บทที่ 2

ทฤษฎีที่ใช้ในการแก้ปัญหา

ในบทนี้จะกล่าวถึงรายละเอียดของทฤษฎีที่ใช้ในการแก้ปัญหาในงานวิจัย ซึ่งแบ่งออกเป็น 5 ส่วนหลัก ได้แก่ ภาษาที่ใช้ในการจัดการกระแสข้อมูล (ESL) [1] ภาษาที่ใช้ในการจัดการกระแสข้อมูลที่มีลักษณะเป็นเหตุการณ์ [5] ภาษาในการสอบถาม พาสทรี [7] และการเพิ่มประสิทธิภาพในการสอบถาม (Query Optimization)

2.1 ภาษาในการสอบถามอีเอสแอล

อีเอสแอล (Expressive Stream Language หรือ ESL) [1] เป็นภาษาที่ถูกพัฒนาขึ้น เพื่อให้ฐานข้อมูลสามารถทำงานร่วมกับข้อมูลที่เป็นกระแส (Stream) ได้ โดยภาษาอีเอสแอลมีความสามารถในการใช้งานแบบ แอด ฮ๊อก (Ad hoc) ในฐานข้อมูล รวมถึงความสามารถในการค้นหา คัดกรอง และรวบรวมข้อมูล และเนื่องจากภาษาอีเอสแอลได้รับการพัฒนามาจากภาษาเอสคิวแอล จึงมีการใช้งานในลักษณะเดียวกับภาษาเอสคิวแอล โดยมีเป้าหมายเพื่อลดขนาดของการใช้งานในภาษาเอสคิวแอลกับกระแสข้อมูลที่จัดการได้ยาก ภาษาอีเอสแอลมีความสามารถในการทำงานร่วมกับกระแสข้อมูล ซึ่งนำข้อมูลเข้ามายังฐานข้อมูลอย่างต่อเนื่องไม่จำกัด

สมมุติให้มหาวิทยาลัยแห่งหนึ่งมีการเก็บคะแนนการทำกิจกรรมของนักศึกษา มีการใช้ระบบชี้เฉพาะด้วยคลื่นวิทยุ โดยทำการแจกบัตรกิจกรรมที่มีการใส่แทก (Tag) การชี้เฉพาะด้วยคลื่นวิทยุอยู่ภายในเพื่อบันทึกข้อมูลการทำกิจกรรมของนักศึกษา เมื่อนักศึกษาเข้าทำกิจกรรมในแต่ละฐานเสร็จ อาจารย์ผู้คุมฐานกิจกรรมจะทำการให้คะแนนการทำกิจกรรม โดยการนำบัตรกิจกรรมมาทำการบันทึกคะแนนพร้อมทั้งทาบบัตรเพื่อระบุตัวนักศึกษาที่ทำกิจกรรมผ่านเครื่องอ่านการชี้เฉพาะด้วยคลื่นวิทยุ (RFID Reader) และในแต่ละกิจกรรมจะมีคะแนนเต็ม 30 คะแนน

```
CREATE STREAM StartCheck (
        tagID int ,
        studentID int ,
        start_point real,
        start_time timestamp )
ORDER BY start_time;
SOURCE 'PORT4445';

CREATE STREAM CheckPoint (
        tagID int,
        teacherID int ,
        rating_point real,
        current_time timestamp)
ORDER BY current_time ;
SOURCE 'PORT4446';
```

รูปที่ 2.1 ตัวอย่างการสร้างและเก็บกระแสข้อมูลในภาษาอีเอสแอล

จากรูปที่ 2.1 เป็นตัวอย่างคำสั่งในการสร้างและเก็บกระแสข้อมูลในภาษาอีเอสแอล คำสั่งที่หนึ่งเป็นการเก็บข้อมูลบัตรชี้เฉพาะด้วยคลื่นวิทยุของนักศึกษา โดยจะมีการเก็บคะแนนเริ่มต้น และเวลาการเริ่มทำกิจกรรม มีการสร้างตารางชื่อ StartPoint เพื่อเก็บข้อมูลรหัสบัตรชี้เฉพาะด้วยคลื่นวิทยุ (tagID) รหัสนักศึกษา (studentID) คะแนนเริ่มต้น (start_point) และเวลาการเริ่มทำกิจกรรม (start_time) ซึ่งข้อมูลดังกล่าวมีลักษณะเป็นกระแสที่ได้จากการเริ่มทำกิจกรรมของนักเรียนเรียงตามเวลาการเริ่มทำกิจกรรมนั้นๆ คำสั่งที่สองเป็นการเก็บข้อมูลการให้คะแนนการทำกิจกรรมของนักเรียนจากอาจารย์ผู้คุมฐานกิจกรรม โดยการสร้างตารางชื่อ CheckPoint เพื่อเก็บข้อมูลรหัสบัตรชี้เฉพาะด้วยคลื่นวิทยุ (tagID) รหัสอาจารย์ (teacherID) คะแนนที่ได้ (rating_price) และเวลาที่ให้คะแนน (current_time) ซึ่งข้อมูลดังกล่าวมีลักษณะเป็นกระแสเรียงตามเวลาการให้คะแนนของอาจารย์ โดยทั้ง

