รหัสโครงการ SUT7-705-49-24-48

รายงานการวิจัย

การเพิ่มประสิทธิภาพการประมวลผลข้อคำถามด้วยวิวข้อมูลและโมเดลจากการทำเหมืองข้อมูล

(Query optimization with materialized views and data mining models)

ได้รับทุนอุดหนุนการวิจัยจาก

มหาวิทยาลัยเทคโนโลยีสุรนารี

ผลงานวิจัยเป็นความรับผิดชอบของหัวหน้าโครงการวิจัยแต่เพียงผู้เดียว

รหัสโครงการ **SUT7-705-49-24-48**

รายงานการวิจัย

การเพิ่มประสิทธิภาพการประมวลผลข้อคำถามด้วยวิวข้อมูลและโมเดลจากการทำเหมืองข้อมูล

(Query optimization with materialized views and data mining models)

ผู้วิจัย

หัวหน้าโครงการ

รองศาสตราจารย์ ดร.นิตยา เกิดประสพ

สาขาวิชาวิศวกรรมคอมพิวเตอร์

สำนักวิชาวิศวกรรมศาสตร์

ผู้วิจัยร่วม

รองศาสตราจารย์ ดร.กิตติศักดิ์ เกิดประสพ

ได้รับทุนอุดหนุนการวิจัยจากมหาวิทยาลัยเทคโนโลยีสุรนารี ปีงบประมาณ พ.ศ. 2549 และ 2550

ผลงานวิจัยเป็นความรับผิดชอบของหัวหน้าโครงการวิจัยแต่เพียงผู้เดียว

พฤศจิกายน 2553

กิตติกรรมประกาศ

ผู้วิจัยขอขอบคุณมหาวิทยาลัยเทคโนโลยีสุรนารีและสำนักงานคณะกรรมการวิจัยแห่งชาติ ที่ได้จัดสรรงบประมาณให้ในปีงบประมาณ 2549 และ 2550 เพื่อสนับสนุนโครงการวิจัยนี้ ขอขอบคุณผู้ทรงคุณวุฒิที่ได้เสียสละเวลาทำหน้าที่ตรวจข้อเสนอโครงการ และตรวจร่างรายงานการวิจัยฉบับสมบูรณ์ งานวิจัยนี้สำเร็จได้อย่างดีด้วยการมีส่วนร่วมจากนักศึกษาทั้งในระดับปริญญาบัณฑิตและบัณฑิตศึกษาสาขาวิชาวิศวกรรมคอมพิวเตอร์ ที่ได้ทำหน้าที่เป็นผู้ช่วยวิจัยในโครงการวิจัยนี้ โดยเฉพาะนายอภิชัย ฤทธิ์ธงชัยเลิศ และนายจักรพันธ์ มหาวันตัง นักศึกษาปริญญาโทสาขาวิชาวิศวกรรมคอมพิวเตอร์ รุ่นที่ 2 ที่ได้นำแนวคิดของโครงการวิจัยนี้ไปประยุกต์เป็นหัวข้อวิทยานิพนธ์

บทคัดย่อภาษาไทย

การสอบถามข้อมูลจากฐานข้อมูลเป็นงานปกติของระบบฐานข้อมูลส่วนใหญ่ การตั้งข้อคำถามมีวัตถุประสงค์ที่จะค้นหาคำตอบจากฐานข้อมูล ซึ่งในระบบฐานข้อมูลส่วนใหญ่จะประมวลผลโดยไม่มีการปรับเปลี่ยนเงื่อนไขของข้อคำถาม ดังนั้นการได้คำตอบที่ตรงตามความต้องการของผู้ใช้จึงเป็นภาระของผู้ใช้ฐานข้อมูลที่จะต้องตั้งข้อคำถามอย่างถูกต้องและระบุเงื่อนไขอย่างครบถ้วน ซึ่งเป็นสิ่งที่ทำได้ยากในกรณีที่ผู้ใช้ไม่มีความรู้เกี่ยวกับเนื้อหา และโครงสร้างความสัมพันธ์ทั้งหมดของฐานข้อมูล ระบบฐานข้อมูลที่ชาญฉลาดจึงเป็นแนวทางใหม่ของการพัฒนาเทคโนโลยีฐานข้อมูล ระบบฐานข้อมูลที่ชาญฉลาดจะมีส่วนที่ช่วยวิเคราะห์ข้อคำถามของผู้ใช้ และแปลงข้อคำถามนั้นให้อยู่ในรูปแบบที่เหมาะสมมากขึ้นต่อการค้นหาคำตอบที่ตรงกับความต้องการ การปรับปรุงข้อคำถามของผู้ใช้กระทำได้ด้วยการใช้ฐานความรู้ที่เรียนรู้ได้จากข้อมูล และใช้ความรู้นั้นร่วมกับวิวข้อมูลเปลี่ยนโครงสร้างของข้อคำถามให้มีประสิทธิภาพมากขึ้นในการประมวลผลเพื่อค้นหาคำตอบ งานวิจัยนี้ได้นำเสนอวิธีการสร้างฐานความรู้จากข้อมูลที่มีอยู่ และได้ออกแบบวิธีการใช้ความรู้ร่วมกับวิวข้อมูลเพื่อปรับปรุงข้อคำถาม จากผลการทดสอบวิธีการที่ได้นำเสนอขึ้นกับฐานข้อมูลขนาดกลางพบว่าข้อคำถามที่ได้รับการปรับปรุงรูปแบบแล้วสามารถประมวลผลได้เร็วขึ้น

บทคัดย่อภาษาอังกฤษ

Querying a database is a common task for most database systems. To query a database is to find some answers from stored data. Traditional database systems return exactly what is being asked. This is a method of direct query answering and a user is required to construct a query intelligently and properly. To remove the burden of intelligence from the database users, the concept of intelligent or cooperative query answering has emerged. The process of intelligent query answering consists of analyzing the intent of query, rewriting the query based on the intention and other kinds of knowledge, and providing answers in an intelligent way. Intelligent answers could be generalized, neighborhood or associated information relevant to the query. This concept is based on the assumption that some users might not have a clear idea of the database content and schema. Therefore, it is difficult to pose queries correctly to get some useful answers. Producing answers effectively depends largely on users' knowledge about the query language and the database schema. Knowledge, either intentional or extensional, is the key ingredient of intelligence. In order to improve effectiveness and convenience of querying databases, we design a systematic way to analyze user's request and revise the query with virtual mining and materialized views. Virtual mining views are data mining rules discovered from the database. Materialized views are pre-computed data. This research presents the knowledge acquisition method, its implementation, and a systematic method of rewriting query with virtual mining views and materialized views. We perform preliminary efficiency tests of the proposed system. The experimental results demonstrate the effectiveness of our system in answering queries sharing the same pattern as the available knowledge.

สารบัญ

หน้า

สารบัญตาราง

หน้า

สารบัญภาพ

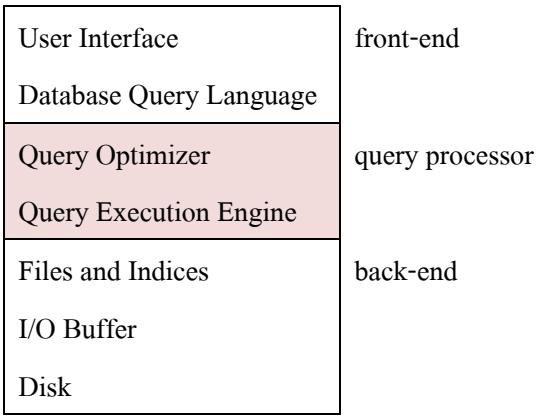
หน้า

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มาของปัญหาการวิจัย

ระบบฐานข้อมูล (database system) ประกอบด้วยสองส่วนประกอบหลัก คือฐานข้อมูล (database) ที่รวบรวมข้อมูลที่สำคัญไว้ในแหล่งเดียวกันด้วยรูปแบบที่เป็นมาตรฐานเช่นตารางข้อมูล และระบบจัดการฐานข้อมูล (Database Management System -- DBMS) ที่เป็นโปรแกรมระบบขนาดใหญ่ทำหน้าที่บันทึกและอ่านข้อมูล ตรวจสอบสิทธิในการเรียกดูข้อมูลของผู้ใช้ ตรวจสอบความถูกต้องและการคงสภาพของข้อมูล สำรองข้อมูล ตลอดจนถึงการกู้คืนข้อมูลเมื่อมีเหตุขัดข้องกับระบบคอมพิวเตอร์หรือระบบฐานข้อมูล

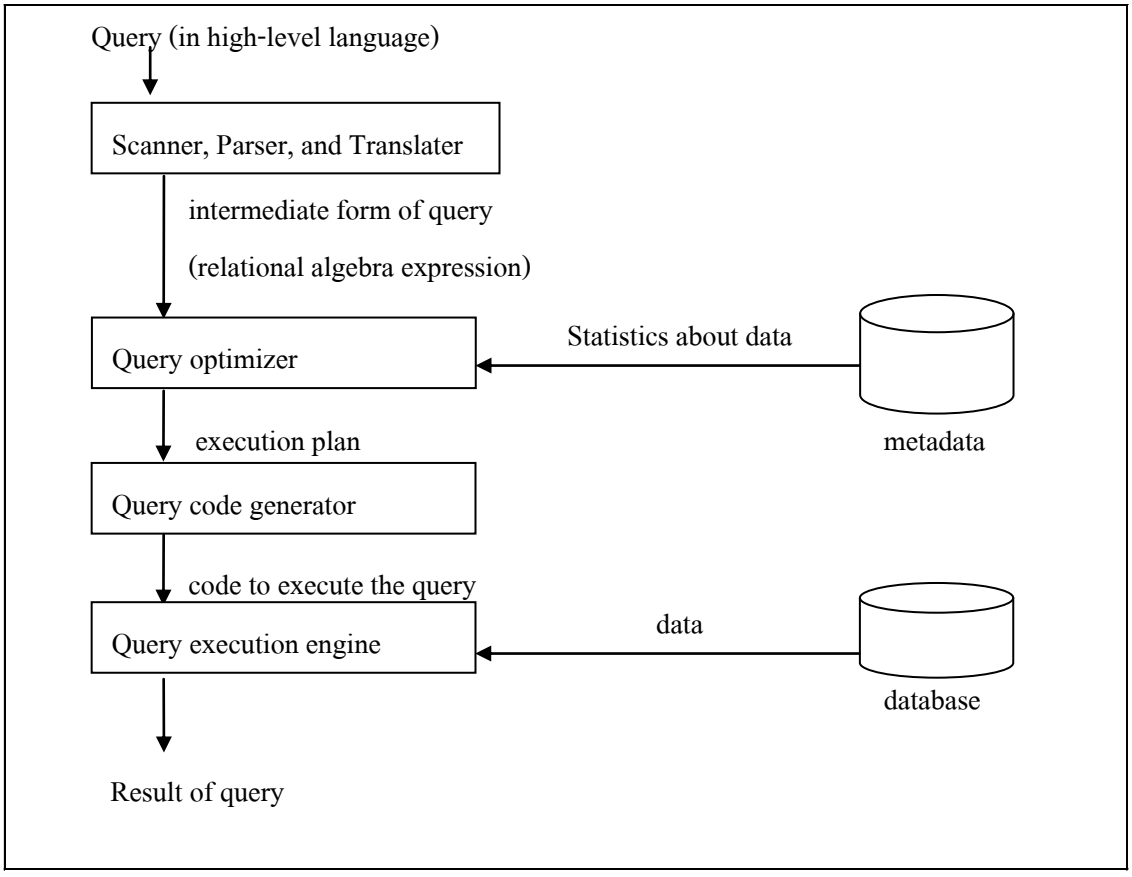
ส่วนประกอบที่สำคัญส่วนหนึ่งของระบบจัดการฐานข้อมูล คือ *ส่วนประมวลผลข้อคำถาม* (query processor) ที่ทำหน้าที่รับข้อคำถาม (query) จากผู้ใช้ ตรวจสอบความถูกต้องของข้อคำถามและแปลความหมาย จนกระทั่งถึงขั้นตอนประมวลผลและแสดงข้อมูลที่เป็นคำตอบของข้อคำถาม ในระบบจัดการฐานข้อมูลขนาดใหญ่ เช่น Oracle, Microsoft SQL Server, IBM DB2 บริษัทผู้สร้างระบบจัดการฐานข้อมูลมักจะผนวกส่วนที่เรียกว่า *ส่วนเพิ่มประสิทธิภาพข้อคำถาม* (query optimizer) ไว้ในส่วนประมวลผลข้อคำถามเพื่อทำหน้าที่ปรับปรุงรูปแบบข้อคำถามให้สามารถประมวลผลได้เร็วขึ้น โดยที่ยังคงให้คำตอบที่ถูกต้องได้เช่นเดิม ส่วนประมวลผลข้อคำถามและส่วนเพิ่มประสิทธิภาพข้อคำถาม (โครงสร้างหลักแสดงดังรูปที่ 1.1) จะทำหน้าที่เป็นส่วนเชื่อมต่อระหว่างส่วนหน้า (front end) ที่ทำหน้าที่ติดต่อกับผู้ใช้ และส่วนหลัง (back end) ที่เป็นโปรแกรมระบบ ที่สำคัญได้แก่ระบบแฟ้มข้อมูลและระบบปฏิบัติการที่จัดการกับการอ่านและบันทึกข้อมูลในดิสก์



รูปที่ 1.1 โครงสร้างของระบบจัดการฐานข้อมูลส่วนประมวลผลข้อคำถาม

โครงสร้างโดยละเอียด (รูปที่ 1.2) ของส่วนประมวลผลข้อคำถาม (query processor) ประกอบด้วย

- Scanner, Parser, and Translator
  ทำหน้าที่แปลความหมายของข้อคำถามในรูปแบบของคำสั่ง SQL รวมถึงตรวจสอบความถูกต้องของข้อคำถาม
- Query optimizer
  ทำหน้าที่ปรับปรุงรูปแบบคำสั่ง SQL ให้มีรูปแบบเหมาะสมยิ่งขึ้นกับการประมวลผล
- Query code generator
  ทำหน้าที่เปลี่ยนคำสั่ง SQL ที่ปรับปรุงแล้วให้อยู่ในรูปแบบคำสั่งที่พร้อมจะประมวลผล
- Query execution engine
  ทำหน้าที่ประมวลผลและแสดงผลลัพธ์ที่ได้จากการประมวลผล



รูปที่ 1.2 โครงสร้างและขั้นตอนการทำงานของส่วนประมวลผลข้อคำถาม

จากโครงสร้างของส่วนประมวลผลข้อคำถามจะเห็นได้ว่าส่วน Scanner, Parser, and Translater ส่วน Query code generator และส่วน Query execution engine จะทำงานร่วมกันเพื่อประมวลผลให้ได้คำตอบที่ถูกต้องต่อข้อคำถาม (ที่มักจะอยู่ในรูปแบบคำสั่ง Structured Query Language หรือ SQL) ของผู้ใช้ ในขณะที่ส่วน Query optimizer ทำหน้าที่เสริมประสิทธิภาพการประมวลผลข้อคำถามด้วยการพยายามแปลงรูปแบบคำสั่ง SQL ให้เป็นรูปแบบที่จะประมวลผลได้เร็วขึ้น ตัวอย่างเช่น ข้อคำถาม Q_1 ที่ผู้ใช้เขียนอยู่ในรูปแบบคำสั่ง SQL ที่มีคำสั่ง SQL ย่อยซ้อนอยู่ชั้นในดังต่อไปนี้

```
Q1  :  SELECT    Emp.Name
       FROM      Emp
       WHERE     Emp.Dept #  IN
                     SELECT    Dept.Dept #
                     FROM      Dept
                     WHERE     Dept.Loc = 'Bangkok'
                           AND    Emp.Emp # = Dept.Mgr
```

จะประมวลผลได้เร็วขึ้นถ้าสามารถปรับให้เป็นรูปแบบคำสั่ง SQL ชั้นเดียวดังต่อไปนี้

```
Q1' :  SELECT    Emp.Name
       FROM      Emp E, Dept D
       WHERE     E.Dept # = D.Dept#
       AND       D.Loc = 'Bangkok'
       AND       E.Emp # = D.Mgr
```

การแปลงรูปแบบคำสั่งจาก Q_1 เป็น Q_1' เป็นวิธีการแปลงที่ใช้ความรู้เกี่ยวกับไวยากรณ์ของคำสั่ง SQL ช่วยในการแปลงให้เป็นคำสั่งที่มีความหมายเทียบเท่ากัน แต่มีขั้นตอนในการทำงานที่เร็วกว่า การแปลงในอีกวิธีการหนึ่งจะใช้ความหมายหรือความรู้เกี่ยวกับสคีมา (schema) ของข้อมูลมาใช้ช่วยในการแปลง ตัวอย่างเช่น จากสคีมาหรือโครงสร้างของข้อมูลเรือเดินสมุทร (Ships) ประกอบกับความรู้เกี่ยวกับสคีมาที่กำหนดไว้ในลักษณะของ semantic rule ช่วยให้สามารถแปลงข้อคำถาม Q_2 เป็นข้อคำถาม Q_2' ที่จะใช้เวลาประมวลผลหาคำตอบเร็วขึ้นดังนี้

```
schema :  Ships (Name, Type, Weight, Registry, Owner Name, Capacity)
                  index on Type
semantic rule  :  IF Ships.Weight > 200 THEN Ships.Type = 'Tanker'
```

Q_2 : SELECT Name
FROM Ships
WHERE Weight > 250

=>

Q_2' : SELECT Name
FROM Ships
WHERE Weight > 250
AND Type = 'Tanker'

การเพิ่มประสิทธิภาพการประมวลผลข้อคำถามดังตัวอย่างข้างต้น ที่เป็นการใช้ความรู้เกี่ยวกับสคีมาในลักษณะของ semantic rules เพื่อแปลงรูปแบบข้อคำถาม เรียกว่า *การเพิ่มประสิทธิภาพข้อคำถามเชิงความหมาย* (semantic query optimization)

นอกจากการใช้ semantic rules ที่กำหนดไว้ในส่วน integrity constraints และใน business rules แล้วเรายังสามารถใช้ความรู้จากแหล่งอื่นมาใช้เพิ่มประสิทธิภาพการประมวลผลข้อคำถามได้อีก

โครงการวิจัยนี้มุ่งความสนใจไปที่การพัฒนาวิธีการค้นหาโมเดลและใช้โมเดลจากการทำเหมืองข้อมูลและวิวข้อมูล (materialized view) เพื่อคัดเลือกและแปลงรูปแบบให้อยู่ในลักษณะของ semantic rules ด้วยจุดมุ่งหมายหลักที่จะนำมาใช้ช่วยในกระบวนการเพิ่มประสิทธิภาพการประมวลผลข้อคำถาม ในกรณีที่อาจมีบางส่วนของคำถามเกี่ยวข้องกับวิวข้อมูลและโมเดลข้อมูล

1.2 งานวิจัยที่เกี่ยวข้อง

Chang และ Lee (1998) ได้เสนอแนวคิดของการนำวิวข้อมูล (materialized views) มาใช้ช่วยในการตอบข้อคำถาม โดยนำเสนอผ่านตัวอย่างต่อไปนี้

กำหนดสคีมาของข้อมูลพื้นฐานให้ดังนี้

Sales (<u>eid</u>, <u>item</u>, <u>vol</u>, <u>date</u>)

Emp (<u>eid</u>, name, dept, salary, age)

Dept (<u>dept</u>, manager, loc)

Item (<u>item</u>, category, size, weight)

วิวข้อมูลคือ ข้อมูลชุดใหม่ที่สร้างจากข้อมูลพื้นฐาน ตัวอย่างชุดคำสั่งต่อไปนี้ทำหน้าที่สร้างวิวข้อมูล V (<u>eid</u>, name, <u>item</u>, category) ที่เก็บรายละเอียดข้อมูลการขายที่มีปริมาณการขายสินค้าเกิน 100 ชิ้น

```
CREATE VIEW V  AS
        (SELECT   E.eid, E.name, I.item, I.category
         FROM     Emp E, Sales S, Item I
```

```
WHERE   E.eid = S.eid
AND     S.item = I.item
AND     S.vol > 100)
```

ในกรณีที่มีข้อคำถาม สอบถามเกี่ยวกับพนักงาน (พร้อมทั้งให้แสดงชื่อแผนกที่สังกัดและชื่อผู้จัดการแผนก) ที่มีความสามารถขายคอมพิวเตอร์ได้มากกว่า 100 เครื่อง

```
Q3  :   SELECT   E.name, E.dept, D.manager
        FROM     Emp E, Sales S, Dept D
        WHERE    E.eid = S.eid
                 AND  E.dept = D.dept
                 AND  S.item = 'computer'
                 AND  S.vol > 100
```

จากกรณีมีวิวข้อมูล V ทำให้สามารถแปลงข้อคำถาม Q_3 ให้อยู่ในรูปแบบอื่นได้ดังนี้

```
Q3' :   SELECT   V.name, E.dept, D.manager
        FROM     V, Emp E, Dept D
        WHERE    V.eid = E.eid
                 AND  V.dept = D.dept
                 AND  V.item = 'computer'
```

ดังนั้นการประมวลผลข้อคำถาม Q_3 จึงมีโอกาสที่จะได้รับการเพิ่มประสิทธิภาพให้ประมวลผลได้เร็วขึ้นด้วยการแปลงเป็น Q_3' ถ้าหากขนาดของวิวข้อมูล V เล็กกว่าขนาดของข้อมูล Sales

ตัวอย่างข้างต้นนี้เป็นการแนะนำแนวคิดของการนำวิวข้อมูลมาใช้ประโยชน์ในการเพิ่มประสิทธิภาพการประมวลผลข้อมูล แต่ปัญหาที่ยังต้องมีการค้นคว้าหาคำตอบหรือหาแนวทางเพิ่มเติมคือ ในกรณีที่เงื่อนไขของข้อคำถามไม่ตรงพอดีกับลักษณะของวิวข้อมูลควรจะใช้วิวข้อมูลในการปรับปรุงข้อคำถามหรือไม่? และอย่างไร?

ในส่วนของเทคโนโลยีการทำเหมืองข้อมูล อยู่ในระยะเริ่มต้นของการผนวกเหมืองข้อมูลเข้ากับฐานข้อมูล (Microsoft Corporation, 2000; IBM, 2001) โดยกำหนดรูปแบบไวยากรณ์ให้ผู้ใช้สามารถสั่งสร้างโมเดลเหมืองข้อมูลได้ดังตัวอย่างต่อไปนี้ (Chaudhuri et al., 2004)

```
CREATE MINING MODEL Risk_Class                    //Name of model
        (Customer_ID  LONG     KEY,               //Source column
         Gender       TEXT     DISCRETE,          //Source column
         Risk         TEXT     DISCRETE PREDICT,  //Prediction column
         Purchases    DOUBLE   DISCRETIZED(),     //Source column
         Age          DOUBLE   DISCRETIZED()      //Source column
        )
USING [Decision_tree]                             //Mining algorithm
```

คำสั่งข้างต้นเป็นการใช้คำสั่งบน Microsoft SQL Server ให้สร้างโมเดลข้อมูลประเภทโมเดลเพื่อการทำนาย ด้วยอัลกอริทึม Decision-tree โมเดลจะทำนายระดับความเสี่ยง (Risk) ของลูกค้าโดยใช้ข้อมูลจากค่ารหัสลูกค้า (Customer_ID), เพศ (Gender), ปริมาณการซื้อ (Purchases), และอายุของลูกค้า (Age) ผู้ใช้สามารถใช้โมเดลที่ได้เขียนข้อความสั่งในภาษา SQL ให้แสดงรหัสของลูกค้าทุกคนที่มีระดับความเสี่ยงที่คาดหมายว่าอยู่ในระดับต่ำ (Risk = "low") ดังตัวอย่างต่อไปนี้

```
SELECT     D.Customer_ID, M.Risk
FROM       [Risk_Class] M
PREDICTION JOIN
                (SELECT      Customer_ID, Gender, Age, sum(Purchases) as SP
                 FROM        Customer D
                 GROUP BY    Customer_ID, Gender, Age) as D
          ON    M.Gender = D.Gender    and
                M.age = D.age          and
                M.Purchases = T.SP
                    WHERE M.Risk = "low"
```

จากตัวอย่างจะเห็นได้ว่าการใช้ประโยชน์โมเดลจากการทำเหมืองข้อมูลยังอยู่ในระดับจำกัดเพียงการกำหนดเงื่อนไขในการสอบถามข้อมูล โครงการวิจัยนี้จึงเสนอแนวทางที่จะขยายขอบเขตการใช้งานโมเดลจากการทำเหมืองข้อมูลไปสู่ระดับการใช้โมเดลช่วยเพิ่มประสิทธิภาพการสอบถามข้อมูล

ในส่วนของวิวข้อมูล (materialzed view) ที่สามารถนิยามได้ว่า คือข้อมูลที่มีมุมมองเฉพาะหรือมีข้อมูลเฉพาะด้านที่ตรงกับความสนใจของผู้ใช้ วิวข้อมูลจะสร้างขึ้นจากข้อมูลพื้นฐานที่เก็บอยู่ในฐานข้อมูล ประโยชน์หลักของวิวข้อมูล คือ ใช้ตอบคำถามเฉพาะเรื่องที่ผู้ใช้ส่วนใหญ่สนใจและมักจะสอบถามอยู่เสมอ การเก็บข้อมูลดังกล่าวไว้เป็นการถาวรในฐานข้อมูลร่วมกับข้อมูลพื้นฐานจึงช่วยให้การตอบข้อคำถามทำได้รวดเร็ว ความเร็วในการตอบข้อคำถามเป็นประเด็นสำคัญในงานประเภทคลังข้อมูล (Agarwal *et al*., 1996; Gray *et al*., 1996) และการประมวลผลสารสนเทศแบบกระจาย (Levy *et al*., 1996)

นักวิจัยได้พยายามหาวิธีใช้ประโยชน์จากวิวข้อมูลมาเป็นระยะเวลานาน (Chen and Rossopoulos, 1994: Chaudhuri *et al*., 1995: Gupta *et al*., 1995: Qian, 1996; Srivastava *et al*., 1996) แต่วิธีการต่างๆ ที่เสนอยังมีข้อจำกัดที่วิวข้อมูลจะต้องมีโครงสร้างที่ตรงพอดีกับเงื่อนไขในข้อคำถาม แนวทางวิจัยของโครงการวิจัยนี้พยายามลดข้อจำกัดดังกล่าว โดยพยายามใช้เกณฑ์คัดเลือกวิวเพื่อพิจารณาวิวข้อมูลที่มีความใกล้เคียงกับข้อคำถามมากที่สุด นอกจากวิวข้อมูลแล้วยังมีการผนวกโมเดลจากการทำเหมืองข้อมูล เข้ามาช่วยในกระบวนการแปลงรูปแบบข้อคำถามให้สามารถประมวลผลได้เร็วที่สุด โดยยังคงความถูกต้องของผลลัพธ์ในการตอบข้อคำถาม

1.3 วัตถุประสงค์ของการวิจัย

- พัฒนาแนวทางการนำวิวข้อมูล และโมเดลข้อมูลที่ได้จากการทำเหมืองข้อมูลมาใช้ช่วยเพิ่มประสิทธิภาพการประมวลผลข้อคำถาม
- กำหนดวิธีการคัดเลือกวิวข้อมูล และวิธีการใช้วิวข้อมูลเพื่อแปลงรูปแบบคำสั่งสอบถามข้อมูล
- ออกแบบรูปแบบโมเดลข้อมูลจากการทำเหมืองข้อมูลที่เหมาะสมกับงาน query optimization และพัฒนาวิธีการแปลงโมเดลข้อมูลให้เป็น semantic rules
- ทดสอบวิธีการใช้วิวข้อมูลและโมเดลจากการทำเหมืองข้อมูลเพื่อปรับปรุงประสิทธิภาพการประมวลผลข้อคำถาม

1.4 ขอบเขตของการวิจัย

งานวิจัยนี้เป็นการพัฒนาแนวทางในการแสวงหาวิวข้อมูลและความรู้เกี่ยวกับข้อมูลในลักษณะของโมเดลเพื่อการทำนาย (predictive model) แปลงเป็น semantic rules เพื่อนำมาใช้ช่วยเพิ่มขีดความสามารถของ semantic query optimizer ดังนั้นการออกแบบ การวิเคราะห์ และการ

ทดสอบจะจำกัดเฉพาะในส่วน query optimizer โดยไม่ได้มีการ implement ในส่วน Scanner, Code generator, หรือ Execution engine

การทดสอบการใช้ semantic rules เพื่อช่วยในกระบวนการประมวลผลข้อคำถามจะกระทำบนฐานข้อมูลนิรนัย (deductive database) เนื่องจากมีรูปแบบที่เหมาะสมกับการทำงานกับวิวข้อมูลและโมเดลข้อมูลที่อยู่ในลักษณะของกฎ

1.5 ประโยชน์ที่ได้รับจากการวิจัย

โครงการวิจัยนี้มีจุดมุ่งหมายที่จะพัฒนาแนวทางใหม่ในการประมวลผลข้อคำถามในงานฐานข้อมูล ดังนั้นผลผลิตที่จะได้จากการวิจัยนี้จะเป็นองค์ความรู้ใหม่ที่ช่วยพัฒนาความก้าวหน้าของวงการฐานข้อมูลและการสอบถามข้อมูล รวมถึงการเรียกใช้ข้อมูลของ search engine ต่างๆ

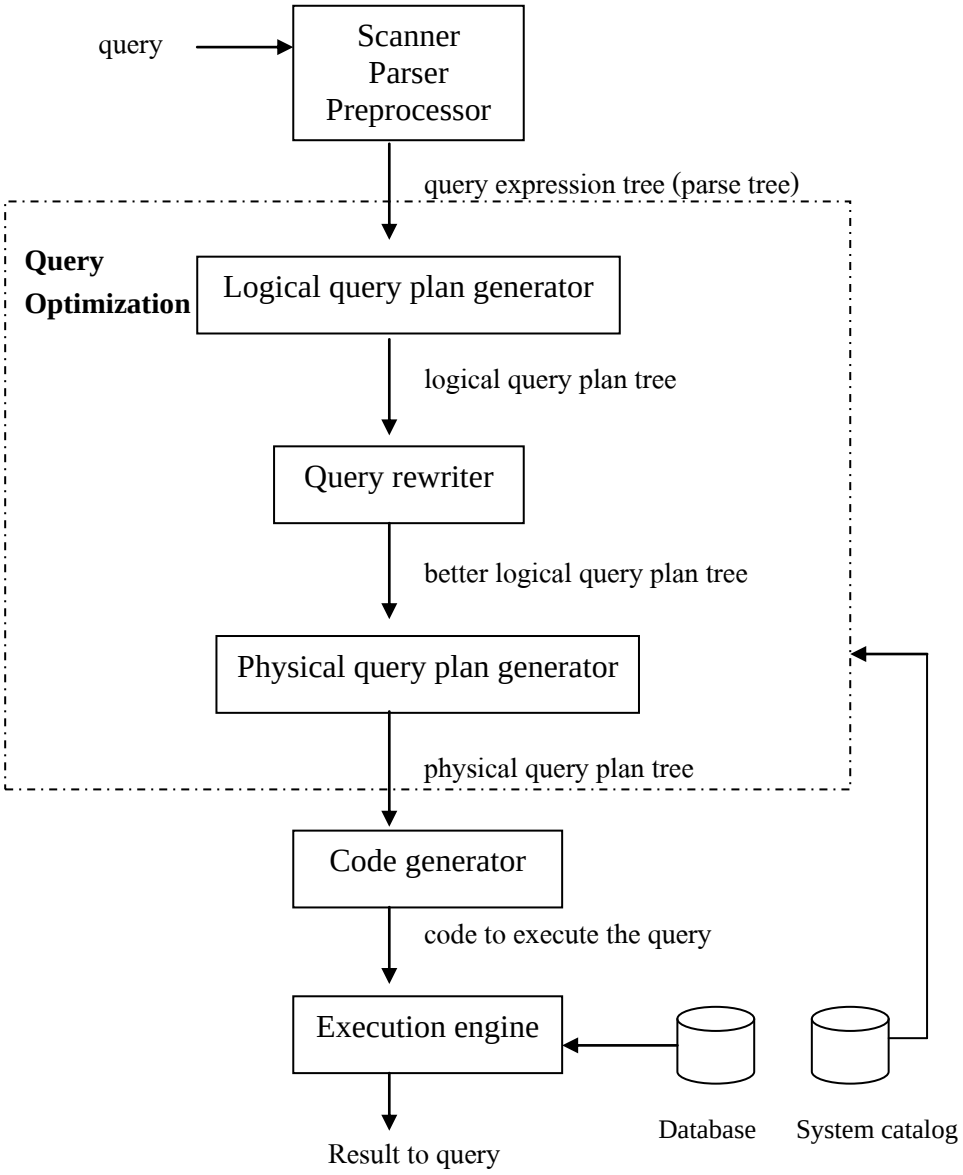
ผลการวิจัยในส่วนการพัฒนาเทคนิคการเพิ่มประสิทธิภาพการประมวลผลข้อคำถามที่เป็นองค์ความรู้ใหม่จะเผยแพร่ด้วยการเขียนบทความวิจัย นำเสนอ และตีพิมพ์ในเอกสารการประชุมวิชาการระดับประเทศและนานาชาติ นอกจากนี้ยังจะเผยแพร่ผลงานผ่านอินเทอร์เน็ตโดยให้นักวิจัยและผู้สนใจทั่วไปดาวน์โหลดได้จากเว็บไซต์ของหน่วยปฏิบัติการวิจัยด้านวิศวกรรมข้อมูลและการค้นหาความรู้ มหาวิทยาลัยเทคโนโลยีสุรนารี

นอกจากนี้โครงการวิจัยนี้ยังช่วยพัฒนาความสามารถและศักยภาพของผู้วิจัยและผู้ช่วยวิจัยที่เป็นนักศึกษาปริญญาโทสาขาวิชาวิศวกรรมคอมพิวเตอร์ ซึ่งจะเป็นการพัฒนากำลังคนระดับสูงด้านวิทยาศาสตร์และเทคโนโลยีของประเทศชาติ

บทที่ 2
การประมวลผลข้อคำถาม

2.1 ขั้นตอนการประมวลผลข้อคำถาม

การประมวลผลข้อคำถาม (query processing) เป็นกระบวนการที่ประกอบด้วยหลายขั้นตอน (รูปที่ 2.1) เพื่อเปลี่ยนข้อคำถามที่ผู้ใช้ส่งเข้ามาให้เป็นชุดคำสั่งที่สามารถดำเนินการกับฐานข้อมูลเพื่อหาคำตอบให้กับข้อคำถาม



รูปที่ 2.1 ขั้นตอนการประมวลผลข้อคำถาม

กระบวนการประมวลผลข้อคำถามนี้ มักจะต้องเกี่ยวข้องกับการเรียกใช้ข้อมูลที่เก็บอยู่ในดิสก์เพื่อนำมาประมวลผลในระบบคอมพิวเตอร์ ขั้นตอนที่ต้องใช้เวลามากที่สุดคือขั้นตอนของการเข้าถึงข้อมูลในดิสก์ เพื่อถ่ายโอนข้อมูลที่ต้องการมายังหน่วยความจำหลัก ถ้าเราสามารถลดจำนวนครั้งของการติดต่อกับดิสก์ (เพื่อถ่ายโอนข้อมูล) ลงได้ จะช่วยให้ลดเวลาในการประมวลผลข้อคำถามลงได้มาก การกระทำเพื่อลดเวลาในการประมวลผลข้อคำถามนี้เรียกว่า *การหาวิธีเหมาะที่สุด* หรือ *การเพิ่มประสิทธิภาพการประมวลผลข้อคำถาม* (query optimization) ซึ่งมักจะปรากฏเป็นส่วนหนึ่งในกระบวนการประมวลผลข้อมูล

ขั้นตอนของการประมวลผลข้อคำถาม อธิบายได้ดังนี้

ขั้นตอนที่ 1: ผู้ใช้ส่งข้อคำถามเข้ามายังระบบฐานข้อมูล ข้อคำถามนี้มักจะอยู่ในรูปแบบของภาษาระดับสูง เช่น SQL ตัวอย่างต่อไปนี้แสดงการสอบถามรายชื่อพนักงานที่ทำหน้าที่เป็นผู้ดูแลโปรเจ็คขนาดใหญ่ที่มีมูลค่าตั้งแต่ 1 ล้านบาทขึ้นไป

Schema

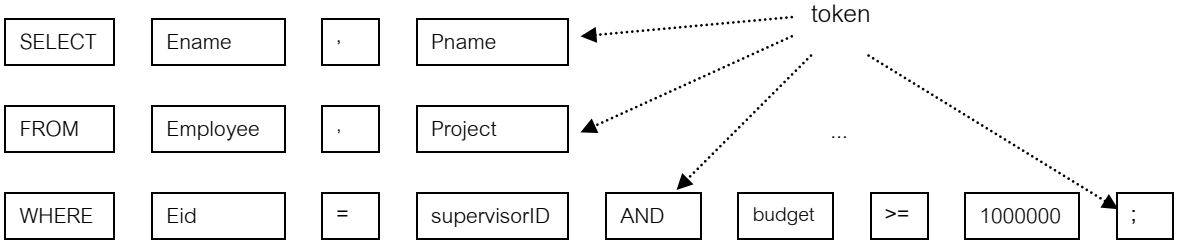
Employee (Eid, Ename, birthdate, salary)

Project (Pno, Pname, budget, supervisorID)

SQL query (Q1)

```
SELECT   Ename, Pname
FROM     Employee, Project
WHERE    Eid  =  SupervisorID AND budget > = 1000000 ;
```

ขั้นตอนที่ 2: โปรแกรมสแกนเนอร์ (scanner) จะจัดแยกส่วนข้อคำถามเป็นหน่วยย่อย (รูปที่ 2.2) ที่เรียกว่า โทเค็น (token) เพื่อใช้ตรวจสอบความถูกต้องของชื่อและคำสงวนต่างๆ เมื่อตรวจสอบความถูกต้องแล้วจะส่งโทเค็นเหล่านี้ไปให้โปรแกรมพาสเซอร์



รูปที่ 2.2 การแยกคำเป็นโทเค็นโดยโปรแกรมสแกนเนอร์

ขั้นตอนที่ 3: โปรแกรมพาสเซอร์ (parser) รับโทเค็นมาจากสแกนเนอร์เพื่อตรวจสอบว่าคำสั่ง SQL เขียนถูกไวยากรณ์หรือไม่ (syntactic checking) โครงสร้างข้อมูลที่นำมาใช้ช่วยใน

การตรวจสอบไวยกรณ์นี้คือ โครงสร้างต้นไม้ หรือ ทรี ถ้าคำสั่ง SQL ถูกต้องตามไวยากรณ์จะสามารถสร้างทรีได้สมบูรณ์ และเรียกทรีนี้ว่า พาสทรี (parse tree) ตัวอย่างต่อไปนี้แสดงไวยากรณ์อย่างง่ายของการเขียนคำสั่ง SQL ไวยากรณ์นี้อยู่ในรูปแบบที่เรียกว่า BNF (Backus–Naur Form) และแสดงพาสทรีของคำสั่ง *Q1* ได้ดังรูปที่ 2.3

BNF grammar

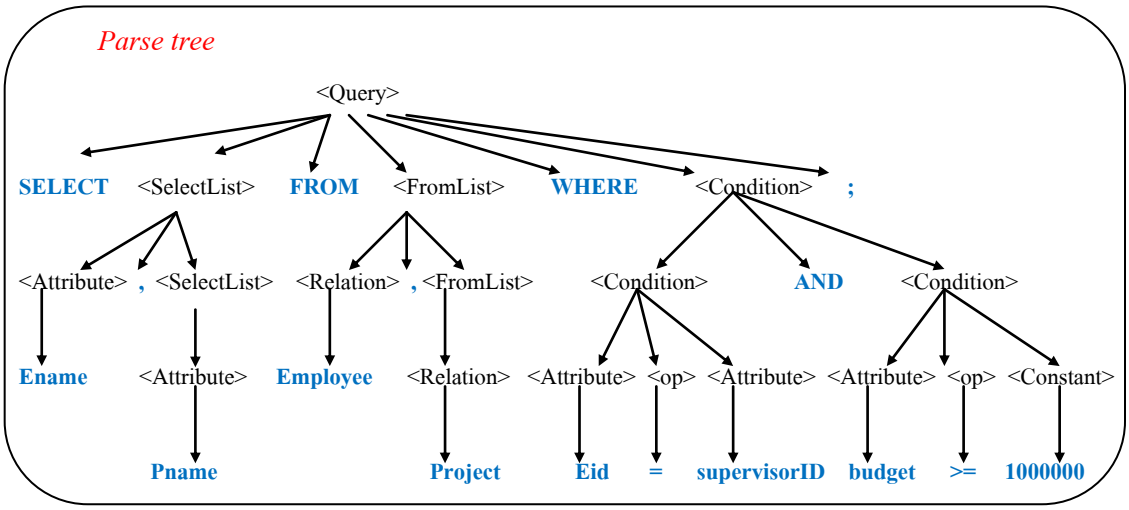
<Query> ::= SELECT <SelectList> FROM <FromList> WHERE <Condition> ;
| SELECT <SelectList> FROM <FromList> ;

<SelectList>::= <Atrribute>
| <Attribute> , <SelectList>

<FromList> ::= <Relation>
| <Relation> , <FromList>

<Condition>::= <Condition> AND <Condition>
| <Attribute> <op> <Attribute>
| <Attribute> <op> <Constant>

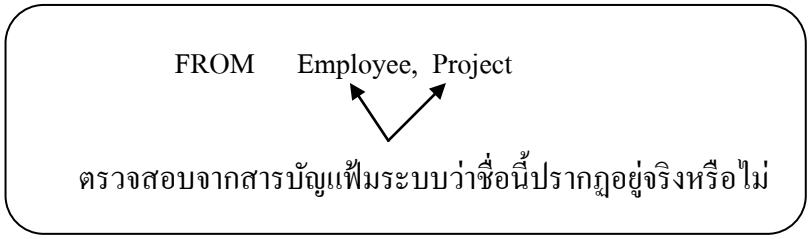
<op> ::= = | <> | > | >= | < | <=



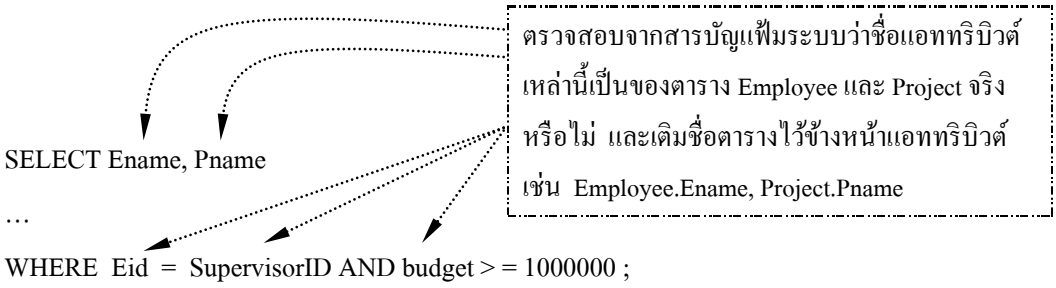
รูปที่ 2.3 ตัวอย่างพาสทรีของคำสั่งที่ถูกไวยากรณ์

ขั้นตอนที่ 4: โปรแกรมประมวลผลก่อน (preprocessor) จะทำหน้าที่ตรวจสอบความหมาย (semantic checking) ของคำสั่ง SQL การตรวจสอบนี้เป็นการตรวจสอบเบื้องต้น ซึ่งส่วนมากจะเป็นการตรวจสอบเกี่ยวกับชื่อได้แก่ การตรวจสอบชื่อตารางข้อมูล (รูปที่ 2.4) ตรวจสอบชื่อแอททริบิวต์ (รูปที่ 2.5) ความสอดคล้องของชนิดข้อมูลและการ

ดำเนินการกับข้อมูล เช่น ตรวจสอบว่า budget เป็นชนิดข้อมูลที่สามารถนำมาเปรียบเทียบค่า “ >= ” กับตัวเลข 1000000 ได้หรือไม่



