งฐานข้อมูลอื่น ๆ อาจจะแสดงผลที่ต่างออกไปได้
4. ในการตรวจหาแอนติแพตเทิร์น เช่น Use Dual-Purpose Foreign Key ยังใช้เวลามาก เนื่องจากมีการตรวจหาในทุกค่าของข้อมูลและทุกตาราง

6.3 ข้อเสนอแนะในการดำเนินงานต่อ

1. ปรับปรุงเครื่องมือนี้เพื่อสนับสนุนประเภทการตรวจหาเอสคิวแอลแอนติแพตเทิร์นในกลุ่มอื่น ๆ

2. ปรับปรุงให้ผู้ดูแลระบบฐานข้อมูลสามารถระบุความรู้เกี่ยวกับโดเมนเพื่อทำให้การตรวจหามีประสิทธิภาพมากขึ้น
3. ปรับปรุงขั้นตอนวิธีการตรวจหาให้สามารถค้นหาได้เร็วมากขึ้น
4. ปรับปรุงให้การแนะนำรีแฟคทอริงฐานข้อมูล ควรเป็นแบบอัตโนมัติมากขึ้น

รายการอ้างอิง

[1] M. Fowler, "Refactoring. Improving the Design of Existing Code.," *Addison-Wesley,* 1999.

[2] E. Fernandes, J. Oliveira, G. Vale, T. Paiva, and E. Figueiredo, "A review-based comparative study of bad smell detection tools," *Proceedings of the* 20*th International Conference on Evaluation and Assessment in Software Engineering - EASE '*16*,* pp. 1-12, 2016.

[3] B. Karwin, "SQL Antipatterns Avoiding the pitfalls of Database Programming," 2010.

[4] S. J. Ambler, "Refactoring Databases: Evolutionary Database Design," 2006.

[5] Microsoft, "Transact-SQL."

[6] W. J. Brown, R. C. Malveau, T. J. Mowbray, and J. Wiley, "AntiPatterns: Refactoring Software , Architectures, and Projects in Crisis," *Crisis,* vol. 3, pp. 281-284, 1998.

[7] amplysoft, "T-SQL คืออะไร มีประโยชน์อย่างไร."

[8] F. A. Fontana, P. Braione, and M. Zanoni, "Automatic detection of bad smells in code: An experimental assessment," *Journal of Object Technology,* vol. 11, pp. 1-38, 2012.

[9] G. H. Pinto and F. Kamei, "What Programmers Say About Refactoring Tools?: An Empirical Investigation of Stack Overflow," *Proceedings of the* 2013 *ACM Workshop on Refactoring Tools,* pp. 33-36, 2013.

[10] A. Yamashita and L. Moonen, "Do developers care about code smells? An exploratory survey," *Proceedings - Working Conference on Reverse Engineering, WCRE,* pp. 242-251, 2013.

[11] K. Nongpong, "Integrating "Code Smells" Detection with Refactoring Tool Support," vol. 3523928, ed, 2012, p. 154.

[12] E. Eessaar, "Innovations and Advances in Computing, Informatics, Systems Sciences, Networking and Engineering," vol. 313, pp. 53-60, 2015.

[13] M. B. P. Domingues, J. R. de Almeida, W. F. Costa, and A. Mauro Saraiva, "A workflow for database refactoring," *International Journal of Innovative Computing, Information and Control,* vol. 10, pp. 2209-2220, 2014.

[14] G. Vial, "Database Refactoring: Lessons from the Trenches," *IEEE Software,* vol. 32, pp. 71-79, 2015.

[15] H. Schink, "Sql-schema-comparer: Support of multi-language refactoring with relational databases," *IEEE 13th International Working Conference on Source Code Analysis and Manipulation, SCAM* 2013, pp. 173-178, 2013.

[16] C. Szabó, L. Samuelis, M. Ivanović, and T. Fesič, "Database refactoring and regression testing of android mobile applications," 2012 *IEEE 10th Jubilee International Symposium on Intelligent Systems and Informatics, SISY* 2012, pp. 135-139, 2012.

ภาคผนวก

ภาคผนวก ก

ทรานแซกเอสคิวแอลของแต่ละแอนติแพตเทิร์น

การตรวจหาเอสคิวแอลแอนติแพตเทิร์น จะมีวิธีการเรียกใช้คำสั่งของไมโครซอฟท์เอสคิวแอลเซิร์ฟเวอร์ ที่ชื่อว่า INFORMATION_SCHEMA บ่อยครั้ง ซึ่งถูกเรียกใช้เพื่อแสดงถึงข้อมูลของสกีมา โดยมีการเรียกใช้ด้วยคำสั่ง INFORMATION_SCHEMA.VIEW_NAME โดยสำหรับงานวิจัยนี้เลือกใช้ชุดคำสั่งในการเขียนทรานแซกเอสคิวแอลเพื่อตรวจหาเอสคิวแอลแอนติแพตเทิร์น ได้แก่ INFORMATION_SCHEMA.TABLE ที่สามารถแสดงรายชื่อตารางทั้งหมดภายในฐานข้อมูลและ INFORMATION_SCHEMA.COLUMN ที่สามารถแสดงรายละเอียดของคอลัมน์ในตารางภายในฐานข้อมูลได้ เช่น DATA_TYPE, CHARACTER_MAXIMUM_LENGTH, CHARACTER_OCTET_LENGTH, NUMERIC_PRECISION,NUMERIC_SCALE และ CONSTRAINT_COLUMN_USAGE ขั้นตอนวิธีการตรวจหาแสดงในตารางที่ 17

ตารางที่ 14 ทรานแซกเอสคิวแอลของแต่ละแอนติแพตเทิร์น

เอสคิวแอลแอนติแพตเทิร์น	ทรานแซกเอสคิวแอล
Format Comma-Separated Lists	(a) Potential zone SELECT TABLE_NAME , COLUMN_NAME FROM INFORMATION_SCHEMA.COLUMNS WHERE DATA_TYPE NOT IN ('datetime','date','numeric','int') ORDER BY TABLE_NAME , ORDINAL_POSITION (b) Dynamic SQL SET @p_stmt = 'INSERT INTO T_CHECK_SEPERATED_LIST ' + ' SELECT '" + @table_name + '","' + @column_name + '" , ' + @column_name + ' ' + ' FROM ' + @table_name + ' WITH (NOLOCK) ' + ' WHERE '+ @column_name + ' LIKE "%' + @p_sep1 +'%"' + ' OR '+ @column_name + ' LIKE "%' + @p_sep2 +'%"' + ' OR '+ @column_name + ' LIKE "%' + @p_sep3 +'%"'

เอสคิวแอล แอนติ แพตเทิร์น	ทรานแซกเอสคิวแอล
Always Depend on One's Parent	(a) Potential zone SELECT T.TABLE_NAME , COLUMN_NAME , DATA_TYPE , CHARACTER_MAXIMUM_LENGTH , CHARACTER_OCTET_LENGTH , NUMERIC_PRECISION , NUMERIC_SCALE FROM INFORMATION_SCHEMA.TABLES T inner join INFORMATION_SCHEMA.COLUMNS C ON (T.TABLE_NAME = C.TABLE_NAME) where T.TABLE_TYPE = 'BASE TABLE' AND T.TABLE_NAME NOT IN ('T_CHECK_DEPEND_PARENT','T_CHECK_DEPEND_PARENT1') AND (C.COLUMN_NAME like '%parent%' or C.COLUMN_NAME like '%upper%') P left outer join INFORMATION_SCHEMA.COLUMNS c on (p.TABLE_NAME = c.TABLE_NAME AND P.DATA_TYPE = C.DATA_TYPE and isnull(p.CHARACTER_MAXIMUM_LENGTH ,0) = isnull(c.CHARACTER_MAXIMUM_LENGTH,0) and isnull(p.CHARACTER_OCTET_LENGTH,0) = isnull(c.CHARACTER_OCTET_LENGTH ,0) and isnull(p.NUMERIC_PRECISION ,0) = isnull(c.NUMERIC_PRECISION,0) and isnull(p.NUMERIC_SCALE ,0) = isnull(c.NUMERIC_SCALE ,0)) where P.COLUMN_NAME <> c.COLUMN_NAME ORDER BY p.TABLE_NAME , p.COLUMN_NAME

เอสคิวแอล แอนติ แพตเทิร์น	ทรานแซกเอสคิวแอล
	(b) Dynamic SQL SET @p_stmt = ' INSERT INTO T_CHECK_DEPEND_PARENT (TABLE_NAME ,PARENT_COLUMN_NAME,CHILD_COLUMN_NAME,PARENT_DATA,CNT_RELATE_DATA) ' + ' SELECT '" + @table_name + "'," + @parent_column_name + "'," + @child_column_name + "" + ', LV1.' + @parent_column_name + ' , COUNT(*) AS TOTAL_REC ' + ' FROM ' + @table_name + ' LV1 INNER JOIN ' + @table_name + ' LV2 ' + ' ON (LV1.' + @parent_column_name + ' = LV2.' + @child_column_name + ') ' + ' GROUP BY LV1.' + @parent_column_name + ', LV2.' + @child_column_name + ' ORDER BY COUNT(*) DESC '
One Size Fits All	(a) Potential zone SELECT DISTINCT TCS.TABLE_NAME , c.COLUMN_NAME FROM (SELECT OBJECT_NAME(parent_object_id) AS TABLE_NAME, type_desc AS CONSTRAINT_TYPE FROM sys.objects WHERE type_desc in ('PRIMARY_KEY_CONSTRAINT' , 'UNIQUE_CONSTRAINT')) TCS LEFT OUTER JOIN INFORMATION_SCHEMA.CONSTRAINT_COLUMN_USAGE cs on (tcs.TABLE_NAME = cs.TABLE_NAME)