สองคำสั่งทั้งสองทำหน้าที่เก็บข้อมูลการทำกิจกรรมจากข้อมูลที่เป็นกระแสที่มีการรับข้อมูลที่เข้ามาอย่างต่อเนื่อง จากการสั่งงานเพียงครั้งเดียว

ภาษาอีเอสแอลสามารถคัดกรองกระแสข้อมูลที่เข้ามาได้ โดยสามารถกำหนดรูปแบบการคัดกรองได้จากการใช้คำสั่งภาษาเอสคิวแอล ดังแสดงในรูปที่ 2.2

```
CREATE STREAM HighScore AS
SELECT tagID, studentID, rating_point, current_time
        FROM CheckPoint WHERE rating_point  > 25
```

รูปที่ 2.2 ตัวอย่างการกรองกระแสข้อมูลในภาษาอีเอสแอล

จากรูปที่ 2.2 เป็นตัวอย่างคำสั่งในการสร้างตาราง HighScore เพื่อเก็บข้อมูลนักศึกษาที่เข้าทำกิจกรรมที่ได้คะแนนในการทำกิจกรรมมากกว่า 25 คะแนน (อ้างอิงจากตาราง CheckPoint) โดยมีการเก็บเวลาการให้คะแนน (current_time) เพื่อป้องกันการซ้ำ และบอกลำดับของข้อมูล สังเกตได้ว่า ภาษาอีเอสแอลสามารถรองรับการเก็บกระแสข้อมูล และสามารถกำหนดเงื่อนไขในการเก็บข้อมูลที่ต้องการได้

ภาษาอีแอสแอลสามารถสร้างตารางเก็บข้อมูลที่มีการนำข้อมูลมาจากการเชื่อมความสัมพันธ์ระหว่างข้อมูลที่เป็นกระแสหรือตารางข้อมูลแบบธรรมดาไม่ใช้กระแสข้อมูลในฐานข้อมูลได้

```
StudentInfo(studentID, Fname, …)
```

รูปที่ 2.3 รูปแบบการเก็บข้อมูลนักเรียน

จากรูปที่ 2.3 เป็นรูปแบบการเก็บข้อมูลนักเรียนจากฝ่ายทะเบียน โดยเก็บรหัสนักศึกษา และชื่อของนักศึกษา ซึ่งเป็นข้อมูลที่มีการบันทึกแบบธรรมดา

```
CREATE STREAM TopStudent
SELECT Fname, tagID, rating_point, current_time
FROM HightScore AS H, StudentInfo AS S
WHERE H.studentID=S.studentID
```

รูปที่ 2.4 ตัวอย่างการเชื่อมตารางกระแสข้อมูล

จากรูปที่ 2.4 แสดงตัวอย่างคำสั่งอีเอสแอลในการเก็บข้อมูลนักศึกษาที่ได้คะแนนมากกว่า 25 คะแนน (อ้างอิงจากตาราง HighScore) จากการทำกิจกรรม โดยการสร้างตาราง TopStudent จากการ

เชื่อมตาราง HightScore ที่มีการเก็บข้อมูลคะแนนที่มากกว่าเกณฑ์ ซึ่งได้จากกระแสข้อมูลกับตาราง StudentInfo ที่มีเก็บข้อมูลนักศึกษา โดยมี studentID เป็นกุญแจหลักที่จะไม่มีการซ้ำกันในตาราง StudentInfo ในการเชื่อมความสัมพันธ์ระหว่างสองตาราง สังเกตได้ว่า ภาษาอีเอสแอลสามารถรวมตารางในฐานข้อมูลจากลำดับความสัมพันธ์ได้

ในการประมวลผล เนื่องจากเป็นกระแสข้อมูล ทำให้ยากต่อการจัดการ และมีความแตกต่างกันในด้านข้อมูล ดังนั้น จึงมีการแบ่งรูปแบบของข้อมูลเป็น 2 ประเภท ดังนี้

1. การรวมฐาน (Base aggregates) คือ ข้อมูลที่ใช้ในการคำนวณค่า ซึ่งค่าดังกล่าวจะถูกเก็บไปยังฐานข้อมูล เมื่อระบบได้รับคำสั่งหยุดการรับข้อมูลหรือการรับข้อมูลเสร็จสิ้น เช่น คะแนนรวมของนักศึกษาที่เข้าทำกิจกรรมเมื่อสิ้นสุดกิจกรรม

```
CREATE STREAM RecentSum
SELECT decay_online_sum(rating_point)
FROM CheckPoint
```

รูปที่ 2.5 ตัวอย่างคำสั่งรวมฐาน

จากรูปที่ 2.5 เป็นตัวอย่างคำสั่งรวมถามโดยผลลัพธ์จะแสดงข้อมูลคะแนนรวมในการทำกิจกรรมของนักศึกษา โดยทำการประมวลผลหลังจากกิจกรรมเสร็จสิ้น ทำให้ข้อมูลนั้นๆ ไม่มีการเปลี่ยนแปลง

