บทที่ 2

ทฤษฎีที่เกี่ยวข้อง

2.1 เอ็กซ์เอ็มแอล [6]

เอกสารเอ็กซ์เอ็มแอล สามารถจัดเก็บได้ทั้งรูปแบบไฟล์ข้อความ (Text File), XML Repository หรือ รูปแบบของฐานข้อมูลมีเหตุผลหลัก 2 ประการซึ่งทำให้องค์กรต่างๆ เสนอให้เก็บข้อมูลเอ็กซ์เอ็มแอล ไว้ในฐานข้อมูล คือ

2.1.1 การจัดเก็บข้อมูลเอ็กซ์เอ็มแอล

การจัดเก็บข้อมูลเอ็กซ์เอ็มแอลขนาดใหญ่เป็นปัญหาด้านฐานข้อมูลเอ็กซ์เอ็มแอล เป็นข้อมูลเหมือนกับข้อมูลทั่วไป แตกต่างเพียงรูปแบบเท่านั้น เหตุผลเดียวกับการเก็บข้อมูลเชิงสัมพันธ์ (Relational Data) ลงในฐานข้อมูล เพราะฐานข้อมูล ช่วยในเรื่องของประสิทธิภาพในการสืบค้นและค้นคืนข้อมูล รองรับการจัดเก็บเพื่อความถาวร การเก็บสำรองข้อมูล และการกู้คืนข้อมูล รองรับการจัดการธุรกรรม ประสิทธิภาพในการประมวลผล ลการปรับขนาดได้ (Scalability)

2.1.2 การบูรณาการข้อมูล

โดยการเก็บข้อมูลเชิงสัมพันธ์และเอกสารเอ็กซ์เอ็มแอล เข้าไว้ด้วยกันทำให้สามารถรวมข้อมูลเอ็กซ์เอ็มแอล เข้ากับข้อมูลเชิงสัมพันธ์ที่มีอยู่เดิม และสามารถเขียน ภาษาสำหรับการสอบถามเอสคิวแอลร่วมกับ เอ็กซ์พาร์ท (Xpath) หรือเอ็กซ์คิวรี ในการสอบถามข้อมูลเดียวกัน นอกจากนี้ยังสามารถนำเสนอข้อมูล เชิงสัมพันธ์ในรูปแบบเอ็กซ์เอ็มแอล และทำในทางกลับกันได้ด้วย ด้วยการบูรณาการข้อมูลนี้ ฐานข้อมูลสามารถรองรับเว็บแอ็พพลิเคชัน (Web Application) เอสโอเอ (SOA) และเว็บเซอร์วิส (Web Services) ได้ดียิ่งขึ้น

2.2 ฐานข้อมูลเอ็กซ์เอ็มแอล

ฐานข้อมูลสำหรับเก็บข้อมูลเอ็กซ์เอ็มแอล มี 2 ประเภท คือ

2.2.1 ฐานข้อมูลแบบ XML-enabled

ฐานข้อมูลแบบ XML-enable จะใช้รูปแบบของระดับจัดการฐานข้อมูลเชิงสัมพันธ์เป็นรูปแบบหลักในการจัดเก็บข้อมูล ดังนั้น จำเป็นต้องมีการแปลงข้อมูลระหว่างข้อมูลเอ็กซ์เอ็มแอล ซึ่งมีการเก็บข้อมูลเป็นลำดับชั้น และข้อมูลเชิงสัมพันธ์ หรือต้องจัดเก็บข้อมูลเอ็กซ์เอ็มแอล ในรูปแบบของ CLOB วิธีการนี้ถูกมองว่าเป็นเทคโนโลยีที่ล้าสมัย แต่ยังคงมีการใช้อยู่ ซึ่งการเก็บเอกสารเอ็กซ์เอ็มแอล ลงฐานข้อมูลด้วยวิธีเก็บข้อมูลในรูปแบบของ CLOB และ Varchar การเก็บเอกสารเอ็กซ์เอ็มแอลเป็น String เมื่อต้องการค้นคืนบาง อีลีเม้น (Element) ของเอกสารเอ็กซ์เอ็มแอล จะต้องเขียนโปรแกรมนำ String นั้นมาประมวลผลให้ได้ผลลัพธ์ตามที่ต้องการ ซึ่งวิธีนี้ จะทำได้ยาก และไม่ยืดหยุ่น