เอสคิวแอล แอนติ แพตเทิร์น	ทรานแซกเอสคิวแอล
	LEFT OUTER JOIN INFORMATION_SCHEMA.COLUMNS c ON (TCS.TABLE_NAME = c.TABLE_NAME) WHERE cs.COLUMN_NAME <> c.COLUMN_NAME AND c.COLUMN_NAME NOT IN (SELECT COLUMN_NAME FROM INFORMATION_SCHEMA.CONSTRAINT_COLUMN_USAGE c2 WHERE C2.TABLE_NAME = tcs.TABLE_NAME) AND DATA_TYPE <> 'datetime' (b) Dynamic SQL SET @p_stmt = 'INSERT INTO T_CHECK_ONE_SIZE_FIT_ALL ' + ' SELECT '" + @table_name + "'," + @column_name + "" + ' ,COUNT(*) AS CNT_TABLE_RECORD , COUNT(DISTINCT ' + @column_name + ') AS CNT_COLUMN_RECORD ' + ' FROM ' + @table_name + ' WITH (NOLOCK) '
Leave Out the Constraints	SELECT distinct T.TABLE_NAME ,isnull(cu.COLUMN_NAME,df.name) as COLUMN_NAME , T.ConstraintType , CASE WHEN C.ConstraintName IS NOT NULL THEN 1 ELSE 0 END AS IS_EXISTS FROM (SELECT T1.TABLE_NAME , cons.ConstraintType FROM INFORMATION_SCHEMA.TABLES T1 , (select 'DEFAULT_CONSTRAINT' as ConstraintType union all select 'FOREIGN_KEY_CONSTRAINT' as ConstraintType union all

เอสคิวแอล แอนติ แพตเทิร์น	ทรานแซกเอสคิวแอล

```
        select 'PRIMARY_KEY_CONSTRAINT' as ConstraintType
union all
        select 'UNIQUE_CONSTRAINT' as ConstraintType union all
        select 'CHECK_CONSTRAINT' as ConstraintType )  as cons


        ) T
left outer join
        (
        SELECT OBJECT_NAME(object_id) AS ConstraintName,
        SCHEMA_NAME(schema_id) AS SchemaName,
        OBJECT_NAME(parent_object_id) AS TableName,
        parent_object_id as table_object_id ,
        type_desc AS ConstraintType
        FROM sys.objects
        WHERE type_desc LIKE '%CONSTRAINT' ) c
        on ( t.TABLE_NAME = c.TableName
        AND T.ConstraintType = C.ConstraintType )
left outer join
INFORMATION_SCHEMA.CONSTRAINT_COLUMN_USAGE cu
on (  t.TABLE_NAME = cu.TABLE_NAME
and  c.ConstraintName = cu.CONSTRAINT_NAME )
left outer join sys.all_columns df
on ( c.table_object_id = df.object_id
and df.default_object_id <> 0   )
 ORDER BY T.TABLE_NAME
```

เอสคิวแอล แอนติ แพตเทิร์น	ทรานแซกเอสคิวแอล
Use a Generic Attribute Table	(a) Potential zone SELECT TABLE_NAME , COLUMN_NAME FROM INFORMATION_SCHEMA.COLUMNS WHERE DATA_TYPE NOT IN ('datetime','date','numeric') ORDER BY TABLE_NAME , ORDINAL_POSITION (b) Dynamic SQL SET @p_stmt = 'INSERT INTO T_CHECK_ONE_CONSTRAINT_2 ' + ' SELECT '' + @table_name + '','' + @column_name + '' + ' ,COUNT(*) AS CNT_TABLE_RECORD , COUNT(DISTINCT ' + @column_name + ') AS CNT_COLUMN_RECORD ' + ' FROM ' + @table_name + ' WITH (NOLOCK) '
Use Dual-Purpose Foreign Key	(a) Potential zone SELECT DISTINCT table_name , COLUMN_NAME FROM INFORMATION_SCHEMA.COLUMNS WHERE DATA_TYPE NOT IN ('datetime','date','numeric','int','tinyint') (b) Dynamic SQL SET @p_stmt = 'INSERT INTO T_CHECK_DUALPURPOSE_FK ' + ' SELECT '' + @t1_table_name + '','' + @t1_column_name + '' , ' + @t1_column_name + ' '

เอสคิวแอล แอนติ แพตเทิร์น	ทรานแซกเอสคิวแอล
	+ ' FROM dbo.[' + @t1_table_name + '] WITH (NOLOCK) ' + ' WHERE '+ @t1_column_name + ' = (select '+ @t1_column_name +' from '+ @t1_table_name +' WHERE ' + @t1_column_name + ' LIKE "%' + @t0_table_name +'%")'
Create Multiple Columns	SELECT a.TABLE_NAME , a.COLUMN_NAME ,a.DATA_TYPE , a.IS_NULLABLE , a.CHARACTER_MAXIMUM_LENGTH , a.CHARACTER_OCTET_LENGTH, a.NUMERIC_PRECISION, a.NUMERIC_SCALE from INFORMATION_SCHEMA.COLUMNS a INNER JOIN (SELECT TABLE_NAME , LEFT(COLUMN_NAME,3) AS PREFIX_COL , DATA_TYPE , IS_NULLABLE , CHARACTER_MAXIMUM_LENGTH , CHARACTER_OCTET_LENGTH, NUMERIC_PRECISION, NUMERIC_SCALE FROM INFORMATION_SCHEMA.COLUMNS GROUP BY TABLE_NAME , LEFT(COLUMN_NAME,3) , DATA_TYPE , IS_NULLABLE , CHARACTER_MAXIMUM_LENGTH , CHARACTER_OCTET_LENGTH, NUMERIC_PRECISION, NUMERIC_SCALE HAVING COUNT(*) > 1) c ON (a.table_name = c.table_name and LEFT(a.COLUMN_NAME,3) = LEFT(c.PREFIX_COL,3)) WHERE ASCII(RIGHT(COLUMN_NAME,1)) BETWEEN 48 AND 57

เอสคิวแอล แอนติ แพตเทิร์น	ทรานแซกเอสคิวแอล
Clone Table or Column	(a) Potential zone INSERT INTO @TMP_TABLE SELECT ROW_NUMBER() OVER(ORDER BY COUNT(C.COLUMN_NAME)) AS _RK , T.TABLE_NAME , COUNT(C.COLUMN_NAME) AS CNT_COL FROM INFORMATION_SCHEMA.TABLES T INNER JOIN INFORMATION_SCHEMA.COLUMNS C ON (T.TABLE_NAME = C.TABLE_NAME) WHERE T.TABLE_TYPE = 'BASE TABLE' GROUP BY T.TABLE_NAME (b) Dynamic SQL SET @p_stmt = ' INSERT INTO T_CHECK_CLONE_TABLE SELECT T1.TABLE_NAME , T1.COLUMN_NAME , ' + CAST (@cnt_col AS VARCHAR) + ' , T2.TABLE_NAME , T2.COLUMN_NAME FROM INFORMATION_SCHEMA.COLUMNS T1 LEFT OUTER JOIN INFORMATION_SCHEMA.COLUMNS T2 ON (T1.ORDINAL_POSITION = T2.ORDINAL_POSITION AND T1.DATA_TYPE = T2.DATA_TYPE AND ISNULL(T1.CHARACTER_MAXIMUM_LENGTH ,0) = ISNULL(T2.CHARACTER_MAXIMUM_LENGTH, 0) AND ISNULL(T1.CHARACTER_OCTET_LENGTH ,0) = ISNULL(T2.CHARACTER_OCTET_LENGTH, 0) AND ISNULL(T1.NUMERIC_PRECISION ,0) = ISNULL(T2.NUMERIC_PRECISION, 0) AND ISNULL(T1.NUMERIC_SCALE,0) = ISNULL(T2.NUMERIC_SCALE , 0)

เอสคิวแอล แอนติ แพตเทิร์น	ทรานแซกเอสคิวแอล
	AND T2.TABLE_NAME = ''' + @next_table_name + ''') '+ ' WHERE T1.TABLE_NAME = ''' + @table_name + '''and T2.TABLE_NAME is not null '

ภาคผนวก ข

การทดสอบด้วยฐานข้อมูลจำลอง

ตารางที่ 15 รายการทดสอบและโครงสร้างตารางของฐานข้อมูลจำลองสำหรับการทดลอง Format Comma-Separated Lists