การรวมวินโดว์หลายชั้น (Window aggregates) คือ ข้อมูลที่เข้าสู่ฐานข้อมูลอย่างต่อเนื่อง ทำให้ปริมาณข้อมูลไม่แน่นอน ยากต่อการคำนวณ ดังนั้น เมื่อมีการเรียกใช้ข้อมูลดังกล่าว ระบบจะใช้ข้อมูลปัจจุบันที่มีอยู่ เพื่อให้ได้ข้อมูลที่ปัจจุบันที่สุดในขณะนั้น โดยไม่สนใจข้อมูลที่เข้ามาภายหลัง

```
CREATE STREAM MaxPointInActive
SELECT tagID, teacherID,
max(rating_point) OVER(RANGE UNLIMITED PRECEDING) AS     NewTime
        FROM CheckPoint
```

รูปที่ 2.6 ตัวอย่างคำสั่งวินโดว์หลายชั้น

จากรูปที่ 2.6 เป็นตัวอย่างคำสั่งวินโดว์หลายชั้นผลลัพธ์ที่ได้จะแสดงข้อมูลคะแนนที่มากที่สุดในการทำกิจกรรม โดยเมื่อมีการเรียกใช้ข้อมูล ระบบจะทำการคำนวณจากข้อมูลปัจจุบันที่มีอยู่ในขณะนั้น เพื่อให้ได้ข้อมูลที่ปัจจุบันที่สุด

สำหรับการรวมในภาษาอีเอสแอล ได้ถูกปรับปรุงให้มีความเหมาะสมกับการรับกระแสข้อมูลจากหลายเส้นทางพร้อมกัน

```
SELECT tagID, start_point, start_time
FROM StartCheck
WHERE tagID = 1000
UNION
SELECT tagID, teacherId, NewTime
FROM MaxPointInActive
WHERE tagID = 1000
```

รูปที่ 2.7 ตัวอย่างการใช้งานวินโดว์หลายชั้น

จากรูปที่ 2.7 แสดงตัวอย่างคำสั่งวินโดว์หลายชั้นโดยผลลัพธ์จะเป็นการรวมกระแสข้อมูลสองเส้นทาง โดยในกระแสข้อมูลต้องมีรหัสแทก (tagID) เท่ากับ 1000 ในการรวมกระแสข้อมูลจะมีความแตกต่างกับการรวมข้อมูลในฐานข้อมูลปรกติ เนื่องจากมีเวลาเข้ามาเกี่ยวข้อง จึงจำเป็นต้องมีเวลาในการรวมที่ชัดเจน ซึ่งจะใช้ข้อมูลเวลาก่อนได้รับคำสั่งในการทำงาน

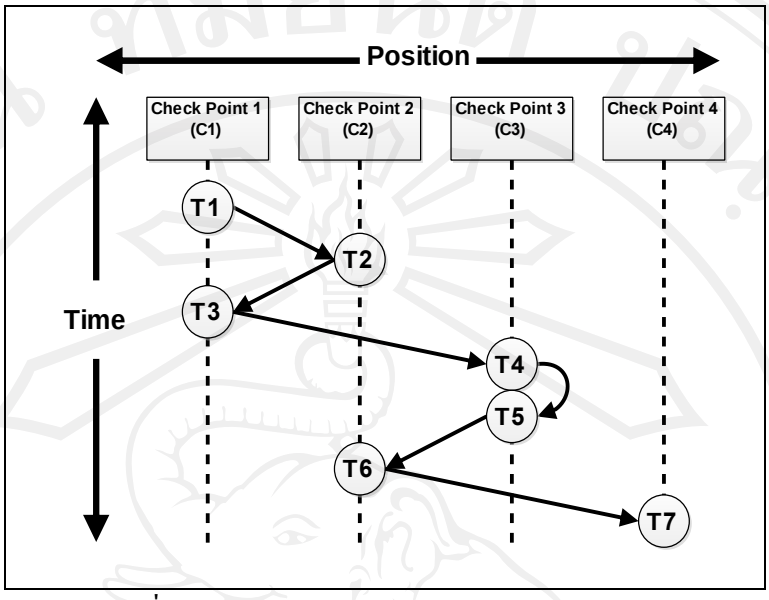
2.2 ภาษาในการสอบถามอีเอสแอลอีเวนท์

อีเอสแอลอีเวนท์ [5] เป็นภาษาที่ได้รับการพัฒนามาจากภาษาอีเอสแอล โดยเพิ่มความสามารถในการจัดการเหตุการณ์ และลำดับขั้นตอนในการทำงาน เพื่อให้มีความเหมาะสมกับกระแสข้อมูล ซึ่งมีการทำงานเป็นลำดับขั้นตอน ตัวอย่างเช่น การขนส่งสินค้าจากโรงงานจากประเทศจีนไปยังผู้รับสินค้า