อีกทางเลือกหนึ่งสำหรับ ฐานข้อมูลแบบ XML-enabled เรียกว่า Shredding หรือ Decomposition คือการแบ่งเอกสารเอ็กซ์เอ็มแอล ที่ต้องการจัดเก็บออกเป็นส่วนย่อย ๆ และแยกไปจัดเก็บในแต่ละคอลัมภ์ในตาราง การใช้วิธีนี้ทำให้โครงสร้างแบบลำดับชั้นของเอกสารเอ็กซ์เอ็มแอล ถูกเปลี่ยนแปลงให้จัดเก็บอยู่ในรูปแบบข้อมูลเชิงสัมพันธ์ ซึ่งไม่มีความยืดหยุ่นเช่นกัน เพราะการเปลี่ยนแปลงต่างๆ ที่ทำกับเอกสารเอ็กซ์เอ็มแอล ซึ่งเก็บไว้ในตารางนั้นเป็นไปได้ยากและอาจต้องมีการสร้างตาราใหม่เพิ่มขึ้นมากมายในกรณีที่มีการเพิ่มข้อมูลใหม่เข้าไปยังเอกสารเอ็กซ์เอ็มแอล อีกทั้งวิธีการนี้ยังมีผล ทำให้การทำงานไม่มีประสิทธิภาพ เพราะหากต้องการได้เอกสารเอ็กซ์เอ็มแอลเดิม (ก่อนการแปลงให้จัดเก็บอยู่ในรูปแบบข้อมูลเชิงสัมพันธ์) กลับคืนมา จำเป็นต้องประมวลผลคำสั่งภาษาสำหรับการสอบถามเอสคิวแอล ซึ่งจะสิ้นเปลืองทรัพยากรของระบบมาก เพราะต้องเชื่อมต่อความสัมพันธ์ของหลายตารางเข้าด้วยกัน

2.2.2 ฐานข้อมูลแบบ Native XML

ใช้รูปแบบในการเก็บและประมวลผลข้อมูลแบบลำดับชั้น สำหรับข้อมูลเอ็กซ์เอ็มแอล โดยไม่มีการแปลงข้อมูลให้จัดเก็บในรูปแบบข้อมูลเชิงสัมพันธ์ และไม่ได้จัดเก็บเอกสารเอ็กซ์เอ็มแอล ไว้ในรูปแบบที่เหมือนกับการจัดเก็บรูปภาพ เราสามารถใช้คำสั่งเอ็กซ์พาร์ท หรือเอ็กซ์คิวรี ในการประมวลผลกับเอกสารเอ็กซ์เอ็มแอลได้โดยตรง ซึ่งเป็นเหตุผลที่เราเรียก ฐานข้อมูลประเภทนี้ว่า ฐานข้อมูลแบบ “Native” XML

2.3 ฐานข้อมูลเชิงเวลา

ฐานข้อมูลเชิงเวลาเป็นฐานข้อมูลที่เก็บข้อมูลที่มีการเปลี่ยนแปลงตามกาลเวลา (Time-varying) เนื่องจากสิ่งของบางอย่างไม่มีการเปลี่ยนแปลงแต่ความสามารถในการวัดเปลี่ยนแปลงไป เช่น ความสูงของระดับน้ำทะเลในเมื่อ 1 ปีที่แล้วกับปัจจุบันไม่เท่ากัน