เลขที่ทดสอบ	เอสคิวแอล	ข้อมูล
1	CREATE TABLE Items (Items_id int CONSTRAINT ufirstkey PRIMARY KEY, item_name VARCHAR(1000), account_id VARCHAR(100))	INSERT INTO public.items(items_id, item_name, account_id) VALUES (1 ,'ballon' ,'12,23'), (2 ,'banana' ,'12,89'), (3 ,'ballon' ,'13,33'), (4 ,'banana' ,'19,36')
2	CREATE TABLE line_Items (L_Items_id int CONSTRAINT usecondkey PRIMARY KEY, L_item_name VARCHAR(100), L_account_id TEXT)	INSERT INTO public.line_items(l_items_id, l_item_name, l_account_id) VALUES (1 ,'ballon' ,'12|23'), (2 ,'banana' ,'12|89'), (3 ,'ballon' ,'13|33'), (4 ,'banana' ,'19|36')
3		INSERT INTO public.empoyee(emp_id, account_name, address1, address2, address3) VALUES ('1', 'somying', 'ซอยจุฬาลงกรณ์,เขตปทุมวัน,แขวงปทุมวัน', '', '');

ตารางที่ 16 ผลการทดสอบ Format Comma-Separated Lists จากการใช้ทรานแซกเอสคิวแอลของผู้วิจัยเทียบกับโพสต์เกรส Eessaar

ผู้วิจัย			
TABLE_NAME	COLUMN_NAME	SEPERATED_LIST	TYPE (FN/FP/TP)
ITEMS	account_id	12,23	TP
ITEMS	account_id	12,89	
ITEMS	account_id	13,33	
ITEMS	account_id	19,36	
line_Items	L_account_id	12|23	TP
line_Items	L_account_id	12|89	
line_Items	L_account_id	13|33	
line_Items	L_account_id	19|36	
EMPOYEE	address1	ซอยจุฬาลงกรณ์, เขตปทุมวัน,แขวงปทุมวัน	FP

Essaar		
table_schema	table_name	TYPE
ITEMS	account_id	TP
line_Items	L_account_id	FN
employee	address	FP

ตารางที่ 17 รายการทดสอบและโครงสร้างตารางของฐานข้อมูลจำลองสำหรับการทดลอง Always Depend on One's Parent

เลขที่ทดสอบ	เอสคิวแอล	ข้อมูล
1	CREATE TABLE PRODUCT_CATEGORY (CATEGORY_ID varchar(20) NOT NULL, CATEGORY_NAME varchar(100) NULL, CATEGORY_DESC varchar(200) NULL, PARENT_ID varchar(20) NULL)	INSERT INTO PRODUCT_CATEGORY (CATEGORY_ID ,CATEGORY_NAME ,CATEGORY_DESC ,PARENT_ID) VALUES ('01','อาหารสดย่อย','',''), ('02','ลูกชิ้น','','01'), ('03','หมู','','01'), ('04','เนื้อ','','01')
2	CREATE TABLE PRODUCTS (PRODUCT_NO INTEGER PRIMARY KEY, NAME TEXT, PRICE NUMERIC , PARENT_ID INTEGER REFERENCES PRODUCTS (PRODUCT_NO),);	INSERT INTO PRODUCTS (PRODUCT_NO ,NAME ,PRICE ,PARENT_ID) VALUES (1,'อาหารสดย่อย','300.00',NULL), (2,'ลูกชิ้น','100.00',1), (3,'หมู','100.00',1), (4,'เนื้อ','100.00',1)

ตารางที่ 18 ผลการทดสอบ Always Depend on One's Parent จากการใช้ทรานแซกเอสคิวแอลของผู้วิจัยเทียบกับโพสต์เกรสของ Eessaar

ผู้วิจัย					
TABLE_NAME	PARENT_COLUMN_NAME	CHILD_COLUMN_NAME	PARENT_DATA	CNT_RELATE_DATA	TYPE (FN/FP/TP)
PRODUCT_CATEGORY	PARENT_ID	CATEGORY_ID	1	3	TP
PRODUCTS	PARENT_ID	PRODUCT_NO	1	3	TP

ESSAR			
TABLE_SCHEMA	TABLE_NAME	CONSTRAINT_NAME	TYPE (FN/FP/TP)
PUBLIC	PRODUCTS	PRODUCTS_PARENT_ID_FKEY	TP
		PRODUCT_CATEGORY	FN

ตารางที่ 19 รายการทดสอบและโครงสร้างตารางของฐานข้อมูลจำลองสำหรับการทดลอง One Size Fits All

เลขที่ทดสอบ	เอสคิวแอล	ข้อมูล
1	CREATE TABLE AGENT (ID int CONSTRAINT zfirstkey PRIMARY KEY, LOGIN varchar(100), account_name varchar(40),	INSERT INTO AGENT (ID ,LOGIN ,account_name ,address1) VALUES (1,'SOMCHI','','),

เลขที่ทดสอบ	เอสคิวแอล	ข้อมูล
	address1 varchar(40))	(2,'SOMYIN','',''), (3,'SOMSRI','','')
2	CREATE TABLE CALL_AGENT (AGENT_ID int CONSTRAINT zsecondkey PRIMARY KEY, AGENT_NO varchar(6), AGENT_NAME varchar(100), MANAGER_NAME varchar(40))	INSERT INTO CALL_AGENT (AGENT_ID ,AGENT_NO ,AGENT_NAME ,MANAGER_NAME) VALUES (1,'500145','SOMCHI','MD001'), (2,'500156','SOMYIN','MD001'), (3,'500178','SOMSRI','MD001')
3	CREATE TABLE LINE_AGENT (ID int CONSTRAINT zthirdkey PRIMARY KEY, LINE varchar(6))	

ตารางที่ 20 ผลการทดสอบ One Size Fits All จากการใช้ทรานแซกเอสคิวแอลของผู้วิจัยเทียบกับโพสต์เกรสของ Eessaar

ผู้วิจัย				
TABLE_NAME	COLUMN_NAME	CNT_TABLE_RECORD	CNT_COLUMN_RECORD	TYPE (FN/FP/TP)
AGENT	LOGIN	3	3	TP
CALL_AGENT	AGENT_NAME	3	3	FP

ผู้วิจัย				
TABLE_NAME	COLUMN_NAME	CNT_TABLE_RECORD	CNT_COLUMN_RECORD	TYPE (FN/FP/TP)
CALL_AGENT	AGENT_NO	3	3	TP
ITEMS	account_id	4	4	TP

Essar	
TABLE_NAME	TYPE (FN/FP/TP)
AGENT	TP
AGENT_NO	FN
LINE_AGENT	FP
ITEMS	FN

ตารางที่ 21 รายการทดสอบและโครงสร้างตารางของฐานข้อมูลจำลองสำหรับการทดลอง Leave Out the Constraints

กรณีทดสอบ	จุดประสงค์
1	ตรวจสอบทุกตารางในฐานข้อมูล

ตารางที่ 22 ผลการทดสอบ Leave Out the Constraints จากการใช้ทรานแซกเอสคิวแอลของผู้วิจัยเทียบกับโพสต์เกรสของ Eessaar

ผู้วิจัย: วิธีวิจัยที่ 1			
TABLE_NAME	COLUMN_NAME	ConstraintType	IS_EXISTS
ACCOUNT_2017	NULL	CHECK_CONSTRAINT	0
ACCOUNT_2017	NULL	DEFAULT_CONSTRAINT	0
ACCOUNT_2017	NULL	FOREIGN_KEY_CONSTRAINT	0
ACCOUNT_2017	NULL	UNIQUE_CONSTRAINT	0
ACCOUNT_2017	account_id	PRIMARY_KEY_CONSTRAINT	1
AGENT	NULL	CHECK_CONSTRAINT	0
AGENT	NULL	DEFAULT_CONSTRAINT	0

ผู้วิจัย: วิธีวิจัยที่ 1			
TABLE_NAME	COLUMN_NAME	ConstraintType	IS_EXISTS
AGENT	NULL	FOREIGN_KEY_CONSTRAINT	0
AGENT	NULL	UNIQUE_CONSTRAINT	0
AGENT	ID	PRIMARY_KEY_CONSTRAINT	1
Attribute_data	NULL	CHECK_CONSTRAINT	0
Attribute_data	NULL	DEFAULT_CONSTRAINT	0
Attribute_data	NULL	FOREIGN_KEY_CONSTRAINT	0
Attribute_data	NULL	UNIQUE_CONSTRAINT	0
Attribute_data	Attr_id	PRIMARY_KEY_CONSTRAINT	1
CALL_AGENT	NULL	CHECK_CONSTRAINT	0
CALL_AGENT	NULL	DEFAULT_CONSTRAINT	0
CALL_AGENT	NULL	FOREIGN_KEY_CONSTRAINT	0
CALL_AGENT	NULL	UNIQUE_CONSTRAINT	0
CALL_AGENT	AGENT_ID	PRIMARY_KEY_CONSTRAINT	1
CONTACT_2018	NULL	CHECK_CONSTRAINT	0
CONTACT_2018	NULL	DEFAULT_CONSTRAINT	0
CONTACT_2018	NULL	FOREIGN_KEY_CONSTRAINT	0
CONTACT_2018	NULL	UNIQUE_CONSTRAINT	0
CONTACT_2018	contact_id	PRIMARY_KEY_CONSTRAINT	1
EMPOYEE	NULL	CHECK_CONSTRAINT	0
EMPOYEE	NULL	DEFAULT_CONSTRAINT	0
EMPOYEE	NULL	FOREIGN_KEY_CONSTRAINT	0
EMPOYEE	NULL	UNIQUE_CONSTRAINT	0
EMPOYEE	emp_id	PRIMARY_KEY_CONSTRAINT	1
EMPOYEE_2010	NULL	CHECK_CONSTRAINT	0
EMPOYEE_2010	NULL	DEFAULT_CONSTRAINT	0
EMPOYEE_2010	NULL	FOREIGN_KEY_CONSTRAINT	0
EMPOYEE_2010	NULL	UNIQUE_CONSTRAINT	0

