# บทที่ 4
# บทสรุปและวิจารณ์

## 4.1 สรุปผลการวิจัย

การค้นพบรูปแบบที่ปรากฏบ่อย หรือ frequent pattern discovery เป็นการค้นหารูปแบบร่วมที่เกิดขึ้นซ้ำๆ ในข้อมูลขนาดใหญ่ เช่น การค้นพบจากฐานข้อมูลทรานแซคชันแผนกซุปเปอร์มาเก็ตของห้างสรรพสินค้าสาขาหาดใหญ่พบว่าลูกค้านิยมซื้อนมผงและชาเขียว ในขณะที่การค้นพบจากทรานแซคชันของสาขานครราชสีมาระบุว่าลูกค้าซื้อนมผงพร้อมกับไข่ไก่ รูปแบบการซื้อสินค้าดังตัวอย่างนี้จะช่วยสะท้อนถึงพฤติกรรมของผู้บริโภคซึ่งอาจจะแตกต่างกันในแต่ละภูมิภาค ความรู้เช่นนี้จะเป็นประโยชน์ต่อผู้บริหารในการวางแผนจัดวางชั้นสินค้า รวมไปถึงการวางแผนจัดรายการกระตุ้นยอดขายสินค้า เช่นจากพฤติกรรมการบริโภคของลูกค้าในเมืองหาดใหญ่ ถ้าผู้จัดการห้างสรรพสินค้าทราบว่ากำไรต่อหน่วยของชาเขียวสูงกว่านมผงมาก การตัดสินใจลดราคานมผงให้ต่ำกว่าราคาจำหน่ายของร้านค้าทั่วไป จึงคาดได้ว่าน่าจะดึงดูดลูกค้าที่ต้องการซื้อนมผงเข้ามาซื้อของในห้างสรรพสินค้า และคาดหมายต่อไปได้ว่าลูกค้าที่ซื้อนมผงจะซื้อชาเขียวร่วมด้วย ซึ่งจะส่งผลให้ยอดขายและกำไรโดยรวมของการจำหน่ายนมผงและชาเขียวสูงขึ้น

การค้นพบความสัมพันธ์หรือความเชื่อมโยงของสินค้าที่ถูกซื้อร่วมกันบ่อยนี้ เรียกว่า การทำเหมืองข้อมูลเพื่อค้นหาความสัมพันธ์ (association mining) ซึ่งอาศัยขั้นตอนพื้นฐานที่สำคัญคือ การค้นหารูปแบบที่ปรากฏบ่อย ลักษณะของรูปแบบที่ปรากฏบ่อย (frequent pattern) มีได้หลากหลายประเภท เช่น รูปแบบการซื้อสินค้าของลูกค้าส่วนใหญ่ในห้างสรรพสินค้า, รูปแบบการค้นหาข้อมูลจากเว็บเพจต่างๆ ของผู้ใช้อินเทอร์เน็ต, รูปแบบการจัดโครงสร้างโมเลกุลในสารประกอบโปรตีน เป็นต้น งานวิจัยด้านการทำเหมืองข้อมูลเพื่อค้นหาความสัมพันธ์ ได้รับความสนใจจากนักวิจัยทั่วโลกอย่างกว้างขวาง เนื่องจากใช้ประโยชน์ได้กับการวิเคราะห์ข้อมูลในหลากหลายสาขา มีการพัฒนาเทคนิคต่างๆ ของการค้นหารูปแบบที่ปรากฏบ่อยให้มีประสิทธิภาพสูง แต่จากความก้าวหน้าของเทคโนโลยีอินเทอร์เนต ทำให้รูปแบบของข้อมูลมีลักษณะเปลี่ยนแปลงไปจากข้อมูลที่มีขนาดใหญ่แต่คงตัว (static) กลายเป็นข้อมูลที่มีลักษณะพลวัต (dynamic) เปลี่ยนแปลงได้ทั้งการเพิ่มปริมาณอย่างต่อเนื่องและการเปลี่ยนรูปแบบการกระจายของข้อมูล ข้อมูลที่มีปริมาณเพิ่มได้ไม่จำกัดเช่นนี้เรียกว่า ข้อมูลสตรีม