ในสหรัฐอเมริกาจำเป็นต้องมีการตรวจสอบสินค้าตลอดเส้นทางตั้งแต่เดินทางออกจากโรงงานผ่านท่าเรือในประเทศจีนไปยังท่าเรือในประเทศสหรัฐอเมริกา สังเกตได้ว่า การทำงานจะเป็นลำดับขั้นตอน ไม่สามารถข้ามขั้นตอนต่างๆได้ หรือ การวิจัยภายในห้องทดลองจำเป็นต้องมีการทำงานเป็นขั้นตอน หากทำข้ามขั้นตอนอาจทำให้เกิดความผิดพลาดในการทำการทดลองได้ เป็นต้น แต่เนื่องจากในปัจจุบันยังไม่มีการพัฒนาภาษาเพื่อใช้งานในลักษณะนี้ ทำให้เกิดความล่าช้าในการทำงาน จึงมีการพัฒนาภาษาอีเอสแอลอีเวนท์ขึ้นเพื่อใช้งาน โดยมีตัวดำเนินการแบ่งเป็น 2 ส่วนคือ

1. การพัฒนาภาษาเอสคิวแอล ให้มีความสามารถในการติดต่อฐานข้อมูลจากคำสั่งเดิมที่มีอยู่ เช่น คำสั่งเพิ่ม (Insert) ลด (Delete) และปรับปรุง (Update) ให้สามารถรองรับข้อมูลที่ได้รับจากการชี้เฉพาะด้วยคลื่นความถี่วิทยุ ซึ่งเป็นข้อมูลเวลาที่มีความต่อเนื่องในรูปแบบกระแสข้อมูลได้ และการพัฒนาภาษาเอสคิวแอลให้สามารถทำกับเงื่อนไขจากคำสั่งเดิม เช่น การรวมข้อมูล (Conjunctions) การปฏิเสธ (Negations) และการเรียงลำดับ (Sequence) ให้สามารถทำเงื่อนไขกับข้อมูลที่เป็นกระแสข้อมูลได้เหมือนคำสั่งเอสคิวแอลแบบปรกติ ตัวอย่างเช่น A and B จะเป็นจริง เมื่อเหตุการณ์ A และเหตุการณ์ B เกิดขึ้นจริง โดยที่ข้อมูลที่เข้ามายังเหตุการณ์เป็นกระแสข้อมูล

2. การเรียงลำดับจับคู่ข้อมูล (Tuple Pairing Mode) เป็นฟังก์ชันที่ได้รับการพัฒนาขึ้นมาภายในภาษาอีเอสแอลอีเวนท์ เพื่อใช้งานกับข้อมูลขนาดใหญ่ที่มีหลายเหตุการณ์เกิดขึ้น โดยจะเป็นระบบจัดการเหตุการณ์ที่เกี่ยวข้องกับเวลา คือ มีการตรวจสอบลำดับข้อมูลที่เข้ามาโดยดูจากเวลา เพื่อตรวจสอบว่าข้อมูลที่ได้รับมีลำดับการทำงานที่ถูกต้องหรือไม่ และแสดงค่าเป็นจริง (True) หรือเท็จ (False) เพื่อนำไปใช้งานในภาษาเอสคิวแอลต่อไป



รูปที่ 2.8 ตัวอย่างชุดข้อมูลจากการทำกิจกรรมค่าย

จากรูปที่ 2.8 เป็นการยกตัวอย่างจากการทำกิจกรรมค่ายของโรงเรียน นักเรียนแต่ละคนต้องทำกิจกรรมทั้งหมด 4 กิจกรรม ในแต่ละฐานกิจกรรมจะมีตัวอ่านการชี้เฉพาะด้วยคลื่นวิทยุ เพื่อใช้ในการตรวจสอบการเข้าทำกิจกรรมจากแทก ซึ่งบรรจุรหัสนักศึกษา สมมุติให้นักเรียนคนหนึ่งผ่านจุดตรวจสอบ เวลาทั้งหมดจะถูกบันทึกผ่านเครื่องอ่านเป็น T1-T7 จากรูปจะสังเกตได้ว่า นักเรียนได้ผ่านเครื่องอ่านที่ 1 (C1) เมื่อเวลา T1 ต่อมาจึงผ่านเครื่องอ่านที่ 2 (C2) เมื่อเวลา T2 กลับมายังเครื่องอ่านที่ 1 อีกครั้ง เมื่อเวลา T3 และในเวลา T7 นักเรียนคนดังกล่าวสิ้นสุดกิจกรรมที่เครื่องอ่านที่ 4 (C4) สมมุติให้ในการผ่านกิจกรรมของนักเรียนแต่ละคนจะต้องผ่านจุดตรวจสอบทั้งหมด 4 จุดคือ C1 C2 C3 และ C4

ในภาษาอีเอสแอลอีเวนท์มีตัวดำเนินการที่ช่วยในการจัดการข้อมูลที่เป็นเหตุการณ์ที่น่าสนใจทั้งหมด 3 ตัวดำเนินการดังนี้

1. ตัวดำเนินการเอสอีคิว (The SEQ Operator) เป็นตัวดำเนินการที่ทำหน้าที่คัดกรองลำดับการทำงานของข้อมูลที่มาจากหลายเส้นทาง เช่น SEQ(C1, C2) จะแสดงค่าเป็นจริง ถ้าเวลาที่บันทึก แสดงว่า C1 เกิดก่อน C2 และตัวดำเนินการนี้ยังสามารถใช้งานได้พร้อมกันจากหลายเส้นทางกระแสข้อมูล ตัวอย่างที่สอง SEQ(C1, C2, C3) จะแสดงค่าเป็นจริง เมื่อ C1 เกิดเป็นอันดับแรก ตามด้วย C2 และ C3 ตามลำดับไป

```
SELECT C1.object_id, C1.time_stamp, C2.time_stamp, C3.time_stamp, C4.time_stamp
FROM C1, C2, C3, C4
WHERE SEQ(C1, C2, C3, C4)
```