ผู้วิจัย: วิธีวิจัยที่ 1			
TABLE_NAME	COLUMN_NAME	ConstraintType	IS_EXISTS
EMPOYEE_2010	emp_id	PRIMARY_KEY_CONSTRAINT	1
EMPOYEE_SHIFT	NULL	CHECK_CONSTRAINT	0
EMPOYEE_SHIFT	NULL	DEFAULT_CONSTRAINT	0
EMPOYEE_SHIFT	NULL	FOREIGN_KEY_CONSTRAINT	0
EMPOYEE_SHIFT	NULL	PRIMARY_KEY_CONSTRAINT	0
EMPOYEE_SHIFT	NULL	UNIQUE_CONSTRAINT	0
ITEMS	NULL	CHECK_CONSTRAINT	0
ITEMS	NULL	DEFAULT_CONSTRAINT	0
ITEMS	NULL	FOREIGN_KEY_CONSTRAINT	0
ITEMS	NULL	UNIQUE_CONSTRAINT	0
ITEMS	Items_id	PRIMARY_KEY_CONSTRAINT	1
LINE_AGENT	NULL	CHECK_CONSTRAINT	0
LINE_AGENT	NULL	DEFAULT_CONSTRAINT	0
LINE_AGENT	NULL	FOREIGN_KEY_CONSTRAINT	0
LINE_AGENT	NULL	UNIQUE_CONSTRAINT	0
LINE_AGENT	ID	PRIMARY_KEY_CONSTRAINT	1
line_Items	NULL	CHECK_CONSTRAINT	0
line_Items	NULL	DEFAULT_CONSTRAINT	0
line_Items	NULL	FOREIGN_KEY_CONSTRAINT	0
line_Items	NULL	UNIQUE_CONSTRAINT	0
line_Items	L_Items_id	PRIMARY_KEY_CONSTRAINT	1
LINK_CODE	NULL	CHECK_CONSTRAINT	0
LINK_CODE	NULL	DEFAULT_CONSTRAINT	0
LINK_CODE	NULL	FOREIGN_KEY_CONSTRAINT	0
LINK_CODE	NULL	UNIQUE_CONSTRAINT	0
LINK_CODE	LINK_ID	PRIMARY_KEY_CONSTRAINT	1
LOOKUP_CODE	NULL	CHECK_CONSTRAINT	0

ผู้วิจัย: วิธีวิจัยที่ 1			
TABLE_NAME	COLUMN_NAME	ConstraintType	IS_EXISTS
LOOKUP_CODE	NULL	DEFAULT_CONSTRAINT	0
LOOKUP_CODE	NULL	FOREIGN_KEY_CONSTRAINT	0
LOOKUP_CODE	NULL	UNIQUE_CONSTRAINT	0
LOOKUP_CODE	CODE_ID	PRIMARY_KEY_CONSTRAINT	1
ORDER_2011	NULL	CHECK_CONSTRAINT	0
ORDER_2011	NULL	DEFAULT_CONSTRAINT	0
ORDER_2011	NULL	FOREIGN_KEY_CONSTRAINT	0
ORDER_2011	NULL	UNIQUE_CONSTRAINT	0
ORDER_2011	order_id	PRIMARY_KEY_CONSTRAINT	1
ORDER_2012	NULL	CHECK_CONSTRAINT	0
ORDER_2012	NULL	DEFAULT_CONSTRAINT	0
ORDER_2012	NULL	FOREIGN_KEY_CONSTRAINT	0
ORDER_2012	NULL	UNIQUE_CONSTRAINT	0
ORDER_2012	order_id	PRIMARY_KEY_CONSTRAINT	1
ORDER_2013	NULL	CHECK_CONSTRAINT	0
ORDER_2013	NULL	DEFAULT_CONSTRAINT	0
ORDER_2013	NULL	FOREIGN_KEY_CONSTRAINT	0
ORDER_2013	NULL	UNIQUE_CONSTRAINT	0
ORDER_2013	order_id	PRIMARY_KEY_CONSTRAINT	1
ORDER_A	NULL	CHECK_CONSTRAINT	0
ORDER_A	NULL	DEFAULT_CONSTRAINT	0
ORDER_A	NULL	FOREIGN_KEY_CONSTRAINT	0
ORDER_A	NULL	UNIQUE_CONSTRAINT	0
ORDER_A	order_id	PRIMARY_KEY_CONSTRAINT	1
ORDER_B	NULL	CHECK_CONSTRAINT	0
ORDER_B	NULL	DEFAULT_CONSTRAINT	0
ORDER_B	NULL	FOREIGN_KEY_CONSTRAINT	0

ผู้วิจัย: วิธีวิจัยที่ 1			
TABLE_NAME	COLUMN_NAME	ConstraintType	IS_EXISTS
ORDER_B	NULL	UNIQUE_CONSTRAINT	0
ORDER_B	order_id	PRIMARY_KEY_CONSTRAINT	1
ORDER_Y2015	NULL	CHECK_CONSTRAINT	0
ORDER_Y2015	NULL	DEFAULT_CONSTRAINT	0
ORDER_Y2015	NULL	FOREIGN_KEY_CONSTRAINT	0
ORDER_Y2015	NULL	UNIQUE_CONSTRAINT	0
ORDER_Y2015	order_id	PRIMARY_KEY_CONSTRAINT	1
ORDER_Y2016	NULL	CHECK_CONSTRAINT	0
ORDER_Y2016	NULL	DEFAULT_CONSTRAINT	0
ORDER_Y2016	NULL	FOREIGN_KEY_CONSTRAINT	0
ORDER_Y2016	NULL	UNIQUE_CONSTRAINT	0
ORDER_Y2016	order_id	PRIMARY_KEY_CONSTRAINT	1
ORDER_Y2017	NULL	CHECK_CONSTRAINT	0
ORDER_Y2017	NULL	DEFAULT_CONSTRAINT	0
ORDER_Y2017	NULL	FOREIGN_KEY_CONSTRAINT	0
ORDER_Y2017	NULL	UNIQUE_CONSTRAINT	0
ORDER_Y2017	order_id	PRIMARY_KEY_CONSTRAINT	1
ORDERX_2014	NULL	CHECK_CONSTRAINT	0
ORDERX_2014	NULL	DEFAULT_CONSTRAINT	0
ORDERX_2014	NULL	FOREIGN_KEY_CONSTRAINT	0
ORDERX_2014	NULL	UNIQUE_CONSTRAINT	0
ORDERX_2014	order_id	PRIMARY_KEY_CONSTRAINT	1
PRODUCT_CATEGORY	NULL	CHECK_CONSTRAINT	0
PRODUCT_CATEGORY	NULL	DEFAULT_CONSTRAINT	0
PRODUCT_CATEGORY	NULL	FOREIGN_KEY_CONSTRAINT	0
PRODUCT_CATEGORY	NULL	PRIMARY_KEY_CONSTRAINT	0
PRODUCT_CATEGORY	NULL	UNIQUE_CONSTRAINT	0

ผู้วิจัย: วิธีวิจัยที่ 1			
TABLE_NAME	COLUMN_NAME	ConstraintType	IS_EXISTS
PRODUCTS	NULL	CHECK_CONSTRAINT	0
PRODUCTS	NULL	DEFAULT_CONSTRAINT	0
PRODUCTS	NULL	UNIQUE_CONSTRAINT	0
PRODUCTS	PARENT_ID	FOREIGN_KEY_CONSTRAINT	1
PRODUCTS	PRODUCT_NO	PRIMARY_KEY_CONSTRAINT	1

ผู้วิจัย: วิธีวิจัยที่ 2				
TABLE_NAME	COLUMN_NAME	CNT_TABLE_RECORD	CNT_COLUMN_RECORD	TYPE (FN/FP/TP)
AGENT	LOGIN	3	3	TP
CALL_AGENT	AGENT_NAME	3	3	TP
CALL_AGENT	AGENT_NO	3	3	TP
Comments	issue_type	3	3	TP
EMPOYEE	account_name	1	1	TP
EMPOYEE	address1	1	1	FP
EMPOYEE	address2	1	1	FP
EMPOYEE	address3	1	1	FP
ITEMS	account_id	4	4	TP

Essar: วิธีวิจัยที่ 1						
primary_table_schema	primary_table_name	primary_column_name	dependent_column_schema	dependent_table_name	dependent_column_name	TYPE (FN/FP/TP)
public	agent	id	public	line_agent	id	TP