รูปที่ 2.4 ตัวอย่างการตรวจสอบความถูกต้องของชื่อข้อมูล

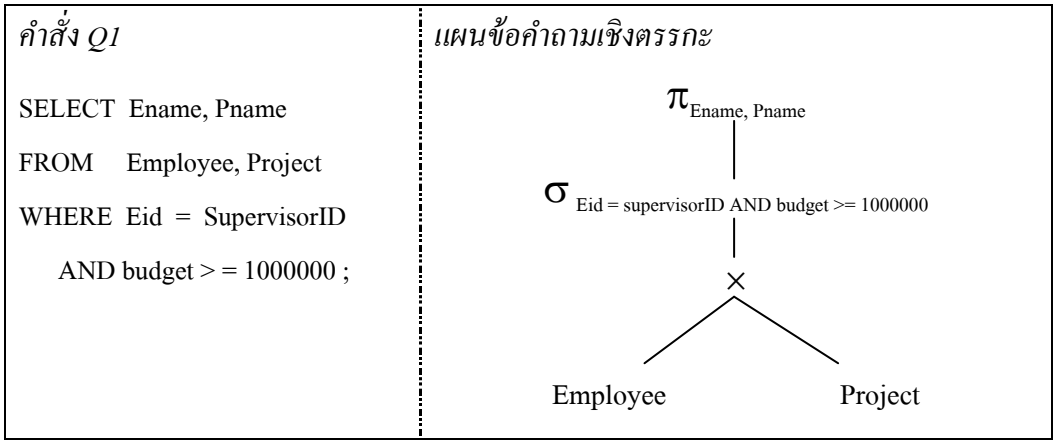


รูปที่ 2.5 ตัวอย่างการตรวจสอบความถูกต้องของชื่อแอททริบิวต์

ขั้นตอนที่ 5: โปรแกรมสร้างแผนข้อคำถามเชิงตรรกะ (logical query plan generator) จะทำหน้าที่แปลงพาสทรีให้เป็น logical query plan tree ซึ่งจะอยู่ในรูปของต้นไม้นิพจน์พีชคณิต วิธีการแปลงจะทำจากส่วนล่างของพาสทรีขึ้นไปหาโหนดรากที่อยู่ด้านบน โดยมีขั้นตอนดังนี้

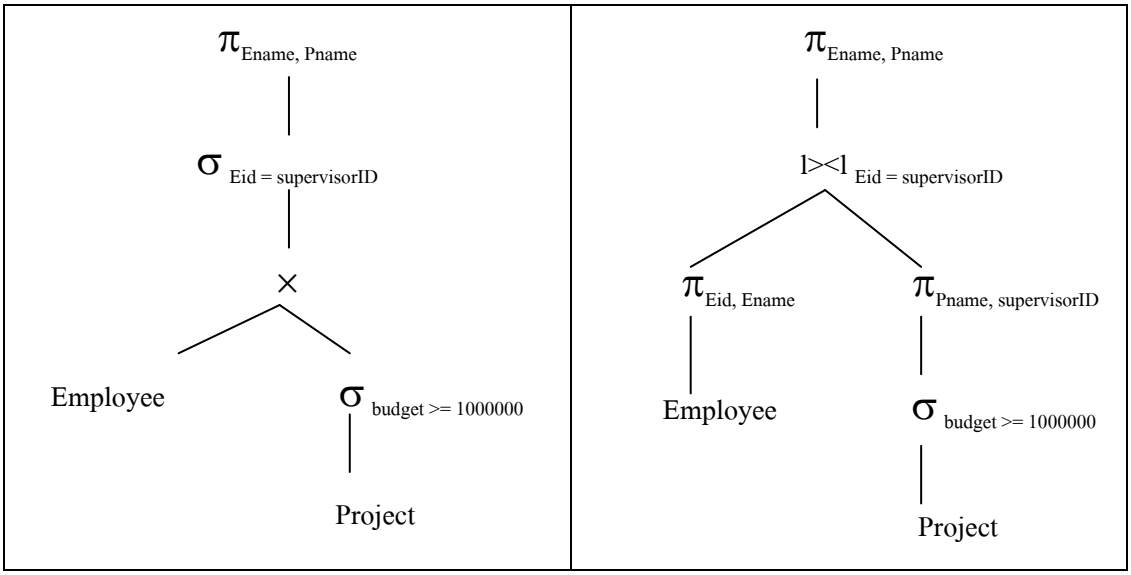
- นำชื่อตารางใน <FromList> มาทำ Cartesian product ($\times$)
- เปลี่ยน <Condition> ให้อยู่ในรูปแบบของคำสั่ง selection (σ)
- เปลี่ยน <SelectList> ให้อยู่ในรูปแบบของคำสั่ง projection (π)

ตัวอย่างการสร้างแผนข้อคำถามเชิงตรรกะจากคำสั่งสอบถามข้อมูล *Q1* แสดงได้ดังรูปที่ 2.6



รูปที่ 2.6 ตัวอย่างการสร้างแผนข้อคำถามเชิงตรรกะจากคำสั่งสอบถามข้อมูล

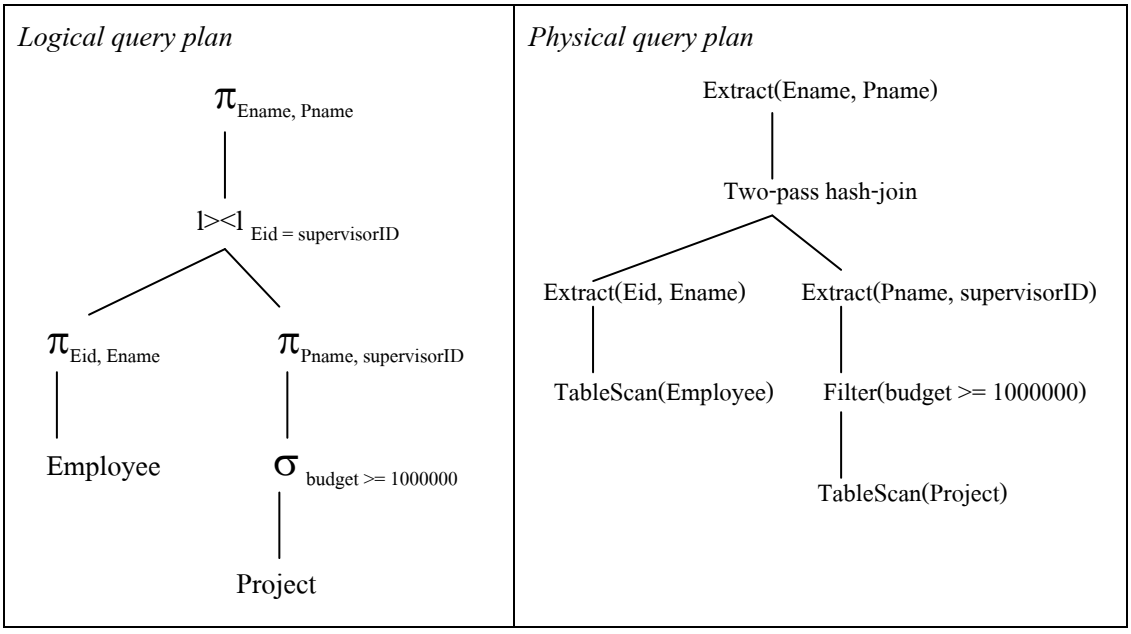
โครงสร้างต้นไม้ที่ได้นี้เรียกว่า แผนข้อคำถามเชิงตรรกะ (logical query plan) ที่เรียกชื่อเช่นนี้เพราะเป็นแบบแผนที่ใช้ประมวลผลข้อคำถาม โดยการประมวลผลจะกระทำจากส่วนล่างสุดของต้นไม้ไล่เป็นระดับชั้นขึ้นไปจนกระทั่งถึงโหนดราก เมื่อประมวลผลการกระทำที่โหนดรากเสร็จ ก็จะได้ผลลัพธ์ที่เป็นคำตอบของข้อคำถามที่ผู้ใช้ส่งเข้ามา แผนข้อคำถามเชิงตรรกะข้างต้นยังไม่ใช่แบบแผนที่ดีที่สุดเพราะสามารถปรับปรุงให้ดีขึ้นได้ดังรูปที่ 2.7



รูปที่ 2.7 แผนข้อคำถามเชิงตรรกะแบบที่สองและสามสำหรับประมวลผลคำสั่ง *Q1*

จะเห็นได้ว่าจากพาสทรีหนึ่งต้นที่ส่งมาจากขั้นตอนที่ 4 โปรแกรมสร้างแผนข้อคำถามเชิงตรรกะ สามารถสร้างทรีที่เป็นแผนข้อคำถามเชิงตรรกะได้เป็นจำนวนมาก แผนข้อคำถามเหล่านี้สามารถให้คำตอบที่ถูกต้องได้เหมือนกันทั้งหมด แต่แตกต่างกันที่เวลาที่ใช้ในการดำเนินการประมวลผลตามแผนข้อคำถาม และเนื้อที่ของหน่วยความจำที่ต้องใช้ในระหว่างการประมวลผล โปรแกรมประมวลผลข้อคำถามส่วนมากจึงต้องมีโปรแกรมที่เรียกว่า *โปรแกรมหาวิธีเหมาะที่สุด* (query optimizer) หรือ *โปรแกรมเพิ่มประสิทธิภาพการประมวลผลข้อคำถาม* เพื่อทำหน้าที่สร้างและคัดเลือกแผนข้อคำถามเชิงตรรกะที่เหมาะสมที่สุด

ขั้นตอนที่ 6: โปรแกรมสร้างแผนข้อคำถามเชิงกายภาพ (physical query plan generator) จะรับแผนข้อคำถามเชิงตรรกะที่คัดเลือกมาแล้วในขั้นตอนก่อนหน้านี้ เพื่อแปลงให้เป็นแผนข้อคำถามเชิงกายภาพ (physical query plan) ซึ่งก็คือ ลำดับขั้นตอนของวิธีประมวลผลที่จะเกิดขึ้นจริง แสดงตัวอย่างได้ดังรูปที่ 2.8



รูปที่ 2.8 แผนข้อคำถามเชิงตรรกะและเชิงกายภาพสำหรับประมวลผลคำสั่ง *Q1*

ชื่อที่ปรากฏในแผนข้อคำถามเชิงกายภาพ เช่น TableScan(Employee) หรือ Extract(Eid, Ename) เป็นชื่ออัลกอริทึมที่ถูกเลือกมาใช้ในการประมวลผลข้อคำถามนี้ และเช่นเดียวกับขั้นตอนก่อนหน้านี้ ที่แผนข้อคำถามเชิงกายภาพสามารถมีได้มากกว่า 1 แบบแผนขึ้นอยู่กับการเลือกอัลกอริทึมมาใช้ ดังนั้นจึงเป็นหน้าที่ของโปรแกรมเพิ่มประสิทธิภาพการประมวลผลข้อคำถามที่จะคัดเลือกแบบแผนที่เหมาะสมที่สุด โดยใช้เวลาประมวลผลและเนื้อที่หน่วยความจำที่ เป็นเกณฑ์ในการพิจารณาคัดเลือก

ขั้นตอนที่ 7: โปรแกรมสร้างรหัสคำสั่ง (code generator) จะทำหน้าที่เปลี่ยนอัลกอริทึมต่างๆ ที่ปรากฏในแผนข้อคำถามเชิงกายภาพให้เป็นรหัสคำสั่ง (เช่น คำสั่งภาษาซี) ที่พร้อมจะถูกดำเนินงานได้โดยระบบคอมพิวเตอร์

ขั้นตอนที่ 8: ตัวกระทำการ (execution engine) จะควบคุมการคอมไพล์และการดำเนินงานชุดคำสั่งที่ส่งมาจากขั้นตอนที่ 7 ในขั้นตอนนี้จะต้องมีการติดต่อกับฐานข้อมูลในดิสก์และผลลัพธ์จากการดำเนินงานคือคำตอบของข้อคำถามที่ผู้ใช้ส่งเข้ามา

2.2 อัลกอริทึมพื้นฐานสำหรับประมวลผลข้อคำถาม

คำสั่งที่ผู้ใช้ใช้สร้างข้อคำถามมักจะอยู่ในลักษณะของภาษาระดับสูง เช่น SQL ซึ่งจะต้องถูกแปลงให้อยู่ในรูปแบบของภาษาฟอร์มอล เช่น พีชคณิตสัมพันธ์ (relational algebra) หรือแคลคูลัสสัมพันธ์ (relational calculus) ทั้งนี้เพื่อความสะดวกในการปรับปรุงข้อคำถามให้มีประสิทธิภาพดีขึ้น นิพจน์ในพีชคณิตสัมพันธ์มักจะประกอบด้วยปฏิบัติการหลักๆ คือ selection, projection, และ join ปฏิบัติการแต่ละประเภทมีเทคนิคที่จะใช้ประมวลผลได้หลายเทคนิคขึ้นอยู่กับขนาดของข้อมูล และโครงสร้างที่ใช้สนับสนุนการเข้าถึงข้อมูล (เช่น ใช้ดรรชนี, ใช้ตารางแฮช)

การอธิบายอัลกอริทึมต่างๆ ของปฏิบัติการพื้นฐาน (หรือคำสั่งในระดับแผนข้อคำถามเชิงตรรกะ) และการวิเคราะห์อัลกอริทึม (Elmasri and Navathe, 2000; Garcia-Molina *et al.*, 2000; Ramakrishnan and Gehrke, 2000; Silberschatz *et al.*, 2002) จะอ้างอิงข้อมูลพื้นฐาน 3 ตารางได้แก่ Employee, Works_In และ Project โดยแต่ละตารางมีแอททริบิวต์และส่วนประกอบทางกายภาพดังต่อไปนี้

Employee

Eid	Ename	gender	birthdate	address	salary	dept_code

สมมุติให้ 1 เรคคอร์ดมีขนาด = 50 ไบต์
จำนวนข้อมูลทั้งหมด = 40,000 เรคคอร์ด
มี primary index บน Eid และ
Secondary index บน salary และ dept_code

กำหนดให้เนื้อที่เก็บข้อมูล 1 บล็อก = 4,000 ไบต์
ดังนั้นหนึ่งบล็อกบันทึกข้อมูลได้ = 80 เรคคอร์ด
$\therefore$ ต้องใช้เนื้อที่ = 40000/80
= 500 บล็อก

Works_In

Eid	Pno	role	start_date	finish_date

สมมุติให้ 1 เรคคอร์ดมีขนาด = 40 ไบต์
จำนวนข้อมูลทั้งหมด = 100,000 เรคคอร์ด

กำหนดให้เนื้อที่เก็บข้อมูล 1 บล็อก = 4,000 ไบต์
ดังนั้นหนึ่งบล็อกบันทึกข้อมูลได้ = 100 เรคคอร์ด
∴ ต้องใช้เนื้อที่ = 100000/100 = 1,000 บล็อก

Project

Pno	Pname	Psupervisor	budget

สมมุติให้ 1 เรคคอร์ดมีขนาด = 40 ไบต์
จำนวนข้อมูลทั้งหมด = 1,000 เรคคอร์ด
มี hash index บน Pno

กำหนดให้เนื้อที่เก็บข้อมูล 1 บล็อก = 4,000 ไบต์
∴ ต้องใช้เนื้อที่ดิสก์ = 10 บล็อก

2.2.1 คำสั่ง selection

อัลกอริทึม S1: ค้นหาเชิงเส้น (ข้อมูลไม่เรียงลำดับตามค่าค้นหา ไม่มีดรรชนี)

1. ค้นหาข้อมูลทีละเรคคอร์ด
2. เปรียบเทียบข้อมูลว่าตรงกับเงื่อนไขในข้อคำถามหรือไม่
   ถ้าตรงกับเงื่อนไข ให้บันทึกเรคคอร์ด (หรือทูพเพิล) นี้ลงในตารางที่จะแสดงเป็นผลลัพธ์
   ถ้าไม่ตรงกับเงื่อนไข ให้ค้นหาเรคคอร์ดต่อไป

ตัวอย่างข้อคำถาม

ภาษา SQL :

```
SELECT  *
FROM    Employee
WHERE   birthdate > 31/12/1950 ;
```

พีชคณิตสัมพันธ์ (relational algebra) :

$\sigma_{\text{birthdate}>31/12/1950}(\text{Employee})$

จำนวนครั้งที่ต้องติดต่อกับดิสก์ = จำนวนครั้งที่ต้องอ่านทุกบล็อกของตาราง Employee
= 500 ครั้ง

อัลกอริทึม S2: ค้นหาแบบทวิภาค (ข้อมูลเรียงลำดับตามค่าค้นหา ไม่มีดรรชนี)

1. แบ่งค้นข้อมูลทีละครึ่งเพื่อหาเรคคอร์ดที่ตรงกับเงื่อนไขของข้อคำถาม
2. เมื่อเจอเรคคอร์ดที่ต้องการ บันทึกเรคคอร์ด (หรือทูพเพิล) ลงในตารางผลลัพธ์
3. ค้นหาเรคคอร์ดถัดไป จนกว่าข้อมูลจะมีค่าไม่ตรงกับเงื่อนไขของข้อคำถาม บันทึกเรคคอร์ดที่ข้อมูลมีค่าตรงกับเงื่อนไขของข้อคำถาม ลงในตารางผลลัพธ์

ตัวอย่างข้อคำถาม

```
SELECT  *
FROM    Employee
WHERE   Ename = 'Smith' ;
```

$$\sigma_{Ename = 'Smith'}(Employee)$$

ในขั้นตอนที่ (1) จำนวนครั้งที่ติดต่อกับดิสก์

$$= \lceil \log_2 (\text{จำนวนบล็อกทั้งหมดของตาราง Employee}) \rceil$$

$$= \log_2(500) \approx 9 \text{ ครั้ง}$$

ในขั้นตอนที่ (3) จำนวนครั้งที่ติดต่อกับดิสก์ มีค่าที่เป็นไปได้ตั้งแต่ศูนย์ครั้ง (ไม่มีเรคคอร์ดอื่นที่ Ename = 'Smith') จนถึง 500 ครั้ง (ทุกเรคคอร์ดมี Ename = 'Smith') ดังนั้น

จำนวนครั้งทั้งหมดที่ต้องติดต่อกับดิสก์ = จำนวนครั้งที่ติดต่อกับดิสก์ในขั้นตอนที่ (1) + จำนวนครั้งที่ติดต่อกับดิสก์ในขั้นตอนที่ (3)

อัลกอริทึม S3: ค้นหาด้วยดรรชนี (มีดรรชนี B^+-tree สำหรับค่าที่ต้องการค้นหา)

1. ใช้ดรรชนีที่มีค้นหาค่าที่ต้องการ
2. ตามตัวชี้ไปยังตารางข้อมูลหลักและแสดงข้อมูลทุกเรคคอร์ดที่มีค่าตรงกับเงื่อนไขของข้อคำถาม

ตัวอย่างข้อคำถามที่ 1 : ข้อมูลเรียงลำดับตามค่าค้นหา

```
SELECT  *
FROM    Employee
WHERE   Eid = '12345' ;
```

$$\sigma_{Eid = '12345}(Employee)$$

สมมุติให้โครงสร้างต้นไม้ของ B^+- tree สำหรับข้อมูล Employee มี 2 ระดับชั้น

จำนวนครั้งที่ต้องติดต่อกับดิสก์ = จำนวนระดับชั้นของโครงสร้างดรรชนี + จำนวนบล็อกที่มีข้อมูลที่ต้องการ

$$= 2 + 1 = 3 \text{ ครั้ง}$$

ตัวอย่างข้อคำถามที่ 2 : ข้อมูลไม่เรียงลำดับตามค่าค้นหา

```
SELECT   *
FROM     Employee
WHERE    salary > 50000 ;
```

$$\sigma_{salary>50000}(Employee)$$

จำนวนครั้งที่ต้องติดต่อกับดิสก์ = จำนวนระดับชั้นของโครงสร้างดรรชนี + จำนวนบล็อกที่มีข้อมูลที่ต้องการ

= 2 + จำนวนบล็อกที่มีข้อมูลที่ต้องการ (มีค่าแปรผันได้ตั้งแต่ 1 ถึง 500)

อัลกอริทึม S4: ค้นหาด้วยตารางแฮช (มีตารางแฮชสำหรับค่าที่ต้องการค้นหา)

1. ใช้ฟังก์ชันแบบแฮชคำนวณหา bucket ที่บรรจุค่าที่ต้องการค้นหา
2. ตามตัวชี้ของ bucket ไปยังตารางข้อมูลหลัก

ตัวอย่างข้อคำถาม

```
SELECT   *
FROM     Project
WHERE    Pno = '09' ;
```

$$\sigma_{Pno = '09'}(Project)$$

จำนวนครั้งที่ต้องติดต่อกับดิสก์ = จำนวนบล็อกที่บรรจุ bucket ที่ต้องการ + จำนวนบล็อกที่มีข้อมูลที่ต้องการ

= 1 + 1

= 2 ครั้ง

อัลกอริทึม S5: ค้นหาด้วยดรรชนี (กรณีมีหลายเงื่อนไขย่อยเชื่อมด้วย AND)

1. ใช้ดรรชนีของเงื่อนไขย่อยค้นหาข้อมูล
2. นำข้อมูลที่ได้จากขั้นตอนที่ 1 มาคัดเลือกต่อด้วยเงื่อนไขย่อยที่เหลือ

ตัวอย่างข้อคำถาม

```
SELECT    *
FROM      Employee
WHERE     salary > 50000 AND
          birthdate > 31/12/1950 AND
          gender = 'M' ;
```

$$\sigma_{\text{salary} > 50000 \ \wedge \ \text{birthdate} > 31/1/1950 \ \wedge \ \text{gender} = \text{'M'}} (\text{Employee})$$

จำนวนครั้งที่ต้องติดต่อกับดิสก์ = จำนวนครั้งที่ต้องติดต่อกับดิสก์ของขั้นตอนที่ (1)

อัลกอริทึม S6: ค้นหาด้วยการทำ intersection ตัวชี้เรคคอร์ดของดรรชนี
(กรณีมีหลายเงื่อนไขย่อยเชื่อมด้วย AND และแต่ละเงื่อนไขย่อยมีดรรชนีที่ชี้ไปยังเรคคอร์ด)

1. ใช้ดรรชนีของแต่ละเงื่อนไขย่อยค้นหาข้อมูล ผลลัพธ์ที่ได้คือตัวชี้เรคคอร์ด ซึ่งมักจะอยู่ในรูปของ record ID หรือ RID
2. นำ RID ที่ได้จากขั้นตอนที่ 1 ของแต่ละเงื่อนไขย่อยมาทำอินเตอร์เซ็คชั่น ผลลัพธ์ที่ได้คือ RID ของเรคคอร์ดที่มีข้อมูลตรงกับที่ระบุในเงื่อนไขทั้งหมด เมื่อตามตัวชี้นี้ไปจะได้ข้อมูลที่ต้องการ

ตัวอย่างข้อคำถาม

```
SELECT    *
FROM      Employee
WHERE     dept_code = '05' AND
          salary > 30000 ;
```

$$\sigma_{\text{dept\_code} = \text{'05'} \ \wedge \ \text{salary} > 30000} (\text{Employee})$$

จำนวนครั้งที่ต้องติดต่อกับดิสก์ = จำนวนระดับชั้นของโครงสร้างดรรชนี dept_code +
จำนวนระดับชั้นของโครงสร้างดรรชนี Salary +
จำนวนบล็อกของข้อมูลที่มีค่าตรงกับทุกเงื่อนไขย่อย

ในกรณีที่เงื่อนไขย่อยเชื่อมด้วย OR ทำได้ในทำนองเดียวกันกับอัลกอริทึม S6 เพียงแต่เปลี่ยนจากการทำอินเตอร์เซ็กชั่นเป็นการทำยูเนี่ยน (union)

2.2.2 คำสั่ง projection

อัลกอริทึม P1: ผลลัพธ์ของการโปรเจ็ครวมแอททริบิวต์ที่เป็นค่าคีย์ จึงไม่มีข้อมูลซ้ำ

1. เปิดแฟ้มข้อมูล
2. คัดเลือกเฉพาะแอททริบิวต์ที่ระบุในคำสั่ง แสดงออกมาเป็นผลลัพธ์

ตัวอย่างข้อคำถาม

SELECT Eid, Ename, salary FROM Employee ;	$\pi_{\text{Eid, Ename, salary}}$ (Employee)

จำนวนครั้งที่ต้องติดต่อกับดิสก์ = จำนวนบล็อกทั้งหมดของตาราง Employee
= 500 ครั้ง

อัลกอริทึม P2: ผลลัพธ์ของการโปรเจ็คอาจมีข้อมูลซ้ำ กำจัดข้อมูลซ้ำด้วยเทคนิคการเรียงลำดับ

1. เปิดแฟ้มข้อมูล คัดเลือกเฉพาะแอททริบิวต์ที่ระบุในคำสั่ง เก็บไว้ในตารางชั่วคราว T
2. เรียงลำดับข้อมูลในตาราง T โดยใช้ทุกแอททริบิวต์ประกอบกันเป็นคีย์ในการเรียงลำดับ
3. นำข้อมูลที่เรียงลำดับแล้วมาเปรียบเทียบทูพเพิลที่อยู่ติดกัน ถ้าซ้ำกันให้กำจัดตัวซ้ำออกไป

ตัวอย่างข้อคำถาม

SELECT DISTINCT Ename, dept_code FROM Employee ;	$\delta(\pi_{\text{Eid, Ename, salary}}$ (Employee))

กรณีที่ข้อมูลมีจำนวนไม่มาก สามารถนำข้อมูลทั้งหมดจากดิสก์มาไว้ในหน่วยความจำหลักเพื่อคัดเลือกแอททริบิวต์ และเรียงลำดับใหม่เพื่อกำจัดข้อมูลซ้ำ
จำนวนครั้งที่ต้องติดต่อกับดิสก์ = จำนวนบล็อกทั้งหมดของตาราง Employee
= 500 ครั้ง

กรณีที่ข้อมูลมีมากเกินกว่าจะนำข้อมูลทั้งหมดมาเรียงลำดับในหน่วยความจำหลักได้ จะต้องใช้วิธีการเรียงลำดับภายนอก (external sorting) โดยแบ่งการทำงานเป็น 2 ช่วง

ช่วงที่ 1: เรียงลำดับข้อมูล

(1) อ่านข้อมูลจากดิสก์มาด้วยจำนวนบล็อกเท่ากับจำนวนบัฟเฟอร์ที่ว่างอยู่

(2) พิจารณาทีละเรคคอร์ดเพื่อตัดแอททริบิวต์ที่ไม่ต้องการออกไป (คัดไว้เฉพาะแอททริบิวต์ที่ระบุในข้อคำถาม)

(3) เรียงลำดับข้อมูล

(4) บันทึกบล็อกที่ข้อมูลเรียงลำดับแล้ว กลับไปที่ดิสก์

(5) ทำซ้ำขั้นตอนที่ (1)-(4) จนกว่าจะหมดข้อมูล (ถ้าทำซ้ำ 3 รอบจะได้ข้อมูล 3 ส่วนย่อยที่แต่ละส่วนย่อยมีข้อมูลเรียงลำดับแล้ว)

ช่วงที่ 2: กำจัดข้อมูลซ้ำ

(1) นำข้อมูลบล็อกแรกของแต่ละส่วนย่อยจากดิสก์มาที่บัฟเฟอร์ในหน่วยความจำหลัก

(2) พิจารณาเปรียบเทียบทูพเพิลแรกของแต่ละบัฟเฟอร์ เพื่อนำทูพเพิลที่มีค่าน้อยที่สุดส่งออกเป็นผลลัพธ์ ถ้ามีค่าซ้ำ จะเลือกมาเพียงทูพเพิลเดียว ทูพเพิลที่ได้รับการพิจารณาแล้ว จะถูกลบออกไปจากบัฟเฟอร์ และเลื่อนทูพเพิลถัดไปมาที่ต้นบัฟเฟอร์แทนที่ จากนั้นทำซ้ำในขั้นตอนที่ (2) ถ้าข้อมูลบัฟเฟอร์ใดหมดจะนำข้อมูลบล็อกถัดไปของส่วนย่อยนั้น ๆ มาจากดิสก์

จำนวนครั้งที่ต้องติดต่อกับดิสก์ =

จำนวนบล็อกทั้งหมดของตาราง Employee (ช่วงโปรเจ็คและเรียงลำดับข้อมูล) +
จำนวนบล็อกทั้งหมดของตาราง Employee (บันทึกส่วนย่อยลงดิสก์) +
จำนวนบล็อกทั้งหมดของตาราง Employee (ดึงแต่ละส่วนย่อยมากำจัดตัวซ้ำและสร้างผลลัพธ์)

= 500 + 500 + 500

= 1500 ครั้ง

อัลกอริทึม P3: ผลลัพธ์ของการโปรเจ็คอาจมีข้อมูลซ้ำ กำจัดข้อมูลซ้ำด้วยเทคนิคแฮชชิง

ช่วงที่ 1: กระจายข้อมูลไปไว้ในที่ฝากข้อมูล (bucket)

1. อ่านข้อมูลจากดิสก์มาทีละบล็อก
2. พิจารณาทีละเรคคอร์ด เพื่อตัดแอททริบิวต์ที่ไม่ต้องการออกไป (คัดไว้เฉพาะแอททริบิวต์ ที่ระบุในข้อคำถาม)

3. ใช้แอททริบิวต์ที่เหลือทั้งหมดของเรคคอร์ดเป็นค่าคีย์ นำไปผ่านฟังก์ชันแบบแฮชเพื่อหาตำแหน่งที่ฝากข้อมูล ถ้าที่ฝากข้อมูลนั้นเต็มให้ถ่ายโอนข้อมูลจากที่ฝากข้อมูลไปยังดิสก์เพื่อทำให้ที่ฝากข้อมูลในหน่วยความจำหลักมีที่ว่าง
4. ทำซ้ำขั้นตอนที่ 1-3 จนหมดข้อมูล
5. ถ่ายโอนข้อมูลจากที่ฝากข้อมูลแต่ละที่ในหน่วยความจำไปยังแต่ละที่ฝากข้อมูลในดิสก์

ช่วงที่ 2: กำจัดข้อมูลซ้ำ

1. อ่านข้อมูลจากดิสก์มาคราวละหนึ่งที่ฝากข้อมูล
2. ในช่วงของการกระจายข้อมูลไปไว้ในที่ฝากข้อมูล เรคคอร์ดที่ซ้ำกันจะถูกแฮชไปไว้ในที่ฝากข้อมูลเดียวกัน ดังนั้นเมื่อตรวจดูข้อมูลทีละหนึ่งที่ฝากข้อมูลและกำจัดข้อมูลซ้ำออกไปสิ่งที่เหลือคือผลลัพธ์ของการทำ projection ที่ไม่มีข้อมูลซ้ำ

ตัวอย่างข้อคำถาม

```
SELECT   DISTINCT Ename, dept_code
FROM     Employee ;
```

$\delta(\pi_{\text{Eid, Ename, salary}}(\text{Employee}))$

จำนวนครั้งที่ต้องติดต่อกับดิสก์

= จำนวนบล็อกทั้งหมดของตาราง Employee (ช่วงกระจายข้อมูล) +
จำนวนบล็อกทั้งหมดของตาราง Employee (บันทึกแต่ละที่ฝากข้อมูลลงดิสก์) +
จำนวนบล็อกทั้งหมดของตาราง Employee (อ่านแต่ละที่ฝากข้อมูลจากดิสก์มากำจัดข้อมูลซ้ำ)

= 500 + 500 + 500
= 1500 ครั้ง

ระบบจัดการฐานข้อมูล Informix ใช้เทคนิคแฮชชิงในการทำ projection ในขณะที่ IBM DB2 และ Oracle ใช้เทคนิคการเรียงลำดับ ส่วน Microsoft SQL server ใช้ทั้งสองเทคนิค

2.2.3 คำสั่ง join

อัลกอริทึม J1: Simple nested loops join

R (X, Y) l><l S (Y, Z)

1. อ่านข้อมูลแต่ละทูพเพิล r จากความสัมพันธ์ R
2. สำหรับแต่ละ r ให้อ่านข้อมูลแต่ละทูพเพิล s จากความสัมพันธ์ S
   ถ้า r. Y = s.Y ให้เชื่อม r และ s เป็นหนึ่งทูพเพิลบันทึกลงในตารางที่เป็นผลลัพธ์

ตัวอย่างข้อคำถาม

```
SELECT  *                                  |
FROM    Employee E, Works_In W             |    Employee l><l Works_In
WHERE   E.Eid = W.Eid                      |
```

จำนวนครั้งที่ต้องติดต่อกับดิสก์

= จำนวนครั้งที่ต้องอ่านทุกบล็อกของ Employee +
จำนวนทูพเพิลทั้งหมดของ Employee × จำนวนบล็อกของWork_In
(เนื่องจากเมื่ออ่านแต่ละทูพเพิล e ของตาราง Employee ต้องอ่านทุกบล็อกของ Works_In เพื่อค้นหาทูพเพิลที่สามารถ join กับ e ได้)

= 500 + 40,000 × 1,000

= 40,000,500 ครั้ง

ถ้าการอ่านข้อมูลจากดิสก์แต่ละครั้งใช้เวลาเฉลี่ย 10 มิลลิวินาที การประมวลผลคำสั่ง join นี้จะใช้เวลาประมาณ 111 ชั่วโมง

ถ้าเราเปลี่ยนอัลกอริทึมให้อ่านข้อมูลจากตาราง Employee มาคราวละบล็อก และเรียกข้อมูลจากตาราง Works_In ขึ้นมาทีละบล็อกเพื่อ join ทูพเพิลในบล็อกของ Employee และ Works_In ที่มีค่า Eid ตรงกัน (แทนที่จะพิจารณาทีละทูพเพิลของ Employee เพื่อ join กับ Works_In) จะทำให้

จำนวนครั้งที่ต้องติดต่อกับดิสก์ = จำนวนครั้งที่ต้องอ่านทุกบล็อกของ Employee +
จำนวนบล็อกทั้งหมดของ Employee × จำนวนบล็อกของ Works_In

= 500 + 500 × 1,000

= 500,500 ครั้ง

ซึ่งจะทำให้เวลาในการประมวลผลลดลงเหลือประมาณ 1 ชั่วโมง 23 นาที

อัลกอริทึม J2: Block nested loops join

R (X, Y) l><l S (Y, Z)

อัลกอริทึม J1 ใช้เนื้อที่เพียง 2 บัฟเฟอร์เพื่อบรรจุข้อมูล 1 บล็อกจากตาราง R และ 1 บล็อกจากตาราง S ถ้าเรามีเนื้อที่บัฟเฟอร์มากกว่านั้น เช่น M บัฟเฟอร์ จะสามารถปรับปรุงอัลกอริทึมให้ทำงานได้เร็วขึ้น โดยบรรจุข้อมูลจากตาราง R ลงใน M-1 บัฟเฟอร์ และบรรจุ 1 บล็อกของ S ลงในอีก 1 บัฟเฟอร์ที่เหลืออยู่ ดังอัลกอริทึมต่อไปนี้

1. แบ่งข้อมูลตาราง R ออกเป็นส่วนละ M-1 บล็อก เพื่อบรรจุลงใน M-1 บัฟเฟอร์
2. อ่านแต่ละส่วนของ R มาไว้ในบัฟเฟอร์และจัดเป็นโครงสร้างที่จะช่วยให้ค้นหาข้อมูลได้รวดเร็ว (เช่น ใช้ตารางแฮช หรือใช้ search tree) โดยมีแอททริบิวต์ร่วม (ได้แก่ Y) เป็นค่าคีย์
3. สำหรับแต่ละส่วนของ R ให้อ่านทีละบล็อกของ S มาไว้ในหนึ่งบัฟเฟอร์ที่เหลืออยู่ เพื่อพิจารณาทูพเพิลของ S ที่สามารถ join กับ R และแสดงเป็นผลลัพธ์ของการ join

ถ้าให้ B(R) คือจำนวนบล็อกทั้งหมดของ R และ B(S) คือจำนวนบล็อกทั้งหมดของ S จากอัลกอริทึม เราแบ่ง R ออกเป็นส่วนละ M-1 บล็อก ดังนั้น R จะมี $\frac{B(R)}{M-1}$ ส่วน ในแต่ละส่วนของ R เราอ่าน M-1 บล็อกจาก R และอ่านทุกบล็อกจาก S เพื่อนำข้อมูลมา join กับแต่ละส่วนของ R ดังนั้น จำนวนครั้งทั้งหมดที่ต้องติดต่อกับดิสก์คือ $\frac{B(R)}{M-1}(M-1+B(S))$

ตัวอย่างข้อคำถาม

SELECT * FROM Employee E, Works_In W WHERE E.Eid = W.Eid	Employee l><l Works_In

ถ้ามีเนื้อที่ว่างในหน่วยความจำ 101 บัฟเฟอร์
จำนวนครั้งที่ต้องติดต่อกับดิสก์

= (จำนวนบล็อกทั้งหมดของ Works_In /100) × (100 + จำนวนบล็อกทั้งหมดของ Works_In)

= $\frac{500}{100}(100+1{,}000)$ = 5,500 ครั้ง

ถ้าเราเปลี่ยนลำดับของการ join เป็น Works_In l><l Employee

จำนวนครั้งที่ต้องติดต่อกับดิสก์

= (จำนวนบล็อกทั้งหมดของ Employee /100) × (100 + จำนวนบล็อกทั้งหมดของ Employee)

$$= \frac{1000}{100}(100+500)$$

= 6,000 ครั้ง

จะเห็นได้ว่าการเลือกตารางที่มีขนาดเล็กกว่าเพื่อแบ่งเป็นส่วนตามจำนวนบัฟเฟอร์ว่างที่มีอยู่ (นั่นคือเป็นลูปชั้นนอกตามอัลกอริทึม J2) และเลือกตารางที่มีขนาดใหญ่กว่าเพื่ออ่านทีละบล็อกมาบรรจุใน 1 บัฟเฟอร์ที่เหลืออยู่ (เป็นลูปชั้นในตามอัลกอริทึม J2) จะใช้เวลาในการประมวลผลที่เร็วกว่า

อัลกอริทึม J3 : Indexed nested loops join

R (X, Y) l><l S(Y, Z)

สมมุติให้มีดรรชนีบนแอททริบิวต์ Y ของตาราง S

1. อ่านข้อมูล R ทีละบล็อก และพิจารณาทุกทูพเพิล r ในบล็อก
2. ใช้ค่าแอททริบิวต์ Y ของทูพเพิล r (r.Y) เป็นคีย์ค้นหา (ด้วยดรรชนี) ทูพเพิลของ S ที่มีค่าแอททริบิวต์ Y ตรงกับ r.Y เพื่อ join ทูพเพิลและแสดงเป็นผลลัพธ์

จำนวนครั้งที่ต้องติดต่อกับดิสก์ = จำนวนบล็อกทั้งหมดของ R + จำนวนข้อมูลใน S ที่มีค่าแอททริบิวต์ Y ตรงกับ r.Y

การค้นหาข้อมูลใน S ด้วย B^+- tree index จากโหนดรากไปถึงโหนดใบ จะต้องอ่านข้อมูลจากดิสก์ประมาณ 2 ถึง 4 ครั้ง (เพราะโดยทั่วไปโครงสร้าง B^+-tree มักจะมี 2 ถึง 4 ระดับ) และการค้นหาจากโหนดใบไปยังตัวข้อมูล จะต้องอ่านข้อมูลจากดิสก์กี่ครั้งนั้นขึ้นอยู่กับประเภทของดรรชนีว่าเป็นชนิดเข้ากลุ่ม (clustering) หรือไม่เข้ากลุ่ม (nonclustering) ถ้าเป็นชนิดไม่เข้ากลุ่ม จะต้องใช้เวลาค่อนข้างมากเพราะข้อมูลที่ต้องการอาจกระจายอยู่ในหลายบล็อก

ตัวอย่างข้อคำถาม

SELECT * FROM Employee E, Works_In W WHERE E.Eid = W.Eid	Employee l><l Works_In

กรณีที่ 1: สมมุติให้มี B^+-tree index บนแอททริบิวต์ W.Eid และเป็นดรรชนีชนิดเข้ากลุ่ม

(1) อ่านข้อมูลทั้งหมดของตาราง Employee ข้อมูลทั้งหมดนี้เก็บอยู่ใน 500 บล็อก จึงต้องติดต่อกับดิสก์ 500 ครั้ง

(2) พิจารณาแต่ละค่าของ E.Eid นำค่า Eid นี้ไปค้นหาค่าที่ตรงกันใน B^+-tree index ของ Works_In ถ้า B^+-tree มี 3 ระดับชั้น (จากโหนดรากถึงโหนดใบ) จะต้องติดต่อกับดิสก์ 3 ครั้ง และต้องอ่านดิสก์อีก 1 ครั้ง เพื่อดึงข้อมูลบล็อกที่ต้องการจากตาราง Works_In มาเพื่อทำการ join รวมแล้วการ join สำหรับ E.Eid หนึ่งค่าจะต้องติดต่อกับดิสก์ 4 ครั้ง E.Eid มีทั้งหมด 40,000 ค่า (เนื่องจากตาราง Employee มีข้อมูล 40,000 ทูพเพิล) จึงต้องติดต่อกับดิสก์ = 40,000 x 4 = 160,000 ครั้ง

(3) รวมจำนวนครั้งที่ต้องติดต่อกับดิสก์ในข้อ (1) และ (2) = 500 + 160,000 = 160,500 ครั้ง

กรณีที่ 2: สมมุติให้มี B^+-tree index บนแอททริบิวต์ W.Eid แต่เป็นดรรชนีชนิดไม่เข้ากลุ่ม

(1) จำนวนครั้งที่ต้องติดต่อกับดิสก์เพื่ออ่านข้อมูล Employee = 500 ครั้ง

(2) พนักงานหนึ่งคนสามารถทำงานหลายโปรเจ็คได้ Employee มีข้อมูล 40,000 ทูพเพิล และ Works_In มีข้อมูล 100,000 ทูพเพิล ถ้าการกระจายพนักงานไปทำงานในโปรเจ็คต่างๆ เป็นไปอย่างสม่ำเสมอ พนักงาน 1 คน จะทำงานโดยเฉลี่ย = $\frac{100,000}{40,000}$ = 2.5 โปรเจ็ค และเนื่องจากดรรชนีบน W.Eid เป็นแบบไม่เข้ากลุ่มจึงมีโอกาสที่ข้อมูลการทำงานในโปรเจ็คต่างๆของพนักงานหนึ่งคนจะกระจายไปอยู่ต่างบล็อกกัน การคำนวณจำนวนครั้งที่ต้องติดต่อกับดิสก์เพื่อการ join จึงต้องแยกพิจารณาเป็นสองส่วน คือ

ส่วนที่ 1 การค้นหาในส่วนดรรชนีของ B^+-tree

B^+-tree มี 3 ระดับ และข้อมูล Employee มี 40,000 ทูพเพิล จึงต้องติดต่อกับดิสก์ = 40,000 × 3 = 120,000 ครั้ง

ส่วนที่ 2 การค้นหาในส่วนข้อมูล

มีข้อมูล Employee จำนวน 40,000 ทูพเพิล ข้อมูลการร่วมงานในแต่ละโปรเจ็คของพนักงานแต่ละคนอาจจะกระจายอยู่ใน 2.5 บล็อก ดังนั้นจึงต้องอ่านข้อมูลจากบล็อก = 40,000 × 2.5 = 100,000 บล็อก

รวมการติดต่อกับดิสก์ (ในทั้งสองส่วน) = 120,000 + 100,000 = 220,000 ครั้ง

(3) รวมจำนวนครั้งที่ต้องติดต่อกับดิสก์ในข้อ (1) และ (2) = 500 + 220,000 = 220,500 ครั้ง

กรณีที่ 3 : สมมุติให้มี B^+-tree index บนแอททริบิวต์ E.Eid

(สลับลำดับการ join เป็น Works_In l><l Employee)

(1) อ่านข้อมูลทั้งหมดของตาราง Works_In ข้อมูลบรรจุอยู่ใน 1,000 บล็อก จึงต้องติดต่อกับดิสก์ 1,000 ครั้ง

(2) นำแต่ละค่าของ W.Eid มาค้นหาผ่านดรรชนีของตาราง Employee ถ้าดรรชนีมี 3 ระดับชั้น (จากโหนดรากถึงโหนดใบ) จะต้องติดต่อกับดิสก์ 3 ครั้ง และต้องอ่านดิสก์อีก 1 ครั้ง เพื่อดึงข้อมูลบล็อกที่ต้องการจากตาราง Employee เนื่องจากแอททริบิวต์ Eid เป็นกุญแจหลักของตาราง Employee จึงไม่มีค่า Eid ซ้ำ ดังนั้นแต่ละครั้งของการ join จะมี E.Eid อย่างมากที่สุดเพียงค่าเดียวที่จะนำไป join กัน W.Eid ได้ (จึงไม่สำคัญว่าดรรชนีจะเป็นแบบเข้ากลุ่มหรือไม่เข้ากลุ่ม)