ในช่วงทศวรรษที่ผ่านมานักวิจัยในสาขาการทำเหมืองข้อมูลเพื่อค้นหาความสัมพันธ์ เริ่มให้ความสนใจกับลักษณะของข้อมูลสตรีม และพยายามปรับปรุงเทคนิคการค้นหารูปแบบที่ปรากฏบ่อยที่ใช้งานได้ดีกับข้อมูลคงตัวให้ทำงานกับข้อมูลสตรีมได้ แต่เทคนิคเหล่านั้นยังมีข้อจำกัดและใช้ได้กับเพียงบางแอพพลิเคชัน ผู้วิจัยจึงได้ศึกษาแนวทางการโปรแกรมเชิงฟังก์ชันด้วยภาษาฮาสเกล (Haskell) และภาษาเออแลง (Erlang) และแนวทางการโปรแกรมเชิงตรรกะด้วยภาษาโปรล็อก (Prolog) เพื่อพัฒนาเทคนิคการค้นหารูปแบบที่ปรากฏบ่อยด้วยภาษาที่เหมาะสมทำงานได้กับหลากหลายแอพพลิเคชัน จาก

ผลการศึกษาในเบื้องต้นพบว่าภาษาทั้งในเชิงฟังก์ชันและเชิงตรรกะ มีลักษณะเด่นบางประการที่เหมือนกันคือเป็นภาษาเชิงประกาศ (declarative) ที่มีรูปแบบการเขียนคำสั่งใกล้เคียงกับซูโดโค้ดที่นิยมใช้ในขั้นตอนการออกแบบโปรแกรม ลักษณะการโปรแกรมเชิงประกาศจะแตกต่างจากการโปรแกรมเชิงกระบวนคำสั่ง ตรงที่รูปแบบคำสั่งจะสั้นกว่าและใช้วิธีการประกาศแพทเทิร์นของอินพุทและเอาท์พุท โดยไม่ต้องระบุรายละเอียดขั้นตอน ข้อแตกต่างนี้จะเห็นได้ชัดเจน ดังตัวอย่างต่อไปนี้ที่แสดงการเขียนโปรแกรมเรียงลำดับข้อมูลแบบ quick sort ด้วยวิธีการโปรแกรมเชิงประกาศ (ด้วยภาษาฮาสเกล ภาษาเออแลง และ ภาษาโปรล็อก) เทียบกับวิธีการโปรแกรมเชิงกระบวนคำสั่ง (ด้วยภาษาซี)

Haskell

```
sort []     = []
sort (x:xs) =  sort [y | y<-xs, y<x ] ++ [x] ++ sort [y | y<-xs, y>=x]
```

Erlang

```
quickSort([ ]) -> [ ];
quickSort([Pivot | T]) ->
                        quickSort([ X || X <- T, X < Pivot ])
                        ++ [Pivot] ++
                        quickSort([ X || X <- T, X >= Pivot ]).
```

Prolog

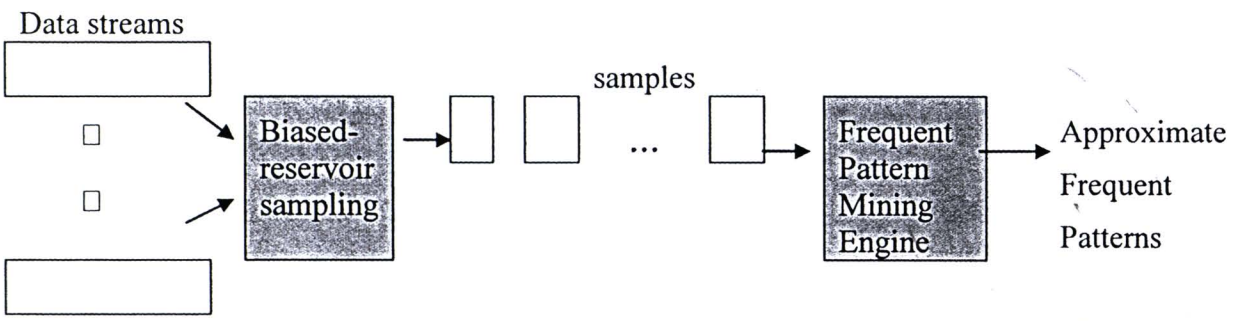
```
qs([],[]).
qs([ X | Xs]) :- part(X, Xs, Littles, Bigs),
                 qs( Littles, Ls),
                 qs( Bigs, Bs),
                 append(Ls, [X| Bs], Ys).
part(_, [], [], []).
part(X, [Y|Xs], [Y|Ls], Bs) :- X>Y, part(X, Xs, Ls, Bs).
part(X, [Y|Xs], Ls, [Y|Bs]) :- X=<Y, part(X, Xs, Ls, Bs).
```

C

```
int partition(int y[], int f, int l);
void quicksort(int x[], int first, int last) {
        int pivIndex = 0;
        if(first < last) {
                        pivIndex = partition(x,first, last);
                        quicksort(x,first,(pivIndex-1));
                        quicksort(x,(pivIndex+1),last);
        }  }
int partition(int y[], int f, int l) {
        int up, down, temp, cc, piv = y[f];
        up = f;
        down = l;
        do {
                while (y[up] <= piv && up < l)  { up++; }
                while (y[down] > piv  )  { down--; }
                if (up < down ) {   temp = y[up];
                                    y[up] = y[down];
                                    y[down] = temp;    }
        } while (down > up);
        temp = piv;
        y[f] = y[down];
        y[down] = piv;
        return down;     }
```

การโปรแกรมเชิงประกาศจะใช้เทคนิคการระบุแพทเทิร์นของข้อมูลและตัวแปร เพื่อกำหนดรูปแบบอินพุทและเอาท์พุท จากนั้นใช้กลไกการทำให้แพทเทิร์นตรงกัน หรือ pattern matching เพื่อกำหนดค่าให้กับตัวแปรและสร้างเอาท์พุท การวนลูปใช้วิธีการเรียกตัวเองซ้ำในลักษณะของ recursive กลไกการทำงานด้วย pattern matching เช่นนี้พบว่าเหมาะสมกับการนำไปใช้ในการพัฒนาโปรแกรมที่มี symbolic computation ค่อนข้างสูงเช่นงานด้านการทำเหมืองข้อมูล

ในโครงการวิจัยการค้นพบโดยประมาณของรูปแบบที่ปรากฏบ่อยในข้อมูลสตรีมนี้ ผู้วิจัยเลือกใช้ภาษาเออแลงเป็นภาษาในการพัฒนาโปรแกรมต้นแบบ เนื่องจากมีรูปแบบการเขียนคำสั่งที่ยืดหยุ่น รองรองการทำงานกับข้อมูลปริมาณมากได้ ดังนั้นจึงมีเหมาะสมที่จะใช้ทำงานกับข้อมูลสตรีม ที่ข้อมูลถูกส่งเข้ามาประมวลผลอย่างต่อเนื่องได้ โดยการค้นพบโดยประมาณกับข้อมูลสตรีมจะให้ผลลัพธ์ในลักษณะของค่าโดยประมาณ แนวคิดการประมวลผลกับสตรีมแสดงได้ดังรูปที่ 4.1




รูปที่ 4.1 การค้นพบรูปแบบโดยประมาณจากข้อมูลสตรีม

จากผลการทดสอบกับข้อมูลรหัสพันธุกรรม พบว่าเทคนิคการค้นพบโดยประมาณนี้สามารถค้นหารูปแบบหรือแพทเทิร์น ได้ประสิทธิภาพใกล้เคียงกับการค้นพบจากข้อมูลทั้งหมด แต่จะใช้เวลาประมวลผลและใช้หน่วยความจำในการเก็บข้อมูลที่น้อยกว่า นอกจากนี้รูปแบบของแพทเทิร์นที่ได้ยังเป็นรูปแบบที่สั้น ทำความเข้าใจและแปลความหมายได้ค่อนข้างง่าย

## 4.2 แนวทางการพัฒนาในอนาคต

จากความสำเร็จของโปรแกรมค้นพบโดยประมาณของรูปแบบที่ปรากฏบ่อยในข้อมูลสตรีม ผู้วิจัยมีแผนงานวิจัยที่จะปรับปรุงเทคนิคการโปรแกรมให้เป็นการประมวลผลแบบคู่ขนาน (parallel processing) เพื่อรองรับกับข้อมูลสตรีมปริมาณมาก โดยจะต้องมีการออกแบบเทคนิค load balancing ของคำสั่งและข้อมูล เพื่อให้เกิดประสิทธิภาพการประมวลผลสูงสุด นอกจากนั้นยังจะต้องออกแบบการรวมผลลัพธ์จากการประมวลผลของแต่ละ processor เพื่อให้สามารถรวบรวมแพทเทิร์ท หรือรูปแบบที่ปรากฏบ่อย และแปลความหมายของแทพเทิร์นเหล่านั้นได้อย่างถูกต้อง