Essar: วิธีวิจัยที่ 1						
primary_table_schema	primary_table_name	primary_column_name	dependent_column_schema	dependent_table_name	dependent_column_name	TYPE (FN/FP/TP)
public	empoyee	emp_id	public	empoyee_2010	emp_id	FP
public	empoyee	emp_id	public	empoyee_shift	emp_id	(ซ้ำ)
public	empoyee_2010	emp_id	public	empoyee_shift	emp_id	FP
public	empoyee_2010	emp_id	public	empoyee	emp_id	(ซ้ำ)
public	line_agent	id	public	agent	id	TP
public	order_2011	order_id	public	order_b	order_id	FP
public	order_2011	order_id	public	order_y2016	order_id	FP
public	order_2011	order_id	public	order_y2015	order_id	FP
public	order_2011	order_id	public	order_2012	order_id	FP
public	order_2011	order_id	public	order_2013	order_id	FP

Essar: วิธีวิจัยที่ 1						
primary_table_schema	primary_table_name	primary_column_name	dependent_column_schema	dependent_table_name	dependent_column_name	TYPE (FN/FP/TP)
public	order_2011	order_id	public	order_y2017	order_id	FP
public	order_2011	order_id	public	orderx_2014	order_id	FP
public	order_2011	order_id	public	order_a	order_id	FP
public	order_2012	order_id	public	order_y2016	order_id	FP
public	order_2012	order_id	public	order_2011	order_id	(ซ้ำ)
public	order_2012	order_id	public	order_y2017	order_id	FP
public	order_2012	order_id	public	order_2013	order_id	FP
public	order_2012	order_id	public	orderx_2014	order_id	FP
public	order_2012	order_id	public	order_a	order_id	FP
public	order_2012	order_id	public	order_b	order_id	FP
public	order_2012	order_id	public	order_y2015	order_id	FP
public	order_2013	order_id	public	order_y2017	order_id	FP

Essar: วิธีวิจัยที่ 1						
primary_table_schema	primary_table_name	primary_column_name	dependent_column_schema	dependent_table_name	dependent_column_name	TYPE (FN/FP/TP)
public	order_2013	order_id	public	order_2012	order_id	(ซ้ำ)
public	order_2013	order_id	public	orderx_2014	order_id	FP
public	order_2013	order_id	public	order_a	order_id	FP
public	order_2013	order_id	public	order_y2016	order_id	FP
public	order_2013	order_id	public	order_b	order_id	FP
public	order_2013	order_id	public	order_2011	order_id	(ซ้ำ)
public	order_2013	order_id	public	order_y2015	order_id	FP
public	order_a	order_id	public	order_2011	order_id	(ซ้ำ)
public	order_a	order_id	public	order_2012	order_id	(ซ้ำ)
public	order_a	order_id	public	order_y2015	order_id	FP
public	order_a	order_id	public	order_2013	order_id	(ซ้ำ)
public	order_a	order_id	public	order_b	order_id	FP

Essar: วิธีวิจัยที่ 1						
primary_table_schema	primary_table_name	primary_column_name	dependent_column_schema	dependent_table_name	dependent_column_name	TYPE (FN/FP/TP)
public	order_a	order_id	public	order_y2016	order_id	FP
public	order_a	order_id	public	order_y2017	order_id	FP
public	order_a	order_id	public	orderx_2014	order_id	FP
public	order_b	order_id	public	orderx_2014	order_id	FP
public	order_b	order_id	public	order_2013	order_id	(ซ้ำ)
public	order_b	order_id	public	order_2012	order_id	(ซ้ำ)
public	order_b	order_id	public	order_2011	order_id	(ซ้ำ)
public	order_b	order_id	public	order_y2017	order_id	FP
public	order_b	order_id	public	order_y2015	order_id	FP
public	order_b	order_id	public	order_y2016	order_id	FP
public	order_b	order_id	public	order_a	order_id	(ซ้ำ)
public	order_y2015	order_id	public	orderx_2014	order_id	FP

Essar: วิธีวิจัยที่ 1						
primary_table_schema	primary_table_name	primary_column_name	dependent_column_schema	dependent_table_name	dependent_column_name	TYPE (FN/FP/TP)
public	order_y2015	order_id	public	order_b	order_id	(ซ้ำ)
public	order_y2015	order_id	public	order_2012	order_id	(ซ้ำ)
public	order_y2015	order_id	public	order_y2017	order_id	(ซ้ำ)
public	order_y2015	order_id	public	order_y2016	order_id	FP
public	order_y2015	order_id	public	order_a	order_id	(ซ้ำ)
public	order_y2015	order_id	public	order_2013	order_id	(ซ้ำ)
public	order_y2015	order_id	public	order_2011	order_id	(ซ้ำ)
public	order_y2016	order_id	public	order_b	order_id	(ซ้ำ)
public	order_y2016	order_id	public	order_y2015	order_id	(ซ้ำ)
public	order_y2016	order_id	public	order_2011	order_id	(ซ้ำ)
public	order_y2016	order_id	public	order_2013	order_id	(ซ้ำ)
public	order_y2016	order_id	public	order_a	order_id	(ซ้ำ)

Essar: วิธีวิจัยที่ 1						
primary_table_schema	primary_table_name	primary_column_name	dependent_column_schema	dependent_table_name	dependent_column_name	TYPE (FN/FP/TP)
public	order_y2016	order_id	public	order_y2017	order_id	FP
public	order_y2016	order_id	public	orderx_2014	order_id	FP
public	order_y2016	order_id	public	order_2012	order_id	(ซ้ำ)
public	order_y2017	order_id	public	order_b	order_id	(ซ้ำ)
public	order_y2017	order_id	public	order_2013	order_id	(ซ้ำ)
public	order_y2017	order_id	public	order_y2015	order_id	(ซ้ำ)
public	order_y2017	order_id	public	order_a	order_id	(ซ้ำ)
public	order_y2017	order_id	public	orderx_2014	order_id	fp
public	order_y2017	order_id	public	order_2011	order_id	(ซ้ำ)
public	order_y2017	order_id	public	order_2012	order_id	(ซ้ำ)
public	order_y2017	order_id	public	order_y2016	order_id	(ซ้ำ)
public	orderx_2014	order_id	public	order_2012	order_id	(ซ้ำ)

Essar: วิธีวิจัยที่ 1						
primary_table_schema	primary_table_name	primary_column_name	dependent_column_schema	dependent_table_name	dependent_column_name	TYPE (FN/FP/TP)
public	orderx_2014	order_id	public	order_2011	order_id	(ซ้ำ)
public	orderx_2014	order_id	public	order_a	order_id	(ซ้ำ)
public	orderx_2014	order_id	public	order_y2017	order_id	(ซ้ำ)
public	orderx_2014	order_id	public	order_b	order_id	(ซ้ำ)
public	orderx_2014	order_id	public	order_y2016	order_id	(ซ้ำ)
public	orderx_2014	order_id	public	order_y2015	order_id	(ซ้ำ)
public	orderx_2014	order_id	public	order_2013	order_id	(ซ้ำ)

Essar: วิธีวิจัยที่ 2		
table_schema	table_name	TYPE (FN/FP/TP)
public	account_2017	TP
public	attribute_data	TP
public	call_agent	TP
public	contact_2018	TP
public	empoyee	TP
public	empoyee_2010	TP
public	empoyee_shift	TP

Essar: วิธีวิจัยที่ 2		
table_schema	table_name	TYPE (FN/FP/TP)
public	items	TP
public	line_agent	TP
public	line_items	TP
public	link_code	TP
public	lookup_code	TP
public	order_2011	TP
public	order_2012	TP
public	order_2013	TP
public	order_a	TP
public	order_b	TP
public	order_y2015	TP
public	order_y2016	TP
public	order_y2017	TP
public	orderx_2014	TP
public	product_category	TP
public	products_cate_price	TP
	agent	FN

ตารางที่ 23 รายการทดสอบและโครงสร้างตารางของฐานข้อมูลจำลองสำหรับการทดลอง Use a Generic Attribute Table

เลขที่ทดสอบ	จุดประสงค์	เอสคิวแอล	ข้อมูล
1		CREATE TABLE LOOKUP_CODE (CODE_ID int CONSTRAINT bfirstkey	INSERT INTO LOOKUP_CODE (CODE_ID,CODE_NAME,CODE_VALUES) VALUES (11,'ORDER_PROCESS','01-COOKING'), (12,'ORDER_PROCESS','02-INPROCESS'),