ตัวอย่างฐานข้อมูลเชิงเวลา

เป็นเรื่องของข้อมูลเกี่ยวกับการปรับเปลี่ยนตำแหน่งและเงินเดือนของพนักงานในบริษัท ปัญหาคือหากต้องการทราบของมูลการเปลี่ยนแปลงเงินเดือน หรือข้อมูลการเปลี่ยนแปลงตำแหน่งของพนักในบริษัท เพื่อนำมาใช้ในการโยกย้ายหรือปรับขึ้นเงินเดือนให้แก่พนักงาน โดยมีเงื่อนไขว่าพนักงานที่จะสามารถขึ้นเงินเดือนได้ต้องไม่เคยมีการปรับขึ้นเงินเดือนมาก่อนภายใน 1 ปี หากไม่มีการเก็บข้อมูลเรื่องของเวลาไว้ด้วย จะทำให้ไม่สามารถทราบช่วงของเวลาที่มีการเปลี่ยนแปลงเงินเดือน หรือ ช่วงเวลาที่มีการเปลี่ยนแปลงตำแหน่ง ดังแสดงให้เห็นดังตารางด้านล่างนี้

ตารางที่ 2.1 ข้อมูลประวัติการทำงานของพนักงานที่ไม่มีข้อมูลเรื่องของเวลา

EMPNO	SALARY	TITLE	DEPTNO
1001	60000	Engineer	D01
1001	70000	Engineer	D01
1001	70000	Sr Engineer	D02
1001	70000	Tech Leader	D02
1002	45000	Engineer	D01
1002	45700	Engineer	D01
1002	60000	Tech Leader	D02
1003	46000	Engineer	D01
1003	60000	Engineer	D01
1003	70000	Sr Engineer	D02
1003	75000	Manager	D02
1004	60000	Engineer	D01
1004	70000	Engineer	D01

ตารางที่ 2.1 ข้อมูลประวัติการทำงานของพนักงานที่ไม่มีข้อมูลเรื่องของเวลา (ต่อ)

EMPNO	**SALARY**	**TITLE**	**DEPTNO**
1005	70000	Sr Engineer	D02
1005	70000	Tech Leader	D02

ดังนั้นหากต้องการให้สามารถทราบระยะเวลาในการปรับเปลี่ยนเงินเดือน หรือตำแหน่งงานของพนักงาน ต้องเพิ่มแอตทริบิวต์ TSTART และ แอตทริบิวต์ TEND เพื่อให้สามารถทราบเวลาที่มีการเปลี่ยนแปลงของข้อมูลเงินเดือน หรือตำแหน่งของพนักงาน ดังที่แสดงให้เห็นในตารางด้านล่างนี้

ตารางที่ 2.2 ข้อมูลประวัติการทำงานของพนักงาน

EMPNO	SALARY	TITLE	DEPTNO	TSTART	TEND
1001	60000	Engineer	D01	1995-01-01	1995-05-31
1001	70000	Engineer	D01	1995-06-01	1995-09-30
1001	70000	Sr Engineer	D02	1995-10-01	1996-01-31
1001	70000	Tech Leader	D02	1996-02-01	1996-12-31
1002	45000	Engineer	D01	1995-01-01	1995-05-31
1002	45700	Engineer	D01	1995-06-01	1995-10-31
1002	60000	Tech Leader	D02	1995-11-01	1996-12-31
1003	46000	Engineer	D01	1995-01-01	1995-06-30
1003	60000	Engineer	D01	1995-06-01	1995-09-30
1003	70000	Sr Engineer	D02	1995-10-01	1996-01-31
1003	75000	Manager	D02	1996-02-01	1996-12-31
1004	60000	Engineer	D01	1995-01-01	1995-05-31
1004	70000	Engineer	D01	1995-06-01	1995-11-30
1005	70000	Sr Engineer	D02	1995-12-01	1996-06-30
1005	70000	Tech Leader	D02	1996-07-01	1996-12-31

จากตารางที่ 2.2 หากผู้ใช้ต้องการสอบถามข้อมูลเงินเดือนของพนักงานรหัส “1001” โดยใช้ภาษาสำหรับการสอบถามเอสคิวเอล (SQL Query) เพื่อสอบถามข้อมูล จะใช้คำสั่งต่อไปนี้

```
SELECT EMPNO, SALARY, TSTART, END
FROM EMPLOYEE_HISTORY
WHERE EMPNO = 1001
```

โดยผลลัพธ์ที่ได้จากสอบถามข้อมูล จากคำสั่งข้างต้นจะทำให้ผู้ใช้งานได้ข้อมูลดังนี้

ตารางที่ 2.3 ผลลัพธ์จากการสอบถามข้อมูลเงินเดือนพนักงาน 1001

EMPNO	SALARY	TSTART	TEND
1001	60000	1995-01-01	1995-05-31
1001	70000	1995-06-01	1995-09-30
1001	70000	1995-10-01	1996-01-31
1001	70000	1996-02-01	1996-12-31

ซึ่งในมุมมองของผู้ใช้งาน ผลลัพธ์ของข้อมูลที่ได้จากการสอบถามดังตารางที่ 6 ทำให้ผู้งานเกิดความสับสนและยากต่อการนำใช้งาน ซึ่งโดยปกติข้อมูลที่ได้จากการสอบถามบนฐานข้อมูลเชิงเวลาควรมีการรวมตัวของแถวข้อมูลด้วย เพื่อให้ผู้ใช้งานได้ข้อมูลที่ง่ายต่อการใช้งาน

ทั้งนี้สำหรับการสอบถามข้อมูลเงินเดือนของพนักงานรหัส “1001” นั้น ควรมีผลลัพธ์ดังแสดงในตารางที่ 7 ซึ่งจะเห็นได้ว่ามีการรวมตัวของแถวข้อมูลตามช่วงเวลา บนแอตทริบิวต์ (Attribute) EMPNO และ SALARY เมื่อ EMPNO เป็นคีย์หลัก (Primary key) ของฐานข้อมูลเชิงเวลา และแอตทริบิวต์ SALARY ซึ่งเป็นแอตทริบิวต์ที่อาจมีค่าแตกต่างกันในเวลาที่แตกต่างกัน และสามารถสังเกตุได้ว่าช่วงเวลาที่พนักงานคนดังกล่าวมีเงินเดือนเท่ากัน จะถูกรวมกลุ่มกันเป็นค่าเดียว เช่น 70000 ซึ่งอยู่ในช่วงเวลาตั้งแต่ 1996-06-01 – 1996-09-30

ตารางที่ 2.4 ผลลัพธ์การสอบถามข้อมูลเงินเดือนพนักงาน 1001 ในกรณีมีการรวมตัวของแถวข้อมูล

EMPNO	SALARY	TSTART	TEND
1001	60000	1995-01-01	1996-05-31
1001	70000	1996-06-01	1996-09-30

จากการสอบถามข้อมูลที่ผ่านมา หากต้องการสอบถามข้อมูลผ่านภาษาสำหรับการสอบถามเอสคิวแอล เพื่อให้ได้ข้อมูลดังตารางที่ 2.4 สามารถสอบถามข้อมูลผ่านภาษาสำหรับการสอบถามเอสคิวแอล ได้จากคำสั่งดังต่อไปนี้

```
WITH Temp(Salary, TSTART, END) AS
(SELECT SALARY, TSTART, END
FROM EMPLOYEE_HISTORY
WHERE EMPNO = 1001 )
SELECT DISTINCT F.Salary, F.TSTART, F.TEND
FROM Temp AS F, Temp AS L
WHERE F.TSTART < L.TEND AND F.Salary = L.Salary
AND NOT EXISTS
    (SELECT * FROM Temp AS M
WHERE M.Salary = F.Salary
AND F.TSTART < M. TSTART AND M.TSTART < L.TEND
AND NOT EXISTS
    (SELECT * FROM Temp AS T1
    WHERE T1.Salary = F.Salary
    AND T1. TSTART < M. TSTART AND M.TSTART <= T1.TEND)
    )
AND NOT EXISTS

    (SELECT * FROM Temp AS T2
    WHERE T2.Salary = F.Salary
```

```
AND
((T2. TSTART < F. TSTART AND F.TSTART <= T2.TEND)
OR
(T2.TSTART < L.TEND AND L.TEND < T2.TEND))
```

จากการสอบถามที่ยกตัวอย่างไปแล้วนั้น แสดงให้เห็นถึงความซับซ้อนและความยุ่งยากในการสอบถามข้อมูล จากฐานข้อมูลเชิงเวลาอีกทั้งในการรวมแถวของข้อมูลจะต้องทำประมวลผลแบบรีเคอร์ซีพ เพื่อให้ได้ผลลัพธ์ที่ถูกต้อง ซึ่งอาจทำให้ต้องใช้เวลานานในการประมวลผล

นอกจากนั้น การสอบถามข้อมูลบนฐานข้อมูลเชิงเวลายังมีความซับซ้อนในการใช้งานเพื่อให้ได้ข้อมูลที่ถูกต้องตามที่ผู้ใช้งานต้องการ อาจต้องใช้ฟังก์ชัน (Function) อื่นนอกเหนือจากการรวมตัวกันของข้อมูล เช่น Overlap Since until เป็นต้น

Modifying Valid-Time State Table

ความยากของการจัดการข้อมูลไม่ได้มีแค่การสอบถามข้อมูลเท่านั้น แต่การเปลี่ยนแปลง (Modify) ข้อมูลก็ยากด้วย

ตัวอย่าง การเพิ่มข้อมูล หากต้องการเพิ่มข้อมูลการเปลี่ยนเงินเดือนที่เริ่มเป็นจริงตั้งแต่วันนี้ สามารถทำได้โดยใช้คำสั่งดังต่อไปนี้

INSERT EMPLOYEE_HISTORY VALUES(‘1005’, 75000, ‘Tech Leader’, ‘D02’, CURRENT_DATE, DATE ‘9999-12-31’)

ซึ่งการกำหนดค่า TEND เท่ากับ 9999-12-31 หรือ NOW หมายถึงปัจจุบันยังได้รับเงินเดือน และอยู่ในตำแหน่ง นั้นอยู่ ดังที่ได้แสดงในตารางที่ 2.5

ตารางที่ 2.5 ผลลัพธ์การเพิ่มข้อมูลประวัติเงินเดือนของพนักงาน

EMPNO	SALARY	TITLE	DEPTNO	TSTART	TEND
1001	60000	Engineer	D01	1995-01-01	1995-05-31
1001	70000	Engineer	D01	1995-06-01	1995-09-30
1001	70000	Sr Engineer	D02	1995-10-01	1996-01-31
1001	70000	Tech Leader	D02	1996-02-01	1996-12-31
1002	45000	Engineer	D01	1995-01-01	1995-05-31
1002	45700	Engineer	D01	1995-06-01	1995-10-31
1002	60000	Tech Leader	D02	1995-11-01	1996-12-31
1003	46000	Engineer	D01	1995-01-01	1995-06-30
1003	60000	Engineer	D01	1995-06-01	1995-09-30
1003	70000	Sr Engineer	D02	1995-10-01	1996-01-31
1003	75000	Manager	D02	1996-02-01	1996-12-31
1004	60000	Engineer	D01	1995-01-01	1995-05-31
1004	70000	Engineer	D01	1995-06-01	1995-11-30
1005	70000	Sr Engineer	D02	1995-12-01	1996-06-30
1005	70000	Tech Leader	D02	1996-07-01	1996-12-31
1005	75000	Tech Leader	D02	1996-12-31	9999-12-31

ตัวอย่าง การลบข้อมูล หากต้องการลบข้อมูลการเปลี่ยนเงินเดือนของพนักงานรหัส ‘1005’ ที่มีเงินเดือน 75000 บาท และมีตำแหน่งเป็น ‘Tech Leader” ไม่สามารถทำได้โดยใช้คำสั่งดังต่อไปนี้

```
DELETE FORM EMPLOYEE_HISTORY WHERE EMPNO = 1005
```

ซึ่งในเรื่องของการแก้ไขข้อมูล (Update) หรือ ลบข้อมูล (Delete) จะมี 2 มุมมองคือ

- General Scenario คือ สามารถ เพิ่ม แก้ไข ลบ ข้อมูลได้ทุกช่วงเวลา
- Restricted Scenario คือ สามารถ เพิ่ม แก้ไข ลบ ได้เฉพาะปัจจุบันเท่านั้น

Current Delete

ตารางที่ 2.6 ตารางประวัติข้อมูลพนักงาน

EMPNO	SALARY	TITLE	DEPTNO	TSTART	TEND
1001	60000	Engineer	D01	1995-01-01	1995-05-31
1001	70000	Engineer	D01	1995-06-01	1995-09-30
1001	70000	Sr Engineer	D02	1995-10-01	1996-01-31
1001	70000	Tech Leader	D02	1996-02-01	1996-12-31
1002	45000	Engineer	D01	1995-01-01	1995-05-31
1002	45700	Engineer	D01	1995-06-01	1995-10-31
1002	60000	Tech Leader	D02	1995-11-01	1996-12-31
1003	46000	Engineer	D01	1995-01-01	1995-06-30
1003	60000	Engineer	D01	1995-06-01	1995-09-30
1003	70000	Sr Engineer	D02	1995-10-01	1996-01-31
1003	75000	Manager	D02	1996-02-01	1996-12-31
1004	60000	Engineer	D01	1995-01-01	1995-05-31
1004	70000	Engineer	D01	1995-06-01	1995-11-30
1005	70000	Sr Engineer	D02	1995-12-01	1996-06-30
1005	70000	Tech Leader	D02	1996-07-01	1996-12-31
1005	75000	Tech Leader	D02	1996-12-31	9999-12-31

จากตารางข้างต้น หากต้องการลบข้อมูลการเปลี่ยนเงินเดือนของพนักงานรหัส ‘1005’ ที่มีเงินเดือน 75000 บาท และมีตำแหน่งเป็น ‘Tech Leader โดยสมมุติว่าวันที่ปัจจุบันคือ 1996-12-31 ถ้าต้องการลบข้อมูลระเบียนดังกล่าวออก จะต้องทำการแก้ไข TEND ของข้อมูลพนักงาน รหัส ‘1005’ แถวที่ 15 เป็นวันที่ปัจจุบัน คือ 1996-12-31 และลบแถวข้อมูลแถวที่ 16 ทิ้ง ได้ดังคำสั่งดังต่อไปนี้

```
UPDATE EMPLOYEE_HISTORY SET TEND = CURRENT_DATE
WHERE EMPNO=1005 AND TEND >= CURRENT_DATE
AND TSTART < CURRENT_DATE

DELETE FORM EMPLOYEE_HISTORY WHERE EMPNO = 1005
AND TSTART > CURRENT_DATE
```

2.4 การทำเหมืองข้อมูล

เป็นการค้นหาความรู้ในฐานข้อมูลขนาดใหญ่ ซึ่งเป็นเทคนิคในการค้นหารูปแบบอะไรบางอย่างที่ซ่อนอยู่ในข้อมูลจำนวนมหาศาล ซึ่งไม่สามารถสังเกตเห็นได้เนื่องจากมีปริมาณข้อมูลที่มาก โดยการทำเหมืองข้อมูลจะอาศัยวิธีการทางสถิติ และการเรียนรู้ของคอมพิวเตอร์

ตัวอย่างของการทำเหมืองข้อมูล เช่น ห้างสรรพสินค้าแห่งหนึ่ง พบว่าลูกค้าร้อยละ 90 ที่ซื้อเบียร์ จะซื้อผ้าอ้อมเด็กด้วย ซึ่งอาจจะสงสัยได้ว่าการซื้อสินค้าดังกล่าวเกี่ยวข้องกันได้อย่างไร การซื้อสินค้าดังกล่าวสามารถอธิบายได้ว่าห้างสรรพสินค้าจะทำการบันทึกข้อมูลการซื้อสินค้าของลูกค้า จากการชำระเงินบริเวณแคชเชียร์ที่ลูกค้าชำระเงิน โดยข้อมูลดังกล่าวจะถูกจัดเก็บลงในฐานข้อมูล ที่มีลักษณะคล้ายกับที่แสดงในใบเสร็จรับเงินที่ได้รับจากแคชเชียร์ ซึ่งข้อมูลเหล่านี้จะอยู่บนฐานข้อมูลขนาดใหญ่ โดยสามารถนำมาวิเคราะห์ และค้นหารูปแบบหรือสารสนเทศที่ซ่อนอยู่ที่มีความน่าสนใจ ในกรณีของผ้าอ้อมเด็กกับเบียร์ เมื่อห้างสรรพสินค้าค้นพบความสัมพันธ์ของข้อมูลดังกล่าวและดำเนินการทดสอบทางสถิติ เช่น การหาค่าความเชื่อมั่นแล้ว สามารถนำไปใช้ในการส่งเสริมการขายใหม่ๆ ได้ เช่น การจัดวางสินค้า การจัดกระเช้าของขวัญ การแขวนผ้าอ้อมเด็กไว้ที่จุดขายเบียร์ หรืออาจจะเป็นโปรโมชั่นในการซื้อเบียร์แล้วแถมผ้าอ้อมก็อาจเป็นไปได้

จากตัวอย่างที่กล่าวไปแล้วนั้น เป็นการนำรูปแบบที่ค้นหาได้ มาปรับใช้เป็นกลยุทธที่จะดูดเงินจากลูกค้าทุกรูปแบบ ซึ่งกรณีนี้เหมือนเป็นการแบ่งกลุ่มลูกค้า ทำให้บริษัทสามารถสื่อสารกับกลุ่มเป้าหมายได้ตรงและแม่ยำมากขึ้น แทนที่จะเหวี่ยงแหหาลูกค้าไปเรื่อยๆ แบบไร้ทิศทาง อีกทั้งยังสามารถประหยัดงบประมาณในการประชาสัมพันธ์ นอกจากนี้ในด้านธุรกิจ สามารถนำไปใช้ในการลดความเสี่ยง การจัดการแบ่งกลุ่มลูกค้า การวิจัยทางการตลาดต่างๆ หรืองานที่เกี่ยวกับการเพิ่มภาพลักษณ์ขององค์กร รักษาและเพิ่มจำนวนฐานลูกค้า

2.5 ขั้นตอนวิธี GSP

เป็นขั้นตอนวิธีในการทำเหมืองข้อมูล อีกเทคนิคหนึ่งที่ใช้ในการหาความสัมพันธ์ระหว่างข้อมูลโดยสนใจลำดับการเกิดของข้อมูลด้วย โดยเทคนิคนี้เป็นการหาความสัมพันธ์ของข้อมูลระหว่างรายการเปลี่ยนแปลง ซึ่งทำให้มีเวลาเข้ามาเกี่ยวข้องด้วย รูปแบบลำดับที่ได้จะแสดงว่า ถ้าเกิดเหตุการณ์อะไรหรือพบกลุ่มของข้อมูลใดตามมาภายหลัง ความเป็นไปได้ที่จะเกิดรูปแบบลำดับนั้นๆ มีค่าเท่ากับค่าสนับสนุน เช่น ถ้ารูปแบบลำดับที่ได้เป็น “<(เบียร์, ผ้าอ้อมเด็ก) (นมสด)> ค่าสนับสนุนเท่ากับ 75” หมายความว่า ลูกค้าจำนวน 75 คนซื้อเบียร์ไปพร้อมกับผ้าอ้อมเด็ก ต่อมาภายหลังจะกลับมาซื้อนมสด โดยที่ลูกค้าเหล่านั้นอาจจะซื้อสินค้าอื่นๆ ก่อนนมสดก็ได้เพียงแต่รูปแบบลำดับนั้นอาจมีค่าสนับสนุนขั้นต่ำน้อยกว่าที่กำหนดไว้ โดยขั้นตอนวิธี GSP ถูกพัฒนาต่อยอดมาจากขั้นตอนวิธี AprioriAll ซึ่งขั้นตอนวิธี GSP สามารถทำงานได้เร็วกว่า และสามารถพิจารณาค่าควบคุมของเวลา คือค่าความแตกต่างของเวลาสูงสุดระหว่างรายการเปลี่ยนแปลง และค่าความแตกต่างของเวลาต่ำสุดระหว่างรายการเปลี่ยนแปลง พร้อมทั้งพิจารณาอนุกรมวิธาน เพื่อให้ได้รูปแบบลำดับที่มากขึ้น

ขั้นตอนวิธี GSP สามารถอิมพลีเม้นต์ได้ดังนี้

```
L=1
While (Result_L != NULL)
        Candidate Generate
        Prune
        L=L+1
```

ตัวอย่างการสืบค้นรูปแบบลำดับ GSP

ตารางที่ 2.7 แสดงตัวอย่างตารางข้อมูล ฐานข้อมูลเชิงเวลาสำหรับสืบค้นรูปแบบลำดับ GSP

Tid	Items
10	A,B, C, D
20	B, E
30	A, B, C, E
40	B, E

จากตารางที่ 2.7 หากต้องการสืบค้นรูปแบบลำดับ ความสัมพันธ์ของข้อมูล บนขั้นตอนวิธี GSP โดยกำหนด ค่าสนับสนุนขั้นต่ำ = 2 ค่าความแตกต่างของเวลาสูงสุด = 0 และค่าความแตกต่างของเวลาต่ำสุด = 0 สามารถดำเนินการสืบค้นได้โดยการกำหนดรูปแบบลำดับที่มีความยาวตั้งแต่ 1 พร้อมทั้งเปรียบเทียบรูปแบบลำดับดังกล่าวกับข้อมูลในฐานข้อมูล เพื่อนับจำนวนค่าสนับสนุนของรูปแบบนั้น เมื่อทำการเปรียบเทียบข้อมูลในฐานข้อมูลเสร็จแล้ว หากค่าสนับสนุนที่ได้มีค่าน้อยกว่าค่าสนับสนุนขั้นต่ำ ต้องทำการตัดรูปแบบลำดับนั้นออกไปเพื่อนำรูปแบบลำดับที่สนุบสนุนไปใช้ในการกำหนดรูปแบบลำดับในขั้นถัดไป โดยต้องทำไปจนกระทั้งรูปแบบลำดับที่มีความยาวเท่ากับ k ไม่สามารถนำไปสร้างรูปแบบลำดับต่อไปได้ ดังที่ได้แสดงดังนี้

ตารางที่ 2.8 การกำหนดรูปแบบลำดับ ขนาดความยาวเท่ากับหนึ่ง และการตัดรูปแบบลำดับที่มีค่าสนับสนุนต่ำกว่าค่าสนับสนุนขั้นต่ำ

Itemset	**sup**
{A}	2
{B}	4
{C}	2
{D}	1
{E}	3

ตารางที่ 2.9 การกำหนดรูปแบบลำดับ ขนาดความยาวเท่ากับสอง และการตัดรูปแบบลำดับที่มีค่าสนับสนุนต่ำกว่าค่าสนับสนุนขั้นต่ำ

Itemset	**sup**
{A,B}	2
{A,C}	0
{A,E}	0
{B,C}	2
{B,E}	2
{C,E}	1

ตารางที่ 2.10 ผลลัพธ์รูปแบบลำดับ ขนาดความยาวเท่ากับสามที่มีค่าสนับสนุนต่ำกว่าค่าสนับสนุนขั้นต่ำ

Itemset	**sup**
{B,C,E}	1