รวมแล้วการ join สำหรับ W.Eid หนึ่งค่าจะต้องติดต่อกับดิสก์ 4 ครั้ง W.Eid มีทั้งหมด 100,000 ทูพเพิล จึงต้องติดต่อกับดิสก์ = 100,000 × 4 = 400,000 ครั้ง

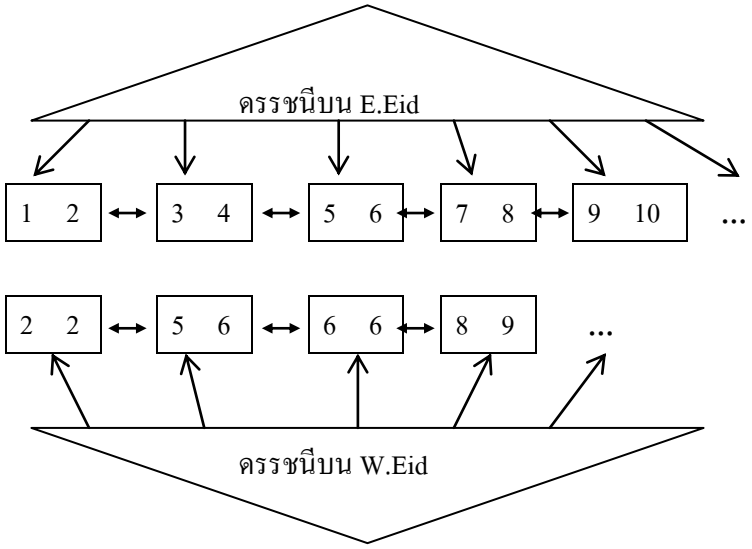
(3) รวมจำนวนครั้งที่ต้องติดต่อกับดิสก์ในข้อ (1) และ (2) = 1,000 + 400,000 = 401,000 ครั้ง

<u>หมายเหตุ</u> จากการวิเคราะห์จำนวนครั้งที่ต้องติดต่อกับดิสก์ในทั้งสามกรณีจะเห็นว่าการอ่านข้อมูลทั้งหมดจากตารางที่จะทำหน้าที่เป็นลูปชั้นนอกในขั้นตอนที่ (1) จะมีค่าน้อยกว่าจำนวนครั้งที่ต้องติดต่อกับดิสก์ในขั้นตอนที่ (2) มาก จึงอาจละการคำนวณในขั้นตอนที่ (1) ได้

กรณีที่ 4 : สมมุติให้มี B^+- tree index บนแอททริบิวต์ E.Eid และ W.Eid

ถ้าดรรชนีบน E.Eid และ W.Eid เรียงลำดับแล้ว สามารถใช้การ join แบบซิกแซก (zig-zag join) ดังตัวอย่างในรูปที่ 2.9 การ join จะใช้การสแกนค่า Eid บนดรรชนีทั้งสอง (E.Eid และ W.Eid) พร้อมกัน เพื่อหาค่า Eid ที่ตรงกันเมื่อพบค่า Eid ที่ตรงกัน (เช่นค่า 2 ดังรูปที่ 2.9) จะอ่านข้อมูลจากตาราง Employee และ Works_In เพื่อมา join กัน จากนั้นจะสแกนค่า Eid บนดรรชนีต่อไป จนกว่าจะพบค่า Eid คู่ต่อไปที่มีค่าตรงกัน การ join ด้วยวิธีนี้จะใช้จำนวนครั้งของการติดต่อกับดิสก์ ไม่เกินค่าผลบวกของจำนวนบล็อกข้อมูลทั้งหมดของทั้งสองตาราง นั่นคือ 500 + 1,000 = 1,500 ครั้ง ซึ่งจะมีค่าน้อยกว่าการ join ด้วยวิธีอื่นๆ ข้างต้น ในการประมวลผลคำสั่ง join ถ้าข้อมูลไม่มีดรรชนีบนแอททริบิวต์ที่จะใช้ในการ join

ระบบจัดการฐานข้อมูลอาจจะพิจารณาสร้างดรรชนีขึ้นเป็นการเฉพาะเพราะอาจจะช่วยให้การประมวลผลคำสั่งนั้นๆ เร็วขึ้น



รูปที่ 2.9 ตัวอย่างการ join แบบซิกแซก

อัลกอริทึม J4: Sort - merge join

R (X,Y) l><l S (Y, Z)

1. เรียงลำดับข้อมูลในตาราง R และ S โดยใช้แอททริบิวต์ Y เป็นหลักในการเรียงลำดับ
2. ตรวจทุกทูพเพิลใน R และ S ไปพร้อมกัน เพื่อหาทูพเพิลที่มีค่าแอททริบิวต์ Y ตรงกันและทำการ join

ตัวอย่างข้อคำถาม

```
SELECT   *
FROM     Employee  E, Works_In  W
WHERE    E.Eid = W.Eid
```

Employee l><l Works_In

ตัวอย่างข้อมูลของตาราง Employee และ Works_In แสดงได้ดังตารางที่ 2.1 และ 2.2 ตามลำดับ และผลลัพธ์ของ Employee l><l Works_In แสดงได้ดังตารางที่ 2.3

ตารางที่ 2.1 ตัวอย่างข้อมูลของตาราง Employee

Eid	Ename	gender	birthdate	address	salary	dept_code
18	Dan	M	N/A	NY	4000	01
19	Jane	F	N/A	LA	5000	04
20	Mark	M	N/A	FL	4500	05
25	Paul	M	N/A	NY	4800	01
27	Bob	M	N/A	LA	6000	04

ตารางที่ 2.2 ตัวอย่างข้อมูลของตาราง Works_In

Eid	Pno	role	start_date	finish_date
19	20	Sup	01/01/00	31/12/01
19	21	SA	01/01/00	31/12/02
25	8	Sup	01/02/99	01/01/01
25	10	SA	01/03/99	31/12/02
25	12	SA	01/01/01	31/12/01

ตารางที่ 2.3 ข้อมูลที่เป็นผลลัพธ์ของการ join ตาราง Employee และ Works_In

Eid	Ename	gender	birth date	address	salary	dept_ code	Pno	role	start_ date	finish_ date
19	Jane	F	N/A	LA	5000	04	20	Sup	01/01/00	31/12/01
19	Jane	F	N/A	LA	5000	04	21	SA	01/01/00	31/12/02
25	Paul	M	N/A	NY	4800	01	8	Sup	01/02/99	01/01/01
25	Paul	M	N/A	NY	4800	01	10	SA	01/03/99	31/12/02
25	Paul	M	N/A	NY	4800	01	12	SA	01/01/01	31/12/01

ในขั้นตอนที่ (1) ของอัลกอริทึม Sort-merge join แต่ละบล็อกของข้อมูล Employee และ Works_In จะถูกอ่านมายังบัฟเฟอร์เพื่อเรียงลำดับข้อมูลตามค่า Eid จากนั้นบล็อกที่ข้อมูลเรียงลำดับตามค่า Eid แล้วจะถูกบันทึกกลับไปยังดิสก์ ดังนั้นแต่ละบล็อกจะถูกอ่านหนึ่งครั้ง และถูกบันทึกหนึ่งครั้ง

ในขั้นตอนที่ (2) แต่ละบล็อกของ Employee และ Works_In จะถูกอ่านมายังบัฟเฟอร์เพื่อทำการ join

ข้อมูล Employee บันทึกอยู่ใน 500 บล็อก และข้อมูล Works_In บันทึกอยู่ใน 1,000 บล็อก ดังนั้นจำนวนครั้งที่ต้องติดต่อกับดิสก์ = 3(500 + 1,000) = 4,500 ครั้ง

ในกรณีที่ตาราง Employee เรียงลำดับข้อมูลตามค่า Eid อยู่แล้ว จะช่วยให้สามารถลดเวลาในขั้นตอนที่ (1) และทำให้จำนวนครั้งที่ต้องติดต่อกับดิสก์ = 500 + 3 × 1,000 = 3,500 ครั้ง

อัลกอริทึม J5: Hash join

R(X, Y) l><l S (Y, Z)

1. อ่านข้อมูลทุกทูพเพิลจากตาราง R ใช้แอททริบิวต์ Y เป็นคีย์ นำไปคำนวณด้วยฟังก์ชันแบบแฮช เพื่อหาตำแหน่งที่ฝากข้อมูล (bucket) ในตารางแฮช และบันทึกตารางแฮชลงดิสก์
2. อ่านข้อมูลทุกทูพเพิลจากตาราง S ใช้แอททริบิวต์ Y เป็นคีย์ นำไปคำนวณด้วยฟังก์ชันเดียวกับที่ใช้ในข้อ (1) เพื่อบรรจุทูพเพิลลงตารางแฮช และบันทึกตารางแฮชของ S ลงดิสก์
3. อ่านแต่ละคู่ bucket ที่หมายเลข bucket ตรงกัน จากตารางแฮชของ R และตารางแฮชของ S เพื่อ join ทูพเพิล (เนื่องจากถ้าทูพเพิล r ที่มีค่า Y = i ถูกแฮชไปอยู่ที่ bucket k ทูพเพิล s ที่มีค่า Y = i ก็จะถูกแฮชไปที่ bucket k เช่นเดียวกัน)

ตัวอย่างข้อคำถาม

```
SELECT  *
FROM    Employee E, Works_In W
WHERE   E.Eid = W.Eid
```

Employee l><l Works_In

จำนวนครั้งที่ต้องติดต่อกับดิสก์

= จำนวนครั้งที่ต้องอ่านทุกบล็อกของ Employee และ Works_In เพื่อทำแฮชชิง + จำนวนครั้งที่ต้องบันทึกทุก bucket ของตารางแฮชของ Employee และ Works_In ลงดิสก์ + จำนวนครั้งที่ต้องอ่านทีละคู่ bucket เพื่อมา join ข้อมูล

= 1,500 + 1,500 + 1,500

= 4,500 ครั้ง

อัลกอริทึม J6: Hybrid hash join

R (X, Y) l><l S (Y, Z)

ถ้าเรามีเนื้อที่บัฟเฟอร์มาก สามารถปรับปรุงอัลกอริทึม hash join ให้ทำงานได้เร็วขึ้นดังนี้

1. ถ้าตาราง S มีขนาดเล็กกว่า ให้อ่านทุกทูพเพิลของ S นำค่า Y ไปผ่านฟังก์ชันแบบแฮชเพื่อหาตำแหน่ง bucket เก็บ bucket จำนวนหนึ่งไว้ในบัฟเฟอร์ (เท่าที่จะมีเนื้อที่บัฟเฟอร์ว่าง) bucket ที่เกินจำนวนบัฟเฟอร์ว่าง ให้บันทึกลงดิสก์เหมือนใน อัลกอริทึม hash join
2. อ่านข้อมูลแต่ละทูพเพิลจากตาราง R นำค่า Y ไปผ่านฟังก์ชันแบบแฮชเพื่อหาตำแหน่ง bucket
   - ถ้าตำแหน่ง bucket ตรงกับ bucket ของตาราง S ที่อยู่ในบัฟเฟอร์ให้ join ข้อมูลทั้งสองทูพเพิลได้ทันที
   - ถ้าตำแหน่ง bucket ไม่ตรงกับ bucket ของตาราง S ที่อยู่ในบัฟเฟอร์ ให้บันทึก bucket ลงดิสก์
3. อ่านแต่ละคู่ bucket ของ S และ R จากดิสก์ มาเพื่อทำการ join

ตัวอย่างข้อคำถาม

```
SELECT   *
FROM     Employee  E, Works_In  W
WHERE    E.Eid = W.Eid
```

Employee l><l Works_In

ข้อมูลในตาราง Employee ใช้เนื้อที่เก็บข้อมูล 500 บล็อก ข้อมูล Works_In ใช้เนื้อที่เก็บข้อมูล 1,000 บล็อก ถ้ามีเนื้อที่ว่างในบัฟเฟอร์ บรรจุข้อมูลได้ 300 บล็อก

(1) แบ่งข้อมูล Employee เป็นสองส่วน ครึ่งแรก (250 บล็อก) เมื่อนำทูพเพิลไปผ่านฟังก์ชันแบบแฮชแล้วเก็บข้อมูลใน bucket ไว้ในบัฟเฟอร์ อีกครึ่งที่เหลือ (250 บล็อก) เมื่อทำแฮชชิงแล้วบันทึก bucket ลงดิสก์ ดังนั้น

จำนวนครั้งที่ต้องติดต่อกับดิสก์ = จำนวนครั้งที่ต้องอ่านข้อมูลบล็อกของ Employee เพื่อทำแฮชชิง + จำนวนครั้งที่ต้องบันทึก bucket ลงดิสก์

= 500 + 250

= 750 ครั้ง

(2) อ่านข้อมูลจากตาราง Works_In เพื่อทำแฮชชิง ถ้าข้อมูลมีการกระจายอย่างสม่ำเสมอ (uniform distribution) จะมีข้อมูลประมาณครึ่งหนึ่งที่สามารถ join กับข้อมูลครึ่งแรกของ Employee ที่คงค้างไว้ในบัฟเฟอร์ และแสดงเป็นผลลัพธ์ แต่จะมีข้อมูลอีกครึ่งที่เหลือ (ประมาณ 500 บล็อก) ที่จะต้องรอ join กับข้อมูล Employee ในดิสก์ ข้อมูล Wors_In ที่ต้องรอ join ในรอบต่อไปจะถูกบันทึกลงดิสก์ ดังนั้น

จำนวนครั้งที่ต้องติดต่อกับดิสก์ = จำนวนครั้งที่ต้องอ่านข้อมูลทุกบล็อกของ Works_In เพื่อทำแฮชชิง + จำนวนครั้งที่ต้องบันทึก bucket ที่ยังไม่ได้ join ลงดิสก์

= 1,000 + 500 = 1,500 ครั้ง

(3) อ่านข้อมูล bucket ที่เหลือในตารางแฮชของ Employee และ Works_In มากเพื่อทำการ join

จำนวนครั้งที่ต้องติดต่อกับดิสก์ = จำนวนครั้งที่ต้องอ่านแต่ละ bucket จาก Employee + จำนวนครั้งที่ต้องอ่านแต่ละ bucket จาก Works_In

= 250 + 500 = 750 ครั้ง

รวมการทำงานทั้ง 3 ขั้นตอน จะได้ว่าจำนวนครั้งที่ต้องติดต่อกับดิสก์

= 750 + 1,500 + 750 = 3,000 ครั้ง

จะเห็นได้ว่าเมื่อเปรียบเทียบกับ hash join วิธี hybrid hash join สามารถลดจำนวนครั้งที่ต้องติดต่อกับดิสก์ลงได้ค่อนข้างมาก ยิ่งถ้ามีจำนวนบัฟเฟอร์ในหน่วยความจำมาก วิธี hybrid hash join จะยิ่งให้ผลดีที่เห็นได้ชัดเจน เช่น ถ้ามีบัฟเฟอร์มากพอที่จะเก็บ bucket ของ Employee ไว้ได้ทั้งหมด จำนวนครั้งที่ต้องติดต่อกับดิสก์จะลดลงเหลือเพียง = 500 + 1,000 = 1,500 ครั้ง

2.3 ภาษาสอบถามข้อมูล SQL และ Datalog

ภาษาที่นิยมใช้สอบถามข้อมูลจากฐานข้อมูลมักจะเป็นภาษา SQL เนื่องจากมีรูปแบบที่ง่ายต่อการจดจำ คำสั่งสอบถามข้อมูลตามมาตรฐานของ SQL มีรูปแบบดังนี้

```
SELECT   [ALL | DISTINCT]  column_list
FROM          table_list
[WHERE        conditional_expression]
[GROUP BY     group_by_column_list]
[HAVING       conditional_expression]
[ORDER BY     order_by_column_list]
```

คำสั่งส่วนที่อยู่ภายในวงเล็บ [ ] สามารถละเว้นได้ และลำดับในการทำงานตามคำสั่ง SQL เป็นดังรูปที่ 2.10

รูปที่ 2.10 ลำดับการประมวลผลคำสั่งสอบถามข้อมูล SQL

ภาษา SQL เป็นภาษาที่ใช้สอบถามข้อมูลจากฐานข้อมูลเชิงสัมพันธ์ที่บันทึกข้อมูลอยู่ในลักษณะของตาราง ข้อมูลที่บันทึกเป็นได้ทั้งข้อมูลฐาน (base table) และวิวข้อมูล (materialized view) แต่ในงานวิจัยนี้จะพัฒนาและทดสอบอัลกอริทึมการเพิ่มประสิทธิภาพการประมวลผลข้อคำถามบนฐานข้อมูลนิรนัย (deductive database) ที่นอกจากจะสามารถบันทึกข้อมูลฐานและวิวข้อมูลแล้ว ยังสามารถบันทึก semantic rules รวมอยู่ในฐานข้อมูลและมีวิธีการใช้งานร่วมกับข้อมูลประเภทอื่นๆได้สะดวกกว่าฐานข้อมูลเชิงสัมพันธ์

ฐานข้อมูลนิรนัยที่ใช้ในงานวิจัยนี้คือฐานข้อมูล DES (Datalog Educational System) เวอร์ชัน 2.0 (released on August 2010) ที่สามารถใช้ทั้งภาษา SQL และดาต้าล็อก (Datalog) เป็นภาษาสำหรับการสอบถามและประมวลผลข้อมูล ภาษาดาต้าล็อกใช้ตรรกศาสตร์อันดับหนึ่งเป็นพื้นฐานของภาษา ข้อมูลที่บันทึกอยู่ในฐานข้อมูลนิรนัยจะประกอบด้วย extensional database และ intensional database

extensional database คือข้อมูลที่อยู่ในลักษณะของข้อความที่เป็นจริง หรือ ground fact เช่น employee(18, ‘Dan’, ‘M’,’N/A’,’NY’,4000,01) ซึ่งเป็นข้อมูลพนักงานคนแรกตามตารางที่ 2.1 ข้อมูลที่เรียกว่า intensional database จะหมายถึงข้อความที่อยู่ในรูปของกฎถ้า-แล้ว เช่น

highly_paid_employee(X) :- employee(X, _, _, _, _, Salary, _), Salary > 5000.

เป็นการระบุลักษณะ (โดยไม่ต้องแจกแจงข้อมูลทุกเรคคอร์ด) ของ highly_paid_employee ว่าหมายถึงพนักงานที่มีเงินเดือนสูงกว่า 5,000 ดอลลาร์ ข้อความในรูปแบบ p(X) :- q(Y) ของดาต้าล็อกจะตรงกับรูปแบบของ p(X) ← q(Y) ในตรรกศาสตร์อันดับหนึ่งซึ่งหมายถึง ถ้า q(Y) เป็นจริงแล้ว p(X) จะเป็นจริง รูปที่ 2.11 เปรียบเทียบการเก็บข้อมูลในลักษณะตารางและวิวของฐานข้อมูลเชิงสัมพันธ์ กับลักษณะของข้อความหรือคลอส (clause) ในฐานข้อมูลนิรนัย

Base table

Eid	Ename	gender	birthdate	address	salary	dept_code
18	Dan	M	N/A	NY	4000	01
19	Jane	F	N/A	LA	5000	04
20	Mark	M	N/A	FL	4500	05
25	Paul	M	N/A	NY	4800	01
27	Bob	M	N/A	LA	6000	04

View

```
CREATE VIEW HIGHLY_PAID_EMPLOYEE
AS SELECT EID
    FROM EMPLOYEE E
    WHERE E.SALARY > 5000;
```

Extensional databases:

employee(18, ‘Dan’, ‘M’,’N/A’,’NY’,4000,01).
employee(19, ‘Jane’, ‘F’,’N/A’,’LA’,5000,04).
employee(20, ‘Mark’, ‘M’,’N/A’,’FL’,4500,05).
employee(25, ‘Paul’, ‘M’,’N/A’,’NY’,4800,01).
employee(27, ‘Bob’, ‘M’,’N/A’,’LA’,6000,04).

Intensional database:

highly_paid_employee(X) :-
employee(X, _, _, _, _, Salary, _),
Salary > 5000.

รูปที่ 2.11 เปรียบเทียบลักษณะข้อมูลในฐานข้อมูลเชิงสัมพันธ์และฐานข้อมูลนิรนัย

การสอบถามข้อมูลด้วยภาษาดาต้าล็อกจะใช้รูปแบบของข้อความหรือคลอส เช่น ถ้าต้องการสอบถามรายชื่อพนักงานที่ทำงานอยู่ในแผนก 01 จะใช้รูปแบบข้อคำถามดังนี้

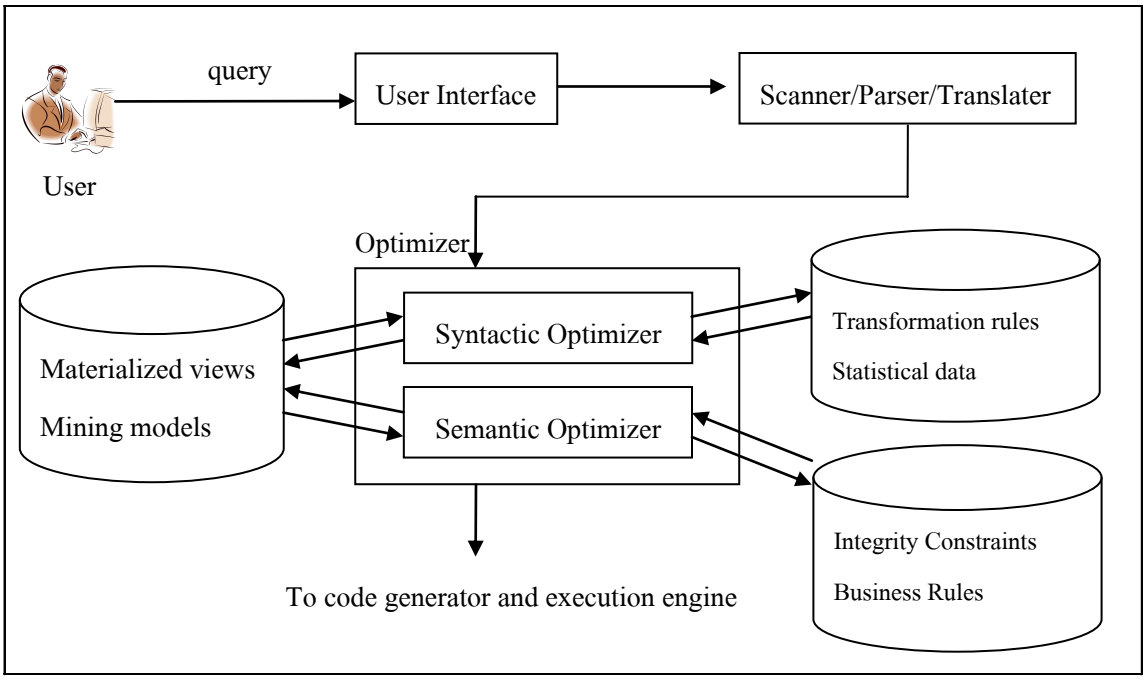
? employee(_, Name, _, _, _, _, 01).
Name = Dan;
Name= Paul.

คำตอบที่ได้คือมีพนักงานในแผนก 01 สองคนชื่อ Dan และ Paul

บทที่ 3
การเพิ่มประสิทธิภาพการประมวลผลข้อคำถาม

3.1 กรอบของงานวิจัย

วิธีการเพิ่มประสิทธิภาพการประมวลผลข้อคำถามสามารถจำแนกเป็น 2 กลุ่มคือ การเพิ่มประสิทธิภาพเชิงไวยากรณ์ (syntactic query optimization) และการเพิ่มประสิทธิภาพเชิงความหมาย (semantic query optimization) วิธีการเพิ่มประสิทธิภาพที่ใช้ในระบบจัดการฐานข้อมูลโดยทั่วไปจะเป็นวิธีการเพิ่มประสิทธิภาพเชิงไวยากรณ์ โครงการวิจัยนี้มีจุดมุ่งหมายที่จะพัฒนา query optimizer ให้สามารถใช้ความหมายและความสัมพันธ์ในข้อมูลมาช่วยเพิ่มประสิทธิภาพการประมวลผลข้อคำถาม ซึ่งวิธีการที่จะพัฒนาขึ้นนี้จัดอยู่ในกลุ่มวิธีการเพิ่มประสิทธิภาพเชิงความหมาย แนวทางการดำเนินงานวิจัย อธิบายได้ด้วยแผนภาพดังรูปที่ 3.1



รูปที่ 3.1 โครงสร้างส่วนเพิ่มประสิทธิภาพการประมวลผลข้อคำถามและข้อมูลที่เกี่ยวข้อง

ขั้นตอนต่างๆของการดำเนินงานวิจัยประกอบด้วย (๑) ศึกษาเทคนิคและวิธีการที่ใช้ในการเพิ่มประสิทธิภาพการประมวลผลข้อคำถามทั้งในเชิงไวยากรณ์และเชิงความหมาย, (๒) ออกแบบและพัฒนาวิธีการสังเคราะห์โมเดลข้อมูลจากฐานข้อมูล และวิธีการแปลงโมเดลให้อยู่ในรูปแบบของกฎหรือ semantic rules ที่เหมาะสมกับการนำไปใช้เพื่อแปลงรูปแบบข้อคำถาม, (๓) ออกแบบวิธีการใช้วิวข้อมูลและโมเดลจากการทำเหมืองข้อมูลที่อยู่ในรูปแบบของกฎ เพื่อแปลงรูปแบบข้อคำถามให้สามารถประมวลผลได้เร็วขึ้น, และ (๔) พัฒนาโปรแกรมต้นแบบ สร้างฐานข้อมูล รวมถึงกำหนดข้อคำถามที่จะใช้ในการทดสอบระบบ

3.2 วิธีการเพิ่มประสิทธิภาพการประมวลผลข้อคำถามด้วยโมเดลและวิวข้อมูล

การเพิ่มประสิทธิภาพการประมวลผลข้อคำถาม (query optimization) ในงานวิจัยนี้เน้นที่การเพิ่มประสิทธิภาพเชิงความหมาย ด้วยการเรียนรู้เพื่อสังเคราะห์โมเดลของข้อมูลและบันทึกโมเดลนั้นในลักษณะของกฎที่เรียกว่า semantic rules หรือ business rules เทคนิคการเรียนรู้โมเดลข้อมูลใช้วิธีการทำเหมืองข้อมูลแบบการค้นหาความสัมพันธ์ (association mining) ที่กำหนดให้แสดงโมเดลในลักษณะของกฎ “ถ้า-แล้ว” หรือ IF antecedent THEN conclusion ซึ่งในฐานข้อมูลนิรนัยจะแสดงกฎในรูปแบบ conclusion :- antecedent ที่มีความหมายเช่นเดียวกับการเขียนข้อความ conclusion ← antecedent

กฎที่ได้เป็นผลลัพธ์จะต้องมีค่าความเชื่อมั่น (confidence) ไม่ต่ำกว่า 1.0 หรือมีค่าความเชื่อมั่น 100% ในขั้นตอนการเรียนรู้เพื่อสร้างโมเดลในงานวิจัยนี้กำหนดให้ค่าสนับสนุน (support) มีค่าไม่ต่ำกว่า 0.7 หรือมีจำนวนข้อมูลอย่างน้อย 70% สนับสนุนกฎที่ได้ (หมายเหตุ ค่า support และ confidence มีค่าต่ำสุดและสูงสุดอยู่ระหว่าง 0.0 ถึง 1.0 ตามลำดับ) โดยค่าสนับสนุนนี้อาจปรับให้แตกต่างจากนี้ได้ตามความเหมาะสมสำหรับแต่ละฐานข้อมูล ค่าสนับสนุนที่ต่ำลงจะให้ผลการเรียนรู้เป็น semantic rules ที่มีจำนวนกฎที่มากขึ้น

นอกจากโมเดลข้อมูลในลักษณะของ semantic rules การพิจารณาเพิ่มประสิทธิภาพการประมวลผลข้อคำถามในงานวิจัยนี้ยังมีการใช้วิวข้อมูล (materialized view) เพื่อช่วยปรับเปลี่ยนรูปแบบข้อคำถามให้สามารถประมวลผลหาคำตอบได้เร็วขึ้น การสร้างวิวข้อมูลจะพิจารณาจากประวัติการตั้งข้อคำถามของผู้ใช้ ข้อคำถามใดที่ถูกใช้บ่อยจะได้รับการคัดเลือกให้สร้างเป็นวิวข้อมูลที่ทำหน้าที่เป็นข้อมูลชุดใหม่ในฐานข้อมูล ที่มีรูปแบบที่เหมาะสมและตรงกับข้อคำถามที่ผู้ใช้สนใจสอบถามบ่อยครั้ง อัลกอริทึมในการปรับปรุงข้อคำถามด้วยโมเดลและวิวข้อมูลแสดงได้ดังรูปที่ 3.2

Algorithm Semantic query optimizer

Input: a database D,
a set of semantic rules S,
a set of materialized views V,
current user's query Q

Output: a new query Q'

Steps:

1. Extract conditions C from the user's query Q
2. For each $c \in C$
3. Search for applicable semantic rules from S by
3.1 assert *c* as a temporary database fact
3.2 search for predicates in S that related to *c*
3.3 report searching result as an answer set A
4. If A is empty Then return Q, Else proceed to the next step
5. Form a new query Q' by
5.1 Construct a head of query clause with C appeared as arguments
5.2 Construct clause body with applicable materialized view from V
5.3 Conjunct a clause body with predicates appeared in A
6. Return a new query Q'

รูปที่ 3.2 อัลกอริทึม Semantic query optimizer

อัลกอริทึม Semantic query optimizer จะรับอินพุตเป็นข้อคำถาม Q แล้วให้ผลลัพธ์เป็นข้อคำถาม Q' ในรูปแบบของคลอส หรือข้อความตรรกศาสตร์อันดับหนึ่งในภาษา Datalog รวมถึง semantic rules และ materialized views ก็จะอยู่ในรูปแบบของคลอสเช่นเดียวกัน ในกรณีที่ข้อคำถาม Q ไม่มีเงื่อนไขใดที่สอดคล้องกับ semantic rules ข้อคำถาม Q' ที่ได้จากอัลกอริทึม จะมีรูปแบบเดียวกับ Q ที่เป็นข้อคำถามเริ่มต้น ทั้งนี้เนื่องจากเป็นกรณีที่ข้อคำถามไม่สามารถปรับปรุงได้ แต่ถ้าเป็นกรณีที่มี semantic rules ที่เกี่ยวข้องกับเงื่อนไขของข้อคำถาม (นั่นคือกรณีที่ answer set ที่ได้จากขั้นตอนที่ 3.1-3.3 ของอัลกอริทึม ไม่ใช่เซตว่าง) ข้อคำถาม Q จะถูกเปลี่ยนรูปแบบเป็น Q' ซึ่งเป็นการถามข้อมูลจาก materialized views และมีเพรดิเคตจาก answer set เป็นเงื่อนไขเพิ่มเติมในข้อคำถาม

3.3 การพัฒนาและการทดสอบอัลกอริทึม

การสังเคราะห์โมเดลจากฐานข้อมูลเพื่อแปลงโมเดลเป็น semantic rules ใช้เทคนิคการทำเหมืองข้อมูลแบบค้นหาความสัมพันธ์ ขั้นตอนนี้ทำแบบแยกส่วนจากขั้นตอนการแปลงข้อคำถาม โดยโปรแกรมค้นหาความสัมพันธ์และแปลงเป็น semantic rules พัฒนาด้วยภาษา Erlang ที่เป็นภาษาเชิงฟังก์ชัน (รหัสต้นฉบับของโปรแกรมปรากฏในภาคผนวก ข) กำหนดการค้นหา rules ให้มีค่า minimum confidence เป็น 1.0 นั่นคือ semantic rules ที่ได้ทั้งหมดจะมีค่าความเชื่อมั่น 100% และในขั้นตอนของการค้นหาแพทเทิร์นของความสัมพันธ์ระหว่างข้อมูลที่ปรากฏบ่อย เพื่อจะแปลงแพทเทิร์นนั้นเป็น semantic rules กำหนดค่า minimum support เป็น 0.7 นั่นคือแพทเทิร์นที่ได้จะต้องมีข้อมูลสนับสนุนมากถึง 70% ค่า minimum support นี้สามารถปรับให้สูงขึ้นหรือต่ำลงได้ตามความเหมาะสมสำหรับแต่ละฐานข้อมูล ตัวอย่างคำสั่งในโปรแกรมที่ระบุค่า minimum support และ minimum confidence แสดงได้ดังรูปที่ 3.3

```
inputSup(AllInput) ->   Total = length(AllInput),
                        {_,Per} = io:read(" input percent> "),   % read minimum support value
                        MinSup = Total*Per/100 .

main() ->  NameList=mylib:read_file( "ipum.NAMES"," ,.\t:|" ),
           mylib:text_file(write,NameList,filetemp),
           [H|Tail]=NameList,
           [F,C|T]=lists:reverse(H),
           %
           % ..... some codes are omitted here ........
           %
           DB = myToSet(AllInput),
           MinSup = inputSup(AllInput),
           mylib:c(20,MinSup),
           mylib:text_file(write,AllInput,allinput),
           Items=my_flat(PossibleValue),
          AllL=apriori(DB, Items,MinSup).

main1() -> {_,[AllL]}=file:consult("set.raw"),          % main1() is for creating rules
           AllAsso2=[list(X)|| {X,_} <-AllL,length(list(X))>1 ],
           AllRuleGen=lists:flatten([genRule(L,length(L),length(L))||L<-AllAsso2]),
           AllRuleConf=[findConf(X,AllL)||X<-AllRuleGen],
           format("~nAllRule=~p ,~nThere are ~p rules ",[AllRuleConf,length(AllRuleConf)]),
           mylib:text_file(write,AllRuleConf,allrule),
           Sorted= lists:sort(fun({_,_,C1},{_,_,C2})->C1>=C2 end,AllRuleConf),
           mylib:text_file(write,Sorted,"allsortedrule.txt"),
           Conf1=lists:filter(fun({A,B,C})->C==1.0 end,Sorted),  % fix confidence value = 1.0
               %% write to file
           {_,IO}=file:open("rules.pl",[write]),
           lists:map(fun(EachR)->transform_to_prolog(EachR,IO) end,Conf1),
           _=file:close(IO),
           mylib:text_file(append,length(Conf1),"allsortedrule.txt"),
           mylib:text_file(append,length(Sorted),"allsortedrule.txt"),
           format("~n----------end main1() process ----------") .
```

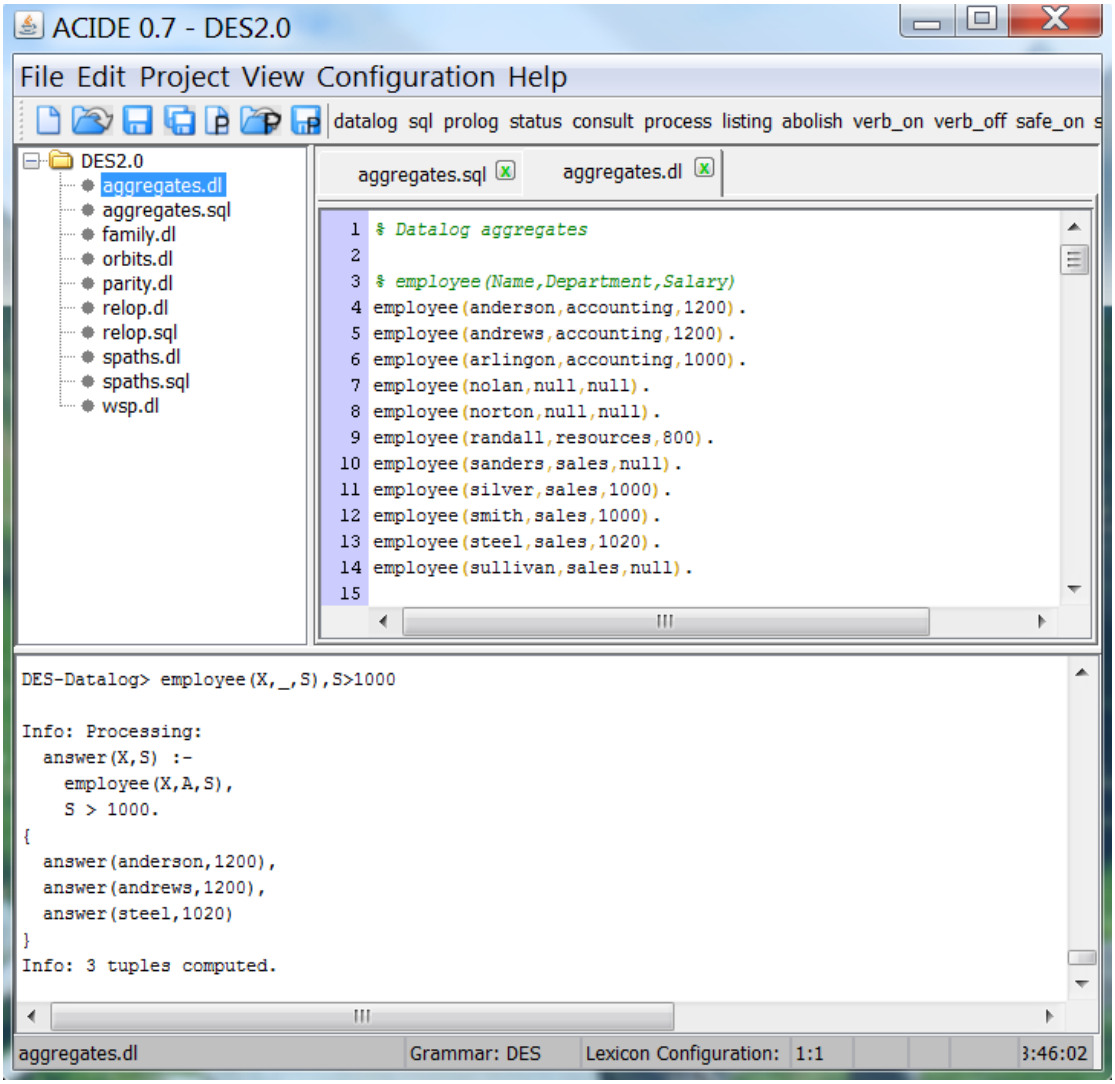
รูปที่ 3.3 ตัวอย่างคำสั่งในโปรแกรมค้นหาความสัมพันธ์จากฐานข้อมูล

การแปลงแพทเทิร์นที่ปรากฏบ่อยเป็น semantic rules กำหนดให้โครงสร้างของ rules อยู่ในรูปแบบของ Horn clause ที่ส่วนหัวของ rules มีเพียงเพรดิเคตเดียว (ในตัวอย่างตามรูปที่ 3.4 ใช้ชื่อเพรดิเคต p) และอาร์กิวเม้นต์จะเป็นอะตอมที่เป็นค่าคงที่ แสดงตัวอย่างของ semantic rules ที่ได้ดังรูปที่ 3.4

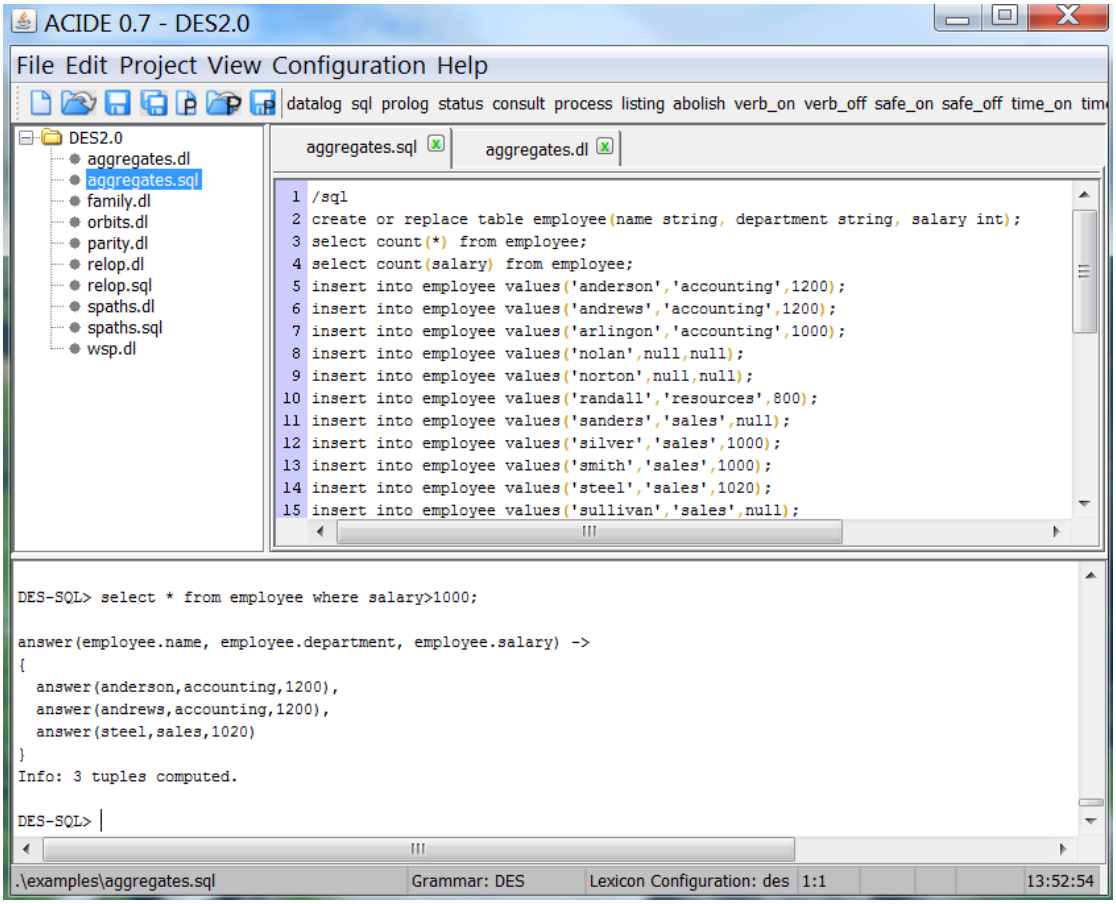
```
p(school_1) :- p(schltype_1).
p(schltype_1) :- p(school_1).
p(farm_1) :- p(gq_1), p(bplg_1).
p(farm_1) :- p(gqtypeg_0), p(bplg_1).
p(farm_1) :- p(migplac5_1), p(bplg_1).
p(farm_1) :- p(stepmom_0), p(bplg_1).
p(farm_1) :- p(steppop_0), p(bplg_1).
p(gqtypeg_0) :- p(bplg_1), p(gq_1).
p(gq_1) :- p(gqtypeg_0), p(bplg_1).
p(farm_1) :- p(gq_1), p(eldch_10).
p(farm_1) :- p(gqtypeg_0), p(eldch_10).
p(nchild_0) :- p(eldch_10), p(farm_1).
p(eldch_10) :- p(farm_1), p(nchild_0).
p(farm_1) :- p(nchild_0), p(eldch_10).
p(schltype_1) :- p(school_1), p(stepmom_0), p(farm_1), p(gq_1), p(gqtypeg_0).
p(school_1) :- p(stepmom_0), p(farm_1), p(gq_1), p(gqtypeg_0), p(schltype_1).
p(farm_1) :- p(gq_1), p(gqtypeg_0), p(schltype_1), p(school_1), p(steppop_0).
```

รูปที่ 3.4 บางส่วนของ semantic rules ที่เป็นผลลัพธ์จากโปรแกรมสังเคราะห์โมเดลจากฐานข้อมูล

การทดสอบอัลกอริทึมแปลงข้อคำถามให้อยู่ในรูปแบบที่มีประสิทธิภาพมากขึ้น จะใช้ตัวอย่างข้อคำถามจำนวน 6 คำถาม ทดสอบกับฐานข้อมูลที่ประกอบด้วยข้อมูล 2 ชุด (หรือ 2 ตาราง ในกรณีที่พิจารณาข้อมูลในแบบฐานข้อมูลเชิงสัมพันธ์) ข้อมูลทั้งสองชุดมีจำนวนเรคคอร์ดเท่ากัน คือ 1,000 เรคคอร์ด การประมวลผลข้อคำถามจะกระทำบนระบบฐานข้อมูล DES (Datalog Educational System) ซึ่งเป็นฐานข้อมูลนิรนัยที่เปิดเผยซอร์สโค้ด สามารถดาวน์โหลดได้จากเว็บไซต์ http//des.sourceforge.net/ ระบบฐานข้อมูล DES ที่ใช้ในการทดลองเป็นเวอร์ชัน 2.0 รันด้วยระบบปฏิบัติการ Windows ระบบฐานข้อมูล DES 2.0 สามารถรองรับการประมวลผลข้อคำถามได้ทั้งในรูปแบบของภาษา Datalog (ตัวอย่างแสดงได้ดังรูปที่ 3.5) และในรูปแบบของภาษา SQL (ดังตัวอย่างในรูปที่ 3.6)