เลขที่ทดสอบ	จุดประสงค์	เอสคิวแอล	ข้อมูล
		PRIMARY KEY, CODE_NAME varchar(25) NULL, CODE_VALUES varchar(25) NULL)	(13,'ORDER_PROCESS','03-SEND'), (21,'ORDER_STAGE','01-NEW'), (22,'ORDER_STAGE','02-INPROCESS'), (23,'ORDER_STAGE','03-CANCEL')
2		CREATE TABLE Attribute_data (Attr_id int CONSTRAINT bsecondkey PRIMARY KEY, Attr_name varchar(250))	
3		CREATE TABLE LINK_CODE (LINK_ID int CONSTRAINT bthirdkey PRIMARY KEY, LINK_CODE_NAME varchar(25) NULL, LINK_CODE_ID varchar(25) NULL)	INSERT INTO LINK_CODE ([LINK_ID],[LINK_CODE_NAME],[LINK_CODE_ID]) VALUES (11,'ORDER_PROCESS','01-COOKING'), (12,'ORDER_PROCESS','02-INPROCESS'), (13,'ORDER_PROCESS','03-SEND'), (21,'ORDER_STAGE','01-NEW'), (22,'ORDER_STAGE','02-INPROCESS'), (23,'ORDER_STAGE','03-CANCEL')

ตารางที่ 24 ผลการทดสอบ Use a Generic Attribute Table จากการใช้ทรานแซกเอสคิวแอลของผู้วิจัยเทียบกับโพสต์เกรสของ Eessaar

ผู้วิจัย				
table_name	column_name	column_data	total_data_dup	TYPE (FN/FP/TP)
CALL_AGENT	MANAGER_NAME	MD001	3	FP
LOOKUP_CODE	CODE_NAME	ORDER_PROCESS	3	TP
LOOKUP_CODE	CODE_NAME	ORDER_STAGE	3	
LOOKUP_CODE	CODE_VALUES	02-INPROCESS	2	TP
LINK_CODE	LINK_CODE_NAME	ORDER_PROCESS	3	TP
LINK_CODE	LINK_CODE_NAME	ORDER_STAGE	3	

Essar		
table_schema	table_name	TYPE (FN/FP/TP)
public	link_code	TP
public	attribute_data	FP
	CODE_NAME	FN
	CODE_VALUES	FN

ตารางที่ 25 รายการทดสอบและโครงสร้างตารางของฐานข้อมูลจำลองสำหรับการทดลอง Use Dual-Purpose Foreign Key

เลขที่ทดสอบ	เอสคิวแอล	ข้อมูล
1	CREATE TABLE Comments (comment_id int PRIMARY KEY, issue_type VARCHAR(20), issue_id int NOT NULL, author int NOT NULL, comment_date date, comment TEXT);	INSERT INTO Comments (comment_id ,issue_type ,issue_id ,author ,comment_date ,comment) VALUES (1,'AGENT',1,1,null,null), (2,'EMPLOYEE',1,1,null,null),

ตารางที่ 26 ผลการทดสอบ Use Dual-Purpose Foreign Key จากการใช้ทรานแซกเอสคิวแอลของผู้วิจัยเทียบกับโพสต์เกรสของ Eessaar

ผู้วิจัย		
TABLE_NAME	COLUMN_NAME	DATA_DUALPURPOSE
Comments	issue_type	AGENT
Comments	issue_type	EMPOYEE

Essar		
ไม่แสดงผลลัพธ์		

ตารางที่ 27 รายการทดสอบและโครงสร้างตารางของฐานข้อมูลจำลองสำหรับการทดลอง Create Multiple Columns

เลขที่ทดสอบ	เอสคิวแอล
1	CREATE TABLE EMPOYEE_SHIFT (emp_id char(5) , shift1 varchar(40), shift2 varchar(40), shift3 varchar(40))
2	CREATE TABLE CONTACT_2018 (contact_id char(5) CONSTRAINT tenthkey PRIMARY KEY, contact_name varchar(40), address1 varchar(40), address2 varchar(40), address3 varchar(40), create_contact_date date);
	CREATE TABLE ACCOUNT_2017 (account_id char(5) CONSTRAINT eleventhkey PRIMARY KEY, account_name varchar(40), address1 varchar(40), address2 varchar(40), address3 varchar(40), create_contact_date date);
3	CREATE TABLE EMPOYEE (emp_id char(5) CONSTRAINT tweleth PRIMARY KEY, account_name varchar(40), address1 varchar(40), address2 varchar(20),

เลขที่ทดสอบ	เอสคิวแอล
	address3 varchar(10), create_contact_date date);
4	CREATE TABLE EMPOYEE_2010 (emp_id char(5) CONSTRAINT thirteenth PRIMARY KEY, account_name varchar(40), address1 varchar(40), address2 varchar(20), address3 varchar(10),);

ตารางที่ 28 ผลการทดสอบ Create Multiple Columns จากการใช้ทรานแซกเอสคิวแอลของผู้วิจัยเทียบกับโพสต์เกรสของ Eessaar

ผู้วิจัย		
TABLE_NAME	COLUMN_NAME	TYPE (FN/FP/TP)
EMPLOYEE	shift1	TP
EMPLOYEE	shift2	
EMPLOYEE	shift3	
ACCOUNT_2017	address1	FP
ACCOUNT_2017	address2	
ACCOUNT_2017	address3	
CONTACT_2018	address1	FP
CONTACT_2018	address2	
CONTACT_2018	address3	

Essar			
table_name	column1	column2	TYPE (FN/FP/TP)
empoyee_shift	shift3	shift1	TP
empoyee_shift	shift3	shift2	
empoyee_shift	shift2	shift1	
empoyee_shift	shift2	shift3	
empoyee_shift	shift1	shift2	
empoyee_shift	shift1	shift3	
account_2017	address1	address2	FP
account_2017	address1	address3	
account_2017	address2	address1	
account_2017	address2	address3	
account_2017	address3	address1	
account_2017	address3	address2	
contact_2018	address2	address1	FP
contact_2018	address2	address3	
contact_2018	address1	address2	
contact_2018	address1	address3	
contact_2018	address3	address1	
contact_2018	address3	address2	

ตารางที่ 29 รายการทดสอบและโครงสร้างตารางของฐานข้อมูลจำลองสำหรับการทดลอง Clone Tables or Columns

เลขที่ทดสอบ	เอสคิวแอล
1	CREATE TABLE ORDER_2011 (order_id char(5) CONSTRAINT firstkey PRIMARY KEY, product_code varchar(40), employee_id integer, order_date date, unit_price numeric (20,2), total_price numeric (20,2));
	CREATE TABLE ORDER_2012 (order_id char(5) CONSTRAINT secondkey PRIMARY KEY, product_code varchar(40), employee_id integer, order_date date, unit_price numeric (20,2), total_price numeric (20,2));
	CREATE TABLE ORDER_2013 (order_id char(5) CONSTRAINT thirdkey PRIMARY KEY, product_code varchar(40), employee_id integer, order_date date, unit_price numeric (20,2), total_price numeric (20,2));
2	CREATE TABLE ORDERX_2014 (order_id char(5) CONSTRAINT forthkey PRIMARY KEY,

เลขที่ ทดสอบ	เอสคิวแอล
	product_code varchar(40), employee_id integer, order_date date, total_price numeric (20,2));
3	CREATE TABLE ORDER_A (order_id char(5) CONSTRAINT fifthkey PRIMARY KEY, product_code varchar(40), employee_id integer, order_date date, total_price numeric (20,2));
	CREATE TABLE ORDER_B (order_id char(5) CONSTRAINT sixthkey PRIMARY KEY, product_code varchar(40), employee_id integer, order_date date, unit_price numeric (20,2), total_price numeric (20,2));
4	CREATE TABLE ORDER_Y2015 (order_id char(5) CONSTRAINT seventkey PRIMARY KEY, product_code varchar(40), employee_id integer, order_date date, unit_price numeric (20,2), total_price numeric (20,2));

เลขที่ ทดสอบ	เอสคิวแอล
	CREATE TABLE ORDER_Y2016 (order_id char(5) CONSTRAINT eightkey PRIMARY KEY, product_code varchar(40), employee_id integer, order_date date, unit_price numeric (20,2), total_price numeric (20,2));
5	CREATE TABLE ORDER_Y2017 (order_id char(5) CONSTRAINT ninthkey PRIMARY KEY, product_code varchar(40), employee_id integer, order_date date);
6	CREATE TABLE CONTACT_2018 (contact_id char(5) CONSTRAINT tenthkey PRIMARY KEY, contact_name varchar(40), address1 varchar(40), address2 varchar(40), address3 varchar(40), create_contact_date date);
	CREATE TABLE ACCOUNT_2017 (account_id char(5) CONSTRAINT eleventhkey PRIMARY KEY, account_name varchar(40), address1 varchar(40), address2 varchar(40), address3 varchar(40),

เลขที่ ทดสอบ	เอสคิวแอล
	create_contact_date date);

ตารางที่ 30 ผลการทดสอบ Clone Tables or Columns จากการใช้ทรานแซกเอสคิวแอลของผู้วิจัยเทียบกับโพสต์เกรสของ Eessaar