รูปที่ 2.9 ตัวอย่างคำสั่งเอสอีคิว

จากรูปที่ 2.9 เป็นตัวอย่างคำสั่งตัวดำเนินการเอสอีคิว ในการแสดงข้อมูลเวลานักศึกษาที่ได้จากการบันทึกจากตัวอย่างจากรูปที่ 2.8 โดยจะทำการแสดงข้อมูลรหัสนักศึกษา และเวลาในแต่ละจุดที่มีการบันทึก โดยนักศึกษาแต่ละคนจะต้องมีการบันทึกเวลาในการทำกิจกรรมครบทุกจุดตามตัวดำเนินการเอสอีคิว

2. ตัวดำเนินการจับคู่ข้อมูล (Tuple Pairing Mode) เนื่องจากตัวดำเนินการเอสอีคิวสามารถลดจำนวนข้อมูลที่มีการเรียงลำดับไม่ถูกต้องได้ แต่ยังไม่ครอบคลุมเงื่อนไขที่ซับซ้อนมากขึ้น จึงมีการสร้างตัวดำเนินการจับคู่ข้อมูลในคำสั่งอีเอสแอลอีเวนท์ ดังนี้

```
SELECT C1.object_id, C1.time_stamp, C2.time_stamp, C3.time_stamp, C4.time_stamp
FROM C1, C2, C3, C4
SEQ(C1, C2, C3, C4)
MODE CONSECUTIVE
```

รูปที่ 2.10 ตัวอย่างคำสั่งตัวดำเนินการจับคู่ข้อมูล

จากรูปที่ 2.10 เป็นตัวอย่างคำสั่งตัวดำเนินการจับคู่ข้อมูล โดยข้อมูลที่จะนำมาแสดงจะต้องมีการเรียงลำดับจาก C1 C2 C3 และ C4 ตามโหมดการจับคู่ที่มีลำดับ

ตัวดำเนินการมีรูปแบบที่น่าสนใจแบ่งเป็น 4 รูปแบบ

2.1 การจับคู่แบบไม่จำกัด (UNRESTRICTED) เป็นการแสดงข้อมูลที่มีการทำงานครบทุกการทำงาน จะแสดงค่าเป็นจริงทั้งหมด เมื่อมีข้อมูลอยู่ในการทำงานครบทุกการทำงานตามที่กำหนด โดยไม่สนใจจำนวนข้อมูลที่เข้ามาซ้ำกัน ผลลัพธ์จะอยู่ในรูปของ (เวลาที่มาถึง : ตำแหน่งเครื่องอ่าน) จากตัวอย่างข้อมูลในรูปที่ 2.8 จะได้ว่า

(T1:C1, T2:C2, T4:C3, T7:C4) **(T3:C1, T2:C2, T4:C3, T7:C4)**

(T1:C1, T6:C2, T4:C4, T7:C4) **(T3:C1, T6:C2, T4:C4, T7:C4)**

(T1:C1, T2:C2, T5:C3, T7:C4) **(T3:C1, T2:C2, T5:C3, T7:C4)**

(T1:C1, T6:C2, T5:C3, T7:C4) **(T3:C1, T6:C2, T5:C3, T7:C4)**

2.2 การจับคู่ที่มีลำดับ (CONSECUTIVE) เป็นการแสดงข้อมูลที่มีการเรียงลำดับการทำงานได้ถูกต้อง จะแสดงค่าเป็นจริงทั้งหมดเมื่อมีการเรียงลำดับขั้นตอนถูกต้องตามที่กำหนด โดยไม่สนใจจำนวนข้อมูลที่เข้ามาซ้ำกัน จากตัวอย่างข้อมูลในรูปที่ 2.8 จะได้ว่า

(T1:C1, T2:C2, T4:C3, T7:C4) **(T1:C1, T2:C2, T5:C3, T7:C4)**

2.3 การจับคู่ที่มีลำดับที่ปัจจุบันที่สุด (RECENT) เป็นการแสดงข้อมูลที่เชื่อถือได้มากที่สุด เนื่องจากในการตรวจสอบเวลาที่บันทึกไว้ ซึ่งอาจมีการซ้ำของข้อมูลเกิดขึ้นได้ มีความคลาดเคลื่อนของข้อมูล ทำให้ข้อมูลที่ได้ไม่มีคุณภาพ จึงมีฟังก์ชันการจับคู่ที่น่าเชื่อถือที่สุดโดยจะแสดงค่าเป็นจริงเมื่อมีการเรียงลำดับขั้นตอนถูกต้องตามที่กำหนด และในแต่ข้อมูลเวลาจะเป็นข้อมูลสุดท้าย เพื่อให้ได้ข้อมูลที่น่าเชื่อถือที่สุด จากตัวอย่างข้อมูลในรูปที่ 2.8 จะได้ว่า

(T1:C1, T2:C2, T5:C3, T7:C4)

2.4 การจับคู่ที่มีลำดับที่เก่าที่สุด (CHRONICLE) เป็นการแสดงข้อมูลที่เก่าที่สุด เนื่องจากในการตรวจสอบเวลาบางกรณีมีความสนใจเวลาที่เก่าที่สุด อาจมีการซ้ำซ้อนของข้อมูล ทำให้เวลาในการตรวจสอบมีความผิดพลาดและได้ข้อมูลที่ไม่มีคุณภาพ จึงมีการสร้างตัวดำเนินการในการจับคู่ที่เก่าที่สุดโดยจะแสดงค่าเป็นจริงเมื่อข้อมูลมีการเรียงลำดับขั้นตอนถูกต้องตามที่กำหนด และในแต่ข้อมูลเวลาจะเป็นข้อมูลแรกสุด จากตัวอย่างข้อมูลในรูปที่ 2.8 จะได้ว่า

(T1:C1, T2:C2, T4:C3, T7:C4)

3. ตัวดำเนินการสไลด์ดิงวินโดว์ (Sliding Windows on SEQ) เนื่องจากในการใช้งานระบบอาจมีการกำหนดช่วงเวลาที่แน่นอน จึงมีการพัฒนาตัวดำเนินการที่ทำหน้าที่กำหนดช่วงเวลาในการทำงานเพื่อเป็นการลดภาระในการตรวจสอบเงื่อนไขการทำงาน ทำให้ระบบมีประสิทธิภาพมากขึ้น ตัวอย่างเช่น ในการทำกิจกรรมของนักเรียนจะต้องมีตารางในการทำกิจกรรม ซึ่งนักเรียนต้องทำกิจกรรมให้อยู่ในเวลาที่กำหนด เพื่อให้สามารถทำกิจกรรมได้ครบทุกกิจกรรม และอาจนำมาคิดเป็น

คะแนนได้ โดยสามารถใช้ตัวดำเนินการสไลด์ดิงวินโดว์ เพื่อตรวจสอบข้อมูลสินค้าได้โดยใช้คำสั่งอีเอสแอลอีเวนท์

```
SELECT C1.object_id, C1.time_stamp, C2.time_stamp, C3.time_stamp, C4.time_stamp
FROM C1, C2, C3, C4
WHERE  SEQ(C1, C2, C3, C4)
        OVER[120 MINUTES PROCESSING C4]
```

รูปที่ 2.11 ตัวอย่างคำสั่งสไลด์ดิงวินโดว์

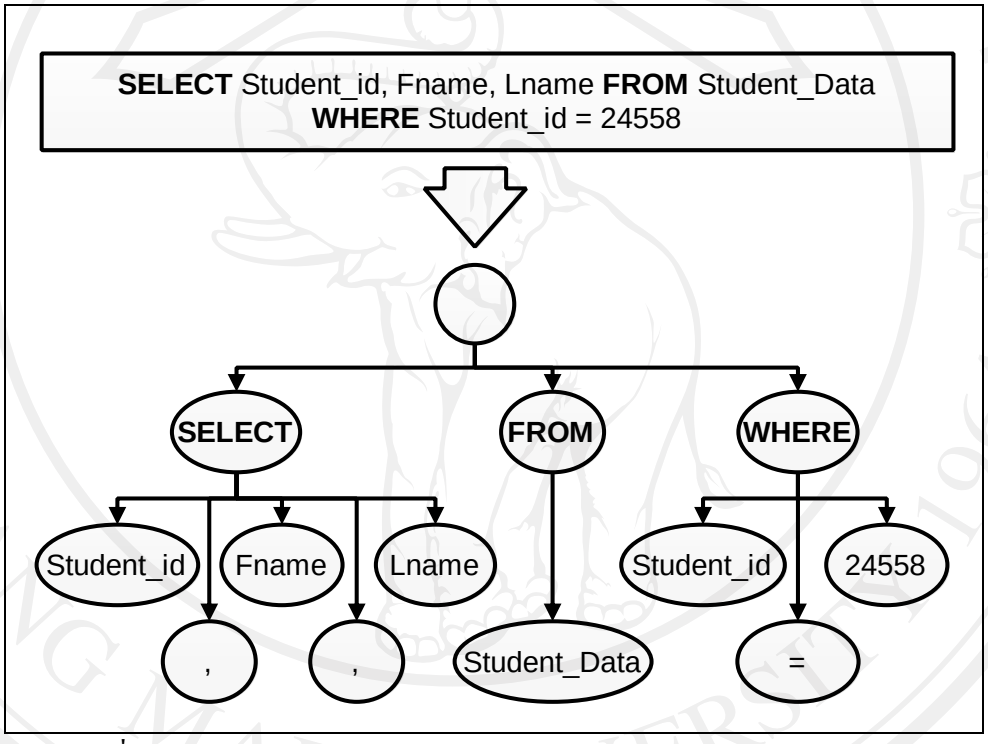
จากรูปที่ 2.11 เป็นตัวอย่างคำสั่งที่มีการใช้งานคำสั่งสไลด์ดิงวินโดว์ โดยกำหนดให้แสดงข้อมูลรหัสสินค้า และข้อมูลเวลาที่สินค้าผ่านในแต่ละจุด โดยสินค้าจะต้องผ่านจุดที่มีการบันทึกเวลาตามลำดับ คือ C1 C2 C3 และ C4 และต้องใช้เวลาตั้งแต่จุดที่มีการบันทึกเวลาจุดแรกถึงจุดสุดท้ายไม่เกิน 3 ชั่วโมง

2.3 ภาษาในการสอบถาม

ภาษาในการสอบถาม (Query language) เป็นภาษาที่ใช้สำหรับสอบถามหรือจัดการข้อมูลในฐานข้อมูล โดยภาษาประเภทนี้ที่ได้รับความนิยมสูงสุด คือ ภาษาเอสคิวแอล คิดค้นโดยนักวิทยาศาสตร์ของไอบีเอ็มในปี 1970 มีรูปแบบคำสั่งที่คล้ายกับประโยคในภาษาอังกฤษมาก ปัจจุบันองค์กร ANSI (American National Standard Institute) ได้ประกาศให้ SQL เป็นภาษามาตรฐานสำหรับระบบการจัดการฐานข้อมูลเชิงสัมพันธ์ (Relational Database Management System หรือ RDBMS) ซึ่งเป็นระบบ DBMS ที่ใช้กันอย่างแพร่หลายที่สุด ระบบการจัดการฐานข้อมูลเชิงสัมพันธ์ทุกระบบจะใช้คำสั่งพื้นฐานของภาษา SQL ได้เหมือนกัน แต่อาจมีคำสั่งพิเศษที่แตกต่างกันบางส่วน เนื่องจากบริษัทผู้ผลิตแต่ละรายพยายามพัฒนา RDBMS ของตนเองให้มีลักษณะที่โดดเด่นมากกว่าระบบอื่น โดยเพิ่มคุณสมบัติที่เกินข้อจำกัดของ ANSI ซึ่งคิดว่าเป็นประโยชน์ต่อผู้ใช้

2.4 พาสทรี

พาสทรี [7] เป็นกระบวนการที่ตัวแปลงโปรแกรม (Compiler) ของคอมพิวเตอร์ใช้ในการวิเคราะห์โครงสร้างรูปประโยคของภาษาโปรแกรมที่มีรูปแบบกฎเกณฑ์เป็นลำดับชั้น (Hierarchical Syntactic Structure) โดยพาสทรีจะมีลักษณะและรูปแบบหลักที่เฉพาะเจาะจง โดยในงานวิจัยนี้ได้ศึกษาการนำพาสทรีมาทำการแยกโครงสร้างของภาษาในการสอบถามคือภาษาเอสคิวแอล เพื่อใช้ในการแปลงภาษาอีเอสแอลต่อไป ดังรูปที่ 2.12

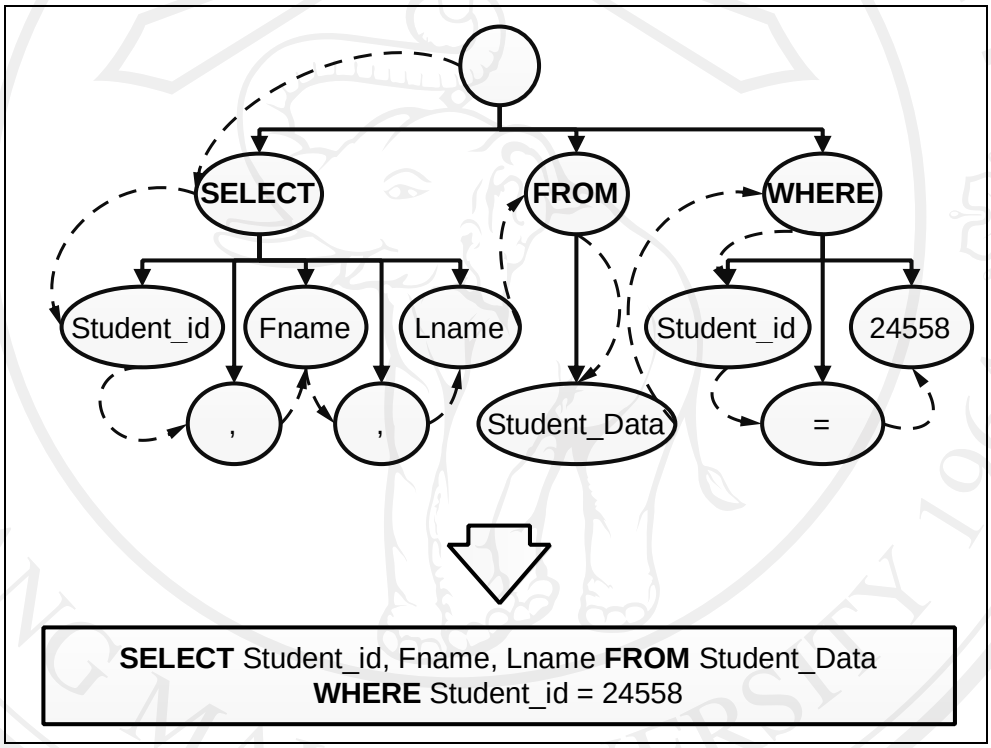


รูปที่ 2.12 ตัวอย่างการแยกโครงสร้างภาษาเอสคิวแอลโดยใช้วิธีการพาสทรี

จากรูปที่ 2.12 เป็นตัวอย่างการแยกโครงสร้างภาษาในการสอบถามเอสคิวแอล ด้วยวิธีการพาสทรี โดยการแยกโครงสร้างให้เป็นรูปต้นไม้ โดยการสร้างโหนดราก (Root) คือโหนดของทรีที่ไม่มีพาเรนต์ (Parent) เพื่อเริ่มต้น ต่อมาจึงสร้างโหนดประเภทอินเทอร์นอล (Internal Node) คือโหนดของทรีที่มีไชด์ (Child) ซึ่งในภาษาอีเอสแอลเป็นภาษาที่มีโครงสร้างที่มีรูปแบบตายตัว คือมีคำหลัก (Keyword) ในโครงสร้างภาษาเพื่อใช้ในการทำงานคือ “SELECT” “FROM” “WHERE” “GROUP” และ “ORDER BY” ในพาสทรีภาษาเอสคิวแอลจึงกำหนดคำเหล่านี้ให้เป็นโหนดประเภทอินเทอร์นอล ต่อมาจึงนำคำที่ไม่ใช้คำหลักมาทำการสร้างเป็นโหนดประเภทเอ็กซ์เทอร์นอล (External Node) คือ

โหนดของทรีที่ไม่มีไชด์ หรือ โหนดใบ (Leaf Node) โดยกำหนดเป็นไซด์โหนดของโหนดอินเทอร์นอลของคำหลักข้างหน้า

เมื่อทำการปรับเปลี่ยนโครงสร้างภาษาแล้วจึงเป็นการรวมโครงสร้างภาษาเอสคิวแอลที่มีรูปแบบเป็นทรีจากการพาสทรี ให้กลับมาเป็นคำสั่งภาษาเอสคิวแอลเพื่อนำไปใช้งานต่อไป โดยจะใช้วิธีการทรีทราเวิร์สซัลคือการทราเวิร์ส(Traverse) โหนดทุกโหนดในทรี ในพาสทรีของภาษาเอสคิวแอลจะใช้วิธีการ พรีออเดอร์ (Pre-order) เพื่อรวมโครงสร้างภาษา ดังรูปที่ 2.13



รูปที่ 2.13 ตัวอย่างการทรีทราเวิร์สซัสพาสทรีของภาษาเอสคิวแอล

PRE-ORDER	ลำดับ	1	2	3	4	5	6
1. แสดงค่าอินเทอร์นอล	เนื้อหาในโหนด	SELECT	Student_id	,	Fname	,	Lname
2. แสดงค่าเอ็กซ์เทอร์นอล	ลำดับในโหนด	1	2	3	4	5	6

ลำดับ	7	8	9	10	11	12
เนื้อหาในโหนด	FROM	Student_Data	WHERE	Student_id	=	24558
ลำดับในโหนด	7	8	9	10	11	12

รูปที่ 2.14 ผลที่ได้จากการทำพรีออเดอร์จากรูปที่ 2.12

จากรูปที่ 2.13 เป็นตัวอย่างการทรีทราเวิร์สซัสพาสทรีของภาษาเอสคิวแอล โดยมีหลักการคือ โหนดใดๆ จะถูกทราเวิร์สก่อนเดสเซนเด็นต์ (Descendent) ของโหนดนั้นๆ ดังรูปที่ 2.14 เป็นตารางแสดงลำดับการ ทราเวิร์สทรีแบบพรีออเดอร์จากรูปที่ 2.13 ซึ่งจะเห็นได้ว่าการทราเวิร์สเข้าไปในทรีจะเริ่มจากโหนดรากของทรีก่อน จากนั้นจะเริ่มทราเวิร์สไปยังไชด์โหนดของราก ซึ่งในที่นี้คือโหนด “SELECT” โดยจะทำการเป็นเวิร์สไปยังโหนดไชด์จนกว่าจะเจอโหนดที่เป็นเอ็กซ์เทอร์นอล ซึ่งในที่นี้ก็คือโหนด “Student_id” “,” “Fname” “,” และ“Lname”

เนื่องจากโหนด “Lname” เป็นโหนดประเภทเอ็กซ์เทอร์นอล ซึ่งไม่มีโหนดไชด์ของตัวเองแล้ว อัลกอริทึมจึงทำการทราเวิร์สไปยังอินเทอร์นอลของโหนด ซึ่งได้แก่โหนด “FROM” และเริ่มทำการทราเวิร์สแบบพรีออเดอร์ต่อไป ทำเช่นนี้ต่อไปเรื่อย จนสิ้นสุด ตามตารางดังรูปที่ 2.14 เห็นได้ว่าการทำ พรีออเดอร์ทรีทราเวิร์สซัล มีลักษณะเหมือนการอ่านปรกติของมนุษย์ จากการคำนวณจากคอมพิวเตอร์ พรีออเดอร์ทรีทราเวิร์สซัล เป็นวิธีการที่เหมาะสมที่สุด