รูปที่ 3.5 ระบบฐานข้อมูลนิรนัย DES ที่รับข้อคำถามในรูปแบบของภาษา Datalog

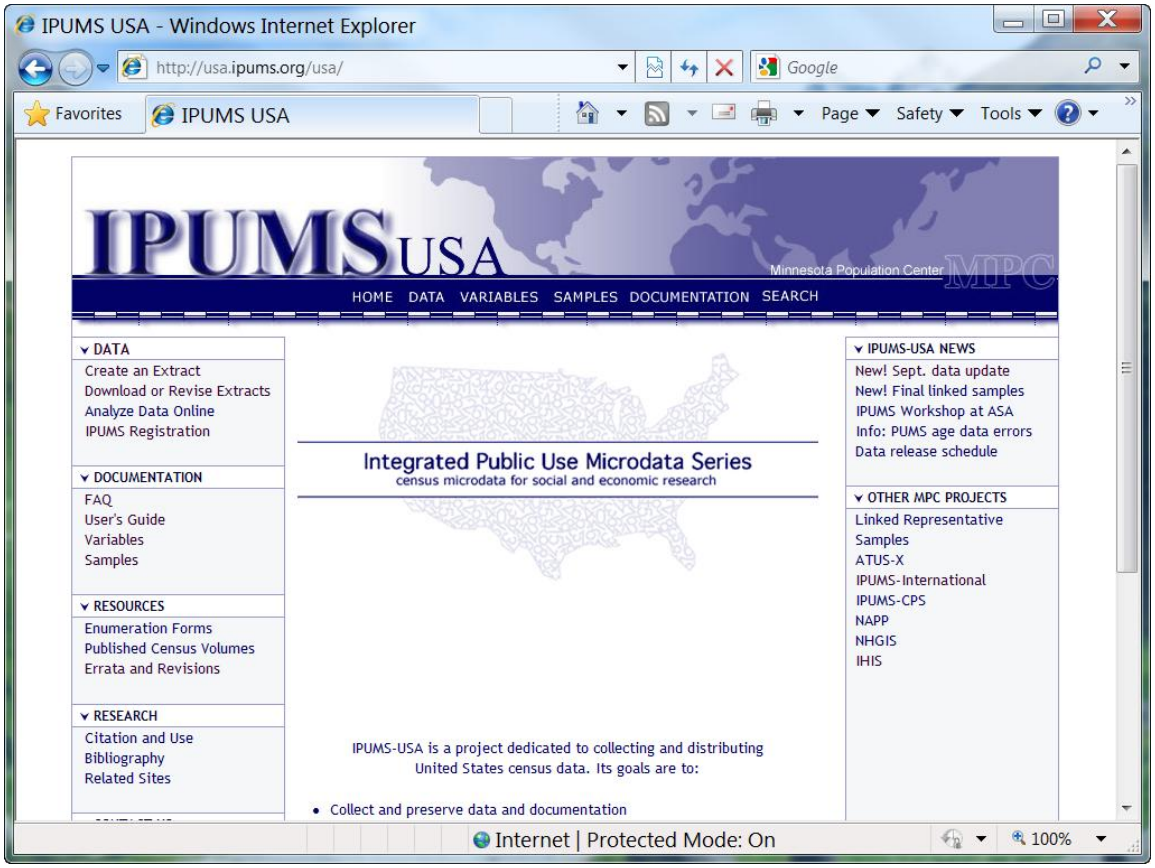


รูปที่ 3.6 ระบบฐานข้อมูลนิรนัย DES ที่รับข้อคำถามในรูปแบบของภาษา SQL

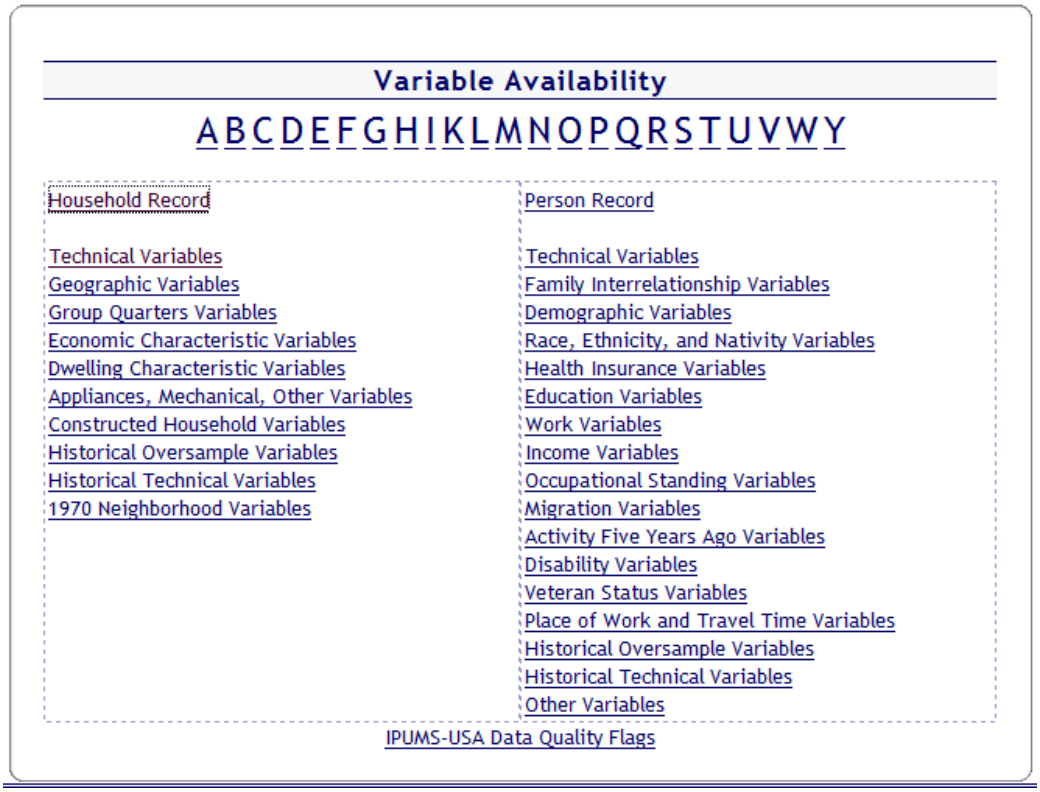
3.4 ข้อมูลที่ใช้ในการทดสอบ

การสร้างฐานข้อมูลเพื่อทดสอบอัลกอริทึม Semantic query optimizer ใช้ข้อมูลสุ่มจำนวน 1,000 เรคคอร์ดจากข้อมูลสำมะโนประชากรปีค.ศ.1999 ของประเทศสหรัฐอเมริกา ซึ่งเป็นข้อมูลที่เผยแพร่เป็นสาธารณะโดย Minnesota Population Center (เว็บไซต์แสดงดังรูปที่ 3.7) ตัวแปรหรือแอททริบิวต์ทั้งหมดของข้อมูลแบ่งเป็นสองกลุ่ม (รูปที่ 3.8) คือรายละเอียดครัวเรือน (household record) และรายละเอียดของบุคคลในครัวเรือน (person record)

ข้อมูลที่ได้คัดเลือกเพื่อนำมาใช้ในการทดสอบอัลกอริทึมการปรับปรุงข้อคำถามของโครงการวิจัยนี้ประกอบด้วยข้อมูล 2 ชุด ข้อมูลชุดแรกเป็นรายละเอียดเกี่ยวกับสภาพที่อยู่อาศัย สมาชิกและความสัมพันธ์ระหว่างสมาชิกในครัวเรือน ข้อมูลชุดแรกนี้ประกอบด้วย 34 แอททริบิวต์ รายชื่อแอททริบิวต์และคำอธิบาย รวมถึงความหมายของรหัสข้อมูลแสดงดังตารางที่ 3.1 และแสดงตัวอย่างข้อมูลในรูปแบบของ Datalog ได้ดังรูปที่ 3.9



รูปที่ 3.7 แหล่งข้อมูลสำมะโนประชากรของประเทศสหรัฐอเมริกา



รูปที่ 3.8 กลุ่มแอททริบิวต์ของข้อมูลสำมะโนประชากร

ตารางที่ 3.1 รายชื่อแอททริบิวต์และคำอธิบายแอททริบิวต์ของข้อมูลชุดที่ 1

ลำดับที่	ชื่อแอททริบิวต์	คำอธิบายแอททริบิวต์	ค่าของข้อมูลที่เป็นไปได้ทั้งหมดและความหมายของรหัสแทนข้อมูล
1	ID	Record-ID	
2	gq	Group quarters status	gq_1=Household, large unit gq_3=Institution gq_4=Other group quarters
3	gqtypeg	Group quarters type -- general	gqtypeg _0=Not available gqtypeg _1=Institution gqtypeg _5=Non-institution
4	farm	Farm status	farm_1=Non-farm farm_2=Farm
5	ownershg	Ownership of dwelling -- general	ownershg _0=Not available ownershg _1=Owned or being bought (loan) ownershg _2=Rented
6	value	Value of housing unit	value_1=0-5,000 value_2=5,001-30,000 value_3=30,001-50,000 value_4=50,001-70,000 value_5=70,001-100,000 value_6=100,001-200,000 value_7=200,001-999,999
7	rent	Monthly rental payment	rent_0=0 or Not available rent_1=1 rent_2=2-200 rent_4=201-400 rent_6=401-600 rent_9=601-900 rent_10=1,000
8	ftotinc	Family total income	ftotinc _1=0-999 ftotinc _2=1,000-9,999 ftotinc _3=10,000-99,999 ftotinc _4=100,000-999,999
9	nfams	Number of families within each household unit	nfams_1=1, nfams_2=2, …, nfams_9=9
10	ncouples	Number of married couples in a household	ncouples_0=0, ncouples_1=1, …, ncouples_4=4
11	nmothers	Number of mothers within each household unit	nmothers_0=0, nmothers_1=1, …, nmothers_4=4

ตารางที่ 3.1 รายชื่อแอททริบิวต์และคำอธิบายแอททริบิวต์ของข้อมูลชุดที่ 1 (ต่อ)

ลำดับที่	ชื่อแอททริบิวต์	คำอธิบายแอททริบิวต์	ค่าของข้อมูลที่เป็นไปได้ทั้งหมดและความหมายของรหัสแทนข้อมูล
12	nfathers	Number of fathers within each household unit	nfathers_0=0, …, nfathers_4=4
13	momloc	Mother's location in the household (a variable that indicates whether the person's mother lived in the same household and, if so, gives the number of mother)	momloc_0=0, monloc_1=1, …, momloc_17=17
14	stepmom	Probable step/adopted mother (whether a person's mother was likely to have been the person's stepmother or adoptive mother)	stepmom_0=0 (no likely stepmother), stepmom_1=1 (probable that the person's mother was a step- or adoptive mother), …, stepmom_7=7
15	momrule	Rule for linking mother	momrule_0= no mother of this person present in the household momrule_1= child-mother link momrule_2= grandchild link momrule_3= plausible child-mother link momrule_4= link by surname similarity momrule_5= same as rule 2 momrule_6= same as rule 3 momrule_7= child-stepmother link
16	poploc	Father's location in the household (a constructed variable to identify social relationship such as stepfather, adoptive father, as well as biological father)	poploc_0=0, poploc_1=1, …, poploc_18=18
17	steppop	Probable step/adopted father	steppop_0=no stepfather present steppop_1=improbable age difference steppop_2=spouse of mother steppop_3=identified stepfather
18	poprule	Rule for linking father	poprule_0=no father link poprule_1=unambiguous father link poprule_2=son/grandchild link poprule_3=preceding male (no intervening person) poprule_7=husband of mother becomes stepfather

ตารางที่ 3.1 รายชื่อแอททริบิวต์และคำอธิบายแอททริบิวต์ของข้อมูลชุดที่ 1 (ต่อ)

ลำดับที่	ชื่อแอททริบิวต์	คำอธิบายแอททริบิวต์	ค่าของข้อมูลที่เป็นไปได้ทั้งหมดและความหมายของรหัสแทนข้อมูล
19	sploc	Spouse's location in household (a constructed variable that indicates whether the person's spouse lived in the same household and, if so, gives the person number of the spouse)	sploc_0=no spouse sploc_1, …, sploc_21
20	sprule	Rule for linking spouse	sprule_0=no spouse link sprule_1=husband-wife link sprule_2=wife-husband link sprule_3=non-adjacent links – consistent relationship to head/age differences sprule_4=adjacent links (husband-wife – no age, other relative conflicts) sprule_5= adjacent links (wife-husband – no age, other relative conflicts)
21	famsize	Number of own family members in household	famsize_1= 1 family member present, …, famsize_20=20 family members present
22	nchild	Number of own children in the household	nchild_0= 0 child present, …, nchild_9= 9 children present
23	nchlt5	Number of own children under age 5 in household	nchlt5_0= 0 young child present, …, nchlt5_5= 5 young children present
24	famunit	Groups of related individuals in the household	famunit_1= 1 family group, …, famunit_9= 9 family groups
25	eldch	Age of eldest own child in household	eldch_1= 0-9 years old eldch_2= 10-19 years old eldch_3= 20-29 years old eldch_4= 30-39 years old eldch_5= 40-49 years old eldch_6= 50-59 years old eldch_7= 60-69 years old eldch_8= 70-79 years old eldch_9= 80-89 years old eldch_10= 90-99 years old
26	yngch	Age of youngest own child in household	yngch_1, …,yngch_10 (same coding as eldch attribute)
27	nsibs	Number of own siblings in the household	nsibs_0= 0 sibling present, …, nsibs_9= 9 siblings present

ตารางที่ 3.1 รายชื่อแอททริบิวต์และคำอธิบายแอททริบิวต์ของข้อมูลชุดที่ 1 (ต่อ)

ลำดับที่	ชื่อแอททริบิวต์	คำอธิบายแอททริบิวต์	ค่าของข้อมูลที่เป็นไปได้ทั้งหมดและความหมายของรหัสแทนข้อมูล
28	relateg	Relationship to household head -- general	relateg_1=head/householder, relateg_2=spouse, relateg_3=child, relateg_4=child-in-law, relateg_5=parent, relateg_6=parent-in-law, relateg_7=sibling, relateg_8=sibling-in-law, relateg_9=grandchild, relateg_10=other relatives, relateg_11=partner, friend, visitor relateg_12=other non-relatives relateg_13=institutional inmates
29	age	Age of a person	age_1= 0-9 years old, age_2= 10-19 years old age_3= 20-29 years old, age_4= 30-39 years old age_5= 40-49 years old, age_6= 50-59 years old age_7= 60-69 years old, age_8= 70-79 years old age_9= 80-89 years old
30	sex	Sex of a person	sex_1=male sex_2=female
31	raceg	Race -- general	raceg_1=White raceg_2=Black/Negro raceg_3=American Indian or Alaska Native raceg_4=Chinese raceg_5=Japanese raceg_6=Other Asian or Pacific Islander raceg_7=Other race
32	marst	Marital status	marst_1=married, spouse present marst_2=married, spouse absent marst_3=separated marst_4=divorced marst_5=widowed marst_6=never married/ single
33	chborn	Number of children ever born to each woman	chborn_0, …, chborn_13
34	bplg	Birthplace -- general	bplg_1=USA. bplg_2=US Possessions bplg_3=North, Central, South America bplg_4=Europe bplg_5=Asia bplg_6=Africa bplg_7=Pacific bplg_8=Unknown

```
table1(1,gq_1,gqtypeg_0,farm_1,ownershg_1,value_6,rent_0,ftotinc_3,nfams_1,ncouples_
         1,nmothers_0,nfathers_0,momloc_0,stepmom_0,momrule_0,poploc_0,steppop_
         0,poprule_0,sploc_2,sprule_1,famsize_2,nchild_0,nchlt5_0,famunit_1,eldch_
         10,yngch_10,nsibs_0,relateg_1,age_6,sex_1,raceg_1,marst_1,chborn_0,bplg_1).%
table1(2,gq_1,gqtypeg_0,farm_1,ownershg_1,value_6,rent_0,ftotinc_3,nfams_1,ncouples_
         1,nmothers_0,nfathers_0,momloc_0,stepmom_0,momrule_0,poploc_0,steppop_
         0,poprule_0,sploc_1,sprule_1,famsize_2,nchild_0,nchlt5_0,famunit_1,eldch_
         10,yngch_10,nsibs_0,relateg_2,age_6,sex_2,raceg_1,marst_1,chborn_4,bplg_1).%
table1(3,gq_1,gqtypeg_0,farm_1,ownershg_2,value_7,rent_9,ftotinc_3,nfams_2,ncouples_
         0,nmothers_1,nfathers_0,momloc_0,stepmom_0,momrule_0,poploc_0,steppop_
         0,poprule_0,sploc_0,sprule_0,famsize_2,nchild_1,nchlt5_0,famunit_1,eldch_
         1,yngch_1,nsibs_0,relateg_1,age_4,sex_2,raceg_1,marst_4,chborn_2,bplg_1).%
table1(4,gq_1,gqtypeg_0,farm_1,ownershg_2,value_7,rent_9,ftotinc_3,nfams_2,ncouples_
         0,nmothers_1,nfathers_0,momloc_1,stepmom_0,momrule_1,poploc_0,steppop_
         0,poprule_0,sploc_0,sprule_0,famsize_2,nchild_0,nchlt5_0,famunit_1,eldch_
         10,yngch_10,nsibs_0,relateg_3,age_1,sex_1,raceg_1,marst_6,chborn_0,bplg_1).%
table1(5,gq_1,gqtypeg_0,farm_1,ownershg_2,value_7,rent_9,ftotinc_3,nfams_2,ncouples_
         0,nmothers_1,nfathers_0,momloc_0,stepmom_0,momrule_0,poploc_0,steppop_
         0,poprule_0,sploc_0,sprule_0,famsize_1,nchild_0,nchlt5_0,famunit_2,eldch_
         10,yngch_10,nsibs_0,relateg_11,age_3,sex_1,raceg_1,marst_6,chborn_0,bplg_1).%
```

รูปที่ 3.9 ตัวอย่างข้อมูลชุดที่ 1

ข้อมูลชุดที่สองเป็นข้อมูลของกลุ่มบุคคลกลุ่มเดียวกับข้อมูลในชุดแรก แต่ข้อมูลในชุดที่สองนี้เป็นข้อมูลเกี่ยวกับการศึกษา สถานภาพการมีงานทำ วิธีที่ใช้เดินทางไปทำงาน และรายได้ ข้อมูลชุดที่สองนี้ประกอบด้วย 27 แอททริบิวต์ รายชื่อแอททริบิวต์และคำอธิบาย รวมถึงความหมายของรหัสข้อมูลแสดงดังตารางที่ 3.2 และแสดงตัวอย่างข้อมูลในรูปแบบของ Datalog ได้ดังรูปที่ 3.10

ตารางที่ 3.2 รายชื่อแอททริบิวต์และคำอธิบายแอททริบิวต์ของข้อมูลชุดที่ 2

ลำดับที่	ชื่อแอททริบิวต์	คำอธิบายแอททริบิวต์	ค่าของข้อมูลที่เป็นไปได้ทั้งหมดและความหมายของรหัสแทนข้อมูล
1	ID	Record-ID	
2	school	School attendance	school_0=not available (N/A) school_1=No, not in school school_2=Yes, in school
3	educrec	Highest year of school or degree completed	educrec_0=N/A or no schooling educrec_1=Nursery school to grade 4 educrec_2=Grade 5,6,7, or 8 educrec_3=Grade 9 educrec_4=Grade 10 educrec_5=Grade 11 educrec_6=Grade 12 educrec_7= 1 year of college educrec_8=2 years of college educrec_9=3 years of college educrec_10=4 years of college educrec_11= 5 or more years of college

ตารางที่ 3.2 รายชื่อแอททริบิวต์และคำอธิบายแอททริบิวต์ของข้อมูลชุดที่ 2 (ต่อ)

ลำดับที่	ชื่อแอททริบิวต์	คำอธิบายแอททริบิวต์	ค่าของข้อมูลที่เป็นไปได้ทั้งหมดและความหมายของรหัสแทนข้อมูล
4	schltype	School type	schltype_0=N/A schltype_1=Not enrolled schltype_2=Public school schltype_3=Private school
5	empstatg	Employment status -- general	empstatg_0=N/A empstatg_1=employed empstatg_2=unemployed empstatg_3=not in labor force
6	labforce	Labor force status	labforce_0=N/A labforce_1=No, not in the labor force labforce_2=Yes, in the labor force
7	occscore	Occupational income score (the median total income in hundreds of dollars)	occscore_1=range 0-9 occscore_2=range 10-19 occscore_3=range 20-29 occscore_4=range 30-39 occscore_5=range 40-49 occscore_6=range 50-79
8	sei	Duncan Socioeconomic Index score to each occupation (SEI is the measure of occupational status based upon the income level and educational attainment associated with each occupation)	sei_1=score 0-9 sei_2=score 10-19 sei_3=score 20-29 sei_4=score 30-39 sei_5=score 40-49 sei_6=score 50-59 sei_7=score 60-69 sei_8=score 70-79 sei_9=score 80-89 sei_10=score 90-99
9	classwkg	Class of worker -- general	classwkg_0=N/A classwkg_1=self-employed classwkg_2=works for wages/salary
10	wkswork2	Weeks worked last year, intervalled	wkswork2_0=N/A wkswork2_1= 1-13 weeks wkswork2_2= 14-26 weeks wkswork2_3= 27-39 weeks wkswork2_4= 40-47 weeks wkswork2_5= 48-49 weeks wkswork2_6= 50-52 weeks

ตารางที่ 3.2 รายชื่อแอททริบิวต์และคำอธิบายแอททริบิวต์ของข้อมูลชุดที่ 2 (ต่อ)

ลำดับที่	ชื่อแอททริบิวต์	คำอธิบายแอททริบิวต์	ค่าของข้อมูลที่เป็นไปได้ทั้งหมดและความหมายของรหัสแทนข้อมูล
11	hrswork2	Hours work last week, intervalled	hrswork2_0=N/A hrswork2_1= 1-14 hours hrswork2_2= 15-29 hours hrswork2_3= 30-34 hours hrswork2_4= 35-39 hours hrswork2_5= 40 hours hrswork2_6= 41-48 hours hrswork2_7= 49-59 hours hrswork2_8= more than 59 hours
12	yrlastwk	Year last worked	yrlastwk_0=N/A yrlastwk_10= worked current year yrlastwk_20= worked previous year yrlastwk_31= worked 2 years prior yrlastwk_33= worked 3-5 years ago yrlastwk_35= worked 6-10 year ago yrlastwk_40= worked more than 10 years ago yrlastwk_50= never worked
13	workedyr	Worked last year	workedyr_0=N/A workedyr_1=No, and did not work in past 5 years workedyr_2=No, but worked 1-5 years ago
14	inctot	Total personal income	inctot_0=less than 0 inctot_1=range 0-999 inctot_2=range 1,000-9,999 inctot_3=range 10,000-99,999 inctot_4=more than 99,999
15	incwage	Wage and salary income	incwage_1= range 0-999 incwage_2= range 1,000-9,999 incwage_3= range 10,000-99,999 incwage_4= more than 99,999
16	incbus	Non-farm business income	incbus_0= less than 0 incbus_1= range 0-999 incbus_2= range 1,000-9,999 incbus_3= range 10,000-99,999 incbus_4= more than 99,999
17	incfarm	Farm income	incfarm_0= less than 0 incfarm_1= range 0-999 incfarm_2= range 1,000-9,999 incfarm_3= range 10,000-99,999 incfarm_4= more than 99,999

ตารางที่ 3.2 รายชื่อแอททริบิวต์และคำอธิบายแอททริบิวต์ของข้อมูลชุดที่ 2 (ต่อ)

ลำดับที่	ชื่อแอททริบิวต์	คำอธิบายแอททริบิวต์	ค่าของข้อมูลที่เป็นไปได้ทั้งหมดและความหมายของรหัสแทนข้อมูล
18	incss	Social security income (social security pensions, survivors benefits, permanent disability insurance)	incss_0= less than 0 incss_1= range 0-999 incss_2= range 1,000-9,999 incss_3= range 10,000-99,999 incss_4= more than 99,999
19	incwelfr	Welfare income (federal/state supplemental security income payments to elderly, blind, or disabled persons with low incomes; families with dependent children)	incwelfr_0= less than 0 incwelfr_1= range 0-999 incwelfr_2= range 1,000-9,999 incwelfr_3= range 10,000-99,999 incwelfr_4= more than 99,999
20	incother	Other income	incother_0= less than 0 incother_1= range 0-999 incother_2= range 1,000-9,999 incother_3= range 10,000-99,999 incother_4= more than 99,999
21	poverty	Poverty status (It expresses each family's total income for the previous year as a percentage of the poverty threshold established by the Social Security Administration.)	Example: if a person's family income is $20,000 and the poverty threshold for such a person is $13,861, then the value of POVERTY for that individual is $20,000/$13,861 * 100 percent, or 144. poverty_1=value 0-99 poverty_2=value 100-199 poverty_3=value 200-299 poverty_4=value 300-399 poverty_5=value 400-499 poverty_6=value 500-599
22	migrat5g	Migration status, 5 years – general	migrat5g_0=N/A migrat5g_1=same house migrat5g_2=moved, place not reported
23	migplac5	State or country of residence 5 years ago	migplac5_0=N/A migplac5_1=USA. migplac5_2=abroad
24	movedin	Number of years ago that the householder moved into the dwelling unit	movedin_0=N/A movedin_1=this year or last year movedin_2= 2-5 years ago movedin_5= 6-10 years ago movedin_6= 11-20 years ago movedin_7= 21-30 years ago movedin_8= more than 30 years ago

ตารางที่ 3.2 รายชื่อแอททริบิวต์และคำอธิบายแอททริบิวต์ของข้อมูลชุดที่ 2 (ต่อ)

ลำดับที่	ชื่อแอททริบิวต์	คำอธิบายแอททริบิวต์	ค่าของข้อมูลที่เป็นไปได้ทั้งหมดและความหมายของรหัสแทนข้อมูล
25	vetstat	Veteran status	vetstat_0=N/A vetstat_1=not a veteran vetstat_2=veteran
26	tranwork	Means of transportation to work	tranwork_0= N/A tranwork_10=auto, truck, or van tranwork_20=motorcycle tranwork_31=bus or trolley bus tranwork_32=streetcar tranwork_33=subway tranwork_34=railroad tranwork_35=taxi tranwork_36=ferry boat tranwork_40=bicycle tranwork_50=walked only tranwork_60=other tranwork_70=worked at home
27	occupation	Occupational classification	occupation_1=professional, technical occupation_2=farmers, operative workers occupation_3=private and service workers occupation_4=at home / at school occupation_5=unknown

```
table2(1,school_1,educrec_7,schltype_1,empstatg_1,labforce_2,occscore_3,sei_2,
        classwkg_2,wkswork2_4,hrswork2_6,yrlastwk_0,workedyr_2,inctot_3,
        incwage_3,incbus_1,incfarm_1,incss_1,incwelfr_1,incother_1,poverty_6,
        migrat5g_1,migplac5_0,movedin_7,vetstat_1,tranwork_10,occupation_2).%
table2(2,school_1,educrec_8,schltype_1,empstatg_1,labforce_2,occscore_2,sei_3,
        classwkg_1,wkswork2_3,hrswork2_6,yrlastwk_0,workedyr_2,inctot_2,
        incwage_2,incbus_2,incfarm_1,incss_1,incwelfr_1,incother_1,poverty_6,
        migrat5g_1,migplac5_1,movedin_0,vetstat_1,tranwork_10,occupation_3).%
table2(3,school_1,educrec_7,schltype_1,empstatg_1,labforce_2,occscore_3,sei_5,
        classwkg_2,wkswork2_6,hrswork2_5,yrlastwk_0,workedyr_2,inctot_3,
        incwage_3,incbus_1,incfarm_1,incss_1,incwelfr_1,incother_1,poverty_2,
        migrat5g_2,migplac5_1,movedin_2,vetstat_1,tranwork_10,occupation_2).%
```

รูปที่ 3.10 ตัวอย่างข้อมูลชุดที่ 2

3.5 ผลการทดสอบการประมวลผลข้อคำถาม

การทดสอบวิธีการปรับปรุงข้อคำถามด้วย semantic rules และผลการประมวลผลข้อคำถามที่ใช้การสอบถามข้อมูลทั้งในกรณีเป็นข้อมูลชุดเดียว และกรณีต้องมีการเชื่อมโยงข้อมูลทั้งสองชุด ซึ่งจะใช้การปรับปรุงข้อคำถามด้วยการสอบถามจากวิวข้อมูล สรุปการประมวลผลและเวลาที่ใช้ได้ดังต่อไปนี้

ข้อคำถามที่ 1 ให้แสดงมูลค่าของที่อยู่อาศัย (value หรือแอททริบิวต์ที่ 6) จากข้อมูลในชุดที่หนึ่ง (table1) ที่เป็นที่อยู่อาศัยประเภทฟาร์ม (farm_2 ในแอททริบิวต์ที่ 4)

```
DES-Datalog> /assert query1(X):-
table1(A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,A11,A12,A13,A14,A15,A16,A
17,A18,A19,A20,A21,A22,A23,A24,A25,A26,A27,A28,A29,A30,A31,A32,
A33,A34),A4=farm_2,X=A6.

DES-Datalog> query1(X).
{
}
Info: 0 tuples computed.
Info: Total elapsed time: 110 ms.
```

ข้อคำถามนี้ได้ผลลัพธ์เป็นเซตว่างเนื่องจากข้อมูลทั้งหมดใน table1 เป็นที่อยู่อาศัยประเภท non-farm (นั่นคือแอททริบิวต์ A4 มีค่าเป็น farm_1 ทั้งหมด)

การปรับปรุงข้อคำถามด้วย semanctic rules

```
DES-Datalog> /assert p(farm_2).
DES-Datalog> p(C).
{
  p(farm_2)
}
Info: 1 tuple computed.
Info: Total elapsed time: 295 ms.
```

ผลการปรับปรุงข้อคำถาม: ไม่ปรากฏเงื่อนไขอื่นที่จะสามารถนำมาใช้ช่วยเพิ่มประสิทธิภาพการประมวลผลข้อคำถาม

เวลาที่ใช้: ใช้เวลาเพิ่มขึ้น 295 มิลลิวินาที

ข้อคำถามที่ 2 ให้แสดงมูลค่าของที่อยู่อาศัย (value หรือแอททริบิวต์ที่ 6) จากข้อมูลในชุดที่หนึ่ง (table1) ที่เป็นที่อยู่อาศัยประเภทที่ไม่ใช่ฟาร์ม (farm_1 ในแอททริบิวต์ที่ 4)

```
DES-Datalog> /assert query2(X):-
table1(A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,A11,A12,A13,A14,A15,A16,A
17,A18,A19,A20,A21,A22,A23,A24,A25,A26,A27,A28,A29,A30,A31,A32,
A33,A34),A4=farm_1,X=A6.

DES-Datalog> query2(X).
{
  query2(value_1),
  query2(value_2),
  query2(value_3),
  query2(value_4),
  query2(value_5),
  query2(value_6),
```

```
  query2(value_7)
}
Info: 7 tuples computed.
Info: Total elapsed time: 1029 ms.
```

ข้อคำถามนี้ได้ผลลัพธ์เป็นมูลค่าที่ปรากฏในข้อมูล table1 ทั้งหมดของที่อยู่อาศัยประเภท non-farm (นั่นคือแอททริบิวต์ A4 มีค่าเป็น farm_1)

การปรับปรุงข้อคำถามด้วย semanctic rules

```
DES-Datalog> /assert p(farm_1).
DES-Datalog> p(C).
{
  p(farm_1)
}
Info: 1 tuple computed.
Info: Total elapsed time: 296 ms.
```

ผลการปรับปรุงข้อคำถาม: ไม่ปรากฏเงื่อนไขอื่นที่จะสามารถนำมาใช้ช่วยเพิ่มประสิทธิภาพการประมวลผลข้อคำถาม

เวลาที่ใช้: ใช้เวลาเพิ่มขึ้น 296 มิลลิวินาที

ข้อคำถามที่ 3 ให้แสดงระดับรายได้ทั้งหมดของครอบครัว (ftotinc หรือแอททริบิวต์ที่ 8) จากข้อมูลในชุดที่หนึ่ง (table1) ที่มีจำนวนสองครอบครัวอาศัยอยู่ในครัวเรือนเดียวกัน (famunit_2)

```
DES-Datalog> /assert query3(X):-
table1(A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,A11,A12,A13,A14,A15,A16,A
17,A18,A19,A20,A21,A22,A23,A24,A25,A26,A27,A28,A29,A30,A31,A32,
A33,A34),A24=famunit_2,X=A8.


DES-Datalog> query3(X).
{
  query3(ftotinc_1),
  query3(ftotinc_2),
  query3(ftotinc_3)
}
Info: 3 tuples computed.
Info: Total elapsed time: 906 ms.
```

การปรับปรุงข้อคำถามด้วย semanctic rules

```
DES-Datalog> /assert p(famunit_2).
DES-Datalog> p(C).
{
  p(famunit_2)
}
Info: 1 tuple computed.
Info: Total elapsed time: 282 ms.
```

ผลการปรับปรุงข้อคำถาม: ไม่ปรากฏเงื่อนไขอื่นที่จะสามารถนำมาใช้ช่วยเพิ่มประสิทธิภาพการประมวลผลข้อคำถาม

เวลาที่ใช้: ใช้เวลาเพิ่มขึ้น 282 มิลลิวินาที

ข้อคำถามที่ 4 ให้แสดงขนาดของครอบครัว (famsize หรือแอททริบิวต์ที่ 21) ที่เป็นครัวเรือนประเภท non_farm (หรือ farm_1) และมีรายได้ของครอบครัวอยู่ระหว่าง 10,000-99,999 หรือในระดับ 3 (ftotinc_3)

```
DES-Datalog> /assert query4(X):-
table1(A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,A11,A12,A13,A14,A15,A16,A
17,A18,A19,A20,A21,A22,A23,A24,A25,A26,A27,A28,A29,A30,A31,A32,
A33,A34),A4=farm_1,A8=ftotinc_3,X=A21.

DES-Datalog> query4(X).
{
  query4(famsize_1),
  query4(famsize_2),
  query4(famsize_3),
  query4(famsize_4),
  query4(famsize_5),
  query4(famsize_6),
  query4(famsize_7)
}
Info: 7 tuples computed.
Info: Total elapsed time: 1028 ms.
```

การปรับปรุงข้อคำถามด้วย semanctic rules

```
DES-Datalog> /assert p(farm_1).
DES-Datalog> /assert p(ftotinc_3).
DES-Datalog> p(C).
{
  p(farm_1),
  p(ftotinc_3),
  p(gq_1),
  p(gqtypeg_0)
}
Info: 6 tuples computed.
Info: Total elapsed time: 594 ms.
```

ผลการปรับปรุงข้อคำถาม:

```
DES-Datalog> /assert query4B(X):-
table1(A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,A11,A12,A13,A14,A15,A16,A
17,A18,A19,A20,A21,A22,A23,A24,A25,A26,A27,A28,A29,A30,A31,A32,
A33,A34),A4=farm_1,A8=ftotinc_3,A2=gq_1,A3=gqtypeg_0,X=A21.

DES-Datalog> query4B(X).
{
  query4B(famsize_1),
  query4B(famsize_2),
  query4B(famsize_3),
  query4B(famsize_4),
  query4B(famsize_5),
  query4B(famsize_6),
  query4B(famsize_7)
}
Info: 7 tuples computed.
Info: Total elapsed time: 94 ms.
```

เวลาที่ใช้: ใช้เวลาลดลง (1028 – (594+94)) = 340 มิลลิวินาที

ข้อคำถามที่ 5 ให้แสดงระดับการศึกษา (educrec หรือแอททริบิวต์ที่ 3 ของข้อมูลชุดที่สอง) และระดับรายได้ (inctot หรือแอททริบิวต์ที่ 15 ของข้อมูลชุดที่สอง) ที่เป็นเพศหญิง (sex_2 ในแอททริบิวต์ที่ 30 ของข้อมูลชุดที่หนึ่ง)

```
DES-Datalog> /assert query5(X,Y):-
table1(A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,A11,A12,A13,A14,A15,A16,A
17,A18,A19,A20,A21,A22,A23,A24,A25,A26,A27,A28,A29,A30,A31,A32,
A33,A34),table2(B1,B2,B3,B4,B5,B6,B7,B8,B9,B10,B11,B12,B13,B14,
B15,B16,B17,B18,B19,B20,B21,B22,B23,B24,B25,B26,B27),A1=B1,A30=
sex_2,X=B3,Y=B15.

DES-Datalog> query5(X,Y).
{
  query5(educrec_0,incwage_4),
  query5(educrec_1,incwage_1),
  query5(educrec_1,incwage_2),
  query5(educrec_1,incwage_3),
  query5(educrec_1,incwage_4),
  query5(educrec_2,incwage_1),
  query5(educrec_2,incwage_3),
  query5(educrec_2,incwage_4),
  query5(educrec_3,incwage_1),
  query5(educrec_3,incwage_2),
  query5(educrec_3,incwage_3),
  query5(educrec_3,incwage_4),
  query5(educrec_4,incwage_1),
  query5(educrec_4,incwage_2),
  query5(educrec_4,incwage_3),
  query5(educrec_4,incwage_4),
  query5(educrec_5,incwage_1),
  query5(educrec_5,incwage_2),
  query5(educrec_5,incwage_3),
  query5(educrec_5,incwage_4),
  query5(educrec_6,incwage_1),
  query5(educrec_6,incwage_2),
  query5(educrec_6,incwage_3),
  query5(educrec_7,incwage_1),
  query5(educrec_7,incwage_2),
  query5(educrec_7,incwage_3),
  query5(educrec_8,incwage_1),
  query5(educrec_8,incwage_2),
  query5(educrec_8,incwage_3),
  query5(educrec_9,incwage_1),
  query5(educrec_9,incwage_2),
  query5(educrec_9,incwage_3),
  query5(educrec_9,incwage_4)
}
Info: 756 tuples computed.

Info: Total elapsed time: 10423 ms.
```

ผลการปรับปรุงข้อคำถามด้วยการสอบถามจากวิวข้อมูล:

```
DES-Datalog> /assert query5B(X,Y):-
view1(V1,V2,V3,V4,V5,V6,V7,V8,V9,V10,V11,V12,V13,V14,V15,V16,V1
7,V18,V19,V20,V21,V22,V23,V24,V25,V26,V27,V28,V29,V30,V31,V32,V
33,V34,V35,V36,V37,V38,V39,V40,V41,V42,V43,V44,V45,V46,V47,V48,
V49,V50,V51,V52,V53,V54,V55,V56,V57,V58,V59),V29=sex_2,X=V35,Y=
V47.

DES-Datalog> query5B(X,Y).
{
  query5B(educrec_0,incwage_4),
  query5B(educrec_1,incwage_1),
  query5B(educrec_1,incwage_2),
```

```
  query5B(educrec_1,incwage_3),
  query5B(educrec_1,incwage_4),
  query5B(educrec_2,incwage_1),
  query5B(educrec_2,incwage_3),
  query5B(educrec_2,incwage_4),
  query5B(educrec_3,incwage_1),
  query5B(educrec_3,incwage_2),
  query5B(educrec_3,incwage_3),
  query5B(educrec_3,incwage_4),
  query5B(educrec_4,incwage_1),
  query5B(educrec_4,incwage_2),
  query5B(educrec_4,incwage_3),
  query5B(educrec_4,incwage_4),
  query5B(educrec_5,incwage_1),
  query5B(educrec_5,incwage_2),
  query5B(educrec_5,incwage_3),
  query5B(educrec_5,incwage_4),
  query5B(educrec_6,incwage_1),
  query5B(educrec_6,incwage_2),
  query5B(educrec_6,incwage_3),
  query5B(educrec_7,incwage_1),
  query5B(educrec_7,incwage_2),
  query5B(educrec_7,incwage_3),
  query5B(educrec_8,incwage_1),
  query5B(educrec_8,incwage_2),
  query5B(educrec_8,incwage_3),
  query5B(educrec_9,incwage_1),
  query5B(educrec_9,incwage_2),
  query5B(educrec_9,incwage_3),
  query5B(educrec_9,incwage_4)
}
Info: 33 tuples computed.
Info: Total elapsed time: 1060 ms.
```

เวลาที่ใช้: ใช้เวลาลดลง (10423 – 1060) = 9363 มิลลิวินาที

ข้อคำถามที่ 6 ให้แสดงประเภทของงาน (classwkg ซึ่งจำแนกได้เป็น classwkg_0=N/A, classwkg_1=self-employed, classwkg_2=works for wages/salary) ลักษณะอาชีพ (occupation ซึ่งจำแนกได้เป็น occupation_1=professional and technical, occupation_2=farmers and operative workers, occupation_3=private and service workers, occupation_4=at home / at school, occupation_5=unknown) และระดับรายได้ (inctot ซึ่งจำแนกได้เป็น inctot_0=less than 0, inctot_1=range 0-999, inctot_2=range 1,000-9,999, inctot_3=range 10,000-99,999, inctot_4=more than 99,999) ของประชากรผิวขาวที่อพยพมาจากยุโรปและเป็นครอบครัวเดี่ยว

```
DES-Datalog> /assert query6(X,Y,Z):-
table1(A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,A11,A12,A13,A14,A15,A16,A
17,A18,A19,A20,A21,A22,A23,A24,A25,A26,A27,A28,A29,A30,A31,A32,
A33,A34),table2(B1,B2,B3,B4,B5,B6,B7,B8,B9,B10,B11,B12,B13,B14,
B15,B16,B17,B18,B19,B20,B21,B22,B23,B24,B25,B26,B27),A1=B1,A9=n
fams_1,A31=raceg_1,A34=bplg_4,X=B9,Y=B27,Z=B14.

DES-Datalog> query6(X,Y,Z).
{
  query6(classwkg_0,occupation_5,inctot_1),
  query6(classwkg_0,occupation_5,inctot_2),
```

```
  query6(classwkg_0,occupation_5,inctot_3),
  query6(classwkg_1,occupation_2,inctot_3),
  query6(classwkg_2,occupation_1,inctot_1),
  query6(classwkg_2,occupation_1,inctot_2),
  query6(classwkg_2,occupation_1,inctot_3),
  query6(classwkg_2,occupation_1,inctot_4),
  query6(classwkg_2,occupation_2,inctot_1),
  query6(classwkg_2,occupation_2,inctot_2),
  query6(classwkg_2,occupation_2,inctot_3)
}
Info: 121 tuples computed.
Info: Total elapsed time: 9852 ms.
```

ผลการปรับปรุงข้อคำถามด้วยการสอบถามจากวิวข้อมูลและโมเดลข้อมูล:

```
DES-Datalog> /assert p(nfams_1)
DES-Datalog> p(C)
{
  p(famunit_1),
  p(farm_1),
  p(nfams_1)
}
Info: 3 tuples computed.
Info: Total elapsed time: 452 ms.
```

```
DES-Datalog> /assert query6B(X,Y,Z):-
view1(V1,V2,V3,V4,V5,V6,V7,V8,V9,V10,V11,V12,V13,V14,V15,V16,V1
7,V18,V19,V20,V21,V22,V23,V24,V25,V26,V27,V28,V29,V30,V31,V32,V
33,V34,V35,V36,V37,V38,V39,V40,V41,V42,V43,V44,V45,V46,V47,V48,
V49,V50,V51,V52,V53,V54,V55,V56,V57,V58,V59),V3=farm_1,V8=nfams
_1,V23=famunit_1,V30=raceg_1,V33=bplg_4,X=V41,Y=V59,Z=V46.

DES-Datalog> query6B(X,Y,Z).
{
  query6B(classwkg_0,occupation_5,inctot_1),
  query6B(classwkg_0,occupation_5,inctot_2),
  query6B(classwkg_0,occupation_5,inctot_3),
  query6B(classwkg_1,occupation_2,inctot_3),
  query6B(classwkg_2,occupation_1,inctot_1),
  query6B(classwkg_2,occupation_1,inctot_2),
  query6B(classwkg_2,occupation_1,inctot_3),
  query6B(classwkg_2,occupation_1,inctot_4),
  query6B(classwkg_2,occupation_2,inctot_1),
  query6B(classwkg_2,occupation_2,inctot_2),
  query6B(classwkg_2,occupation_2,inctot_3)
}
Info: 11 tuples computed.
Info: Total elapsed time: 1015 ms.
```

เวลาที่ใช้: ใช้เวลาลดลง (9852 – (452+1015)) = 8385 มิลลิวินาที

จากผลการทดลองกับข้อคำถามทั้ง 6 รูปแบบ ที่ประกอบด้วย

Q1: ข้อคำถามที่สอบถามข้อมูลเดียว แต่ไม่มีข้อมูลใดสอดคล้องกับเงื่อนไขของข้อคำถาม ทำให้ได้ผลลัพธ์เป็นเซตว่าง

Q2: ข้อคำถามที่สอบถามข้อมูลเดียว และใช้เงื่อนไขเดียวในการคัดเลือกข้อมูล โดยไม่มี semantic rules ใดที่สอดคล้องกับเงื่อนไขนี้ ทำให้ไม่สามารถแปลงรูปแบบข้อคำถามได้

Q3: ข้อคำถามที่สอบถามข้อมูลเดียว และใช้เงื่อนไขเดียวในการคัดเลือกข้อมูล โดยไม่มี semantic rules ใดที่สอดคล้องกับเงื่อนไขนี้ ทำให้ไม่สามารถแปลงรูปแบบข้อคำถามได้

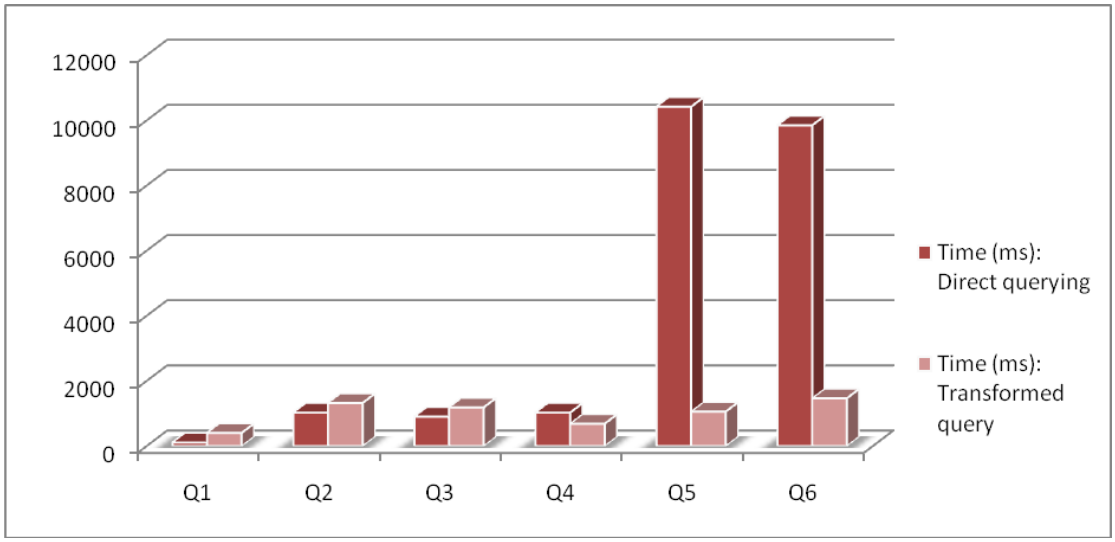
Q4: ข้อคำถามที่สอบถามข้อมูลเดียว แต่ใช้สองเงื่อนไขในการคัดเลือกข้อมูล และมี semantic rules จำนวนสองกฎที่สอดคล้องกับเงื่อนไขนี้ ทำให้แปลงรูปแบบข้อคำถามได้

Q5: ข้อคำถามที่สอบถามสองข้อมูล และใช้เงื่อนไขเดียวในการคัดเลือกข้อมูล ข้อคำถามนี้มีเงื่อนไขที่สอดคล้องกับวิวข้อมูลที่สามารถนำมาใช้ช่วยตอบข้อคำถามนี้ได้

Q6: ข้อคำถามที่สอบถามสามข้อมูล และใช้สี่เงื่อนไขในการคัดเลือกข้อมูล ข้อคำถามนี้มี semantic rules จำนวนสองกฎที่สอดคล้องกับเงื่อนไขทำให้แปลงรูปแบบข้อคำถามได้ และเงื่อนไขยังสอดคล้องกับวิวข้อมูลที่สามารถนำมาใช้ช่วยตอบข้อคำถามนี้ได้

เวลาที่ใช้ในการประมวลผลข้อคำถามทั้งหกข้อคำถาม แสดงเป็นภาพเปรียบเทียบได้ดังรูปที่ 3.11 การเปรียบเทียบเป็นการแสดงเวลาที่ใช้ในการสอบถามโดยตรงจากฐานข้อมูลโดยไม่ต้องมีขั้นตอนการแปลงข้อคำถาม และเวลาที่ใช้ในการสอบถามข้อมูลจากฐานข้อมูลโดยที่ข้อคำถามต้องผ่านขั้นตอนการแปลงรูปแบบข้อคำถามด้วยวิวข้อมูล และ/หรือโมเดลจากการทำเหมืองข้อมูลที่อยู่ในรูปแบบของ semantic rules

จากภาพจะเห็นได้ว่าในกรณีที่ข้อคำถามมีเงื่อนไขที่ไม่สอดคล้องกับ semantic rules หรือไม่ตรงกับโครงสร้างของวิวข้อมูล การประมวลผลข้อคำถามที่ต้องผ่านขั้นตอนการพิจารณาแปลงรูปแบบข้อคำถาม จะใช้เวลามากขึ้นกว่าการสอบถามโดยตรงจากฐานข้อมูล (กรณีข้อคำถามที่ 1, 2 และ 3) แต่เมื่อข้อคำถามมีเงื่อนไขที่สอดคล้องกับโครงสร้างวิวข้อมูล หรือตรงกับเพรดิเคตของ semantic rules การค้นหาข้อมูลจากฐานข้อมูลจะทำได้รวดเร็วขึ้นมาก (กรณีข้อคำถามที่ 4, 5 และ 6) เมื่อพิจารณาเวลาโดยเฉลี่ยของทั้งหกข้อคำถาม การแปลงข้อคำถามด้วยวิวข้อมูลและโมเดลข้อมูลจะลดเวลาการประมวลผลและการค้นหาข้อมูลในฐานข้อมูลลงได้มากถึง 2869.16 มิลลิวินาที



รูปที่ 3.11 เปรียบเทียบเวลาการประมวลผลข้อคำถามระหว่างการสอบถามโดยตรงจากฐานข้อมูลและการแปลงข้อคำถามด้วยวิวข้อมูลและโมเดลข้อมูลก่อนการสอบถามจากฐานข้อมูล

บทที่ 4
บทสรุป

4.1 สรุปผลการวิจัย

ฐานข้อมูล (database) คือ แหล่งที่รวมข้อมูลที่มีความสัมพันธ์เกี่ยวข้องกัน เช่น ข้อมูลพนักงานบริษัท ข้อมูลสำมะโนประชากรและเคหะ เป็นต้น การบันทึกข้อมูล การสอบถาม และการเปลี่ยนแปลงแก้ไขข้อมูลผ่านระบบคอมพิวเตอร์ จะต้องใช้โปรแกรมจำนวนมากช่วยอำนวยความสะดวกให้งานดังกล่าวทำได้อย่างรวดเร็วและถูกต้อง ชุดของโปรแกรมที่ช่วยอำนวยความสะดวกในการสร้างและใช้งานฐานข้อมูลเรียกว่า **ระบบจัดการฐานข้อมูล** (database management system – DBMS) หน้าที่หลักของระบบจัดการฐานข้อมูล คือ อำนวยความสะดวกให้ผู้ใช้สามารถสร้างฐานข้อมูลใหม่และกำหนดเค้าร่าง (schema) หรือโครงสร้างเชิงตรรกะของข้อมูลในฐานข้อมูลได้โดยง่าย ช่วยให้ผู้ใช้สามารถสอบถามข้อมูลด้วยภาษาสอบถาม (query language) หรือเปลี่ยนแปลงแก้ไขข้อมูลด้วยภาษาดำเนินการข้อมูล (data manipulation language – DML) มีกลไกที่ช่วยให้การเก็บและการดำเนินการกับข้อมูลจำนวนมากทำได้อย่างถูกต้องและมีประสิทธิภาพ และมีการป้องกันข้อมูลเสียหายจากอุบัติเหตุหรือจากบุคคลภายนอกที่ไม่มีสิทธิ์ใช้ข้อมูล มีการจัดการในกรณีที่ผู้ใช้หลายคนเรียกใช้งานฐานข้อมูลพร้อมๆ กัน โดยมิให้การดำเนินการกับข้อมูลของผู้ใช้แต่ละคนไปมีผลกระทบต่อผู้ใช้อื่นที่ใช้ฐานข้อมูล ณ เวลาเดียวกันนั้น

การประมวลผลในงานฐานข้อมูลแบ่งได้คร่าวๆ เป็นสองประเภทคือ การประมวลผลข้อคำถาม (query processing) และการประมวลผลด้วยรายการเปลี่ยนแปลง (transaction processing) ข้อคำถามมักจะเป็นการปฏิบัติการสั้นๆ กับข้อมูล เช่น การสอบถามข้อมูลที่ตรงกับความสนใจของผู้ใช้งาน ในขณะที่รายการเปลี่ยนแปลงเป็นการรวมปฏิบัติการกับข้อมูลหลายอย่างให้เป็นหนึ่งรายการ และถือหนึ่งรายการนี้เป็นเสมือนการทำงานหนึ่งอย่างที่ต้องทำให้เสร็จในคราวเดียว งานวิจัยนี้เน้นที่ส่วนของการสอบถามข้อมูล และพัฒนาเทคนิคการประมวลผลข้อคำถามให้สามารถประมวลผลได้รวดเร็วขึ้น

การประมวลผลข้อคำถาม (query processing) เป็นกระบวนการที่ประกอบด้วยหลายขั้นตอน เช่นการตรวจสอบไวยากรณ์และความถูกต้องเบื้องต้นของข้อคำถาม การแปลงข้อคำถามให้เป็นพาสทรี การเปลี่ยนพาสทรีให้เป็นแผนข้อคำถามเชิงตรรกะและเชิงกายภาพ เพื่อเปลี่ยนข้อคำถามที่ผู้ใช้ส่งเข้ามาให้เป็นชุดคำสั่งที่สามารถดำเนินการกับฐานข้อมูล เพื่อหาคำตอบที่ถูกต้องให้กับข้อคำถาม กระบวนการนี้มักจะต้องเกี่ยวข้องกับการเรียกใช้ข้อมูลที่เก็บอยู่ในดิสก์เพื่อนำมาประมวลผลในระบบคอมพิวเตอร์ ขั้นตอนที่ต้องใช้เวลามากที่สุดคือขั้นตอนของการเข้าถึงข้อมูลใน

ดิสก์ เพื่อถ่ายโอนข้อมูลที่ต้องการมายังหน่วยความจำหลัก ถ้าเราสามารถลดจำนวนครั้งของการติดต่อกับดิสก์ (เพื่อถ่ายโอนข้อมูล) ลงได้ จะช่วยให้ลดเวลาในการประมวลผลข้อคำถามลงได้มาก การกระทำเพื่อลดเวลาในการประมวลผลข้อคำถามนี้เรียกว่า *การหาวิธีเหมาะที่สุด* หรือ *การเพิ่มประสิทธิภาพการประมวลผลข้อคำถาม* (query optimization) ซึ่งมักจะปรากฏเป็นส่วนหนึ่งในกระบวนการประมวลผลข้อมูล

การเพิ่มประสิทธิภาพการประมวลผลข้อคำถามมักจะใช้วิธีการวิเคราะห์เงื่อนไขในข้อคำถาม เพื่อเปลี่ยนรูปแบบการถามให้ใช้การเข้าถึงข้อมูลด้วยจำนวนครั้งที่น้อยที่สุด การเพิ่มประสิทธิภาพการประมวลผลข้อคำถามแบ่งได้เป็นสองกลุ่มคือ การเพิ่มประสิทธิภาพเชิงไวยากรณ์ (syntactic query optimization) และการเพิ่มประสิทธิภาพเชิงความหมาย (semantic query optimization) การเพิ่มประสิทธิภาพเชิงไวยากรณ์เป็นการใช้ความรู้เกี่ยวกับจำนวนข้อมูล และโครงสร้างประกอบของข้อมูล เช่น การมีดรรชนีบนบางแอททริบิวต์ ช่วยในการคัดเลือกแผนการประมวลผลและอัลกอริทึมการประมวลผลที่จะใช้หน่วยความจำ และจำนวนครั้งที่ต้อติดต่อกับดิสก์น้อยที่สุด โดยข้อมูลประกอบการตัดสินใจเหล่านี้ปรากฏอยู่ในสารบัญแฟ้มระบบ (system catalog) ทำให้การเพิ่มประสิทธิภาพการประมวลผลข้อคำถาม ด้วยการปรับปรุงข้อคำถามในเชิงไวยากรณ์ทำได้ไม่ยากนัก แต่การเพิ่มประสิทธิภาพเชิงความหมายเป็นการวิเคราะห์ข้อมูลเพื่อนำความรู้ที่ได้มาปรับปรุงข้อคำถามให้มีเงื่อนไขของข้อคำถามที่แตกต่างจากเดิม โดยคาดหมายว่าเงื่อนไขใหม่จะใช้เวลาการประมวลผลที่ลดลง แต่จะให้คำตอบที่ถูกต้องเช่นเดียวกับข้อคำถามเดิมที่ผู้ใช้กำหนด การวิเคราะห์หาความรู้ที่เกี่ยวข้องมาช่วยปรับปรุงข้อคำถามในเชิงความหมาย เป็นงานที่ทำได้ยากเนื่องจากจะต้องเทียบเคียงความหมายของข้อมูล

ในงานวิจัยนี้ได้นำเสนอเทคนิคในการเพิ่มประสิทธิภาพการประมวลผลข้อคำถามในเชิงความหมายด้วยการนำความรู้เกี่ยวกับโมเดลข้อมูล และวิวข้อมูลมาช่วยในการเปลี่ยนแปลง หรือเพิ่มเติมเงื่อนไขในข้อคำถามเดิมของผู้ใช้ ให้มีความจำเพาะเจาะจงมากขึ้นซึ่งจะช่วยให้ลดเวลาการค้นหาคำตอบลงได้ โมเดลข้อมูลที่ใช้ในงานวิจัยนี้เป็นโมเดลเชิงสัมพันธ์ที่ปรับปรุงเทคนิคมาจากวิธีการทำเหมืองข้อมูลแบบการค้นหาความสัมพันธ์ (association mining) แต่จำกัดรูปแบบโมเดลให้อยู่ในลักษณะของ Horn clause

ในฐานข้อมูลเชิงสัมพันธ์ (relational database) ข้อมูลแต่ละเรื่องจะถูกบันทึกอยู่ในแต่ละตารางความสัมพันธ์ (relation) ตารางที่มีข้อมูลบันทึกอยู่จริงจะเรียกว่า**ตารางฐาน** (base table) ผู้ใช้งานฐานข้อมูลแต่ละคนอาจจะใช้ข้อมูลเพียงบางส่วนของตาราง ถ้าเราสร้างตารางขึ้นมาใหม่จากตารางฐาน ให้มีเฉพาะข้อมูลที่เกี่ยวข้องกับผู้ใช้แต่ละฝ่าย จะช่วยให้ลดขนาดและความซับซ้อนของข้อมูลลงได้มาก ตารางที่สร้างขึ้นใหม่นี้เรียกว่า **ทรรศนะ** หรือ **วิว** (view) โดยอาจจะสร้างจากตารางฐานตารางเดียว หรือจากหลายตารางฐานก็ได้ ตารางข้อมูลที่เรียกว่าวิวนี้ จะถูกสร้างขึ้นใช้งานชั่วคราวเพื่อสนับสนุนความต้องการใช้งานข้อมูลที่แตกต่างกันของผู้ใช้แต่ละคน วิวมีลักษณะ

เป็นตารางเหมือนตารางฐาน ต่างกันแต่เพียงว่าข้อมูลในวิวจะไม่ถูกบันทึกไว้อย่างถาวรในฐานข้อมูล ทุกครั้งที่ผู้ใช้เรียกใช้งานวิวตารางชั่วคราวก็จะถูกสร้างขึ้นโดยคัดลอกข้อมูลส่วนที่ต้องการมาจากตารางฐานเมื่อเลิกใช้งานตารางชั่วคราวนี้ก็จะหายไป แต่มีวิวบางชนิดที่ข้อมูลถูกบันทึกไว้อย่างถาวร เรียกวิวชนิดนี้ว่า materialized view เพื่อเน้นว่าวิวประเภทนี้มีข้อมูลบันทึกอยู่จริง ต่างจากวิวทั่วไปที่บันทึกเฉพาะคำอธิบายว่าจะสร้างวิวได้จากตารางฐานใดด้วยเงื่อนไขใด วิวข้อมูลจะต้องใช้เนื้อที่ในดิสก์ในการบันทึกข้อมูลเช่นเดียวกับตารางฐานข้อมูล ดังนั้นวิวข้อมูลจะมีประโยชน์เมื่อข้อคำถามของผู้ใช้เรียกใช้วิวนั้นๆบ่อยครั้ง ในงานวิจัยนี้ใช้ประโยชน์เพิ่มเติมจากวิวข้อมูล ด้วยการพิจารณานำวิวข้อมูลมาใช้ร่วมกับโมเดลข้อมูลในการปรับปรุงข้อคำถามของผู้ใช้ให้สามารถประมวลผลหาคำตอบได้รวดเร็วขึ้น

วัตถุประสงค์หลักของงานวิจัยนี้ต้องการพัฒนาแนวทางการใช้ประโยชน์วิวข้อมูลและโมเดลที่ได้จากการทำเหมืองข้อมูลเพื่อเพิ่มประสิทธิภาพการประมวลผลข้อคำถาม จากการพัฒนาระบบต้นแบบและทดสอบประมวลผลด้วยข้อคำถามตัวอย่างจำนวนหกข้อคำถาม กับฐานข้อมูลจริงซึ่งเป็นข้อมูลประชากรในปีค.ศ.1999 ของประเทศสหรัฐอเมริกา (http://usa.ipums.org/usa/) จำนวน 1,000 เรคคอร์ด พบว่าในกรณีที่ข้อคำถามไม่ซับซ้อนมีเงื่อนไขเพียงเงื่อนไขเดียว การสอบถามโดยตรงกับฐานข้อมูลจะทำได้รวดเร็วกว่าการใช้เทคนิคแปลงข้อคำถามที่พัฒนาขึ้น แต่เมื่อข้อคำถามมีความซับซ้อนมากขึ้นและมีเงื่อนไขที่ต้องพิจารณามาก การใช้เทคนิคแปลงข้อคำถามด้วยวิวข้อมูลและโมเดลข้อมูลจะลดเวลาการประมวลผลลงได้มาก โดยการเปรียบเทียบเวลาการประมวลผลข้อคำถามแสดงโดยสรุปได้ดังตารางที่ 4.1

ตารางที่ 4.1 เปรียบเทียบเวลาการสอบถามข้อมูลโดยตรงจากฐานข้อมูล กับเวลาที่ใช้ในการแปลงข้อคำถามและค้นหาข้อมูลในฐานข้อมูล

ลักษณะข้อคำถาม	เวลาที่ใช้ (มิลลิวินาที)		ประสิทธิภาพที่เพิ่มขึ้น (เวลา)	เปอร์เซ็นต์ของการเพิ่มประสิทธิภาพ
	การสอบถามโดยตรงจากฐานข้อมูล	การแปลงข้อคำถามและสอบถามข้อมูลจากฐานข้อมูล		
Q1: ask one information with a single condition, null answer	110	405	-295	-268.18%
Q2: ask one information with a single condition	1,029	1,325	-296	-28.76%
Q3: ask one information with a single condition	906	1,188	-282	-31.12%
Q4: ask one information with two conditions	1,028	688	340	33.07%
Q5: ask two information with a single condition	10,423	1,060	9,363	89.83%
Q6: ask three information with four conditions	9,852	1,467	8,385	85.10%

4.2 ข้อจำกัดของระบบและข้อเสนอแนะ

เทคนิคการปรับปรุงข้อคำถามเชิงความหมายที่นำเสนอในงานวิจัยนี้ยังจำกัดอยู่เพียงการปรับเปลี่ยนตารางข้อมูลจากตารางฐานเป็นการใช้ตารางจากวิวข้อมูล และใช้โมเดลข้อมูลเป็นฐานความรู้เฉพาะในการเพิ่มเงื่อนไขในข้อคำถามที่ปรับปรุงใหม่ โดยไม่ได้พิจารณาการเปลี่ยนเงื่อนไขในข้อคำถามเดิมของผู้ใช้ และไม่ได้พิจารณาตัดเงื่อนไขที่ไม่เป็นประโยชน์ออกจากข้อคำถาม

การปรับปรุงงานวิจัยนี้สามารถทำได้ต่อไปในแนวทางของการวิเคราะห์ข้อคำถามของผู้ใช้เพื่อค้นหาความตั้งใจของผู้ใช้ว่าต้องการคำตอบเกี่ยวกับอะไร และใช้ฐานความรู้ของระบบเปลี่ยนเงื่อนไขของข้อคำถามเดิมที่อาจจะไม่ถูกต้อง ให้เป็นเงื่อนไขใหม่ที่ถูกต้องและเหมาะสมต่อการประมวลผลอย่างมีประสิทธิภาพมากขึ้น นอกจากนี้ยังอาจเพิ่มแนวทางการค้นหาคำตอบที่เป็นคำตอบโดยประมาณ หรือเป็นการค้นหาคำตอบอื่นที่อาจจะเกี่ยวข้อง แต่ผู้ใช้มิได้กำหนดเงื่อนไขไว้ในข้อคำถามเดิม

จากประสบการณ์การพัฒนาโปรแกรมค้นหาโมเดลจากฐานข้อมูล พบว่าโมเดลในรูปแบบของ Horn clause ที่ได้มีปริมาณมากเกินไป บางคลอสมีข้อความบางส่วนที่ซ้ำซ้อนกับคลอสอื่น และข้อความในหลายคลอสมีลักษณะวนซ้ำ (recursive) ดังนั้นการปรับปรุงประสิทธิภาพของโมเดลเพื่อสามารถนำไปใช้เป็นฐานความรู้ในระบบประมวลผลข้อคำถาม ด้วยการวิเคราะห์โครงสร้างแต่ละกฎและความสัมพันธ์ซึ่งกันและกันในระหว่างกฎ จึงเป็นสิ่งที่จำเป็นต่อการพัฒนาระบบที่เสนอขึ้นให้สามารถใช้ประโยชน์ได้จริงในฐานข้อมูลขนาดใหญ่

บรรณานุกรม

S. Agrawal, R. Agrawal, P.M. Deshpand, A. Gupta, J.F. Noughton, R. Ramakrishnan, & S.Sarawagi, On the computation of multidimensional aggregates, *Proceedings of VLDB*, pp. 506-512, 1996.

J. Chang & S. Lee, Query reformulation using materialized views in data warehousing environment, *Proceedings of First ACM International Workshop on Data Warehousing and OLAP (DOLAP)*, pp. 54-59, 1998.

S. Chaudhuri, S. Krishnamurthy, S. Potamianos, & K. Shim, Optimizing queries with materialized views, *Proceedings of International Conference on Data Engineering*, pp. 190-200, 1995.

S. Chaudhuri, V. Narasayya, & S. Sarawagi, Extracting predicates from mining models for efficient query evaluation, *ACM Transactions on Database Systems*, 29(3), pp. 508-544, 2004.

C.M. Chen & N. Rossopoulos, The implementation and performance evaluation of the ADMS query optimizer: Integrating query result caching and matching, *Proceedings of EDBT*, pp. 323-336, 1994.

Datalog Educational System, version 2.0, http://www.fdi.ucm.es/profesor/fernan/DES/

R. Elmasri & S.B. Navathe, *Fundamental of Database Systems*, third edition, Reading, MA: Addison-Wesley, 2000.

H. Garcia-Molina, J.D. Ullman, & J. Widom, *Database System Implementation*, Upper Saddle River, NJ: Prentice Hall, 2000.

J. Gray, A. Bosworth, A. Layman, & H. Pirahesh, Data cube: A relational operator generalizing group-by, cross-tab and sub-totals, *Proceedings of ICDE*, pp. 152-159, 1996.

A. Gupta, V. Harinarayan, & D. Guass, Aggregate-query processing in data warehousing environments, *Proceedings of VLDB*, pp. 358-369, 1995.

IBM, *IBM intelligent miner scoring, administration and programming for DB2 version 7.1*, New York: IBM, 2001.

A.Y. Levy, A. Rajaramam, & J.J. Ordille, Querying heterogeneous information sources using source descriptions, *Proceedings of VLDB*, pp. 251-262, 1996.

Microsoft Corporation, *OLE DB for data mining*, Redmond, WA: Microsoft Corporation, 2000.

X. Qian, Query folding, *Proceedings of ICDE*, pp. 48-55, 1996.

R. Ramakrishnan & J. Gehrke, *Database Management Systems*, second edition, Singapore: McGraw Hill, 2000.

A. Silberschatz, H.F. Korth, & S. Sudarshan, *Database System Concepts*, fourth edition, Singapore: McGraw Hill, 2002.

D. Srivastava, S. Das, H.V. Jagadish, & A.Y. Levy, Answering queries with aggregation using views, *Proceedings of VLDB*, pp. 318-329, 1996.

ภาคผนวก

ภาคผนวก ก

ผลผลิตของงานวิจัย: บทความวิจัยในเอกสารการประชุมวิชาการ

1. N. Kerdprasop and K. Kerdprasop (2008). The design of an inductive database framework. *Proceedings of 1st Rajamangala University of Technology Conference*, Trang, Thailand, August 27-29.
2. N. Kerdprasop, N. Pannurat, and K. Kerdprasop (2008). Intelligent query answering with virtual mining and materialized views. *World Academy of Science, Engineering and Technology*, Volume 48, December 2008, pp. 84-87.
3. N. Kerdprasop and K. Kerdprasop (2007). Semantic knowledge integration to support inductive query optimization. *Lecture Notes in Computer Science*, Volume 4654 Data Warehousing and Knowledge Discovery (DaWak), September, pp.157-169.
4. K. Kerdprasop, N. Kerdprasop, and A. Ritthongchailert (2007). Query answering in relational inductive databases. *Proceedings of 18th International Workshop on Database and Expert Systems Applications (DEXA)*, Regensburg, Germany, September 3-7, pp. 329-333.
5. A. Ritthongchailert, N. Kerdprasop and K. Kerdprasop (2007). Semantic query optimization with association rule induction. *Proceedings of 33rd Congress on Science and Technology of Thailand*, Walailak University, Nakhon Srithammarat, Thailand, October 18-20.
6. J. Mahawantang, N. Kerdprasop and K. Kerdprasop (2007). Materialized view selection for query rewriting. *Proceedings of 33rd Congress on Science and Technology of Thailand*, Walailak University, Nakhon Srithammarat, Thailand, October 18-20.

การออกแบบกรอบแนวคิดของฐานข้อมูลเชิงอุปนัย

The Design of an Inductive Database Framework

นิตยา เกิดประสพ และ กิตติศักดิ์ เกิดประสพ

Nittaya Kerdprasop and Kittisak Kerdprasop

บทคัดย่อ

ฐานข้อมูลเชิงอุปนัยแตกต่างจากฐานข้อมูลเชิงสัมพันธ์ที่ใช้อยู่โดยทั่วไป ตรงที่ฐานข้อมูลเชิงอุปนัยบันทึกทั้งข้อมูลและแพทเทิร์น(หรือรูปแบบ)ข้อมูล ในระยะแรกของการนำเสนอแนวคิดเกี่ยวกับฐานข้อมูลเชิงอุปนัยมีวัตถุประสงค์หลักเพียงเพื่อให้ฐานข้อมูลสามารถสนับสนุนงานด้านการทำเหมืองข้อมูล แต่ผู้วิจัยมีแนวคิดที่จะออกแบบให้ทั้งระบบฐานข้อมูลและระบบการทำเหมืองข้อมูลสนับสนุนซึ่งกันและกัน โดยในการออกแบบมุ่งเน้นให้สามารถค้นหารูปแบบจากข้อมูล และสามารถนำทั้งข้อมูลและรูปแบบข้อมูลมาใช้ช่วยในการปรับปรุงรูปแบบข้อคำถามและตอบข้อคำถามของผู้ใช้ โดยผลการทดลองเบื้องต้นยืนยันว่าแนวความคิดนี้เป็นประโยชน์ในการเพิ่มประสิทธิภาพการตอบข้อคำถามของระบบฐานข้อมูล

คำสำคัญ : ฐานข้อมูลเชิงอุปนัย, การทำเหมืองข้อมูล

ABSTRACT

Inductive databases can be viewed as a natural extension of traditional databases to contain not only persistent data but also the generalization of stored data, which are called patterns. The idea of inductive databases has been proposed originally as a support system for the knowledge discovery or data mining process. We perceive the concept of inductive databases in a different angle. In stead of designing yet another inductive database system, we are looking for the deployment of an existing inductive query language and environment to support the database tasks. We focus on the task of query answering which has a high potential of being a beneficiary of the stored patterns in inductive databases. Our experimental results of query rewriting technique using induced patterns as a semantic knowledge confirm this advantage.

Key words : Inductive databases, Data mining

หน่วยวิจัยวิศวกรรมข้อมูลและการค้นหาความรู้ สาขาวิชาวิศวกรรมคอมพิวเตอร์ มหาวิทยาลัยเทคโนโลยีสุรนารี

Data Engineering and Knowledge Discovery Research Unit, School of Computer Engineering, Suranaree Univ. of Tech.

_Corresponding author. E-mail: nittaya@sut.ac.th

บทนำ

การทำเหมืองข้อมูล (data mining) หรือการค้นหาความรู้จากฐานข้อมูล (knowledge discovery in databases) เป็นงานวิจัยในสาขาใหม่ที่ได้รับความสนใจอย่างมากจากนักคอมพิวเตอร์ในช่วงทศวรรษที่ผ่านมา ความนิยมนี้มีสาเหตุหลักมาจากการบันทึกข้อมูลลงในระบบฐานข้อมูลกระทำกันอย่างแพร่หลาย ทำให้เกิดข้อมูลอิเล็กทรอนิกส์เป็นปริมาณมหาศาล แต่การใช้ประโยชน์จากข้อมูลเหล่านี้ยังมีน้อยมาก เนื่องจากข้อจำกัดด้านกำลังคนที่จะทำหน้าที่วิเคราะห์ข้อมูลให้ได้ผลผลิตเป็นความรู้ใหม่ที่จะใช้ประโยชน์ได้ต่อไป ดังนั้นเมื่อมีการพัฒนาเทคนิคการทำเหมืองข้อมูลให้สามารถค้นหารูปแบบข้อมูล (patterns or models) ได้โดยอัตโนมัติ จึงทำให้เกิดความคาดหวังว่างานวิเคราะห์เพื่อหารูปแบบข้อมูล จะทำได้รวดเร็วขึ้นและลดภาระของนักวิเคราะห์ข้อมูลให้น้อยลง ผลที่ตามมาคือจะสามารถนำรูปแบบข้อมูลที่ค้นพบไปใช้ประโยชน์ด้านต่าง ๆ ได้รวดเร็วทันต่อความต้องการ

ตัวอย่างความสำเร็จของการใช้ประโยชน์จากเทคโนโลยีการทำเหมืองข้อมูล ได้แก่ การค้นพบพฤติกรรมผู้บริโภคจากฐานข้อมูลการชำระเงิน ณ จุดขาย (point-of-sale) ของลูกค้าในห้างสรรพสินค้าวอลมาร์ต ประเทศสหรัฐอเมริกา การค้นพบรูปแบบพฤติกรรมนี้นำไปสู่การวางแผนที่ดีขึ้นทั้งในด้านการจัดวางสินค้า การจัดการสินค้าคงคลัง รวมไปถึงการวางแผนขนส่งเพื่อกระจายสินค้าไปยังสาขาต่าง ๆ ทำให้วอลมาร์ตสามารถบริหารต้นทุนสินค้าได้อย่างมีประสิทธิภาพ

ถึงแม้จะปรากฏรายงานจำนวนมากที่ชี้ให้เห็นถึงความสำเร็จ และประโยชน์ที่ได้จากการทำเหมืองข้อมูลกับข้อมูลหลากหลายประเภท เช่น ข้อมูลธุรกิจ ข้อมูลทางวิทยาศาสตร์ ข้อมูลทางการแพทย์โดยเฉพาะข้อมูลจีโนมหรือรหัสพันธุกรรม แต่กระบวนการทำเหมืองข้อมูลยังเป็นเทคโนโลยีที่ต้องอาศัยผู้ชำนาญทางด้านนี้โดยเฉพาะ ในปัจจุบันยังไม่สามารถพัฒนาระบบให้ใช้งานได้ง่ายสำหรับผู้ใช้ทั่วไป เนื่องจากการทำเหมืองข้อมูลเป็นกระบวนการทำซ้ำ ที่ต้องมีการปรับปรุงทั้งโครงสร้างและเนื้อหาข้อมูลในแต่ละรอบของการทำงานเพื่อการค้นหารูปแบบที่มีความถูกต้องสูง และเป็นรูปแบบที่น่าสนใจนำไปใช้ให้เกิดประโยชน์ได้จริง

ความก้าวหน้าของการทำเหมืองข้อมูลในปัจจุบัน ถึงแม้ว่ายังไม่สามารถพัฒนากระบวนการไปสู่ลักษณะสำเร็จรูปที่ใช้งานได้ทันทีในแบบ plug-and-play ได้ แต่ก็ได้มีความพยายามที่จะรวบรวมขั้นตอนต่างๆ ของการทำเหมืองข้อมูลที่ในปัจจุบันเป็นขั้นตอนที่แยกจากกัน แต่ละขั้นตอนเป็นอิสระสามารถใช้วิธีการจัดการที่ต่างกันได้ ให้มาอยู่ในระบบปิดที่มีพื้นฐานแนวคิดเดียวกันมีวิธีการจัดการกับข้อมูลและรูปแบบข้อมูลที่เป็นมาตรฐานเดียวกัน โดยในปี ค.ศ.1996 T. Imielinski และ H. Mannila [22] ได้เสนอแนวคิดให้มีการเพิ่มฟังก์ชันการเรียนรู้รูปแบบข้อมูล (pattern mining) ลงในระบบจัดการฐานข้อมูล (database management system – DBMS) และเพิ่มชุดคำสั่ง SQL ให้โปรแกรมเมอร์สามารถระบุเทคนิคในการทำ mining สามารถบันทึกรูปแบบข้อมูลที่ค้นพบเก็บไว้ในฐานข้อมูล และสามารถใช้ภาษา SQL สอบถามรูปแบบข้อมูลที่ต้องการได้

T. Imielinski และ H. Mannila เสนอว่าการรวมความสามารถด้าน pattern mining เข้ากับ DBMS นี้ควรจะเป็นลักษณะ tightly coupling แทนการเพิ่มโมดูลในการทำ mining เป็นระดับชั้นเหนือ DBMS ในลักษณะ loosely coupling การขยายขีดความสามารถของระบบจัดการฐานข้อมูลให้สามารถจัดเก็บและทำงานได้กับทั้งข้อมูล (data) และรูปแบบข้อมูล (patterns) ทำให้เกิดเป็นแนวคิดของฐานข้อมูลชนิดใหม่เรียกว่า **ฐานข้อมูลเชิงอุปนัย** (inductive databases)

ในช่วงต้นทศวรรษที่ 2000 จนกระทั่งถึงปัจจุบันนักพัฒนาโปรแกรมได้มีความพยายามจะปรับปรุงระบบจัดการฐานข้อมูลเชิงสัมพันธ์ที่นิยมใช้อยู่ในปัจจุบัน เช่น IBM DB2, Microsoft SQL Server และ Oracle ให้มีฟังก์ชันและชุดคำสั่งที่ทำงานกับรูปแบบข้อมูลได้ แต่ผลที่ได้ยังไม่เป็นฐานข้อมูลเชิงอุปนัยที่สมบูรณ์ เนื่องจากฟังก์ชันและชุดคำสั่งที่เพิ่มเข้ามายังเป็นลักษณะ loosely coupling แยกส่วนจาก DBMS นอกจากนี้ชุดคำสั่งที่ใช้ระบุเทคนิคการทำ mining ส่วนใหญ่ยังเป็นเพียงแนวคิดหรือโปรแกรมต้นแบบ นอกจากนี้วิธีการแทนและเก็บบันทึกรูปแบบข้อมูลยังมีวิธีการที่แตกต่างกันในแต่ละระบบ งานวิจัยนี้จึงได้เสนอแนวทางการออกแบบระบบจัดการฐานข้อมูลเชิงอุปนัย ที่ใช้มาตรฐานเดียวกันทั้งในส่วนการจัดการกับข้อมูลและส่วนจัดการกับรูปแบบข้อมูล

ลักษณะของฐานข้อมูลเชิงอุปนัย

งานวิจัยส่วนใหญ่ในสาขาการทำเหมืองข้อมูล มีมุมมองเกี่ยวกับฐานข้อมูลว่าเป็นเพียงแหล่งป้อนข้อมูลเข้าสู่ mining phase ของกระบวนการทำเหมืองข้อมูล จึงมักจะพัฒนา mining engine เป็นโมดูลที่แยกจากระบบจัดการฐานข้อมูล (database management system – DBMS) ในปี 1996 T. Imielinski และ H. Mannila [22] ได้เสนอแนวคิดของการปรับปรุง DBMS ให้เป็น KDDMS (knowledge and data discovery management system) เพื่อรองรับได้ทั้งงานด้านการจัดการฐานข้อมูลและการทำเหมืองข้อมูล ภาษาที่ใช้ในการสอบถามและจัดการกับข้อมูล จะต้องได้รับการปรับปรุงให้มีประสิทธิภาพมากกว่าภาษา SQL ที่ใช้ในฐานข้อมูลทั่วไป และโครงสร้างของข้อมูลในฐานข้อมูลจะมีความซับซ้อนขึ้นมากกว่าเป็นเพียงรูปแบบของเรคคอร์ด (หรือทูพเพิล) และรีเลชันที่ใช้อยู่ในฐานข้อมูลปกติ ฐานข้อมูลในรูปแบบใหม่นี้เรียกว่า *ฐานข้อมูลเชิงอุปนัย* (inductive databases) โดยสื่อความหมายถึงแหล่งรวมข้อมูลและรูปแบบข้อมูลที่ค้นพบหรืออุปนัย (induced) มาจากข้อมูล

หลังจากที่ H. Mannila ได้ร่วมเสนอแนวคิดพื้นฐานเกี่ยวกับฐานข้อมูลเชิงอุปนัย ในปีต่อมาเขาได้เสนอรายละเอียดเพิ่มเติมเกี่ยวกับโครงสร้าง และนิยามอย่างเป็นทางการของฐานข้อมูลเชิงอุปนัย [31, 32, 33] ดังนี้

นิยามที่ 1 ฐานข้อมูลเชิงอุปนัย

ฐานข้อมูลเชิงอุปนัย คือ คู่ลำดับ (R, P) โดย R คือความสัมพันธ์หรือรีเลชันในฐานข้อมูล และ P คือแพทเทิร์นของความสัมพันธ์ในฐานข้อมูล โดย P จะอยู่ในรูปแบบของ (Q_R, e) เมื่อ Q_R คือแพทเทิร์นที่ได้จากการสอบถามข้อมูลในฐานข้อมูล และ e เป็นฟังก์ชันประเมินคุณสมบัติของแพทเทิร์น

ข้อมูล *R* และแพทเทิร์น *P* ในรูปที่ 1 แสดงตัวอย่างของฐานข้อมูลเชิงอุปนัย (ปรับปรุงจากตัวอย่างของ J.-F. Boulicaut และคณะ [6, 7]) แพทเทิร์น *P* สร้างขึ้นจากความสัมพันธ์ที่เป็นจริง (แทนด้วยค่า 1) ของแอททริบิวต์ X, Y, Z ในรีเลชัน *R* แพทเทิร์นในตัวอย่างนี้เป็นลักษณะของกฎความสัมพันธ์ (association rules) เช่น $X \Rightarrow Y$ แทนความสัมพันธ์ว่าเมื่อ X มีค่าเป็น 1 แล้ว Y มีค่าเป็น 1 โดยความสัมพันธ์นี้ปรากฏใน 1 เรคคอร์ดจากข้อมูลทั้งหมด 4 เรคคอร์ด จึงมีค่า support = 1/4 = 0.25 และ X มีค่าเป็น 1 จำนวน 3 เรคคอร์ด แต่ในจำนวนนี้ Y มีค่าเป็น 1 เหมือน X เพียงเรคคอร์ดเดียว กฎนี้จึงมีความถูกต้อง หรือมีค่า confidence = 1/3 = 0.33 ดังนั้นในตัวอย่างนี้ Q_R = { $LHS \Rightarrow RHS \mid LHS, RHS \subseteq R$ } และ *e* = (support, confidence)

R

X	Y	Z
1	0	0
1	1	1
1	0	1
0	1	1

P

pattern	support	confidence
X⇒Y	0.25	0.33
X⇒Z	0.50	0.66
Y⇒X	0.25	0.50
Y⇒Z	0.50	1.00
Z⇒X	0.50	0.66
Z⇒Y	0.50	0.66
XY⇒Z	0.25	1.00
XZ⇒Y	0.25	0.50
YZ⇒X	0.25	0.50

รูปที่ 1 ตัวอย่างของข้อมูลและรูปแบบข้อมูลในฐานข้อมูลเชิงอุปนัย

รูปแบบข้อมูลหรือแพทเทิร์น *P* ในรูปที่ 1 เกิดขึ้นจากการใช้คำสั่งเพื่อระบุเกณฑ์ต่างๆ และลักษณะของแพทเทิร์นที่ต้องการ การค้นหาแพทเทิร์นในฐานข้อมูลเชิงอุปนัยนิยามได้ดังต่อไปนี้

นิยามที่ 2 การค้นหาแพทเทิร์นในฐานข้อมูลเชิงอุปนัย

กำหนดให้ r คือข้อมูล (instances) ในรีเลชัน *R* การค้นหาแพทเทิร์นจากคลาสของ L (แทน language ในรูปแบบที่ต้องการ เช่น กฎความสัมพันธ์) คือการค้นหาแพทเทิร์น p ที่ตรงตามเงื่อนไข q (r, p) และเรียกเซตของแพทเทิร์นที่ได้นี้ว่า ทฤษฎี (แทนด้วย Th)

$$Th (L, r , q) = \{ p \in L \mid q (r, p) \text{ is true } \}$$

จากนิยามข้างต้นเงื่อนไข q (r, p) คือ inductive query ที่ใช้ระบุการค้นหาแพทเทิร์น p ที่ผู้ใช้สนใจ จากมุมมองนี้การทำเหมืองข้อมูลจึงถูกพิจารณาว่าเป็นลักษณะหนึ่งของการสอบถาม (querying) ในกรอบคิดของฐานข้อมูลเชิงอุปนัย

งานวิจัยในช่วงระยะเวลาสิบปีที่ผ่านมาของการพัฒนาฐานข้อมูลเชิงอุปนัย แบ่งกลุ่มงานวิจัยได้เป็นสองกลุ่มใหญ่ [41] คือ กลุ่มแรกเน้นการค้นคว้าวิจัยในเชิงทฤษฎีเพื่อกำหนดรากฐานให้กับฐานข้อมูลเชิงอุปนัย โดยจะเน้นการกำหนดรูปแบบมาตรฐานของโครงสร้างข้อมูลในฐานข้อมูล และกำหนดคลาส

มาตรฐานของ inductive queries งานวิจัยในกลุ่มที่สองซึ่งเป็นกลุ่มที่มีนักวิจัยเข้าร่วมเป็นจำนวนมาก จะเน้นในแง่มุมของการใช้งานจริงโดยพยายามปรับปรุง DBMS ที่ใช้อยู่ในปัจจุบัน ให้สามารถทำเหมืองข้อมูลได้ด้วยการเพิ่มโมดูลต่าง ๆ เพื่อการทำ pattern mining และเพิ่มเติมชุดคำสั่ง SQL ให้สามารถสั่งการค้นหาแพทเทิร์นและจัดการกับแพท เทิร์น เช่น การบันทึก การแก้ไขเปลี่ยนแปลง และการสอบถาม

ในกลุ่มของนักวิจัยที่สนใจในแนวทางเชิงทฤษฎีพื้นฐานของฐานข้อมูลเชิงอุปนัยมี L. De Raedt และคณะ [14, 15] เป็นทีมงานหลักในด้านการออกแบบเชิงทฤษฎีโดยใช้ first-order logic เป็นคณิตศาสตร์พื้นฐานของแนวคิด นักวิจัยในกลุ่มนี้จะเน้นการออกแบบโมเดลของฐานข้อมูล [14] และกำหนดทฤษฎีของภาษาเพื่อการสอบถามข้อมูลและแพทเทิร์นจากฐานข้อมูล [15] รวมถึงการกำหนดรูปแบบพีชคณิตเพื่อการประมวลผล inductive queries [29] พีชคณิตนี้สามารถปรับปรุงจาก relational algebra ด้วยการเพิ่มเติมส่วน evaluation function เพื่อประเมินคุณภาพของแพทเทิร์น และส่วนของการปฏิบัติการกับแพทเทิร์น

งานวิจัยของกลุ่มที่เน้นการใช้งานจริงของฐานข้อมูลเชิงอุปนัย ซึ่งเป็นแนวทางที่ได้รับความสนใจจากนักวิจัยจำนวนมาก งานวิจัยส่วนใหญ่ในกลุ่มนี้มีแนวทางที่คล้ายกัน คือมุ่งเน้นไปที่การปรับปรุงภาษา SQL ในลักษณะของการต่อขยาย (SQL-extensions) ภาษาในลักษณะนี้ที่เกิดขึ้นในระยะแรกและได้รับการอ้างอิงถึงมาก ได้แก่ ภาษา DMQL (data mining query language) พัฒนาโดย J. Han และคณะ [21] และภาษาที่ใช้ชุดคำสั่ง MINE RULE พัฒนาโดย R. Meo และคณะ [34, 35, 36] ตัวอย่างในรูปที่ 2 (ดัดแปลงจากตัวอย่างใน [5]) แสดงคำสั่ง DMQL ระบุการค้นหาแพทเทิร์นประเภท characteristic rules จากฐานข้อมูล university_database และกำหนดลักษณะที่เกี่ยวข้อง คือ gpa, birth_place, grant ของนักศึกษาบัณฑิตศึกษาในสาขาคอมพิวเตอร์

```
use database university_database find characteristic rules
related to GPA, birth_place, grant, count (*)%
from students
where  status = "graduate" and major = "cs" with noise threshold = 0.05
```

รูปที่ 2 ตัวอย่างคำสั่งค้นหารูปแบบข้อมูลของภาษา DMQL

การเพิ่มเติมคำสั่ง SQL ของ R. Meo และคณะ ใช้วิธีสร้างคำสั่ง MINE RULE เพื่อค้นหากฎความสัมพันธ์ (association rules) ดังตัวอย่างชุดคำสั่งในรูปที่ 3 (ดัดแปลงจากตัวอย่างใน [5]) ที่แสดงคำสั่งเพื่อใช้ค้นหาความสัมพันธ์ของสินค้า (item) จากรีเลชัน transaction(Date, CustID, Item, Value) โดยมีเงื่อนไขว่าสินค้านั้นจะต้องมีมูลค่าสูงกว่า 100 และลูกค้าซื้อสินค้าในคราวเดียวกันมากกว่า 4 ชิ้น

```
MINE RULE Associations AS
  SELECT DISTINCT 1..n Item AS BODY, 1..1 Item AS HEAD, SUPPORT, CONFIDENCE
   WHERE BODY.Value > 100 AND HEAD.Value > 100
   FROM transaction
   GROUP BY CustID HAVING COUNT(Item) > 4
   CLUSTER BY Date HAVING BODY.Date < HEAD.Date
EXTRACTING RULE WITH SUPPORT: 0.2, CONFIDENCE: 0.5
```

รูปที่ 3 ตัวอย่างคำสั่ง MINE RULE เพื่อการค้นหากฎความสัมพันธ์

ทั้งภาษา DMQL และ MINE RULE มีความสามารถในการค้นหาแพทเทิร์น แต่ยังไม่สามารถสอบถามแพทเทิร์น ความสามารถนี้ได้รับการพัฒนาเพิ่มเติมใน MSQL ที่พัฒนาโดย T. Imielinski และคณะ [23] ภาษา MSQL จำกัดการค้นหาแพทเทิร์นเฉพาะในประเภท association rules และในงานวิจัยต่อมาของทีมงานวิจัยจำนวนมาก [2, 8, 40, 45] ก็จำกัดความสนใจของการพัฒนาฐานข้อมูลเชิงอุปนัยเพื่อค้นหาเฉพาะแพทเทิร์นประเภทนี้เช่นเดียวกัน ทั้งนี้เนื่องจากเป็นแพทเทิร์นที่ใช้มากในงานทางด้านธุรกิจ

ใน DMX (data mining extensions) ของระบบจัดการฐานข้อมูล Microsoft SQL Server [42] นักพัฒนาระบบได้บรรจุคำสั่งในการค้นหาแพทเทิร์นประเภท classification แต่การทำงานกับแพทเทิร์นยังไม่มีขีดความสามารถในระดับการใช้คำสั่งซ้อนกัน นอกจากการขยายขีดความสามารถของ SQL ในฐานข้อมูลเชิงสัมพันธ์แล้ว ยังได้มีงานวิจัยที่พัฒนาภาษาในรูปแบบอื่น เช่น ODMQL (object data mining query language) [16] ที่ใช้กับฐานข้อมูลเชิงวัตถุ PMML (predictive model markup language) [38] ที่ใช้กับฐานข้อมูลเว็บ และจากความสนใจในการผนวกรวมฟังก์ชันการทำเหมืองข้อมูลเข้ากับชุดคำสั่ง SQL ทำให้มีการกำหนดมาตรฐานเป็น ISO SQL/MM [24] และกำหนดมาตรฐานส่วนเชื่อมต่อกับ Java [25]

ตั้งแต่ระยะแรกของการเกิดแนวคิดเกี่ยวกับฐานข้อมูลเชิงอุปนัย นอกจากการใช้ SQL เป็นพื้นฐานในการปรับปรุงเพิ่มเติมคำสั่งในการทำ pattern mining แล้ว ยังมีนักวิจัยในอีกกลุ่มหนึ่งที่ใช้ภาษาเชิงตรรกะเป็นพื้นฐานในการพัฒนาชุดคำสั่งเพื่อการค้นหาแพทเทิร์น นักวิจัยในกลุ่มนี้ ได้แก่ L. Dehaspe และคณะ [11,12] ที่พัฒนาคำสั่งและกระบวนการค้นหา frequent patterns โดยอาศัยพื้นฐานจากภาษา Datalog ซึ่งเป็นภาษาที่ใช้ทำงานกับฐานข้อมูลเชิงนิรนัย (deductive databases) แต่เนื่องจากข้อจำกัดของฐานข้อมูลเชิงนิรนัยที่จะมีประสิทธิภาพด้อยลงเมื่อข้อมูลมีปริมาณมาก C. Goh และคณะ [20] จึงได้เสนอวิธีการสุ่มเพื่อคัดเลือกเฉพาะข้อมูลตัวแทนมาใช้ในขั้นตอนการหา characteristic patterns การเพิ่มความสามารถด้านอุปนัยกับฐานข้อมูลเชิงนิรนัยได้รับความสนใจมาอย่างต่อเนื่อง โดยมีการนำเสนอแนวคิดทั้งในเชิงทฤษฎีพื้นฐาน [1, 3, 30, 39] และในด้านการออกแบบ query เพื่อค้นหาและจัดการกับแพทเทิร์น [13, 17, 19, 26, 44]

ในระยะหลังงานวิจัยด้านฐานข้อมูลเชิงอุปนัย ทั้งในกลุ่มที่พัฒนาทฤษฎีพื้นฐานและกลุ่มที่เน้นการประยุกต์ใช้งานจริง เริ่มลดความชัดเจนของการแบ่งแยกขอบเขตของงาน แนวทางของงานวิจัยเริ่มพัฒนาไปสู่การออกแบบระบบฐานข้อมูลเชิงอุปนัย IDBMS (inductive database management system) ที่มีทฤษฎีพื้นฐานรองรับและเพิ่มความสามารถให้เหมาะสมกับการใช้งานจริงโดยออกแบบ query ที่ใช้ในการค้นหาและจัดการกับแพทเทิร์นได้มากกว่าหนึ่งประเภท งานวิจัยในลักษณะการออกแบบระบบ IDBMS มีทั้งที่ใช้ first-order logic เป็นพื้นฐานของการออกแบบ [18] และที่เป็นลักษณะผสมผสานระหว่างวิธีการทางตรรกะที่มีการทำงานเชิงประกาศ (declarative) และวิธีการโปรแกรมเชิงสั่งงาน (imperative) โดยการเก็บข้อมูลและแพทเทิร์นจะอยู่ในฐานข้อมูลเชิงสัมพันธ์ ระบบในลักษณะผสมนี้ส่วนใหญ่เพิ่งจะเกิดขึ้นและขณะนี้อยู่ในระหว่างการพัฒนา ได้แก่ ระบบ CINQ [9], Psycho [10, 37, 43] และ ConQuest [4]

ในงานวิจัยนี้ ผู้วิจัยมีความสนใจในการพัฒนาระบบฐานข้อมูลเชิงอุปนัยที่ใช้ first-order logic เป็นพื้นฐานของการกำหนดกรอบแนวคิดและการออกแบบการทำงานของระบบ เนื่องจากวิธีการทางตรรกะนี้มี

ขีดความสามารถในเชิงประกาศสูงกว่าภาษา SQL เช่น คำสั่งดังตัวอย่างในรูปที่ 4 (ดัดแปลงจาก [19]) แสดงวิธีการระบุการค้นหาสินค้าที่ลูกค้านิยมซื้อคู่กันบ่อยๆ ของการซื้อของในคราวเดียวกัน การค้นหาสินค้ากระทำกับฐานข้อมูล transaction(date, customer, item, price, quantity) และระบุเงื่อนไขเพิ่มเติมว่าสินค้าที่ถูกซื้อคู่กันบ่อยนี้จะต้องถูกซื้อโดยลูกค้าจำนวนมากกว่า 10 คน

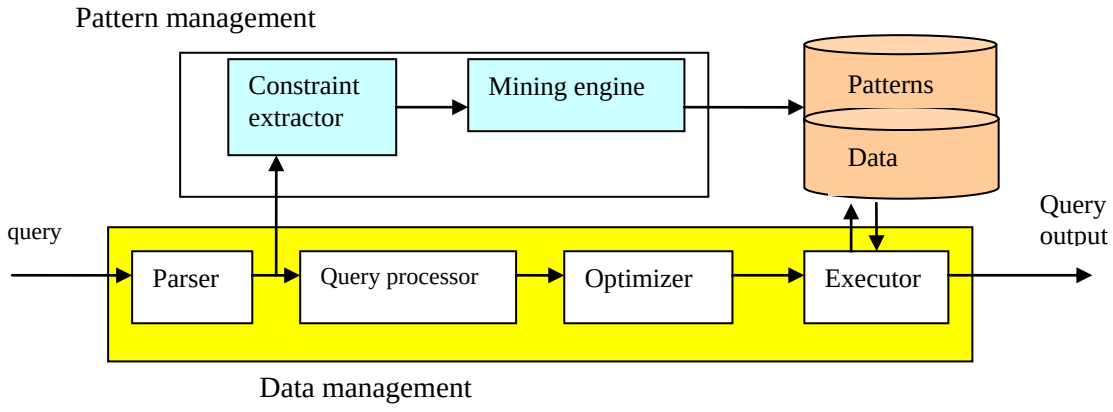
```
pair( I1, I2, count(C)) :- transaction(D, C, I1, _, _),
                           transaction(D, C, I2, _, _),
                           I1 \== I2.
ans(I1, I2) :- pair(I1, I2, C),  C > 10.
```

รูปที่ 4 ตัวอย่างคำสั่งค้นหา frequent pattern ในภาษาเชิงประกาศ

จากตัวอย่างคำสั่งในรูปที่ 4 จะเห็นได้ว่าลักษณะของการทำ pattern matching ในภาษาเชิงประกาศ ทำได้ง่ายกว่าในภาษาเชิงสั่งงาน ภาษาเชิงประกาศที่ใช้หลักการตรรกศาสตร์จึงเหมาะกับงาน pattern mining นอกจากความสามารถในเชิงประกาศแล้ว first-order logic ยังเหมาะสมสำหรับการพัฒนาการทำเหมืองข้อมูล ไปสู่ความสามารถในด้านการค้นหาแพทเทิร์นที่มีการระบุเงื่อนไข (constraint pattern mining) และการค้นหาแพทเทิร์นจากหลายความสัมพันธ์หรือหลายฐานข้อมูลในลักษณะ multi-relation mining

กรอบแนวคิดของการออกแบบฐานข้อมูลเชิงอุปนัย

ในการออกแบบฐานข้อมูลเชิงอุปนัย นอกจากส่วนของการค้นหาและจัดการกับแพทเทิร์นแล้ว ผู้วิจัยยังได้เสนอกรอบแนวคิด (framework) เกี่ยวกับการออกแบบและพัฒนา IDBMS เพิ่มเติมจากที่นักวิจัยต่าง ๆ ได้เสนอไว้แล้ว คือ ให้มีการเพิ่มส่วนเชื่อมโยงของการใช้ประโยชน์จากแพทเทิร์น ป้อนกลับไปยังส่วนประมวลผลข้อคำถาม เพื่อเพิ่มประสิทธิภาพของกระบวนการ query answering และ semantic query optimization [27, 28] องค์ความรู้ใหม่ที่ได้จากงานวิจัยนี้จะช่วยให้สามารถพัฒนาระบบฐานข้อมูลไปสู่ความสามารถทั้งในเชิงอุปนัยและนิรนัยได้สมบูรณ์มากขึ้น กรอบแนวคิดของการออกแบบฐานข้อมูลเชิงอุปนัยแสดงเป็นแผนภาพได้ดังรูปที่ 5



รูปที่ 5 กรอบแนวคิดแสดงส่วนประกอบของระบบจัดการฐานข้อมูลเชิงอุนัย

กรอบแนวคิดของระบบจัดการฐานข้อมูลเชิงอุปนัย (inductive database management system -- IDBMS) ประกอบด้วยส่วนประกอบหลักสองส่วน คือ ส่วนจัดการกับข้อมูล (data management) และส่วนจัดการกับรูปแบบข้อมูล (pattern management) ในส่วนที่จัดการกับรูปแบบข้อมูลมีการออกแบบโครงสร้างมาตรฐานของการแทนข้อมูลและรูปแบบข้อมูล (data and pattern representation) เนื่องจากในปัจจุบันรูปแบบข้อมูลมีได้หลายลักษณะ เช่น generalized rule-based patterns, clustering patterns, association patterns, time-series patterns การกำหนดรูปแบบให้เป็นมาตรฐานเดียวกันเป็นสิ่งจำเป็นสำหรับระบบปิด เพื่อประโยชน์ในการเรียกใช้รูปแบบข้อมูลซ้อนกัน (nested patterns) รวมถึงให้ query สามารถทำงานกับรูปแบบข้อมูลใหม่ ๆ ที่จะเพิ่มเติมขึ้นมาในอนาคตได้

ส่วนจัดการกับข้อมูลได้มีการออกแบบและพัฒนา operators ในการประมวลผล query โดยรูปแบบของ query พัฒนาเพิ่มเติมจาก query ที่ใช้ในภาษา Datalog ซึ่งมีทฤษฎีพื้นฐานจาก first-order logic ชุดคำสั่งที่ออกแบบนี้จะสอบถามได้ทั้งข้อมูลและรูปแบบข้อมูล โดยในงานวิจัยนี้จะพิจารณารูปแบบข้อมูลใน 3 กลุ่มหลัก คือ classification patterns, clustering patterns และ association patterns ในส่วนของระบบการจัดการกับรูปแบบข้อมูล ได้รับการออกแบบให้สามารถทำการค้นหารูปแบบข้อมูลแบบกำหนดเงื่อนไข (constraint pattern mining) การออกแบบวิธีการ update ข้อมูลและรูปแบบข้อมูล จะพิจารณาวิธีการปรับปรุงรูปแบบข้อมูลในแนวทางของ incremental mining ซึ่งจะให้ประสิทธิภาพที่ดีกว่าในแบบ batch

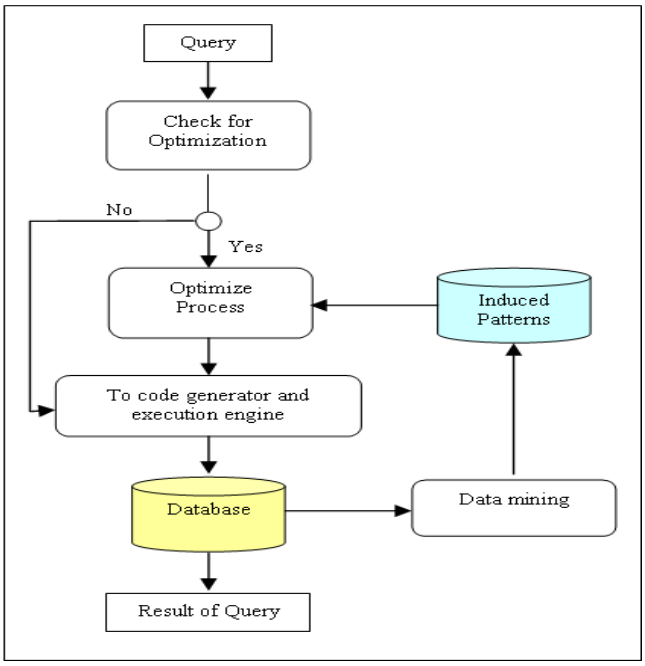
การทดสอบความสามารถของฐานข้อมูลเชิงอุปนัย

การออกแบบระบบฐานข้อมูลเชิงอุปนัยของงานวิจัยนี้ มีจุดมุ่งหมายหลักที่จะให้สามารถนำรูปแบบข้อมูลมาใช้เป็นฐานความรู้เพื่อการปรับปรุงการตอบข้อคำถาม (query optimization) ทำได้ดีขึ้น ขั้นตอนการปรับปรุงข้อคำถามแสดงได้ดังรูปที่ 6 การทดสอบประสิทธิภาพการตอบข้อคำถาม ใช้ข้อมูล customers ซึ่งเป็นข้อมูลสังเคราะห์ มีโครงสร้าง (schema) ดังนี้

customers (customerID, name, address, city, country, birthdate, marital, gender, education, member_card, total_children, occupation, houseowner)

และรูปแบบข้อมูลในลักษณะของ association rules ที่ค้นพบได้จากข้อมูล customers แสดงได้ดังนี้

gender = m	⇒	*marital = s*
total_children = 0	⇒	*marital = s*
total_children = 0	⇒	*gender = m*
gender = m	⇒	*total_children = 0*
houseowner = no	⇒	*marital = s*
member_card = bronze	⇒	*occupation= skilled_manual*
marital = m	⇒	*gender = f*
marital = m	⇒	*houseowner = yes*
city = los_angeles	⇒	*houseowner = yes*
city = nation_city	⇒	*occupation= skilled_manual*



รูปที่ 6 ขั้นตอนการปรับปรุงการตอบข้อคำถามของระบบจัดการฐานข้อมูลเชิงอุนัย

รูปแบบข้อมูลเหล่านี้ถูกนำไปใช้ในการแปลงรูปแบบข้อคำถาม (query rewriting) โดยใช้ตัวอย่างข้อคำถาม Q1 ถึง Q5 สอบถามข้อมูลลูกค้าด้วยเงื่อนไขต่าง ๆ ผลการทดสอบแสดงการปรับปรุงข้อคำถาม (Q1', Q2', Q3', Q4', Q5') ด้วยรูปแบบข้อมูล (pattern) และแสดงเวลาที่ใช้ในการตอบแต่ละข้อคำถาม

Q1: SELECT * FROM customers WHERE city = 'santa cruz' AND gender = 'f';
Pattern: *city = santa cruz* $\Rightarrow$ *gender = m*
Q1': None: detection of unsatisfiable condition
Answer: null
Time(ms):

query	Test#1	Test#2	Test#3	Test#4	Test#5
Q1	50	50	51	50	51
Q1'	0	0	0	0	0

Q2: SELECT * FROM customers WHERE city = 'santa cruz' AND gender = 'm' AND marital = 'm';
Pattern: *city = santa cruz* $\Rightarrow$ *gender = m, marital=s*
Q2': None: detection of unsatisfiable condition
Answer: null
Time(ms):

query	Test#1	Test#2	Test#3	Test#4	Test#5
Q2	109	103	104	101	104
Q2'	0	0	0	0	0

Q3: SELECT * FROM customers WHERE city = 'los angeles' AND houseowner = 'yes';
Pattern: *city = los angeles* $\Rightarrow$ *houseowner = yes*
Q3': SELECT * FROM customers WHERE city = 'los angeles';
Answer: Q3 = 27,660 tuples; Q3'= 27,660 tuples

Time(ms):

query	Test#1	Test#2	Test#3	Test#4	Test#5
Q3	1336	1442	1406	1429	1422
Q3’	1126	1303	1268	1273	1230

Q4: SELECT * FROM customers WHERE gender = ‘m’ AND marital = ‘s’
AND total_children = ‘0’;

Pattern: *gender = m ⇒ marital= s, total_children = 0*

Q4’: SELECT * FROM customers WHERE gender = ‘m’;

Answer: Q4 = 71,916 tuples; Q4’= 71,916 tuples

Time(ms):

query	Test#1	Test#2	Test#3	Test#4	Test#5
Q4	4559	4043	4665	4043	4440
Q4’	3377	3717	3489	3690	3404

Q5: SELECT * FROM customers WHERE city = ‘santa cruz’ AND gender = ‘m’
AND member_card = ‘bronze’;

Pattern: *city = santa cruz ⇒ gender = m*

Q5’: SELECT * FROM customers WHERE gender = ‘m’ AND member_card = ‘bronze’;

Answer: Q5 = 16,596 tuples; Q5’= 16,596 tuples

Time(ms):

query	Test#1	Test#2	Test#3	Test#4	Test#5
Q5	1313	1908	1185	1749	1375
Q5’	936	1274	1074	962	951

จากผลการทดสอบจะเห็นได้ว่า รูปแบบข้อมูลช่วยค้นพบความขัดแย้งในเงื่อนไขของข้อคำถาม Q1 และ Q2 ทำให้สามารถตอบได้ทันทีว่าไม่มีคำตอบ (null answer) ในขณะที่ข้อคำถาม Q3, Q4, Q5 สามารถถูกปรับปรุงให้มีเงื่อนไขลดลงได้ด้วยรูปแบบข้อมูลที่ค้นพบ ทำให้ใช้เวลาลดลงในการตอบข้อคำถามเหล่านั้น

สรุปและข้อเสนอแนะ

งานวิจัยนี้นำเสนอแนวทางในการขยายขีดความสามารถของระบบฐานข้อมูลที่ใช้งานอยู่ในปัจจุบันให้มีขีดความสามารถเพิ่มขึ้นโดยผนวกฟังก์ชันของการทำเหมืองข้อมูล การเพิ่มฟังก์ชันนี้จะมีประโยชน์ในการค้นหาความสัมพันธ์ในลักษณะของ association rules, functional dependencies, semantic constraints และความสัมพันธ์ในรูปแบบอื่น ๆ จากข้อมูลที่เก็บไว้ในฐานข้อมูล ความสัมพันธ์ที่ค้นพบนี้จะเรียกว่ารูปแบบข้อมูล หรือ แพทเทิร์นของข้อมูล ฐานข้อมูลที่บันทึกทั้งข้อมูลและรูปแบบข้อมูลนี้เรียกว่าฐานข้อมูลเชิงอุปนัย

ในการออกแบบระบบจัดการฐานข้อมูลเชิงอุปนัย จึงต้องมีทั้งส่วนจัดการกับข้อมูล และ ส่วนจัดการกับรูปแบบข้อมูล ในงานวิจัยนี้เน้นการออกแบบฐานข้อมูลเพื่อให้สามารถค้นหารูปแบบข้อมูลจากฐานข้อมูลและใช้รูปแบบข้อมูลเพิ่มประสิทธิภาพการตอบข้อคำถาม โดยผลการทดลองเบื้องต้นยืนยันข้อสมมุติฐานว่าแนวความคิดนี้สามารถนำไปใช้ประโยชน์ได้จริง การพัฒนางานวิจัยนี้ต่อไปในอนาคตจึงเป็นแนวทางของการขยายขอบเขตงานวิจัยให้ครอบคลุมความสามารถอื่นของระบบฐานข้อมูล เช่น การจัดการทรานแซคชัน การ update ข้อมูล และการใช้งานจริงกับฐานข้อมูลขนาดใหญ่

กิตติกรรมประกาศ

งานวิจัยนี้ได้รับการสนับสนุนงบประมาณจากสำนักงานคณะกรรมการวิจัยแห่งชาติ สำนักงานคณะกรรมการการอุดมศึกษาและสำนักงานกองทุนสนับสนุนการวิจัย (สกว., รหัสโครงการ RMU-5080026) หน่วยวิจัยด้านวิศวกรรมข้อมูลและการค้นหาความรู้ เป็นหน่วยปฏิบัติการวิจัยที่ได้รับการสนับสนุนการดำเนินงานและงบประมาณบางส่วนจากมหาวิทยาลัยเทคโนโลยีสุรนารี

เอกสารอ้างอิง

[1] M. Aragao and F. Fernandes. Logic-based integration of query answering and knowledge discovery. *Proc. 6th Int. Conf. on Flexible Query Answering Systems*, pp. 68-83, 2004.

[2] M. Botta, J.-F. Boulicaut, C. Masson, and R. Meo. A comparison between query languages for the extraction of association rules. *Proc. 3rd Int. Conf. on Data Warehousing and Knowledge Discovery (DaWaK'02)*, pp. 1-10, 2002.

[3] I. Bartolini, P. Ciaccia, I. Ntoutsi, M. Patella, and Y. Theodoridis. A unified and flexible framework for comparing simple and complex patterns. *Proc. 8th European Conf. on Principles and Practice of Knowledge Discovery in Databases (PKDD'04)*, pp. 496-499, 2004.

[4] F. Bonchi, F. Giannotti, C. Lucchese, S. Orlando, R. Perego, and R. Trasarti. ConQuest: A constraint-based querying system for exploratory pattern discovery. *Proc. IEEE Int. Conf. on Data Engineering (ICDE'06)*, pp. 159-160, 2006.

[5] F. Bonchi, F. Giannotti, C. Lucchese, S. Orlando, R. Perego, and R. Trasarti. On interactive pattern mining from relational databases. *Proc. 5th Int. Workshop on Knowledge Discovery in Inductive Databases (KDID'06)*, pp. 42-62, 2006.

[6] J.-F. Boulicaut, M. Klemettinen, and H. Mannila. Querying inductive databases: A case study on the MINE RULE operator. *Proc. 2nd European Symposium on Principles of Data Mining and Knowledge Discovery (PAKDD'98)*, pp. 194-202, 1998.

[7] J.-F. Boulicaut, M. Klemettinen, and H. Mannila. Modeling KDD processes within the inductive database framework. *Proc. 1st Int. Conf. on Data Warehousing and Knowledge Discovery (DaWaK'99)*, pp. 293-302, 1999.

[8] T. Calders, B. Goetals, and A. Prado. Integrating pattern mining in relational databases. *Proc. 10th European Conf. on Principles and Practice of Knowledge Discovery in Databases (PKDD'06)*, pp. 454-461, 2006.

[9] CINQ project, 2003. http://www.cinq-project.org.

[10] B. Catania, A. Maddalena, and M. Mazza. Psycho: A prototype system for pattern management. *Proc. 31st Int. Conf. on Very Large Data Bases (VLDB'05)*, pp. 1346-1349, 2005.

[11] L. Dehaspe. Frequent pattern discovery in first order logic. *PhD Thesis*, Katholieke Universiteit Leuven, 1998.

[12] L. Dehaspe and H. Toivonen. Discovery of frequent datalog patterns. *Data Mining and Knowledge Discovery*, 3(1):7-36, 1999.

[13] L. De Raedt. A logical database mining language. *Proc. 10th Int. Conf. on Inductive Logic Programming*, pp. 78-92, 2000.

[14] L. De Raedt. A perspective on inductive databases. *SIGKDD Explorations*, 4(2):69-77, 2003.

[15] L. De Raedt M. Jaeger, S. Lee, and H. Mannila. A theory of inductive query answering. *Proc. IEEE Int. Conf. on Data Mining (ICDM'02)*, pp. 123-130, 2002.

[16] M. Elfeky, A. Saad, and S. Fouad. ODMQL: Object data mining query language. *Proc. Int. Symposium on Objects and Databases*, pp. 128-140, 2000.

[17] F. Giannott and G. Manco. Querying inductive databases via logic-based user-defined aggregates. *Proc. 3rd European Conf. on Principles of Data Mining and Knowledge Discovery (PKDD'99)*, pp. 125-135, 1999.

[18] F. Giannott, G. Manco, D. Pedreschi, and F. Turini. Experiences with a logic-based knowledge discovery support environment. *Proc. 6th Italian Congress of Artificial Intelligence*, pp. 202-213, 2000.

[19] F. Giannott, G. Manco, and F. Turini. Specifying mining algorithms with iterative user-defined aggregates. *IEEE Transactions on Knowledge and Data Engineering*, 16(10):1232-1246, 2004.

[20] C. Goh, M. Tsukamoto, and S. Nishio. Knowledge discovery in deductive databases with large deduction results: The first step. *IEEE Transactions on Knowledge and Data Engineering*, 8(6):952-956, 1996.

[21] J. Han, Y. Fu, W. Wang, K. Koperski, and O. Zaiane. DMQL: A data mining query language for relational databases. *Proc. ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery*, pp. 27-34, 1996.

[22] T. Imielinski and H.Mannila. A database perspective on knowledge discovery. *Communications of the ACM*, 39(11):58-46, 1996.

[23] T. Imielinski and V. Virmani. MSQL: A query language for database mining. *Data Mining and Knowledge Discovery*, 3(4):373-408, 1998.

[24] ISO SQL/MM Part 6, 2001. http://www.sql-99.org/sc32/WG4/Progression_Documents/FCD/fcd-datamining-2001-05.pdf.

[25] Java Data Mining API, 2003. http://www.jcp.org/jsr/detail/73.prt.

[26] B. Jeudy and J.-F. Boulicaut, Constraint-based discovery and inductive queries: Application to association rule mining. *Proc. ESF Exploratory Workshop on Pattern Detection and Discovery*, pp. 110-124, 2002.

[27] N. Kerdprasop and K. Kerdprasop. Semantic knowledge integration to support inductive query optimization. *Proc. 8th Int. Conf. on Data Warehousing and Knowledge Discovery (DaWaK'07)*, pp. 157-169, 2007.

[28] K. Kerdprasop, N. Kerdprasop, and A. Ritthongchailert. Query answering in relational inductive databases. *Proc. 18th Int. Workshop on Database and Expert Systems Applications (DEXA'07)*, pp. 329-333, 2007.

[29] S. Lee and L. De Raedt. An algebra for inductive query evaluation. *Proc. IEEE Int. Conf. on Data Mining (ICDM'03)*, pp. 147-154, 2003.

[30] G. Manco. Foundations of a logic-based framework for intelligent data analysis. *PhD Thesis*, University of Pisa, 2001.

[31] H. Mannila. Inductive databases and condensed representations for data mining. *Proc. Int. Symposium on Logic Programming*, pp. 21-30, 1997.

[32] H. Mannila. Theoretical frameworks for data mining. *SIGKDD Explorations*, 1(2):30-32, 2000.

[33] H. Mannila and H. Toivonen. Levelwise search and borders of theories in knowledge discovery. *Data Mining and Knowledge Discovery*, 1(3):241-258, 1997.

[34] R. Meo, G. Psaila, and S. Ceri. A new SQL-like operator for mining association rules. *Proc. 23rd Int. Conf. on Very Large Data Bases (VLDB'96)*, pp. 122-133, 1996.

[35] R. Meo, G. Psaila, and S. Ceri. A tightly-coupled architecture for data mining. *Proc. IEEE Int. Conf. on Data Engineering (ICDE'98)*, pp. 316-322, 1998.

[36] R. Meo, G. Psaila, and S. Ceri. An extension to SQL for mining association rules. *Data Mining and Knowledge Discovery*, 2(2):195-224, 1998.

[37] PANDA project, 2002. http://dke.cti.gr/panda.

[38] Predictive Model Markup Language (PMML), 2003. http://www.dmg.org/pmmlspecs_v2/pmml_v2_0.html

[39] S. Rizzi, E. Bertino, B. Catania, M. Golfarelli, M. Halkidi, M. Terrovitis, P. Vassiliadis, M. Vazirgiannis, and E. Vrachnos. Towards a logical model for patterns. *Proc. 22nd Int. Conf. on Conceptual Modeling (ER'03)*, pp. 77-90, 2003.

[40] S. Sarawagi, S. Thomas, and R. Agrawal. Integrating association rule mining with relational database systems: Alternatives and implications. *Data Mining and Knowledge Discovery*, 4(2-3):89-125, 2000.

[41] A. Siebes. Data mining in inductive databases. *Proc. 4th Int. Workshop on Knowledge Discovery in Inductive Databases (KDID'05)*, pp. 1-23, 2005.

[42] Z. Tang and J. MacLennan. *Data Mining with SQL Server 2005*. John Wiley & Sons, 2005.

[43] M. Terrovitis, P. Vassiliadis, S. Skiadopoulos, E. Bertino, B. Catania, A. Maddalena, and S. Rizzi. Modeling and language support for the management of pattern-bases. *Data and Knowledge Engineering*, 62(2):368-397, 2007.

[44] I. Toroslu and M. Yetisgen-Yildiz. Data mining in deductive databases using query flocks. *Expert Systems with Applications*, 28(3):395-407, 2005.

[45] H. Wang and C. Zaniolo. ATLaS: A native extension of SQL for data mining. *Proc. 3rd SIAM Int. Conf. on Data Mining*, pp. 130-144, 2003.

Intelligent Query Answering with Virtual Mining and Materialized Views

Nittaya Kerdprasop, Natthapon Pannurat and Kittisak Kerdprasop

***Abstract*—**Querying a database is a common task for traditional database systems. Producing answers effectively depends largely on users' knowledge about the query language and the database schema. In order to improve effectiveness and convenience of querying databases, we design a multi-agent system working cooperatively in an intelligent way to analyze user's request and revise the query with virtual mining and materialized views. Virtual mining views are data mining rules discovered from the database and materialized views are pre-computed data. This paper presents work in progress on the implementation and preliminary efficiency tests of the proposed system. The experimental results demonstrate the effectiveness of our multi-agent system in answering queries sharing the same pattern.

***Keywords*—**Multi-agent, query answering, mining views.

I. Introduction

TO query a database is to find some answers from stored data. Traditional database systems return exactly what is being asked. This is a method of direct query answering and a user is required to construct a query intelligently and properly. To remove the burden of intelligence from the database users, the concept of intelligent or cooperative query answering has emerged [Chu and Chen, 1994; Han et al, 1996].

The process of intelligent query answering consists of analyzing the intent of query, rewriting the query based on the intention and other kinds of knowledge, and providing answers in an intelligent way [Lin et al, 2004]. Intelligent answers could be generalized, neighborhood or associated information relevant to the query. This concept is based on the assumption that some users might not have a clear idea of the database content and schema. Therefore, it is difficult to pose queries correctly to get some useful answers.

Knowledge, either intentional or extensional, is the key ingredient of intelligence. Many researchers [Han et al, 1996; Lin et al, 2004; Aragao and Fernandes, 2004] propose to integrate data mining techniques as a knowledge discovery engine to serve an intelligent query answering purpose. We extend this idea by incorporating both virtual mining and materialized views in the query answering system.

Virtual mining views [Calders et al, 2006; Blockeel et al, 2008] are data mining rules discovered from databases and stored as tables, whereas materialized views are view relations computed and stored in the database as well. We consider virtual mining and materialized views as semantic constraints capable of transforming queries to be processed intelligently.

We design a query answering system using multi-agent technology for its advantages of scalability, autonomy and learnability. The rest of the paper is organized as follows. Section 2 presents the related work. Section 3 explains architecture of the proposed query answering system. Section 4 discusses the implementation and some experimental results. Section 5 concludes the paper and indicates our future work.

II. Related work

Evaluating queries efficiently and intelligently requires an important step of query rewriting and modification. Query rewriting is a basic step in query processing aiming at transforming a given query into another more efficient one that uses less time and resources to execute. A rewritten query normally produces the same answer set as the original query.

Query modification [Chaudhuri, 1990] interprets query rewriting in a more relaxing way as a query refining process to produce answers that might be a superset of the expected answers. The advantage of query relaxation is the increased possibility of obtaining desired answers when users have limited knowledge about the problem domain and the database schema.

Early research in query modification [Chaudhuri, 1990; Chu and Chen, 1994] has focused on rewriting the query using generalization concept, neighborhood, and type abstraction hierarchy. The work of Han et al [1996] is among the early research in intelligent query answering that incorporates data mining techniques to rewrite users' queries. Their query relaxation approach employed the notion of generalization to build concept hierarchy.

Lin et al [2004] proposed to integrate neighborhood information and data mining rules discovered from the databases to rewrite the queries. Muslea [2004] introduced the LOQR algorithm to learn some knowledge about the problem domain using a small subset of the database. Then the learned information is used to relax the constraints in the query that originally returns an empty answer.

Aragao and Fernandes [2004] proposed a unified foundation

Nittaya Kerdprasop is a principal researcher of DEKD research unit and an associate professor at the School of Computer Engineering, Suranaree University of Technology, 111 University Ave., Nakhon Ratchasima 30000, Thailand (e-mail: nittaya@sut.ac.th, nittaya.k@gmail.com).

Natthapon Pannurat is a master degree student of School of Computer Engineering, Suranaree University of Technology.

Kittisak Kerdprasop is a director of DEKD research unit and an associate professor at the School of Computer Engineering, Suranaree University of Technology.

for query answering and knowledge discovery. The combined system is called CIDS (Combined Inference Database Systems).

The integration of knowledge discovery and query answering system is also the basis of our research. However, we propose to extend the idea by incorporating not only the knowledge discovered from databases (or virtual mining views), but also the materialized views in the process of query rewriting and answering.

Materialized views are pre-computed data that are stored in the database. Answering queries using views has long been extensively studied [Halevy, 2001; Afrati, 2001; Gou, 2006]. Materialized views can provide useful information in query processing especially in the context of web searching applications. We thus design our query answering system to employ both learned knowledge and materialized views to refine the given query.

III. A Framework of Query Answering System

The proposed query answering system composed of a number of autonomous agents working cooperatively to pursue the common goal of rewriting and producing answers in an intelligent way. The framework of the system is depicted in figure 1.

User interface is a front-end agent to get query from the user and return a final answer set. Constraint extractor is responsible for extracting constraints from the original query and inputs these constraints to the mining agent and the materialized view (MV) manager. A mining agent is thus driven by the query constraints to discover knowledge such as association rules [Agrawal and Srikant, 1994] that are relevant to the given query. MV manager is an agent responsible for view creation, selection and modification.

The data storage thus contains three kinds of information: base relations (data), materialized views created by the MV manager, and virtual mining views discovered by the mining agent. Materialized view definitions and virtual mining views are to be used as semantic constraints by the query rewriter in transforming the given query. Some queries can be answered at this stage, whereas the more complicate ones are sent to the query executor in which base relations and materialized data might be accessed.

IV. Implementation and Experimental Results

We have implemented the mining agent and the MV manager to be triggered by the query constraints. The extracted constraints are employed to guide the association rule mining process [Agrawal and Srikant, 1994] as well as to drive the materialized view creation if the frequency count monitored by the MV manager is above the threshold value. In the preliminary experimentation we set the threshold to be 5 (which means the same constraint has occurred in the queries more than 5 times) and mining agent has been set to find association rules that are 100% accurate and represented in simple form (i.e., one clause in an antecedent part and one clause in the conclusion part). We test efficiency in answering queries of the created materialized views and the discovered association rules using synthetic data. The database contains two base relations of automobile details (data taken from the UCI repository http://www.ics.uci.edu/~mlearn/MLRepository.html) and customer profiles. The examples of discovered association rules, materialized view definitions, original queries and revised queries are given as in figure 2.

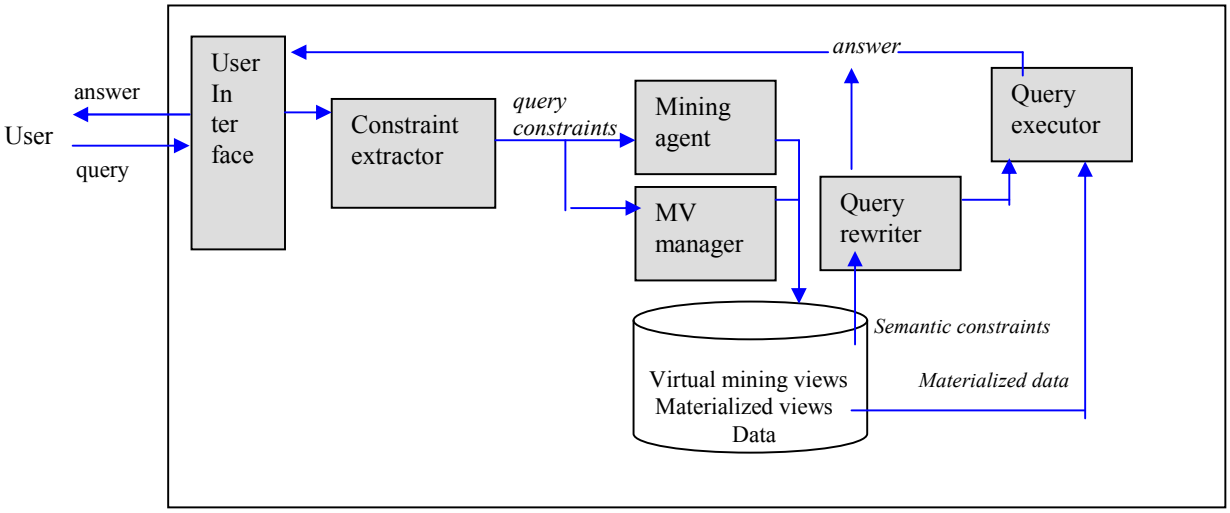


Fig. 1 A multi-agent system for query answering

Association rules:

IF fuel_type = diesel THEN fuel_system = idi
IF engine_type = ohc THEN engine_location = front
IF num_cylinders = four THEN engine_location = front

Materialized view definitions:

MV1(CustomerID, Name, Phone, Address)
MV2(CustomerID, Name, OwnCar)
MV3(CustomerID, Name, Phone, Address, OwnCar)

Given query

```
Q1: SELECT *   FROM automobile
    WHERE   fuel_type = 'diesel'
     AND  fuel_system = 'idi'
     AND  num_cylinders = 'four'
     AND  engine_location = 'front'
```

Transformed query

```
Q1': SELECT *   FROM automobile
     WHERE   fuel_type = 'diesel'
     AND   num_cylinder = 'four'
```

```
Q2:  SELECT  CustomerID, Name,
             Address, automobile.Make
     FROM    automobile, customer
     WHERE  customer.carID = automobile.ID
```

Transformed query

```
Q2': SELECT  CustomerID, Name, Address, OwnCar
     FROM    MV3
```

Fig. 2 Examples of rules and view definitions used in query answering

TABLE I
Processing time (in millisecond) of given queries and transformed queries

	Q_{1A}	Q_{1A}'	Q_{1B}	Q_{1B}'	Q_{1C}	Q_{1C}'	Q_{2A}	Q_{2A}'	Q_{2B}	Q_{2B}'
Time	1132	1541	1011	997	1173	1004	5431	9158	6022	5518
Gain	none		1.38%		14.41%		none		8.37%	

We generated more than twenty queries tested on the sample database. Query processing time of the given queries and the revised queries are observed and shown some results as in table 1. Processing time of the revised queries includes association rule mining time and time to create materialized views. Queries Q_{1A}, Q_{1B}, Q_{1C} ask the database with quite similar constraints. This is also the case for queries Q_{2A} and Q_{2B}. It can be noticed from the results that there is no profit on transforming the given query at first occasion because it takes much time on mining for association rules or creating materialized views. However, processing time reduction is getting better for the subsequent queries asking with almost the same constraints. It could be inferred from the experimental results that the proposed architecture of query answering would be a payoff for the situation that users asking the same or similar constraints over the databases. The longer the database persists and users keep on asking the same things, the higher gain in terms of processing time is expected.

I. CONCLUSION AND FUTURE WORK

We design and implement a query answering system to provide an integrated, flexible and efficient platform supported by a community of agents. To answer queries effectively the mining agent and the MV manager are two key players to derive useful knowledge relevant to the given query. Query rewriter supported by intelligent transformation rules and co-operated with query executor is expected to produce answers in an intelligent way. The preliminary experimental results satisfy the expectation. We are, however, improving the capability of these agents to analyze the user's intent and preferences to better providing associated information. Extending the scope of this project towards the distributed environment is also the direction of our future work.

ACKNOWLEDGMENT

This work was supported by the Thailand Research Fund under grant RMU-5080026 and the National Research Council of Thailand. The authors are with the Data Engineering and Knowledge Discovery (DEKD) research unit which is supported by research fund from Suranaree University of Technology.

REFERENCES

[1] Afrati, F. et al, 2001. Generating efficient plans for queries using views. *Proceedings of ACM SIGMOD International Conference on Management of Data*, pp. 319-330.

[2] Agrawal, R. and Srikant, R., 1994. Fast algorithm for mining association rules. *Proceedings of 20th International Conference on Very Large Data Bases*, pp. 487-499.

[3] Aragao, M. and Fernandes, A., 2004. Logic-based integration of query answering and knowledge discovery. *Proceedings of 6th International Conference on Flexible Query Answering Systems*, pp. 68-83.

[4] Blockeel, H. et al, 2008. Mining views: database views for data mining. *Proceedings of 24th IEEE International Conference on Data Engineering*, pp. 1608-1611.

[5] Calders, T. et al, 2006. Integrating pattern mining in relational databases. *Proceedings of 10th European Conference on Principles and Practice of Knowledge Discovery in Databases*, pp. 454-461.

[6] Chaudhuri, S., 1990. Generalization and a framework for query modification. *Proceedings of 6th IEEE International Conference on Data Engineering*, pp. 138-145.

[7] Chu, W. and Chen, Q., 1994. A structured approach for cooperative query answering. *IEEE Transactions on Knowledge and Data Engineering*, Vol.6, pp. 738-749.

[8] Gou, G. et al, 2006. Query evaluation using overlapping views: completeness and efficiency. *Proceedings of ACM SIGMOD International Conference on Management of Data*, pp. 37-48].

[9] Halevy, A., 2001. Answering queries using views: a survey. *The VLDB Journal*, Vol.10, No.4, pp. 270-294.

[10] Han, J. et al, 1996. Intelligent query answering by knowledge discovery techniques. *IEEE Transactions on Knowledge and Data Engineering*, Vol.8, No.3, pp. 373-390.

[11] Lin, T. et al, 2004. Intelligent query answering based on neighborhood systems and data mining techniques. *Proceedings of the International Database Engineering and Applications Symposium*, pp. 91-96.

[12] Muslea, I., 2004. Machine learning for online query relaxation. Proceedings of 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 246-255.

Nittaya Kerdprasop is an associate professor at the school of computer engineering, Suranaree University of Technology, Thailand. She received her B.S. from Mahidol University, Thailand, in 1985, M.S. in computer science from the Prince of Songkla University, Thailand, in 1991 and Ph.D. in computer science from Nova Southeastern University, USA, in 1999. She is a member of ACM and IEEE Computer Society. Her research of interest includes Knowledge Discovery in Databases, AI, Logic Programming, Deductive and Active Databases.

Natthapon Pannurat is a master degree student of School of Computer Engineering, Suranaree University of Technology. His master thesis is the application of data mining technique, association analysis particularly, to assist database design and normalization.

Kittisak Kerdprasop is an associate professor at the school of computer engineering, Suranaree University of Technology, Thailand. He received his bachelor degree in Mathematics from Srinakarinwirot University, Thailand, in 1986, master degree in computer science from the Prince of Songkla University, Thailand, in 1991 and doctoral degree in computer science from Nova Southeastern University, USA., in 1999. His current research includes Data mining, Artificial Intelligence, Functional Programming, Computational Statistics.

Semantic Knowledge Integration to Support Inductive Query Optimization

Nittaya Kerdprasop and Kittisak Kerdprasop

Data Engineering and Knowledge Discovery Research Unit,
School of Computer Engineering, Suranaree University of Technology,
111 University Avenue, Nakhon Ratchasima 30000, Thailand
{nittaya,kerdpras}@sut.ac.th

Abstract. We study query evaluation within a framework of inductive databases. An inductive database is a concept of the next generation database in that the repository should contain not only persistent and derived data, but also the patterns of stored data in a unified format. Hence, the database management system should support both data processing and data mining tasks. Having provided with a tightly-coupling environment, users can then interact with the system to create, access, and modify data as well as to induce and query mining patterns. In this paper, we present a framework and techniques of query evaluation in such an environment so that the induced patterns can play a key role as semantic knowledge in the query rewriting and optimization process. Our knowledge induction approach is based on rough set theory. We present the knowledge induction algorithm driven by a user's query and explain the method through running examples. The advantages of the proposed techniques are confirmed with experimental results.

1 Introduction

Since the emerging of knowledge discovery in databases (KDD), or data mining, as a new multi-disciplinary research area in the 1990's [9], it has been soon realized that the current database system should be extended or re-designed to support the KDD process. Imielinski and Mannila [13] have argued that existing KDD techniques are simply file mining because the inductive learning tools are built on top of the databases assuming a loose coupling between the two components. To fulfill a database mining concept, the mining engine has to be tightly coupled with the database system. In recent years, this idea has been realized and several research work along this line have been developed [2, 20, 23]. The tightly integration of databases with data mining gives rise to the new concept of inductive databases [5, 7, 19].

An inductive database is a database that contains not only data, but also patterns which are generalized information induced from data. By providing this tightly integration framework of data management system and pattern discovery engine, users can access patterns in the same manner as querying data. To achieve this aim a number of SQL-based inductive query languages, such as DMQL [11], MINE RULE [4], MSQL [14], have been proposed and implemented. Most of these languages are an SQL extension with some primitives to support the data mining task, that is, users can pose queries to induce, access and update patterns.

I.Y. Song, J. Eder, and T.M. Nguyen (Eds.): DaWaK 2007, LNCS 4654, pp. 157–169, 2007.

Besides the front-end functionalities, we propose that the induced patterns can also be useful in the back-end part of query answering. The induced patterns are viewed as a repository of semantic knowledge highly beneficial to the optimization process. The purpose of query optimization is to rewrite a given query into an equivalent one that uses less time and resources. Equivalence is defined in terms of identical answer sets. Query optimization utilizes syntactic and logic equivalence transformation of a given query expression. Semantic query optimization (SQO), on the contrary, uses not only syntactic transformations, but also semantic knowledge, such as integrity constraints and various forms of data generalization, to transform a query into an optimized one.

Early work on SQO [10, 15] transforms query by reasoning via heuristics such as index and restriction introduction, join elimination, contradiction detection. Since the introduction of SQO concept in 1981 [15], semantic-based transformation techniques have been developed constantly. Some proposed techniques in the literature are resolution refutation method [6], knowledge deduction [24], knowledge induction [25]. Recently, the interest on SQO has moved toward the setting of intelligent query answering [12, 17], which is defined as a procedure that can answer incorrect or incompletely specified query cooperatively and intelligently. The intelligence is obtained by analyzing the intent of a query and provide some generalized or associated answers. Necib and Freytag [21] propose an ontology-based optimization approach to rewrite a query into another one which is not necessary equivalent but can provide more meaningful result satisfying the user's intention.

Our research follows the direction of intelligent query answering with the emphasis on semantic-based optimization. We consider acquiring semantic knowledge using a rough set approach. By means of a rough set theory certain knowledge as well as rough (or vague) knowledge can be induced from the database content. The main purpose of this paper is to illustrate the idea of inducing and integrating certain and rough knowledge in the query rewriting and optimization process to produce an intelligent answer. We present the optimization process within the framework of inductive database systems that both data content and patterns are stored in the databases. Unlike previous work on inductive databases that express queries using a logic-based language [3, 4, 8], we formalize our idea based on a structured query language (SQL) as it is a typical format used extensively in most database systems.

The remainder of this paper is organized as follows. In section 2, we review the two important foundations of our work, that is, the relational inductive databases and rough set theory. We present our framework and algorithm of semantic knowledge induction using rough set concept in section 3. Section 4 illustrates the steps in query optimization with some experimental results. Section 5 concludes the paper and discusses the plausible extension of this research.

2 Preliminaries

2.1 Inductive Database Concept

Inductive databases can be viewed as an extension of the traditional database systems in that the databases do not only store data, but they also contain patterns of those data. Mannila [18] formalized a framework of inductive database $\mathcal{I}$ as a pair ($\mathcal{R}$, $\mathcal{P}$) where $\mathcal{R}$ is a database relation and $\mathcal{P}$ is a nested relation of the form ($\mathcal{Q}_{\mathcal{R}}$, e) in which $\mathcal{Q}_{\mathcal{R}}$ is a set

of patterns obtained from querying the base data and e is the evaluation function measuring some metrics over the patterns. As an example, consider the database (adapted from [3]) consisting of one base relation, $\mathcal{R}$. The induced patterns $\mathcal{P}$ are a set of rules represented as an implication LHS $\Rightarrow$ RHS; therefore, $\mathcal{Q}_{\mathcal{R}}$ = { LHS $\Rightarrow$ RHS | LHS, RHS $\subseteq \mathcal{R}$ } and the rule's quality metrics are support and confidence [1]. An inductive database $\mathcal{I} = (\mathcal{R}, \mathcal{P})$ containing one base relation $\mathcal{R}$ and a set $\mathcal{P}$ of all association patterns induced from $\mathcal{R}$ is shown in figure 1.

$\mathcal{R}$

X	Y	Z
1	0	0
1	1	1
1	0	1
0	1	1

$\mathcal{P}$

pattern	support	confidence
X $\Rightarrow$ Y	0.25	0.33
X $\Rightarrow$ Z	0.50	0.66
Y $\Rightarrow$ X	0.25	0.50
Y $\Rightarrow$ Z	0.50	1.00
Z $\Rightarrow$ X	0.50	0.66
Z $\Rightarrow$ Y	0.50	0.66
XY $\Rightarrow$ Z	0.25	1.00
XZ $\Rightarrow$ Y	0.25	0.50
YZ $\Rightarrow$ X	0.25	0.50

Fig. 1. An example of inductive database instance

Given the framework of an inductive database $\mathcal{I}$, users can query both the stored data (the part of $\mathcal{I}.\mathcal{R}$ in figure 1) as well as the set of patterns (the $\mathcal{I}.\mathcal{P}$ part). Formalization of inductive queries to perform data mining tasks has been studied by several research groups [5, 8]. We are, however, interested in the concept of inductive databases from a different perspective. Instead of using a sequence of queries and operations to create the induced patterns such as association rules [1], we shift our focus towards the induction of precise and rough rules and then deploy the stored information to support query answering. We unify the pattern representation to the relation format normally used in relational databases and call it relational inductive databases.

2.2 Rough Set Theory

The notion of rough sets has been introduced by Zdzislaw Pawlak in the early 1980s [22] as a new concept of set with uncertain membership. Unlike fuzzy set, uncertainty in rough set theory does not need probability or the value of possibility to deal with vagueness. It is rather formalized through the simple concepts of lower and upper approximation, which are in turn defined on the basis of set. Rough set concepts are normally explained within the framework of a decision system. The basic idea is partitioning universe of discourse into equivalence classes.

Definition 1. A *decision system* is any system of the form $\mathcal{A}$ = <U, A, d>, where U is a non-empty finite set of objects called the universe, A is a non-empty finite set of conditions, and d $\notin$ A is the decision attribute.

Definition 2. Given a decision system $\mathcal{A}$ = <U, A, d>, then with any B ⊆ A there exists an *equivalence* or *indiscernibility relation* $I_{\mathcal{A}}$(B) such that $I_{\mathcal{A}}$(B) = {(x, x') ∈ UxU | ∀a ∈ B [a(x) = a(x')] }.

Table 1. A students' grading decision table

	Conditions			*Decision*
	age	score1	score2	grade
s1	19	0-20	0-20	fail
s2	19	0-20	21-40	fail
s3	20	0-20	41-60	fail
s4	20	0-20	41-60	fail
s5	19	0-20	81-100	pass
s6	19	41-60	41-60	pass
s7	19	21-40	61-80	pass
s8	20	21-40	21-40	pass

From the data samples in table 1, the followings are equivalent relations.

I(age) = {{s1, s2, s5, s6, s7}, {s3, s4, s8}}
I(score1) = {{ s1, s2, s3, s4, s5}, {s6}, {s7, s8}}
I(score2) = {{s1},{s2, s8},{s3, s4, s6}, {s5}, {s7}}
I(age, score1) = {{s1, s2, s5},{s3, s4},{s6},{s7},{s8}}
I(age, score2) = {{s1},{s2},{s3, s4},{s5},{s6},{s7},{s8}}
I(score1, score2) = {{s1},{s2},{s3, s4},{s5},{s6},{s7},{s8}}
I(age, score1, score2) = {{s1},{s2},{s3, s4},{s5},{s6},{s7},{s8}}

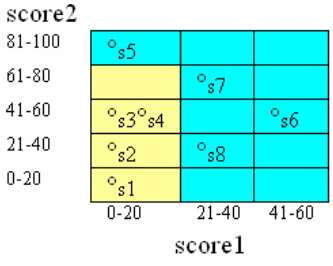
Equivalence relations partition the universe into groups of similar objects based on the values of some attributes. The question often arises is whether one can remove some attributes and still preserve the same equivalence relations. This question leads to the notion of reduct [16].

Definition 3. Let $\mathcal{A}$ = <U, A, d> be a decision system and P, Q ⊆ A, P ≠ Q be two different sets of conditions. The set P is the *reduct* of set Q if P is minimal (i.e. no redundant attributes in P) and the equivalence relations defined by P and Q are the same.

It can be seen from the listed equivalence relations that *I*(age, score2) = *I*(score1, score2) = *I*(age, score1, score2). Therefore, (age, score1) and (score1, score2) are reducts of (age, score1, score2). The intersection of all reducts produces *core attributes*. According to our example, score2 is a core attribute. A reduct table of (score1, score2) and its partitions are shown in figure 2(a). If we are interested in the decision criteria for the pass grade, we can infer decision rules from the reduct table in figure 2(a) as follows.

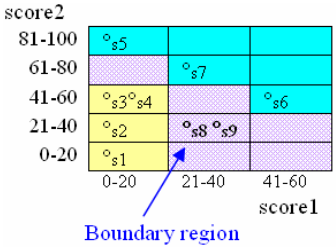
IF (score1 = 0-20 ∧ score2 = 81-100) THEN grade = pass
IF (score1 = 21-40 ∧ score2 = 21-40) THEN grade = pass
IF (score1 = 21-40 ∧ score2 = 61-80) THEN grade = pass
IF (score1 = 41-60 ∧ score2 = 41-60) THEN grade = pass

	score1	score2	grade
s1	0-20	0-20	fail
s2	0-20	21-40	fail
s3	0-20	41-60	fail
s4	0-20	41-60	fail
s5	0-20	81-100	pass
s6	41-60	41-60	pass
s7	21-40	61-80	pass
s8	21-40	21-40	pass



(a) equivalence relations on 8 students

	score1	score2	grade
s1	0-20	0-20	fail
s2	0-20	21-40	fail
s3	0-20	41-60	fail
s4	0-20	41-60	fail
s5	0-20	81-100	pass
s6	41-60	41-60	pass
s7	21-40	61-80	pass
s8	21-40	21-40	**pass**
s9	21-40	21-40	**fail**



(b) equivalence relations on 9 students

Fig. 2. A reduct table and (a) its partition into equivalence relations, each one is represented by a rectangular region, (b) equivalence relations with conflicting cases of s8 and s9

Suppose we are given additional information of the ninth student as shown in figure 2(b), then the above decision rules for the passing grade is no longer valid. It can be seen from figure 2(b) that s8 and s9 are in the same equivalence relation but their grades are different. It is such conflicting cases that inspires the rough set concept. Given the two decision sets of pass/fail, the uncertain cases such as s8 and s9 can be approximated their set membership by means of lower and upper approximation [16].

Definition 4. Let $\mathcal{A}$ = <U, A, d> be a decision system, B $\subseteq$ A, X $\subseteq$ U be objects of interest and $[x]_B$ denote the equivalence class of $I_{\mathcal{A}}(B)$. The *B-lower approximation* and *B-upper approximation* of X, denoted by *bX* and *BX* respectively, are defined by $bX = \{x \mid [x]_B \subseteq X\}$ and $BX = \{x \mid [x]_B \cap X \neq \varnothing\}$. The area between B-lower approximation and B-upper approximation is called *B-boundary region* of X, *BN,* and defined as $BN = BX - bX$.

The lower approximation of X is the set of all objects that certainly belong to X. This set is also called *B-positive region* of X. The *B-negative region* of X is defined as U−BX, or the set of all objects that definitely not belong to X. The *B-boundary region* of X is the set of all objects that cannot be classified as not belonging to X.

Given the information as shown in figure 2(b), B = {score1, score2} and X = {s5, s6, s7, s8} be set of students with passing grade, then *bX* = {s5, s6, s7} and *BX* = {s5, s6, s7, s8, s9}. The boundary region *BN* = {s8, s9}. *B-negative region* of X is {s1, s2, s3, s4} or the set of all students who definitely fail the exam.

If the boundary region is empty, it is a *crisp* (precise) set; otherwise, the set is *rough*. The set of passing students in figure 2(a) is a crisp set, whereas it is a rough set in figure 2(b). Decision rules generated from a rough set comprise of certain rules generated from the positive and negative regions, and possible rules generated from the boundary region.

Such method to generate decision rules is static because the decision attribute is defined in advance. Within the framework of query answering that decision attributes are usually not known in advance, the classical static rough set methodology is certainly impractical. We thus propose in the next section our method of dynamic rule induction driven by the query predicates.

3 Certain and Rough Knowledge Induction

3.1 A Framework for Semantic Knowledge Induction

In the typical environment of database systems, the size of data repository can be very large. With the classical rough set method that all prospective decisions have to be pre-specified, the number of generated rules can be tremendous. We thus propose a dynamic approach by taking predicate in the user's query to be a decision attribute at query processing time. By this scheme, we can limit the induction to only relevant decision rules and these rules are subsequently used as semantic knowledge in the process of query rewriting and optimization. The framework of our approach is shown in figure 3.

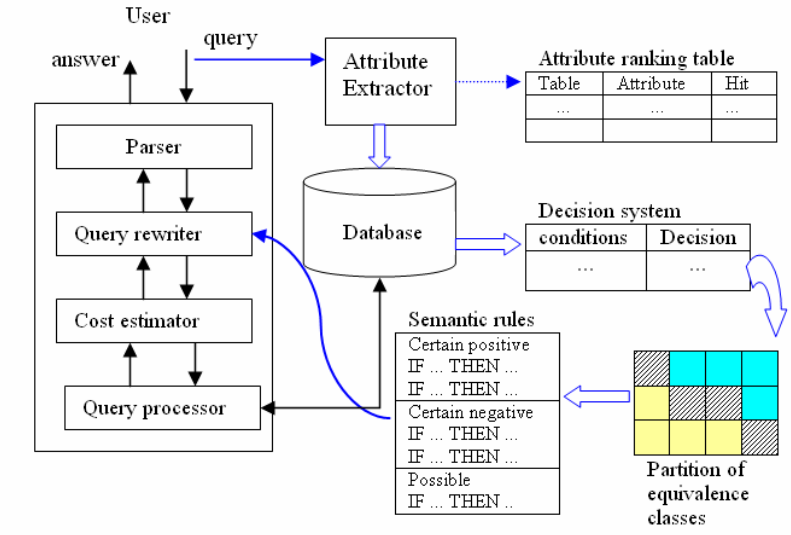


Fig. 3. A framework of query-driven induction for rough and precise knowledge

In our proposed framework, rule induction is invoked by user's query. Once the query has been posted, the component named *attribute extractor* has been called to extract the table's and attribute's name from the query. The *attribute ranking table* is

also created to collect the history of attributes used in the queries. The column *hit* counts the number of times that attributes has been used. The approach of inducing rules based on query's attribute is described in the algorithm as shown in figure 4.

Algorithm. Query-driven semantic rule induction

Input: User's query, a database and background knowledge
Output: Certain and rough semantic rules

1. Call *Attribute Extractor* to extract table names T_i and attribute names A_j from the query
2. Access the *attribute ranking table* and update the hit counter identified by each T_i and A_j and sort the counter in descending order
3. Create a decision table $\mathcal{A}$ = <U, A, d> where d = A_i, A = a set of attributes in T_i, U = a set of tuples in T_i
4. Pre-process $\mathcal{A}$ by
   - removing attributes with number of distinct values = $|\mathcal{A}|$
   - discretizing attributes with real values
5. Partition U into equivalence classes and search for the first reduct *R*
6. From R, identify *bX*, *BX*, *BN* regions, then generate certain positive, certain negative, and possible rules
7. Generalize all three classes of rules using available background knowledge
8. Return the final rule set

Fig. 4. A query-driven semantic rule induction algorithm

3.2 Running Examples

We use the student data shown in table 1 with additional record <s9, 20, 21-40, 21-40, fail> as our running example. The information on interval order that 81-100 > 61-80 > 41-60 > 21-40 > 0-20 is used as background knowledge for rule generalization.

Example 1. Suppose there is a query asking whether the score1 = 55 is high enough for the passing grade.
Method:

(1) This query asks about grade with score1 as a condition. Hence, a reduct table as in figure 2(b) is constructed.
(2) Then, the following rules are generated.

Certain positive rules: IF (score1=0-20 ∧ score2=81-100) THEN grade = pass
IF (score1=21-40 ∧ score2=61-80) THEN grade = pass
IF (score1=41-60 ∧ score2=41-60) THEN grade = pass

Certain negative rules: IF (score1=0-20 ∧ score2=0-20) THEN grade = fail
IF (score1=0-20 ∧ score2=21-40) THEN grade = fail
IF (score1=0-20 ∧ score2=41-60) THEN grade = fail

Possible rules: IF (score1=21-40 ∧ score2=21-40) THEN grade = pass

(3) The three classes of rules are generalized according to the given background knowledge. The final rules are as follow.

R1: IF (score1 > 20 ∧ score2 > 60) THEN grade = pass
R2: IF (score1 > 40 ∧ score2 > 40) THEN grade = pass
R3: IF (score1 > 20 ∧ score2 > 20) THEN grade = possibly pass

Notice that with the given information there is no matching rules from the negative class and R2 can be applied to answer this query.

Answer:

IF score2 > 40 THEN grade = pass.
IF score2 > 20 THEN grade = possibly pass.

Example 2. From the response of example 1, suppose the user wants to know further that based on the information of her first score, could her second score be predicted.
Method:

(1) The query asks for the value of score2, given the value of score1=55. Thus, a decision attribute is score2 and a decision table is as shown in table 2.

Table 2. A d ecision table with respect to query 2

	Conditions			*Decision*
	age	**score1**	**grade**	**score2**
s1	19	0-20	fail	0-20
s2	19	0-20	fail	21-40
s3	20	0-20	fail	41-60
s4	20	0-20	fail	41-60
s5	19	0-20	pass	81-100
s6	19	41-60	pass	41-60
s7	19	21-40	pass	61-80
s8	20	21-40	pass	21-40
s9	20	21-40	fail	21-40

(2) There is no reduct. So, all conditional attributes are used in the approximation of *bX*, *BX*, and *BN* regions. The decision objectives (*X*) are five sets of students whose score2 values are in the range 0-20, 21-40, 41-60, 61-80, and 81-100, respectively. From the approximation, these rules are induced:

Certain rules: IF (age=19 ∧ grade=pass ∧ score1 =0-20) THEN score2 = 81-100
IF (age=19 ∧ grade=pass ∧ score1=21-20) THEN score2 = 61-80
IF (age=20 ∧ grade=fail ∧ score1 = 0-20) THEN score2 = 41-60
IF (age=19 ∧ grade=pass ∧ score1 = 0-20) THEN score2 = 41-60
IF (age=20 ∧ score1 = 21-40) THEN score2 = 21-40

Possible rules: IF (age=19∧score1=20∧grade=fail) THEN score2 = 0-20 ∨ 21-40

(3) Generalized rules are as follow.

R1: IF (score1 = 0-20) THEN score2 = 81-100
R2: IF (score1 = 21-40) THEN score2 = 61-80
R3: IF (score1 = 0-20 ∨ 41-60) THEN score2 = 41-60
R4: IF (age=20 ∧ score1 = 21-40) THEN score2 = 21-40
R5: IF (age=19 ∧ score1 = 20 ∧ grade=fail) THEN possibly score2 = 0-40

Answer:

IF score1 = 55 THEN score2 = 41-60.

4 Query Optimization in Inductive Databases

With the mechanism to induce semantic rules, we can then integrate the knowledge to rewrite and optimize queries in the framework of relational inductive database. We first explain the steps in query transformation, then show the results of our experimentation.

4.1 A Method of Query Rewriting and Optimization

We study intelligent query answering in inductive databases in a simplified framework of relational databases. We illustrate our idea through examples on a business database containing eight relations: customers, orders, products, categories, order_details, suppliers, shippers, and employees. To keep this section concise, we will consider only customers relation with schema as follows.

customers (customerID, name, address, city, state, postalcode, country, phone, fax, birthdate, marital, gender, education, member_card, total_children, occupation, houseowner, num_car)

Our objective is to turn this simple database $\mathcal{R}$ into an inductive database $\mathcal{I} = (\mathcal{R}, \mathcal{P})$ by inducing a set of patterns $\mathcal{P}$ from the base tables. The processes of inductive database creation (the $\mathcal{P}$ part of the database $\mathcal{I}$) and query rewriting composed of the following steps.

Step 1: Preprocess the base table by removing irrelevant attributes, i.e. those without inherent patterns such as phone number, customerID. After the attribute elimination step, the customers table contains information as shown in figure 5.

City	Marital	Gender	Education	Member_card	Total_children	Occupation	House owner	Num_car
los angeles	s	m	bachelor	golden	0	management	yes	1
san diego	s	m	partial college	normal	0	skilled manual	no	1
nation city	m	f	bachelor	silver	3	skilled manual	yes	2
santa cruz	s	m	bachelor	silver	0	skilled manual	no	0

Fig. 5. Some examples from a set of customer instances

Step 2: Perform a semantic rule induction using the algorithm explained in figure 4. Suppose the query asks about *marital*, *gender*, *total_children*, and *houseowner* attributes, some of the induced patterns are shown in figure 6.
Step 3: Transform the induced rules into a tabular form (as shown in figure 7).
Step 4: Evaluate user's query for the possibility of null answer set detection. If the query contains unsatisfiable constraints that can be detected in an early stage using the induced patterns, then the subsequent processing is unnecessary. For example, given the query: *SELECT * FROM customers*

WHERE address = 'Bangkok' AND member_card = 'silver';

and the induced pattern: *IF address = 'Bangkok' THEN member_card = 'gold'*,
then the answer 'No' can be returned instantly.

IF gender = m THEN *marital = s*
IF total_children = 0 THEN
marital = s
IF total_children = 0 THEN
gender = m
IF gender = m THEN
total_children = 0
IF marital = m THEN
houseowner = yes

Fig. 6. The patterns of customers data represented as semantic rules

tablename_1	column_1	value_1	tablename_2	column_2	value_2
customers	gender	m	customers	marital	s
customers	total_child	0	customers	marital	s
customers	total_child	0	customers	gender	m
customers	gender	m	customers	total_child	0
customers	total_child	0	customers	marital	s

Fig. 7. The pattern table with some sample values

Step 5: Rewrite query using the induced patterns. The conjunctive conditions C1∧C2∧C3∧... in the where clause of the SQL query are matched against the patterns in an iterative manner. In the first iteration C1 is matched against the antecedent part of the patterns to search for the one with the consequent part that is unifiable with either C2 or the rest of the conditions. If this is the case, the unified condition is considered redundant and thus, can be removed. The subsequent iterations are performed on the remaining conditions. Consider the following query *Q*, and the induced patterns *P*.

Q: SELECT * FROM customers
WHERE total_children = '0' AND marital = 's' AND gender = 'm';

P: *IF total_childern = 0 THEN marital = s* (rule 1)
IF gender = m THEN total_childern = 0 (rule 2)

First iteration: the antecedent and consequent parts of rule 1 can match with the first and second conditions of the query. These two conditions are redundant. Thus, the where clause can be simplified to: *WHERE total_children = '0' AND gender = 'm';*

Second iteration: rule 2 states the fact regarding the association between gender = m and total_children = 0. Therefore, it can be applied to the query Q which can be finally rewritten as

Q': SELECT * FROM customers
WHERE gender = 'm';

4.2 Experimentation and Results

The proposed technique of query answering and refinement has been tested on a customer database implemented on MS SQL Server 2000. The patterns are induced from the customers table and four different queries pertaining to the customers have been tested on the database. We observe the returned answer set (number of tuples) as well as the query response time. We perform the experiments on the PC with CPU speed 3.2 GHz, 512 MB main memory and 80 GB HD. The query evaluation results are reported (in figure 8) comparatively between the original query processing and the answering from the query rewritten using the induced patterns.

Query Q1 illustrate the case of unsatisfiable query in which the condition of the query is found conflicting with the existing data content. The query asks for female customers who live in santa cruz. But the induced pattern states that there is no such

Query	Original query form	Pattern applied	Transformed query
Q1	SELECT * FROM customers WHERE city = 'santa cruz' AND gender = 'm' AND marital = 'm';	*IF city = santa cruz* *THEN gender = m* $\wedge$ *marital=s* *(9 patterns are induced)*	None: detection of unsatisfiable condition
Q2	SELECT * FROM customers WHERE city = 'los angeles' AND houseowner = 'yes';	*IF city = los angeles* *THEN houseowner = yes* *(2 patterns are induced)*	SELECT * FROM customers WHERE city = 'los angeles';
Q3	SELECT * FROM customers WHERE gender = 'm' AND marital = 's' AND total_children = '0';	*IF gender = m* *THEN marital= s* $\wedge$ *total_children = 0* *(4 patterns are induced)*	SELECT * FROM customers WHERE gender = 'm';
Q4	SELECT * FROM customers WHERE city = 'santa cruz' AND gender = 'm' AND member_card = 'bronze';	*IF city = santa cruz* *THEN gender = m* *(7 patterns are induced)*	SELECT * FROM customers WHERE gender = 'm' AND member_card = 'bronze';

Size of answer sets (number of tuples) and response time (millisecond):

	Q1		Q2		Q3		Q4	
	size	time	size	time	size	time	size	time
Original query	0	104	27,660	1406	71,916	4665	16,596	1185
Transformed query	0	0	27,660	1268	71,916	3489	16,596	1074
Gain		100%		9.8%		25.2%		9.4%

Fig. 8. Experimental results of asking four queries on a customer database

customers; most customers in santa cruz are male. Therefore, this query can be answered instantly (i.e., the response time is 0). Queries Q2, Q3, and Q4 are the examples of queries with redundant predicates. Once redundancy has been removed, the query response time can be reduced.

5 Conclusions

Our query answering scheme presented in this paper is based on the setting of inductive databases. An inductive database is the concept proposed as the next generation of database systems. Within the framework of an inductive database system, data and patterns which are discovered from data are stored together as database objects. In such tightly coupling architecture patterns are considered first-class objects in that they can be created, accessed, and updated in the same manner as persistent data. We present the framework and techniques of query rewriting and answering that use stored patterns as semantic knowledge. Our knowledge induction process is based on the rough set theory. We propose the algorithm to induce rough and precise semantic rules. We limit the number of discovered rules by inducing only rules that are relevant to user's need. Relevancy is guided by query predicates. The intuitive idea of our knowledge induction algorithm is illustrated trough running examples.

In the query optimization process, we take into account two major techniques of transformations: semantically redundant predicate elimination and detection of unsatisfiable conditions, i.e. conditions that never been true. We plan to extend our work on additional rewriting techniques and experiments with different kinds of queries such as range queries, top-k queries. The test on effectiveness with real-world large database is also our future research.

Acknowledgements

This research has been supported by grants from the National Research Council. The second author is supported by the grant from Thailand Research Fund (TRF – grant number RMU5080026). The Data Engineering and Knowledge Discovery Research Unit is fully supported by the research grants from Suranaree University of Technology.

References

1. Agrawal, R., Imielinski, T., Swami, A.: Mining association rules between sets of items in large databases. In: Proc. ACM SIGMOD, pp. 207–216. ACM Press, New York (1993)
2. Agrawal, R., Shim, K.: Developing tightly-coupled data mining applications on a relational database system. In: Proc. KDD, pp. 287–290 (1996)
3. Bonchi, F.: Frequent pattern queries: Language and optimizations. Ph.D. Thesis, Computer Science Department, University of Pisa, Italy (2003)
4. Boulicaut, J.-F., Klemettinen, M., Mannila, H.: Querying inductive databases: A case study on the MINE RULE operator. In: Żytkow, J.M. (ed.) PKDD 1998. LNCS, vol. 1510, pp. 194–202. Springer, Heidelberg (1998)
5. Boulicaut, J.-F., Klemettinen, M., Mannila, H.: Modeling KDD processes within the inductive database framework. In: Mohania, M.K., Tjoa, A.M. (eds.) DaWaK 1999. LNCS, vol. 1676, pp. 293–302. Springer, Heidelberg (1999)
6. Charkravarthy, U.S., Grant, J., Minker, J.: Logic-based approach to semantic query optimization. ACM Transactions on Database Systems 15(2), 162–207 (1990)
7. De Raedt, L.: A perspective on inductive databases. SIGKDD Explorations 4(2), 69–77 (2002)
8. De Raedt, L., Jaeger, M., Lee, S., Mannila, H.: A theory of inductive query answering. In: Proc. IEEE ICDM, pp. 123–130. IEEE Computer Society Press, Los Alamitos (2002)
9. Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P., Uthurusamy, R.: Advances in Knowledge Discovery and Data Mining. AAAI Press (1996)
10. Hammer, M.M., Zdonik, S.B.: Knowledge base query processing. In: Proc. VLDB, pp. 137–147 (1980)
11. Han, J., Fu, Y., Wang, W., Koperski, K., Zaiane, O.: DMQL: A data mining query language for relational databases. In: Proc. ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery, pp. 27–34. ACM Press, New York (1996)
12. Han, J., Huang, Y., Cercone, N., Fu, Y.: Intelligent query answering by knowledge discovery techniques. IEEE Trans. on Knowledge and Data Engineering 8(3), 373–390 (1996)
13. Imielinski, T., Mannila, H.: A database perspective on knowledge discovery. Communications of the ACM 39(11), 58–64 (1996)

14. Imielinski, T., Virmani, A.: MSQL: A query language for database mining. Data Mining and Knowledge Discovery 2(4), 373–408 (1999)
15. King, J.: QUIST: A system for semantic query optimization in relational databases. In: Proc. VLDB, pp. 510–517 (1981)
16. Komorowski, J., Polkowski, L., Skowron, A.: Rough sets: A tutorial. In: Rough Fuzzy Hybridization: A New Trend in Decision-Making, pp. 3–98. Springer, Heidelberg (1999)
17. Lin, T., Cercone, N., Hu, X., Han, J.: Intelligent query answering based on neighborhood systems and data mining techniques. In: Proc. IEEE IDEAS, pp. 91–96. IEEE Computer Society Press, Los Alamitos (2004)
18. Mannila, H.: Inductive databases and condensed representations for data mining. In: Proc. Int. Logic Programming Symp., pp. 21–30 (1997)
19. Meo, R.: Inductive databases: Towards a new generation of databases for knowledge discovery. In: Proc. DEXA Workshop, pp. 1003–1007 (2005)
20. Meo, R., Psaila, G., Ceri, S.: A tightly-coupled architecture for data mining. In: Proc. IEEE ICDE, pp. 316–323. IEEE Computer Society Press, Los Alamitos (1998)
21. Necib, C., Freytag, J.: Semantic query transformation using ontologies. In: Proc. IEEE IDEAS, pp. 187–199. IEEE Computer Society Press, Los Alamitos (2005)
22. Pawlak, Z.: Rough sets. Int. Jour. Information and Computer Science 11(5), 341–356 (1982)
23. Sarawagi, S., Thomas, S., Agrawal, R.: Integrating association rule mining with relational database systems: Alternatives and implications. In: Proc. ACM SIGMOD, pp. 343–354. ACM Press, New York (1998)
24. Siegel, M., Sciore, E., Salveter, S.: A method for automatic rule derivation to support semantic query optimization. ACM Trans. on Database Systems 17(4), 563–600 (1992)
25. Sun, J., Kerdprasop, N., Kerdprasop, K.: Relevant rule discovery by language bias for semantic query optimization. Jour. Comp. Science and Info. Management 2(2), 53–63 (1999)

Query Answering in Relational Inductive Databases

Kittisak Kerdprasop, Nittaya Kerdprasop and Apichai Ritthongchailert
Data Engineering and Knowledge Discovery Research Unit
School of Computer Engineering, Suranaree University of Technology, Thailand
kerdpras@sut.ac.th, nittaya@sut.ac.th, apichai_ri@hotmail.com

Abstract

Inductive databases can be viewed as a natural extension of traditional databases to contain not only persistent data but also the generalization of stored data, which are called patterns. The idea of inductive databases has been proposed originally as a support system for the knowledge discovery or data mining process. Many SQL-like languages have been designed and implemented to include mining operators in the SQL primitives. We percept the concept of inductive databases in a different angle. In stead of designing yet another inductive database system, we are looking for the deployment of an existing inductive query language and environment to support the database tasks. We focus on the task of query answering which has a high potential of being a beneficiary of the stored patterns in inductive databases. Our experimental results of query rewriting technique using induced patterns as a semantic knowledge confirm this advantage.

1. Introduction

Knowledge discovery in databases (KDD), or data mining, has emerged as a new multi-disciplinary research area in the 1990's [9]. Since then it has been realized that the current database system should be extended or re-designed to support the KDD process. Imielinski and Mannila [11] have argued that existing KDD techniques are simply file mining because the inductive learning tools are built on top of the databases assuming a loose coupling between the two components. In order to gain the full power of KDD as a database mining process, the mining engine has to be tightly coupled with the database system. In recent years, this idea has been realized and several research work along this line have been developed [2, 17, 18, 21, 22]. The tightly integration of databases with data mining gives rise to the new concept of inductive databases [3, 6, 7, 15]. Inductive databases are defined [4, 6] as databases that contain not only data, but also patterns which are generalized information induced from data. By providing this tightly integration framework of data management system and pattern discovery engine, users can access patterns in the same manner as querying data. To achieve this aim a number of SQL-based inductive query languages, such as MINE RULE [16], MSQL [12], DMQL [10], SQL Server 2005 [19], have been proposed and implemented. Most of these languages are an SQL extension with some primitives to support the data mining task, that is, users can pose queries to induce, access and update patterns.

We propose that besides the front-end functionalities the induced patterns should also be useful in the back-end part of query answering. The induced patterns are viewed as a repository of semantic knowledge which has the high potential to support the process of query rewriting and optimization. It is thus the purpose of this paper to illustrate the use of induced knowledge in the query answering process within the framework of inductive database systems. Unlike previous work on inductive databases that express queries using an object query language [20] or a logic-based language [4, 5, 8], we formalize our idea using a structured query language (SQL) of a typical relational database system.

The paper is organized as follows. Section 2 reviews the concept of inductive databases using the framework of relational databases. Section 3 proposes our idea of supporting query answering with the induced patterns. Section 4 shows the running examples and the experimental results. Section 5 concludes the paper.

2. Inductive databases

Inductive databases can be viewed as an extension of the traditional database systems in that the databases do not only store data, but they also contain patterns of those data. Mannila [14] formalized a framework of inductive database $\mathcal{I}$ as a pair ($\mathcal{R}$, $\mathcal{P}$) where $\mathcal{R}$ is a database relation and $\mathcal{P}$ is a nested relation of the form

1529-4188/07 $25.00 © 2007 IEEE
DOI 10.1109/DEXA.2007.61

($\mathcal{Q}_\mathcal{R}$, e) in which $\mathcal{Q}_\mathcal{R}$ is a set of patterns obtained from querying the base data and e is the evaluation function measuring some metrics over the patterns.

As an example, consider the database (adapted from [4]) consisting of one relation with the schema $\mathcal{R}$ = {(X, Y, Z)}; the values of the three attributes are in the domain {0, 1}. The induced patterns $\mathcal{P}$ are a set of association rules [1] represented as an implication LHS ⇒ RHS; therefore, $\mathcal{Q}_\mathcal{R}$ = {LHS ⇒ RHS | LHS, RHS ⊆ $\mathcal{R}$} and the metrics are support (percentage of tuples in $\mathcal{R}$ that contain both LHS and RHS, LHS ∪ RHS) and confidence (support(LHS ∪ RHS) / support(LHS)). An inductive database $\mathcal{I}$ = ($\mathcal{R}$, $\mathcal{P}$) is shown in figure 1 in a tabular format for the $\mathcal{R}$ and $\mathcal{P}$ components.

$\mathcal{R}$

X	Y	Z
1	0	0
1	1	1
1	0	1
0	1	1

$\mathcal{P}$

pattern	support	confidence
X⇒Y	0.25	0.33
X⇒Z	0.50	0.66
Y⇒X	0.25	0.50
Y⇒Z	0.50	1.00
Z⇒X	0.50	0.66
Z⇒Y	0.50	0.66
XY⇒Z	0.25	1.00
XZ⇒Y	0.25	0.50
YZ⇒X	0.25	0.50

Figure 1. An example of inductive database instance.

Given the framework of an inductive database $\mathcal{I}$, users can query both the stored data (the part of $\mathcal{I}.\mathcal{R}$ in figure 1) as well as the set of patterns (the $\mathcal{I}.\mathcal{P}$ part). Formalization of inductive queries to perform data mining tasks has been studied by several research groups [6, 8, 13]. We are, however, interested in the concept of inductive databases from a different perspective. Instead of using a sequence of queries and operations to create the induced patterns such as association rules, we shift our focus towards the deployment of the stored information (i.e., patterns and data) to support the process of query answering.

3. Query answering in inductive databases

In this paper, we study inductive databases in a simplified framework of relational databases that are extended with SQL primitives to perform some data mining tasks such as association rule mining. We illustrate our idea through examples. Suppose we have a business database with schema as follows (the underlined attributes are primary keys and the starred attributes are foreign keys).

product (<u>productID</u>, name, supplierID*, categoryID*, QuantityPerUnit, unitPrice, unitInStock, unitOnOrder, reorderLevel)
suppliers (<u>supplierID</u>, name, contactName, address, city, state_province, postalcode, country, phone, fax, homepage, type)
orders (<u>orderID</u>, customerID*, employeeID*, orderdate, requiredate, productID*, quantity, discount)
customers (<u>customerID</u>, name, address, city, state_province, postalcode, country, phone, fax, birthdate, marital, year_income, gender, education, member_card, total_children, date_open, occupation, houseowner, num_car)

Our objective is to turn this simple database $\mathcal{R}$ = {product, suppliers, orders, customers} into an inductive database $\mathcal{I}$ = ($\mathcal{R}$, $\mathcal{P}$) by inducing a set of patterns $\mathcal{P}$ from the base tables. Each table is capable of being a candidate in the induction process, but we consider only the customers table to keep this paper short. The processes of inductive database creation (the $\mathcal{P}$ part of the database $\mathcal{I}$) and querying composed of the following steps.

Step 1: Preprocess the target table by removing irrelevant attributes, i.e. those without inherent patterns such as phone number, customerID. After the attribute elimination step, the customers table contains information as shown in figure 2.

City	Marital	Gender	Education	Member_card	Total_children	Occupation	House owner	Num_car
los angeles	s	m	bachelor	golden	0	management	yes	1
san diego	s	m	partial college	normal	0	skilled manual	no	1
nation city	m	f	bachelor	silver	3	skilled manual	yes	2
santa cruz	s	m	bachelor	silver	0	skilled manual	no	0
san diego	s	m	partial college	normal	0	skilled manual	no	1
san diego	s	m	partial college	normal	0	manual	no	0
san diego	m	f	bachelor	bronze	1	skilled manual	yes	1
san diego	m	f	partial college	normal	1	manual	yes	0
los angeles	s	m	graduate	golden	0	management	yes	3

Figure 2. A set of customer instances to be used as examples in the association mining process.

Step 2: Perform association rule induction using Apriori algorithm [1] with minimum confidence = 1 and minimum support = 0. Due to the fact that we want to induce all relations that are consistent among the existing customers data, the confidence metric has to be 1.00 or 100% correct. We consider the induced patterns interesting if they are true over the majority of our customer population. Thus, we rank the patterns in descending order of support values. The best ten patterns are shown in figure 3.

gender = m	⇒	*marital = s*
total_children = 0	⇒	*marital = s*
total_children = 0	⇒	*gender = m*
gender = m	⇒	*total_children = 0*
houseowner = no	⇒	*marital = s*
member_card = bronze	⇒	*occupation= skilled_manual*
marital = m	⇒	*gender = f*
marital = m	⇒	*houseowner = yes*
city = los_angeles	⇒	*houseowner = yes*
city = nation_city	⇒	*occupation= skilled_manual*

Figure 3. The best ten patterns of customers data represented as association rules.

Step 3: Transform the induced association rules into a tabular form. The table containing induced patterns is shown in figure 4. We use a fixed format with base relation name in the first column, the antecedent attribute and its value in the second and third columns, respectively. The last three columns contain the relation name, attribute's name and value of the consequent part of the association rule.

tablename_1	column_1	value_1	tablename_2	column_2	value_2
customers	gender	m	customers	marital	s
customers	total_child	0	customers	marital	s
customers	total_child	0	customers	gender	m
customers	gender	m	customers	total_child	0
customers	total_child	0	customers	marital	s

Figure 4. The pattern table with some sample values.

Step 4: Evaluate user's query for the possibility of null answer set detection. If the query contains an unsatisfiable constraint that can be detected in an early stage against the induced patterns, the subsequent query processing is unnecessary. For example, given the query:

SELECT * FROM customers
WHERE address = 'Bangkok'
AND member_card = 'silver';

and the induced pattern:

$$address = \text{'Bangkok'} \Rightarrow member\_card = \text{'gold'}$$

The answer 'No' can be returned instantly.

Step 5: Rewrite query with the induce patterns. The conjunctive conditions $C1 \wedge C2 \wedge C3 \wedge \ldots$ in the where clause of the SQL query are matched against the patterns in an iterative manner. In the first iteration C1 is matched against the antecedent part of the patterns to search for the one with the consequent part that is unifiable with either C2 or the rest of the conditions. If this is the case, the unified condition is considered redundant and thus, can be removed. The subsequent iterations are performed on the remaining conditions. As an example, consider the following query Q, and the set of induced patterns P.

Q: SELECT * FROM customers
WHERE total_children = '0'
AND marital = 's' AND gender = 'm';

P: $total\_childern = 0 \Rightarrow marital = s$ (rule 1)
$gender = m \Rightarrow marital = s$ (rule 2)
$gender = m \Rightarrow total\_childern = 0$ (rule 3)

First iteration: the antecedent and consequent parts of rule 1 can match with the first and second conditions of the query. These two conditions are redundant. Thus, the where clause can be simplified to

WHERE total_children = '0' AND gender = 'm';

Second iteration: rule 3 states the fact regarding the association between gender = m and total_children = 0. Therefore, it can be applied to the query Q which can be finally rewritten as

Q': SELECT * FROM customers
WHERE gender = 'm';

These steps of query rewriting and answering can be summarized as a flow chart as shown in figure 5.

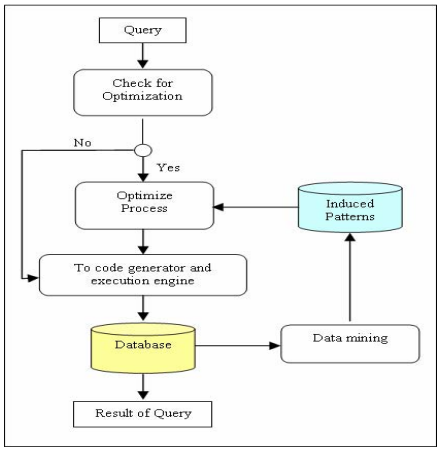


Figure 5. The conceptual diagram of query answering within the framework of inductive databases.

4. Running examples and experimentation

The proposed technique of query answering and refinement has been tested on a business database containing eight base tables implemented on MS SQL Server 2000. The patterns are induced from the

customers table and five different queries pertaining to the customers have been tested on the database. The test are repeated five times on each query. We observe the returned answer set as well as the response time. We do the experiments on the personal computer with CPU speed 3.2 GHz, 512 MB main memory and hard disk 80 GB. The query evaluation results are reported comparatively between the original query processing and the answering from the query rewritten using the induced patterns.

Q1: SELECT * FROM customers
WHERE city = 'santa cruz'
AND gender = 'f';

Pattern: *city = santa cruz ⇒ gender = m*

Q1': None: detection of unsatisfiable condition

Answer: null

Time(ms):

query	Test#1	Test#2	Test#3	Test#4	Test#5
Q1	50	50	51	50	51
Q1'	0	0	0	0	0

Q2: SELECT * FROM customers
WHERE city = 'santa cruz'
AND gender = 'm'
AND marital = 'm';

Pattern: *city = santa cruz ⇒ gender = m, marital=s*

Q2': None: detection of unsatisfiable condition

Answer: null

Time(ms):

query	Test#1	Test#2	Test#3	Test#4	Test#5
Q2	109	103	104	101	104
Q2'	0	0	0	0	0

Q3: SELECT * FROM customers
WHERE city = 'los angeles'
AND houseowner = 'yes';

Pattern: *city = los angeles ⇒ houseowner = yes*

Q3': SELECT * FROM customers
WHERE city = 'los angeles';

Answer: Q3 = 27,660 tuples; Q3'= 27,660 tuples

Time(ms):

query	Test#1	Test#2	Test#3	Test#4	Test#5
Q3	1336	1442	1406	1429	1422
Q3'	1126	1303	1268	1273	1230

Q4: SELECT * FROM customers
WHERE gender = 'm'
AND marital = 's'
AND total_children = '0';

Pattern: *gender = m ⇒ marital= s, total_children = 0*

Q4': SELECT * FROM customers
WHERE gender = 'm';

Answer: Q4 = 71,916 tuples; Q4'= 71,916 tuples

Time(ms):

query	Test#1	Test#2	Test#3	Test#4	Test#5
Q4	4559	4043	4665	4043	4440
Q4'	3377	3717	3489	3690	3404

Q5: SELECT * FROM customers
WHERE city = 'santa cruz'
AND gender = 'm'
AND member_card = 'bronze';

Pattern: *city = santa cruz ⇒ gender = m*

Q5': SELECT * FROM customers
WHERE gender = 'm'
AND member_card = 'bronze';

Answer: Q5 = 16,596 tuples; Q5'= 16,596 tuples

Time(ms):

query	Test#1	Test#2	Test#3	Test#4	Test#5
Q5	1313	1908	1185	1749	1375
Q5'	936	1274	1074	962	951

Queries Q1 and Q2 illustrate the case of unsatisfiable queries in which the conditions of the queries conflict with the existing data content. For instance, Q1 asks for female customers who live in santa cruz. But the induced pattern states that there is no such customers because according to our database most customers in santa cruz are male. Therefore, this query can be answered instantly (i.e., the response time is 0) without wasting the time consulting the stored database. Queries Q3, Q4, and Q5 are the examples of queries with redundant predicates. Once redundancy has been removed, the query response time can be speed up.

5. Conclusions

Inductive databases are the concept proposed originally by Imielinski and Mannila [11] in 1996 as the next generation of database systems. Within the framework of an inductive database system, data and patterns which are discovered from data are stored

together as database objects. In such tightly coupling architecture patterns are considered first-class objects in that they can be created, accessed, and updated in the same manner as persistent data.

In this paper, we present the techniques of query rewriting and answering that use stored patterns as semantic knowledge to facilitate the query evaluation process. We take into account two major techniques of semantically redundant predicate elimination and detection of unsatisfiable conditions, i.e. conditions that never been true. We plan to work on additional rewriting techniques and experiments with different kinds of queries such as range queries, top-k queries in our future research.

Acknowledgements

This research has been partially funded by grants from the National Research Council. The first author has been supported by grant from the Thailand Research Fund (TRF, grant number RMU5080026). The Data Engineering and Knowledge Discovery Research Unit is fully supported by the research grants from Suranaree University of Technology.

References

[1] R. Agrawal, T. Imielinski, and A. Swami, “Mining association rules between sets of items in large databases”, *Proc. ACM SIGMOD*, pp.207-216, 1993.

[2] R. Agrawal and K. Shim, "Developing tightly-coupled data mining applications on a relational database system" *Proc. KDD*, pp.287-290, 1996.

[3] F. Bergadano, “Inductive database relations”, *IEEE Trans. Knowledge and Data Engineering*, vol.5, no.6, pp.969-972, 1993.

[4] F. Bonchi, “Frequent pattern queries: Language and optimizations”, *Ph.D. Thesis*, Computer Science Department, University of Pisa, Italy, 2003.

[5] J.-F. Boulicaut, M. Klemettinen, and H. Mannila, “Querying inductive databases: A case study on the MINE RULE operator”, *Proc. PKDD*, pp.194-202, 1998.

[6] J.-F. Boulicaut, M. Klemettinen, and H. Mannila, “Modeling KDD processes within the inductive database framework”, *Proc. DaWAK*, pp.293-302, 1999.

[7] L. De Raedt, “A perspective on inductive databases”, *ACM SIGKDD Explorations*, vol.4. no.2, pp.69-77, 2002.

[8] L. De Raedt, M. Jaeger, S. Lee, and H. Mannila, "A theory of inductive qury answering", *Proc. IEEE ICDM*, pp.123-130, 2002.

[9] U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, AAAI Press, 1996.

[10] J. Han, Y. Fu, K. Koperski, W. Wang, and O. Zaiane, "DMQL: A data mining query language for relational databases", *Proc. ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery*, 1996.

[11] T. Imielinski and H. Mannila, “A database perspective on knowledge discovery”, *Communications of the ACM*, vol.39, no.11, pp.58-64,1996.

[12] T. Imielinski and A. Virmani, "MSQL: A query language for database mining", *Data Mining and Knowledge Discovery*, vol.2, no.4, pp.373-408, 1999.

[13] S. Lee and L. De Raedt, "An algebra for inductive query evaluation", *Proc. IEEE ICDM*, pp.147-154, 2003.

[14] H. Mannila, "Inductive databases and condensed representations for data mining", *Proc. Int. Logic Programming Symp.*, pp.21-30, 1997.

[15] R. Meo, "Inductive databases: Towards a new generation of databases for knowledge discovery", *Proc. DEXA*, pp.1003-1007, 2005.

[16] R. Meo, G. Psaila, and S. Ceri, "A new SQL-like operator for mining association rules", *Proc. VLDB*, pp.122-133, 1996.

[17] R. Meo, G. Psaila, and S. Ceri, "A tightly-coupled architecture for data mining", *Proc. IEEE ICDE*, pp.316-323, 1998.

[18] S. Sarawagi, S. Thomas, and R. Agrawal, "Integrating association rule mining with relational database systems: Alternatives and implications", *Proc. ACM SIGMOD*, pp.343-354, 1998.

[19] Z. Tang and J. Maclennan, *Data Mining with SQL Server 2005*, John Wiley & Sons, 2005.

[20] A. Trigoni and K. Moody, "Using association rules to add or eliminate query constraints automatically", *Proc. Int. Conf. Scientific Database Management*, pp.124-133, 2001.

[21] D. Tsur and S. Nestorov, "Integrating data mining with relational DBMS: A tightly coupled approach", *Proc. Workshop on Next Generation Information Technologies and Systems*, pp.295-311, 1999.

[22] Y. Wang, C. Tao, Y. Zhao, and Y. Yang, "Crd: A new data mining method in deductive databases", *Proc. Workshop on Deductive Databases and Logic Programming*, 1997.

การปรับปรุงประสิทธิภาพข้อคำถามเชิงความหมายด้วยการอุปนัยกฎความสัมพันธ์

SEMANTIC QUERY OPTIMIZATION WITH ASSOCIATION RULE INDUCTION

อภิชัย ฤทธิ์ธงชัยเลิศ, นิตยา เกิดประสพ และ กิตติศักดิ์ เกิดประสพ

Apichai Rintthongchailert, Nittaya Kerdprasop and Kittisak Kerdprasop

Data Engineering and Knowledge Discovery (DEKD) Research Unit, School of Computer Engiineering, Suranaree University of Technology, Muang, Nakhon Ratchasima, 30000.

บทคัดย่อ: การปรับปรุงประสิทธิภาพข้อคำถามเชิงความหมาย หมายถึง การนำข้อคำถามเดิมมาจัดรูปแบบใหม่ ให้มีรูปประโยคที่แตกต่างกับข้อคำถามเดิม แต่ยังคงให้ผลลัพธ์ที่เหมือนเดิม สิ่งที่แตกต่างกันของทั้งสองข้อคำถาม คือ เวลาที่ใช้ในการประมวลผล เพื่อตอบคำถามนั้นจะใช้เวลาน้อยลงกว่าเดิม ความสมบูรณ์ของการปรับปรุงข้อคำถามเชิงความหมายนี้ จะขึ้นอยู่กับเงื่อนไขหรือกฎข้อบังคับที่จะนำมาเพิ่มหรือลดตัวประโยคเงื่อนไขของข้อคำถามโดยทั่วไปแล้วกฎข้อบังคับที่นำมาใช้ในการปรับปรุงข้อคำถามเชิงความหมายนี้ จะได้มาจากผู้เขียนโปรแกรมฐานข้อมูล ซึ่งอาจจะไม่ครอบคลุมกับข้อมูลทั้งหมดที่มีอยู่ในฐานข้อมูล ดังนั้น ในงานวิจัยนี้จึงนำเอาเทคโนโลยีการขุดค้นความรู้จากฐานข้อมูล ซึ่งเป็นเทคโนโลยีที่เป็นที่รู้จักกันอย่างแพร่หลาย โดยนำมาเฉพาะส่วนของการค้นหากฎความสัมพันธ์ของข้อมูล มาประยุกต์ใช้เพื่อทำงานร่วมกับการปรับปรุงประสิทธิภาพข้อคำถามเชิงความหมาย เพื่อลดเวลาในการประมวลผลข้อคำถามนั้น

Abstract: Semantic query optimization is the process of transforming a given query into a semantically equivalent one that still returns the same answer for any database state satisfying query's constraints. The different of both queries is lower execution cost of the transformed one. The efficiently optimized query depends on semantic constraints or integrity constraints which are used to remove a useless condition in a query's where clause. Basically, integrity constraint is defined by user. It may not cover all data in the database. Therefore, this paper aims at presenting the utilization of a well known data mining technique, association mining, to assist the semantic query optimization process.

Introduction: The increase speed of query execution in the database management system is important for database development. One popular and well known method is query optimization. Some queries may contain confusing condition in where clause or condition may conflict with data in the database such that no answer the exists for that query. Then it is wasteful to execute such query. Therefore, it is our aim to adjust pattern of query before sending it to database management system for execution. The adjustment or transformation is based on semantic constraint induced with a data mining technique.

Methodology: We can find semantic constraint by take a dataset on the database into data mining process. In data mining process, to find association of data. We use apriori algorithm [1,2] in the data mining process. So the results of mining give association rule in the form of IF – THEN rule (cause column to result column). But we adjust pattern of rule to become simple rule that the IF part contains one cause and the THEN part contains one result as follow:

If column_1 = 'value_1' then column_2 = 'value_2'

In applying algorithm, we must specify the minimum confidence value and the minimum support value. In this paper, we define minimum confidence to be 1 or 100% accurate for any induced rule to guarantee correctness in query answering and we define minimum support to be zero for finding all association among data in the database. In query optimization process, we get semantic constraint from database for comparing query condition in the where clause. In this process we divide query into two types [3]: query with redundant condition and query with conflict condition. For the first type of the given query, the condition that happens to be redundant will be removed before processing the query. Figure 1 shows an example of this case.

Original query:	select * from *table_name* where *column_1* = '*A*' and *column_2* = '*B*';
Association rule:	column_1 = 'A' ➔ column_2 = 'B'
Optimized query:	select * from *table_name* where *column* = '*A*';

Figure 1 Query with redundant condition

For the second case, query condition may conflict to the semantic constraint induced from the database. Then, the optimizer can produce immediate answer and save processing time of the database management system. Figure 2 show the example of conflicting case.

Original query:	select * from *table_name* where *column_1* = '*S*' and *column_2* = '*T*';
Association rule:	column_1 = 'S' ➔ column_2 = 'Q'
Optimized query:	*No execution.*

Figure 2 Query with conflicting condition

Results, Discussion and Conclusion: In our experiment, we use Microsoft SQL Server 2000 database, tested on Pentium IV 3.0 GHz with RAM 512 MB machine. The data sets used in our experiment are synthetic data and data taken from the UCI Repository(http://www.ics.uci.edu/~mlearn/MLRepository.html). We compared execution time of original query with the optimize query. Table 1 shows the execution time for the second type of query: query with conflicting condition. Once the conflict was detected, the answer no is given immediately. Therefore, execution time of optimized query is zero. Experimental result for the first type of query (that is, query with redundant condition) are shown in Figure 4.

Query	Query 1	Query 2	Query 3	Query 4	Query 5
Original query	109 ms	103 ms	104 ms	101 ms	104 ms
Optimize query	0 ms	0 ms	0 ms	0 ms	0 ms

Table 1 The result of execution time for queries with conflicting condition

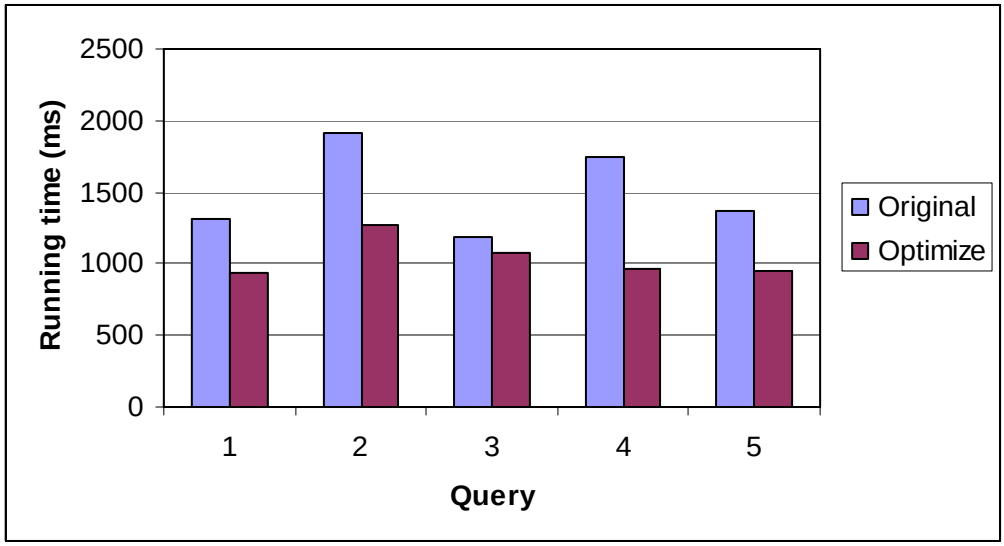


Figure 4 Response time of original queries compared to the optimized queries

This paper demonstrates that the efficiently optimized query depend on semantic constraints can be learned inductively under association data from a database and using data mining technique to fine data association. Experimental results show that there is significantly time different in query with conflict condition to the semantic constraint due to there is no execution query in database management system. Execution time in query with redundant condition depends on amount of data in the database.

References:

1. R. Argawal and R. Srikant, Fast algorithm for mining association rules. Proceedings of the 20th International Conference on Very Large Data Bases Conference, 1994.
2. R. Argawal, T. Imielinski and A. Swami, Mining association rules between set of items in large database. Proceedings of ACM SIGMOD International Conference on Management of Data. 1993.
3. Q. Cheng, J. Gryz, F. Koo, C. Lenung and L. Liu, Implementation of Two Semantic Query Optimization Techniques in DB2 Universal Database. Proceedings of the 25th International Conference on Very Large Data Bases VLDB '99, 1999

Keywords: Semantic query optimization, association rule induction.

Acknowledgements: This work was supported by grants from Nation Research Council of Thailand (NRCT). The authors are members of Data Engineering and Knowledge Discovery (DEKD) Research Unit, which is fully supported by Suranaree University of Technology.

การคัดเลือกวิวข้อมูลเพื่อแปลงรูปแบบข้อคำถาม

MATERIALIZED VIEW SELECTION FOR QUERY REWRITING

จักรพันธ์ มหาวันตัง, นิตยา เกิดประสพ และ กิตติศักดิ์ เกิดประสพ

Jackapan Mahavantang, Nittaya Kerdprasop and Kittisak Kerdprasop

School of Computer Engineering, Suranaree University of Technology, Muang District, Nakhon Ratchasima 30000.

E-mail: jackapan@yahoo.com

บทคัดย่อ: ฐานข้อมูลในปัจจุบันเต็มไปด้วยข้อมูลมากมายทั้งที่เกิดประโยชน์ และไม่เกิดประโยชน์ จึงเกิดการขุดค้นข้อมูลเพื่อหาความรู้จากข้อมูลเหล่านั้น อย่างไรก็ตาม ในการขุดค้นข้อมูลยังมีปัญหาเรื่องการประมวลผลข้อคำถามที่ต้องใช้เวลานาน จึงได้มีความพยายามที่จะเพิ่มประสิทธิภาพของการประมวลผลข้อคำถามด้วยวิธีการต่างๆ วิวข้อมูลได้ถูกนำมาใช้ในการเพิ่มประสิทธิภาพการประมวลผลข้อคำถาม โดยนักวิจัยได้พยายามหาวิธีใช้ประโยชน์จากวิวข้อมูลมาเป็นระยะเวลานาน แต่วิธีการต่างๆที่เสนอยังมีข้อจำกัดที่วิวข้อมูลจะต้องตรงพอดีกับเงื่อนไขในข้อคำถาม แนวทางการวิจัยของโครงการวิจัยนี้พยายามลดข้อจำกัดดังกล่าว โดยเสนอการสร้างวิวข้อมูลและพยายามหาเกณฑ์คัดเลือกวิวเพื่อพิจารณาวิวข้อมูลที่มีความใกล้เคียงกับข้อคำถามมากที่สุด เพื่อแปลงรูปแบบข้อคำถามให้สามารถประมวลผลได้เร็วที่สุด โดยยังคงความถูกต้องของผลลัพธ์ในการตอบข้อคำถาม

Abstract: Modern database contains a wealth of information waiting to be discovered and understood. However, finding and presenting this information in a timely fashion can be a major issue, especially when vast amount of data have to be searched. Materialized views help solve this problem. To realize this potential, the query optimizer should know how and when to exploit materialized views. This paper presents algorithm for determining whether part or all of a query can be estimated from materialized views and describes how it can be combined to rewrite query.

Introduction: Materialized views can provide massive improvements in query processing time, especially for aggregation queries over large data [2]. The materialized view should be thought of as a special kind of view, which physically exists inside in the database [3, 4]. We can improve query execution time by pre-computing expensive joins and aggregation operation prior to execution [6]. Then create a materialized view as a new physical table which consists of pre-computed data that are much smaller in size and able to answer the query rapidly [4]. Compared to the original source the need of physical space is very low, but the increased speed of the answer is substantial. For example, given a database containing a customer relation customer (C_ID, C_NAME, C_PHONE, PROVINCE_ID) and a province relation province (PROVINCE_ID, PROVINCE_NAME). Let c_korat_mv be a materialized view that contains all customers who live in nakhonratchasima (PROVINCE_NAME = nakhonratchasima). Consider the query that asks for customer whose name is jirawan and live in nakhonratchasima. We present accessed data path on a base table, compare to accessed data path on materialized view as follows:

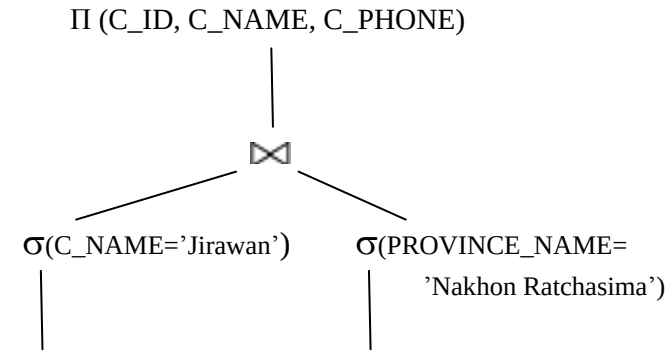


Figure 1 Access data path on base table**.**

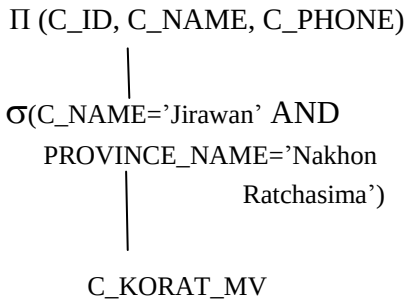


Figure 2 Access data path on materialized view.

In this example, execution time for accessing data on the materialized view is better than accessing data a base table. That means materialized views have been found to be very effective at speeding up query answering [5]. Essential for materialized view usage is view selection. The conceptual idea is selecting views that match the query's constraints as much as possible. According to our sample, a view to be selected is MV1. Therefore, we use this view first, thin apply view MV2.

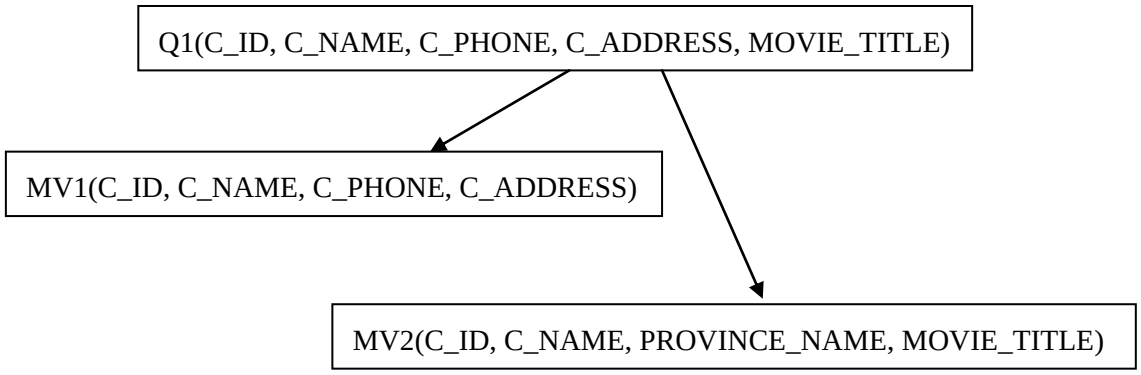


Figure 3 Materialized view selecting example.

Methodology: We use the basic architecture for the automated selection process as proposed in [1] and extend the algorithm for materialized view selection described step by step as follows:

Step1. Select the materialized view that correlated with the query. Materialized view which is correlated with the query can give the desired answer. For example, given query Q1 (C_ID, C_NAME, C_PHONE, C_ADDRESS, MOVIE_TITLE), there are chances that several materialized views can be applied. Assume MV1 (C_ID, C_NAME, C_PHONE, C_ADDRESS), MV2 (C_ID, C_NAME, PROVINCE_NAME, MOVIE_TITLE), MV3 (C_ID, C_NAME, C_PHONE, C_ADDRESS, MOVIE_TITLE) are set of views that can give the answer for Q1**.**

Step2. Select the materialized views which can give the best answer. From step 1 we will get the answers which correlated with query but there are the overlapping data in each answer then in this step we will select the materialized view that can give the best answer. We've get a set of data from this step such as set 1, the materialized view indicates ID code, phone number and address. Set 2, the materialized view indicates ID code and name of the movie and also set 3, the materialized view indicates ID code, phone number, address and name of the movie. The example showed that we don't need to materialized views all the data but we can get the best answer for all queries by selecting the materialized views which can give the best answer. In this step we took advantage of mvBestselect algorithm and Cost-based pruning of syntactically relevant materialized views algorithm [1] present as figure4.

```
M = {}          /* M is the set of materialized views that is useful for at
                        least one query in workload W */

For i = 1 to |W|
        Let Si = Set of materialized views proposed for Qi
        C = MvBestSelect(Qi,Si)
        M = M U C
End For

Return M
```

Figure 4 Cost-based pruning of syntactically relevant materialized views.

mvBestSelect is the algorithm that is use for selecting the materialized view that can give the best answer and used with Cost-based pruning of syntactically relevant materialized views algorithm.

```
Let Si = Set of materialized views proposed for Qi

If (|Si| > 1)
        For each v in Si
                Vm = MaxOfFequency(v)
        Return Vm
End If

Return Si
```

Figure 5 mvBestSelect algorithm.

Qi is the divided query and v is the member of set of the materialized view that can answer the query of Qi, Si is a set of the materialized view which can answer the query of Qi for example:

{MV1, MV2, MV3} is a set that can answer the query of $Q1_1$ (C_ID)

{MV1, MV2, MV3} is a set that can answer the query of $Q1_2$ (C_NAME)

{MV1, MV3} is a set that can answer the query of $Q1_3$ (C_PHONE)

{MV1, MV3} is a set that can answer the query of $Q1_4$ (C_ADDRESS)

{MV3} is a set that can answer the query of $Q1_5$ (MOVIE_TITLE)

Then, M is the materialized view that can give the best answer and M = {MV3}

Results, Discussion and Conclusion: The experiments were run on Pentium4 with CPU speed 3.2 GHz and 512 MB RAM. The databases used for our tests were stored on an internal 80GB hard drive.

Databases: The algorithms presented in this paper have been extensively tested on Oracle Database Sample Schemas 10g [6] (sales history (SH) schema) 0.893 GB database size and 128 MB Table spaces size. The result (as shows in figure 6) reveals that response time of new query is better than original one. This process decreases compute and joins data, and thus shows the better performance.

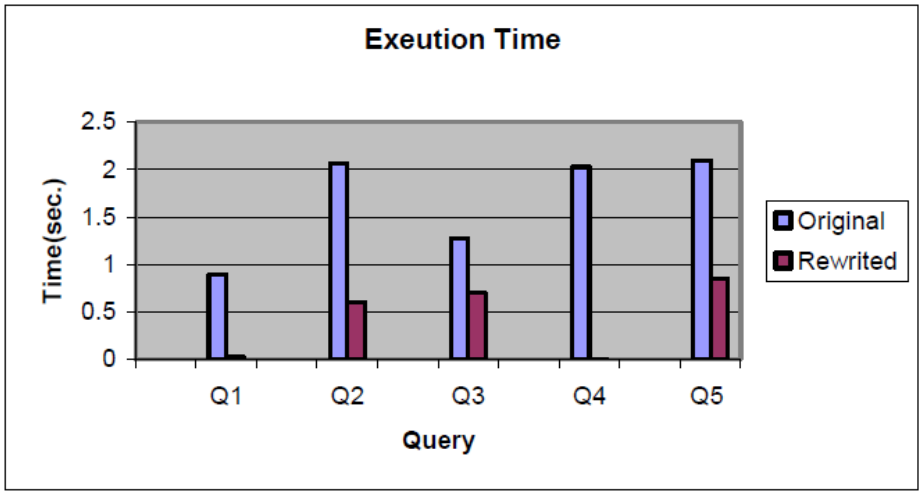


Figure 6 Execution time.

Materialized views can provide massive improvements in query processing time, especially for aggregation queries over large data. Essential for materialized view usage is views selection and physical design.

References:

1. Chaudhuri, S., Narasayya V., Agrawal S., Automated selection of materialized views and indexes for SQL databases, *Proc. 26th Int. Conf. Very Large Databases*, 496-505, 2000.
2. Goldstein, J. and Larson, P., Optimizing queries using materialized views: A practical, scalable polution, *Proc. ACM SIGMOD*, 331-342, 2001.
3. Gupta, A., and Mumick, I. S., Maintenance of materialized views: Problem, techniques, and application, in A. Gupta and I. S. Mumick, Eds., *materialized view*s, 145-158, 1998.
4. http://www.oracle.com/
5. Mistry, H., Prasan Roy, S., Ramamritham, S., Materialized view selection and maintenance using multi-query optimization, *Proc. SIGMOD*, 307-318, 2001.
6. Zhengxin, C., Intelligent data warehousing: From data preparation to data mining, CRC PRESS, 2001.

Keywords: Materialized view selection, query rewriting.

Acknowledgements: This work was supported by grant from National Research Council of Thailand (NRCT). The authors are member of Data Engineering and Knowledge Discovery (DEKD) Research Unit, which is fully supported by Suranaree University of Technology.

ภาคผนวก ข

รหัสต้นฉบับของโปรแกรมสังเคราะห์โมเดลข้อมูล

```
%%%  SQO final report file: asso_table.erl
%%%  input: ipum.NAMES, ipums-1999-1Krecords.bak
%%%  use file: mylib.erl
%%%  output:set.raw , allsortedrule.txt, factc.pl+rules.pl= rules_fact1.pl
-module(asso_table).
-import(lists,[sublist/2,seq/2,sum/1,flatten/1,split/2,nth/2,map/2,last/1]).
-import(io,[format/1,format/2]).
-import(ordsets,[to_list/1,from_list/1,is_subset/2,union/1]).
%%% c(mylib,[export_all]), c(asso_table,[export_all]) .
%%% asso_table:main() .
%%% erlang:spawn_opt(asso_table,main,[],[{min_heap_size,73662860}]).
%%%  70%
%%% erlang:spawn_opt(asso_table,main1,[],[{min_heap_size,73662860}]). % Creat rules with new process
%%% asso_table:main1().  % Creat rules without new process
main() ->
    NameList=mylib:read_file( "ipum.NAMES"," ,.\t:|" ),
    mylib:text_file(write,NameList,filetemp),
    [H|Tail]=NameList, [F,C|T]=lists:reverse(H),NewClass=[F|T], % make_nornal_class
    LL=[NewClass|Tail],
    [A1|A2]=map(fun([H|_])->H end, LL),AttrName=A2++[A1],
    [P1|P2]=lists:map(fun(L)->allPossibleAttr(L) end,LL),%swap class to the last
    PossibleValue=P2++[P1],
    mylib:text_file(write,PossibleValue,filetemp1),
    mylib:text_file(write,AttrName,filetemp2),AllInput=input(AttrName),
        %% write fact to file
    {_,IO}=file:open("factc.pl",[write]),
    lists:map(fun(EachR)->to_prolog(EachR,IO) end,AllInput),
    _=file:close(IO),
    DB=myToSet(AllInput),  MinSup=inputSup(AllInput), mylib:c(20,MinSup),
    mylib:text_file(write,AllInput,allinput),
    Items=my_flat(PossibleValue),
    AllL=apriori1(DB, Items,MinSup),
    format("~n/////////// END \\\\\\\\\\\\\~n").

my_flat([H|T]) -> H++my_flat(T);
my_flat([]) -> [].

to_prolog(Fact,IO) ->
     io:format(IO,"~n~p(",[rec]),
     print_fact(IO,Fact),io:format(IO,").% ",[]).

print_fact(IO,[H]) -> io:format(IO,"~p",[list_to_atom(H)]) ;
print_fact(IO,[H|T]) -> io:format(IO,"~p,",[list_to_atom(H)]),print_fact(IO,T).

apriori1(DB,Items,Min) -> %% findSup(Set,ListOfSet)
   mylib:text_file(append,myToList(DB),apriori1),mylib:text_file(append,Items,apriori1),
   C1=[{from_list([X]),findSup(from_list([X]),DB)}|| X<-Items ],
   CkPrint=[ {to_list(FS),Sup} || {FS,Sup}<-C1],
   L1=[{FS,Sup} || {FS,Sup}<-C1,Sup>=Min],
   LkPrint=[ {to_list(FS),Sup,Sup/length(DB)*100} || {FS,Sup}<-L1],
   mylib:text_file(write,CkPrint,lkfile),
   K=2, LS=[FS||{FS,_}<-L1],
   AllSet=aprioriAll(L1,DB,LS,K,Min),
   mylib:term_file(write,AllSet,"set.raw").

inputSup(AllInput)->Total=length(AllInput),
    {_,Per}=io:read(" input percent> "),
    MinSup=Total*Per/100 .

aprioriAll(AllL,_,[],_,_) -> format("~nfinal set=~p~n",[AllL]),AllL;  % return final Set
aprioriAll(AllL,_,[_],_,_) -> format("~nfinal set=~p~n",[AllL]),AllL;
```

```
aprioriAll(AllL,DB,LS,K,Min) -> Com=combi(LS),
     C_=myDistinct(usedCombi(Com,K)),
     Ck=[{X,findSup(X,DB)}|| X<-C_],mylib:c(35,Min),
     Lk=[ {FS,Sup} || {FS,Sup}<-Ck,Sup>=Min],
     LkS=[FS||{FS,_}<-Lk],
     LkPrint=[ {to_list(FS),Sup,Sup/length(DB)*100} || {FS,Sup}<-Lk],
     format("~nK=~w-~p,  has ~w set ~n  ",[K,LkPrint,length(LkPrint)]),
     aprioriAll(AllL++Lk,DB,LkS,K+1,Min) .

allPossibleAttr([H|T]) -> [H++ET||ET<-T].

input(AttrName) ->  LinesList=mylib:read_file( "ipums-1999-1Krecords.bak"," ," ),
     Zip=map( fun(EachL)->lists:zip(AttrName,EachL) end,LinesList ),
     UsedData=map(fun(LineOfTuple)->concat_line_tuple(LineOfTuple) end,Zip).

% shift([a,b,c]) --> [b,c,a]
shift([H|T]) -> T++[H].

genR(_,Max,Max) -> [];
genR(L,N,Max) -> {H,T}=lists:split(N,L), [{H,T}]++genR(L,N+1,Max).

% genRule([2,3,5],3,3).
genRule(_,0,_)->[];
genRule(L,Count,Len)-> genR(L,1,Len)++genRule(shift(L),Count-1,Len).
findConf({H,B},AllL)  -> {H,B,searchL(set(H++B),AllL)/searchL(set(H),AllL) }.

% main1() is for creating rules.
main1() ->
     format("~n----------START-create rules----------"),
     {_,[AllL]}=file:consult("set.raw"),
     AllAsso2=[list(X)|| {X,_} <-AllL,length(list(X))>1 ],
       %gen Rules
     AllRuleGen=lists:flatten([genRule(L,length(L),length(L))||L<-AllAsso2]),
     AllRuleConf=[findConf(X,AllL)||X<-AllRuleGen],
     format("~nAllRule=~p ,~nThere are ~p rules ",[AllRuleConf,length(AllRuleConf)]),
     mylib:text_file(write,AllRuleConf,allrule),
     Sorted= lists:sort(fun({_,_,C1},{_,_,C2})->C1>=C2 end,AllRuleConf),
     mylib:text_file(write,Sorted,"allsortedrule.txt"),
     Conf1=lists:filter(fun({A,B,C})->C==1.0 end,Sorted),
        %% write to file
        %%% create prolog file Rules+Facts
     {_,IO}=file:open("rules.pl",[write]),
     lists:map(fun(EachR)->transform_to_prolog(EachR,IO) end,Conf1),
     _=file:close(IO),
     mylib:text_file(append,length(Conf1),"allsortedrule.txt"),
     mylib:text_file(append,length(Sorted),"allsortedrule.txt"),
     format("~n----------end main1() process ----------") .
        %%% create prolog file Rules+Facts

transform_to_prolog({Body,Head,Conf},IO) ->
      if (length(Head)==1) -> [Head1]=Head,
                              io:format(IO,"~np(~p):-",[list_to_atom(Head1)]),
                              print_body(IO,Body), io:format(IO,"% Conf=~p",[Conf]);
         true -> io:format("")
      end.

print_body(IO,[H]) -> io:format(IO,"p(~p).",[list_to_atom(H)]) ;
print_body(IO,[H|T]) -> io:format(IO,"p(~p),",[list_to_atom(H)]),print_body(IO,T).

set(X) -> from_list(X).
list(X) -> to_list(X).
```

```
searchL(Set,[{Set,Val}|_])   ->  Val;
searchL(Set,[{_Another,_}|T])  -> searchL(Set,T);
searchL(_Set,[]) ->  1  .% Cannot find Set

concat_line_tuple(LineOfTuple) -> map(fun({A,B})->A++B end, LineOfTuple).

myToSet(L) -> [from_list(X)||X<-L].
myToList(SL) -> [to_list(S)||S<-SL].

findSup(_,[]) -> 0;
findSup(Set,DB) -> [H|T]=DB,
     Cond = is_subset(Set,H),
     if Cond -> 1+findSup(Set,T);
        true -> findSup(Set,T)
     end.

myDistinct(List) -> to_list(from_list(List)).

combi([H|T]) -> [[H,Te] || Te<-T]++ combi(T);
combi([]) -> [].

usedCombi([H|T],K) ->  Union=union(H),
     Len=ordsets:size(Union),
     if Len==K -> [Union|usedCombi(T,K)];
        true -> usedCombi(T,K)
     end ;
usedCombi([],_) -> [].

 %-----------  end of association rule mining program ------------------

-module(mylib).
-compile([export_all]).
    %% extract data from formatted text file ,delimeter= blank,comma,newline
read_file(FileN,Delimeter) ->
     io:format("~nRead from file:~p",[FileN]),
     {ok, Binary} = file:read_file(FileN),
     Lines = string:tokens(erlang:binary_to_list(Binary), "\n\r"),
     ReturnL=lists:map(fun(X) -> string:tokens(X,Delimeter) end,Lines).

my_sort(max,L) -> lists:sort(fun({A,_},{B,_})-> (A >= B ) end,L);
my_sort(min,L) -> lists:sort(fun({A,_},{B,_})-> (A =< B ) end,L).

text_file(Do,Data,FName) ->
     {ok,FP}=file:open(FName,[Do]),
     io:format(FP,"~n%-----~p text file:~p  {Date,Time}=~p ~n",[Do,FName,calendar:local_time()]),
     io:format(FP,"~p",[Data]),
     file:close(FP).

term_file(Do,Data,FName) ->
     {ok,FP}=file:open(FName,[Do]),
     io:format(FP,"%term file:~p  {Date,Time}=~p ~n",[FName,calendar:local_time()]),
     io:format(FP,"~p.",[Data]),
     file:close(FP).

sub_list(L1,L) -> length(L--L1)==length(L)-length(L1).

c(Line,Term) -> io:format("~nin~pCheck=~p~n",[Line,Term]).

% ------------- end of mylib module ------------------------------------
```

ประวัติผู้วิจัย

รองศาสตราจารย์ ดร.นิตยา เกิดประสพ สำเร็จการศึกษาในระดับปริญญาเอกสาขา Computer Science จาก Nova Southeastern University เมือง Fort Lauderdale รัฐฟลอริดา สหรัฐอเมริกา เมื่อปีพุทธศักราช 2542 (ค.ศ. 1999) ด้วยทุนการศึกษาของกระทรวงวิทยาศาสตร์ฯ โดยทำวิทยานิพนธ์ระดับปริญญาเอกในหัวข้อเรื่อง "The application of inductive logic programming to support semantic query optimization" หลังสำเร็จการศึกษาได้ปฏิบัติราชการในตำแหน่งอาจารย์ ประจำสาขาคอมพิวเตอร์ ภาควิชาคณิตศาสตร์ คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ต่อมาในปีพุทธศักราช 2543 ได้มาปฏิบัติงานในตำแหน่งอาจารย์ประจำสาขาวิชาวิศวกรรมคอมพิวเตอร์ มหาวิทยาลัยเทคโนโลยีสุรนารี จนถึงปัจจุบัน งานวิจัยที่ทำในขณะนี้คือ การพัฒนาระบบเหมืองข้อมูลประสิทธิภาพสูงที่สามารถทนต่อข้อมูลรบกวน และการเพิ่มความสามารถในการจัดการความรู้ของระบบเหมืองข้อมูล

รองศาสตราจารย์ ดร.กิตติศักดิ์ เกิดประสพ สำเร็จการศึกษาในระดับปริญญาเอกสาขา Computer Science จาก Nova Southeastern University เมือง Fort Lauderdale รัฐฟลอริดา สหรัฐอเมริกา เมื่อปีพุทธศักราช 2542 (ค.ศ. 1999) ด้วยทุนการศึกษาของทบวงมหาวิทยาลัย (หรือสำนักงานคณะกรรมการอุดมศึกษาในปัจจุบัน) โดยทำวิทยานิพนธ์ระดับปริญญาเอกในหัวข้อเรื่อง "Active database rule set reduction by knowledge discovery" หลังสำเร็จการศึกษาได้ปฏิบัติงานในตำแหน่งอาจารย์ ประจำสาขาวิชาวิศวกรรมคอมพิวเตอร์ สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี ปัจจุบันดำเนินการวิจัยเกี่ยวกับการพัฒนาระบบเหมืองข้อมูลประสิทธิภาพสูงที่สามารถทนต่อข้อมูลรบกวน และการวิจัยพื้นฐานเกี่ยวกับเทคนิคการจัดกลุ่มข้อมูล และการวิเคราะห์ข้อมูลโดยวิธีอัตโนมัติ โดยมีผลงานวิจัยตีพิมพ์ในวารสารวิชาการและเอกสารการประชุมวิชาการ จำนวนมากกว่า 30 เรื่อง ในสาขาฐานข้อมูลแอคทีฟ ฐานข้อมูลนิรนัย การทำเหมืองข้อมูลและการค้นหาความรู้