ผู้วิจัย			
TABLE_NAME	CHK_TABLE_NAME	TOTAL_COLUMN	TYPE (FN/FP/TP)
ACCOUNT_2017	CONTACT_2018	6	FP
ORDER_2011	ORDER_2012	6	TP
ORDER_2011	ORDER_2013	6	TP
ORDER_2011	ORDERX_2014	6	TP
ORDER_2012	ORDER_2013	6	TP
ORDER_2012	ORDERX_2014	6	TP
ORDER_2013	ORDERX_2014	6	TP
ORDER_A	ORDER_B	5	TP
ORDER_Y2015	ORDER_2011	6	TP
ORDER_Y2015	ORDER_2012	6	TP
ORDER_Y2015	ORDER_2013	6	TP
ORDER_Y2015	ORDER_Y2016	6	TP
ORDER_Y2015	ORDERX_2014	6	TP
ORDER_Y2016	ORDER_2011	6	TP
ORDER_Y2016	ORDER_2012	6	TP
ORDER_Y2016	ORDER_2013	6	TP
ORDER_Y2016	ORDERX_2014	6	TP

Essar				
table_schema	table_name	table_schema-2	table_name-2	TYPE (FN/FP/TP)
public	order_2011	public	order_2012	TP
public	order_2011	public	order_2013	TP
public	order_2012	public	order_2011	(ซ้ำ)
public	order_2012	public	order_2013	TP
public	order_2013	public	order_2011	(ซ้ำ)

Essar				
table_schema	table_name	table_schema-2	table_name-2	TYPE (FN/FP/TP)
public	order_2013	public	order_2012	(ซ้ำ)
public	order_y2015	public	order_y2016	TP
public	order_y2016	public	order_y2015	(ซ้ำ)
	ORDER_A		ORDER_B	FN
	ORDER_2011		ORDERX_2014	FN
	ORDER_2012		ORDERX_2014	FN
	ORDER_2013		ORDERX_2014	FN
	ORDER_Y2015		ORDERX_2014	FN
	ORDER_Y2016		ORDERX_2014	FN
	ORDER_Y2015		ORDER_2011	FN
	ORDER_Y2015		ORDER_2012	FN
	ORDER_Y2015		ORDER_2013	FN
	ORDER_Y2016		ORDER_2011	FN
	ORDER_Y2016		ORDER_2012	FN
	ORDER_Y2016		ORDER_2013	FN

ภาคผนวก ค

รายละเอียดการทดลองรีแฟคทอริงฐานข้อมูล

ตารางที่ 31 รีแฟคทอริงฐานข้อมูลด้วย Replace One to Many with Associative Table ของ เอสคิวแอลแอนติแพตเทิร์น Format Comma-Separated Lists

<table>
<tr><th>ก่อนดำเนินการรีแฟคทอริงฐานข้อมูล</th><th>หลังดำเนินการรีแฟคทอริงฐานข้อมูล</th></tr>
<tr><td>

เอสคิวแอลคอมมานด์

```
SELECT ITEMS_ID , ACCOUNT_ID,
LEFT(ACCOUNT_ID,2) AS ACCOUNT_ID1 ,
RIGHT(ACCOUNT_ID,2) AS ACCOUNT_ID2
FROM ITEMS
WHERE ACCOUNT_ID LIKE '%12,%'
```

ผลกระทำการ (Execute)

ITEMS_ID	ACCOUNT_ID	ACCOUNT_ID1	ACCOUNT_ID2
1	12,23	12	23
2	12,89	12	89

</td><td>

เอสคิวแอลคอมมานด์

```
SELECT ITEMS_ID ,
ACCOUNT_ID
FROM ITEMS_TAG
WHERE ACCOUNT_ID = '12'
```

ผลกระทำการ (Execute)

ITEMS_ID	ITEMS_ID	ACCOUNT_ID
1	12	23
2	12	89

</td></tr>
</table>

ตารางที่ 32 รีแฟคทอริงฐานข้อมูลด้วย Introduce New Column ของ เอสคิวแอลแอนติแพตเทิร์น Always Depend on One's Parent

<table>
<tr><th>ก่อนดำเนินการรีแฟคทอริงฐานข้อมูล</th><th>หลังดำเนินการรีแฟคทอริงฐานข้อมูล</th></tr>
<tr><td>

เอสคิวแอลคอมมานด์

```
SELECT CATEGORY_ID , PARENT_ID
FROM DBO.ORD_PRODUCT_CATEGORY
```

</td><td>

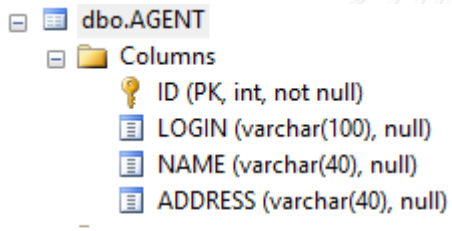
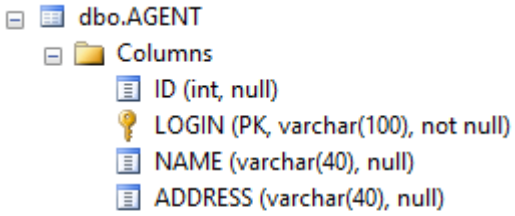
เอสคิวแอลคอมมานด์

```
SELECT  CATEGORY_ID , PATH_ROOT
FROM DBO.ORD_PRODUCT_CATEGORY
```

</td></tr>
</table>

<table>
<tr><th>ก่อนดำเนินการรีแฟคทอริงฐานข้อมูล</th><th>หลังดำเนินการรีแฟคทอริงฐานข้อมูล</th></tr>
<tr><td>ผลกระทำการ (Execute)
<table><tr><th>category_id</th><th>parent_id</th></tr><tr><td>201001071</td><td>200911249</td></tr><tr><td>201001072</td><td>200911249</td></tr><tr><td>201001073</td><td>200911249</td></tr></table></td><td>ผลกระทำการ (Execute)
<table><tr><th>category_id</th><th>path_root</th></tr><tr><td>201001071</td><td>201001071/200911249</td></tr><tr><td>201001072</td><td>201001072/200911249</td></tr><tr><td>201001073</td><td>201001072/200911249</td></tr></table></td></tr>
</table>

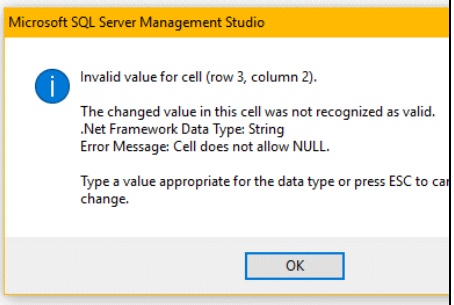
ตารางที่ 33 รีแฟคทอริงฐานข้อมูลด้วย Replace Surrogate Key with Natural Key และ Drop Column ของ เอสคิวแอลแอนติแพตเทิร์น One Size Fits All

<table>
<tr><th>ก่อนดำเนินการรีแฟคทอริงฐานข้อมูล</th><th>หลังดำเนินการรีแฟคทอริงฐานข้อมูล</th></tr>
<tr><td>โครงสร้างตาราง
 
</td><td>โครงสร้างตาราง
 
</td></tr>
</table>

ก่อนดำเนินการรีแฟคทอริงฐานข้อมูล	หลังดำเนินการรีแฟคทอริงฐานข้อมูล
การแก้ไขข้อมูล	**การแก้ไขข้อมูล**

	ID	LOGIN	NAME	ADDRESS
	1	SOMC	SOMCHI	100/423
	2	SOMY	SOMYIN	12
	3	SOMS	SOMSRI	86/63

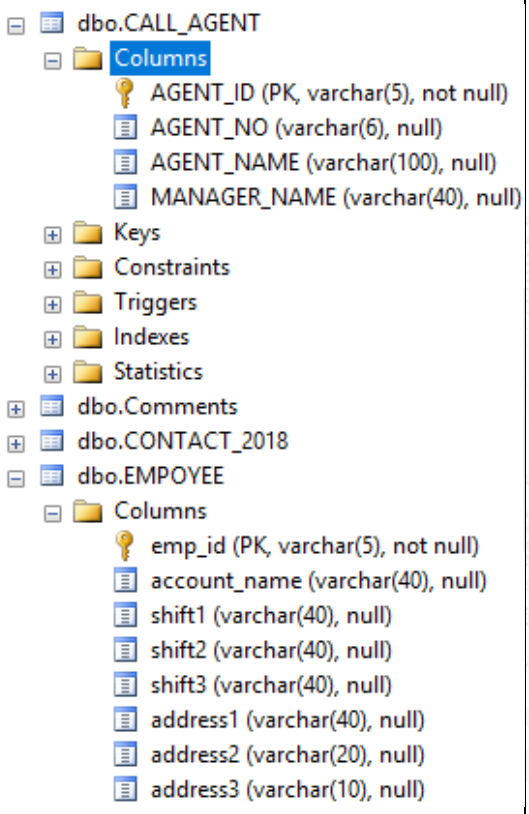
	ID	LOGIN	NAME	ADDRESS
	NULL	SOMCHI	NULL	NULL
	NULL	SOMYIN	NULL	NULL
	NULL	NULL	NULL	NULL
*	NULL	NULL	NULL	NULL



ตารางที่ 34 รีแฟคทอริงฐานข้อมูลด้วย Introduce New Column, Constraint Make column non-nullable , Add foreign key Constraint , Introduce Default Value Replace One to Many with Associative Table ของ เอสคิวแอลแอนติแพตเทิร์น Leave Out the Constraints

<table>
<tr><th>ก่อนดำเนินการรีแฟคทอริงฐานข้อมูล</th><th>หลังดำเนินการรีแฟคทอริงฐานข้อมูล</th></tr>
<tr><td>

โครงสร้างตาราง

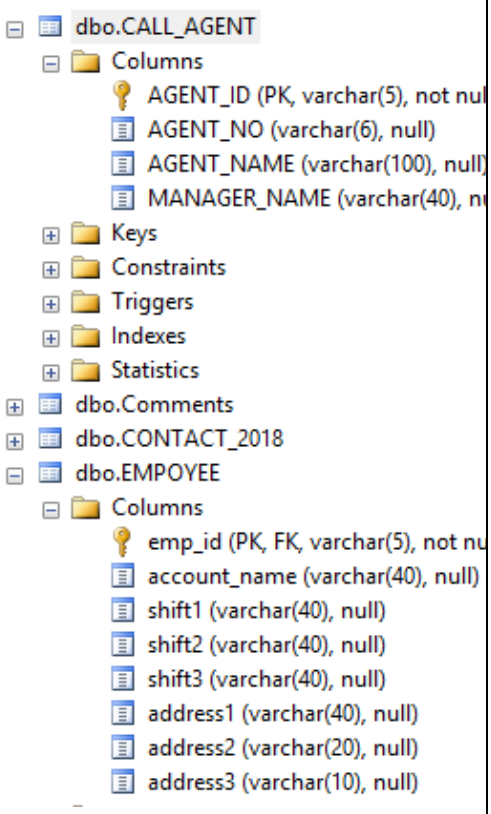


ตัวอย่างข้อมูล

AGENT_ID	AGENT_NO	AGENT_NAME	MANAGER_NAME
1	500145	SOMCHI	MD001
2	500156	SOMYIN	MD001
3	500178	SOMSRI	MD001

</td><td>

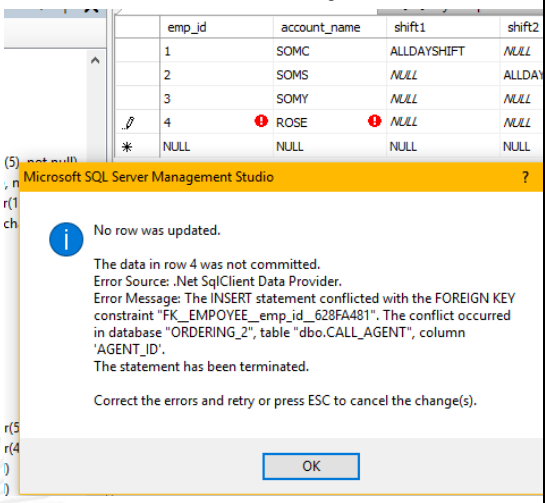
โครงสร้างตาราง



ตัวอย่างข้อมูล

AGENT_ID	AGENT_NO	AGENT_NAME	MANAGER_NAME
1	500145	SOMCHI	MD001
2	500156	SOMYIN	MD001
3	500178	SOMSRI	MD001

</td></tr>
</table>

ก่อนดำเนินการรีแฟคทอริงฐานข้อมูล	หลังดำเนินการรีแฟคทอริงฐานข้อมูล
การแก้ไขข้อมูล ไม่มีการแจ้งเตือนการเชื่อมโยง	**การแก้ไขข้อมูล** 

ตารางที่ 35 รีแฟคทอริงฐานข้อมูลด้วย Introduce New table ของ เอสคิวแอลแอนติแพตเทิร์น Use a Generic Attribute Table

ก่อนดำเนินการรีแฟคทอริงฐานข้อมูล	หลังดำเนินการรีแฟคทอริงฐานข้อมูล

เอสคิวแอลคอมมานด์
`SELECT COUNT(CODE_NAME) FROM LOOKUP_CODE WHERE CODE_NAME = 'ORDER_PROCESS'`
 	
`SELECT COUNT(CODE_NAME) FROM LOOKUP_CODE WHERE CODE_NAME = 'ORDER_STAGE'`
	เอสคิวแอลคอมมานด์
	`SELECT COUNT(*) FROM ORDER_PROCESS`
	
	`SELECT COUNT(*) FROM ORDER_STAGE`

ผลกระทำการ (Execute) Count: 3 Rows Count: 3 Rows	**ผลกระทำการ (Execute)** Count: 3 Rows Count: 3 Rows

ตารางที่ 36 รีแฟคทอริงฐานข้อมูลด้วย Split Table Add foreign key Constraint ของ เอสคิวแอลแอนติแพตเทิร์น Use a Use Dual-Purpose Foreign Key

ก่อนดำเนินการรีแฟคทอริงฐานข้อมูล	หลังดำเนินการรีแฟคทอริงฐานข้อมูล

เอสคิวแอลคอมมานด์
`SELECT LOGIN FROM`
`  ( SELECT ISSUE_ID FROM COMMENTS`
`  WHERE ISSUE_TYPE = 'AGENT'`
`  UNION`
`  SELECT ISSUE_ID FROM COMMENTS`
`  WHERE ISSUE_TYPE = 'EMPOYEE') ISSUE`
`  INNER JOIN AGENT`
`  ON AGENT.ID = ISSUE_ID`
	เอสคิวแอลคอมมานด์
	`SELECT LOGIN FROM COMMENTS_AGENT`
	`  WHERE ISSUE_ID = '1'`
	
	`  SELECT LOGIN FROM COMMENTS_EMPOYEE`
	`  WHERE ISSUE_ID = '2'`

ผลกระทำการ (Execute) LOGIN SOMCHI SOMYIN	**ผลกระทำการ (Execute)** LOGIN SOMCHI SOMYIN

ตารางที่ 37 รีแฟคทอริงฐานข้อมูลด้วย Replace One to many With Associative Table ของ เอสคิวแอลแอนติแพตเทิร์น Create Multiple Columns

ก่อนดำเนินการรีแฟคทอริงฐานข้อมูล	หลังดำเนินการรีแฟคทอริงฐานข้อมูล

เอสคิวแอลคอมมานด์
`SELECT EMP_ID,SHIFT1,SHIFT2,SHIFT3`
`FROM DBO.EMPOYEE`
`WHERE SHIFT1 = 'ALLDAYSHIFT'`
`UNION`
`SELECT EMP_ID,SHIFT1,SHIFT2,SHIFT3`
`FROM DBO.EMPOYEE`
`WHERE SHIFT2 = 'ALLDAYSHIFT'`
`UNION`
`SELECT EMP_ID,SHIFT1,SHIFT2,SHIFT3`
`FROM DBO.EMPOYEE`
`WHERE SHIFT3 = 'ALLDAYSHIFT'`
	เอสคิวแอลคอมมานด์
	`SELECT EMP_ID`
	`WHERE SHIFT =`
	`'ALLDAYSHIFT`

ก่อนดำเนินการรีแฟคทอริงฐานข้อมูล	หลังดำเนินการรีแฟคทอริงฐานข้อมูล
ผลกระทำการ (Execute)	**ผลกระทำการ (Execute)**

EMP_ID	SHIFT1	SHIFT2	SHIFT3
1	ALLDAYSHIFT	NULL	NULL
2	NULL	ALLDAYSHIFT	NULL
3	NULL	NULL	ALLDAYSHIFT

EMP_ID
1
2
3

ตารางที่ 38 รีแฟคทอริงฐานข้อมูลด้วย Merge Table และ Drop Table ของ เอสคิวแอลแอนติแพตเทิร์น Clone Table or Column

ก่อนดำเนินการรีแฟคทอริงฐานข้อมูล	หลังดำเนินการรีแฟคทอริงฐานข้อมูล
เอสคิวแอลคอมมานด์ `SELECT COUNT(ORDER_ID) FROM DBO.ORDER_DETAIL UNION SELECT ORDER_ID FROM DBO.ORDER_DETAIL_2015`	**เอสคิวแอลคอมมานด์** `SELECT COUNT(ORDER_ID) FROM DBO.ORDER_DETAIL`
ผลกระทำการ (Execute)	**ผลกระทำการ (Execute)**

Count
2619 Rows

Count
2619 Rows

ประวัติผู้เขียนวิทยานิพนธ์

นางสาวปุณยนุช ขำนิล เกิดวันที่ 14 มกราคม 2533 สำเร็จการศึกษาระดับปริญญาตรี หลักสูตรวิทยาศาสตรบัณฑิต (วท.บ.) คณะวิทยาศาสตร์ มหาวิทยาลัยเกษตรศาสตร์ ปีการศึกษา 2555

ประสบการณ์ ปัจจุบันเป็นพนักงานองค์กรเอกชน บริษัท เอสซีจี จำกัด ตำแหน่ง นักวิเคราะห์ 4 ปี และเคยอยู่บริษัท ซีทีเอเชีย จำกัด ในตำแหน่ง 1 ปี ตำแหน่ง ผู้ดูแลฐานข้อมูล

เข้าศึกษาต่อระดับปริญญามหาบัณฑิต ปีการศึกษา 2559 หลักสูตรวิทยาศาสตรมหาบัณฑิต (วท.ม.) สาขาวิศวกรรมซอฟต์แวร์ ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย