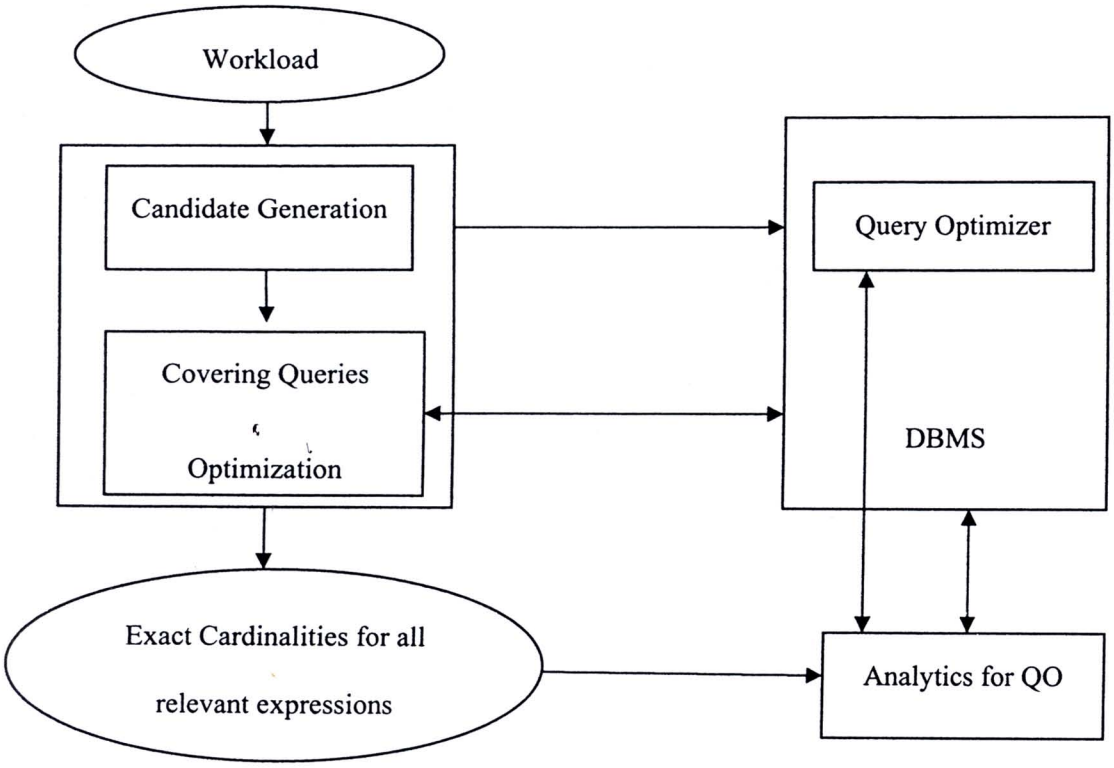
บทที่ 2
การคำนวณจำนวนแถวแบบเอ็กแซกท์

2.1 การคำนวณจำนวนแถวแบบเอ็กแซกท์

เพื่อตอบการสอบถาม (Query) ระบบจัดการฐานข้อมูล (DBMS) จะทำหน้าที่เป็นตัวกลางในการประมวลผลการสอบถามนั้นๆ สิ่งที่ระบบจัดการฐานข้อมูลจะต้องทำเพื่อตอบการสอบถามมีขั้นตอนมากมาย ขั้นตอนสำคัญขั้นตอนหนึ่งได้แก่ การเลือกแผนประมวลผล (Execution Plan) ซึ่งข้อมูลสำคัญข้อมูลหนึ่งที่ระบบจัดการฐานข้อมูลใช้ในการตัดสินใจเลือกแผนประมวลผลที่ดีที่สุด คือ การประเมินจำนวนแถว (Cardinality) โดยวิธีการประเมินจำนวนแถวมีมากมายหลายวิธี ในวิทยานิพนธ์ฉบับนี้สนใจ การประเมินจำนวนแถวแบบเอ็กแซกท์ (Exact Cardinality) ซึ่งขั้นตอนวิธีการนี้เป็นเทคนิคที่ทำให้การเลือกแผนประมวลผลมีประสิทธิภาพมากเนื่องจากวิเคราะห์จากจำนวนแถวที่เป็นค่าจริง ซึ่งเป็นการวิเคราะห์ที่มีความถูกต้องและแม่นยำมากที่สุด นอกจากนั้นในวิทยานิพนธ์ฉบับนี้ยังพิจารณาการแปลงการสอบถาม (Query Transformation) เพื่อที่จะดูผลว่าทำให้ประสิทธิภาพและเวลาในการประมวลผล (Execute) แตกต่างไปจากเดิมอย่างไร

ในงานวิจัย [1] เสนอแนวทางที่จะทำให้ได้แผนประมวลผลที่มีประสิทธิภาพซึ่งได้แก่ การประเมินจำนวนแถวแบบเอ็กแซกท์ โดยมีขั้นตอนการทำงานดังที่แสดงในรูปที่ 2.1 โดยเริ่มแรกจะมีอินพุตคือ เวิร์คโหลด (Workload) ซึ่งประกอบไปด้วยชุดการสอบถามข้อมูล

ในงานวิจัย [1] ได้ทำการประยุกต์ที่จะช่วยให้การประเมินจำนวนแถวแบบเอ็กแซกท์เป็นไปอย่างรวดเร็วด้วยการเพิ่มเทคนิคคือ การหาความสัมพันธ์ของเงื่อนไขในทุกๆ เงื่อนไขในเวิร์คโหลด เพราะฉะนั้นเวิร์คโหลดจะประกอบไปด้วยชุดของการสอบถามข้อมูลรวมกับความสัมพันธ์ของเงื่อนไขที่มีความสัมพันธ์กันในทุกๆ เวิร์คโหลด หลังจากนั้นจะทำการหาความเหมือนของเงื่อนไขที่มีอยู่ในเวิร์คโหลดขณะนั้นแล้วนำเงื่อนไขที่มีความสัมพันธ์ที่คล้ายกันมาจัดกลุ่มแล้วเรียกกลุ่มที่มีความสัมพันธ์เหมือนๆ กันว่า ซิกเนเจอร์ (Signature) โดยสิ่งที่จะนำใช้ในการตัดสินใจว่าเป็นซิกเนเจอร์เดียวกันคือ มีการจอยตารางที่เหมือนกันและมีการจอยพรีดิเคต (Join Predicate) ที่เหมือนกันและมีการจัดกลุ่ม (Aggregate) จากตารางเดียวกัน



รูปที่ 2.1 แสดงโครงสร้างการหาค่าจำนวนแถว

หลังจากที่ได้ซิกเนเจอร์แต่ละซิกเนเจอร์ในเวิร์คโหลดแล้ว จะนำซิกเนเจอร์เหล่านั้นมาทำเคสคิวรี (Case Query) โดยในรูปที่ 2.2 แสดงตัวอย่างเงื่อนไข 2 เงื่อนไขที่มีความคล้ายคลึงกันและอยู่ในซิกเนเจอร์เดียวกัน ซึ่งเทคนิคเคสคิวรีจะเป็นเทคนิคที่นำเงื่อนไขที่มีความสัมพันธ์กันหลายๆ เงื่อนไขนำมารวมกันเพื่อคำนวณจำนวนแถวในครั้งเดียว สาเหตุที่เลือกใช้เทคนิคเคสคิวรีแทนที่จะใช้เทคนิคอื่นๆ นั้นเพราะ สิ่งที่ต้องการคือการนับจำนวนแถวที่เป็นค่าจริงๆ ไม่ใช่ผลลัพธ์ของตัวการสอบถามแต่อย่างใด เทคนิคเคสคิวรีจึงเป็นเทคนิคที่ทำให้การนับจำนวนแถวเป็นไปอย่างรวดเร็วกว่าเทคนิคอื่นๆ ดังแสดงในรูปที่ 2.3

```
e1 = SELECT…. FROM Lineitem
WHERE l_discount < 0.05 and
l_shipdate < '1998-01-01'

e2 =  SELECT…. FROM Lineitem
WHERE l_discount < 0.15 and
l_shipdate < '1997-01-01'
```

รูปที่ 2.2 แสดงเงื่อนไขที่มีความคล้ายคลึงกัน

```
SELECT SUM(a) as card1, SUM(b) as card2
FROM
(
      SELECT      a = CASE When ( l_discount < 0.05
      and l_shipdate < '1998-01-01') then
      1 else 0 end,
      b = CASE When ( l_discount < 0.15
      and l_shipdate < '1997-01-01' then
      1 else 0 end
      FROM        Lineitem
      WHERE       l_discount < 0.15 and
      l_shipdate < '1998-01-01')
```

รูปที่ 2.3 แสดงเคสดิวรี

โดยการทำเคสดิวรีนั้นในงานวิจัย [1] มีขั้นตอนวิธีการทำดังแสดงในรูปที่ 2.4 จากรูปเริ่มต้นด้วยการนำซิกเนเจอร์มาเป็นอินพุต และจะทำการสร้างการจอยพรีดิเคตใหม่ นำจอยพรีดิเคตของแต่ละเงื่อนไขนำมากำหนดให้เป็นตัวแปร $p_{11} \wedge p_{12} \ldots\ldots\ldots p_{1n}$ สำหรับเงื่อนไขที่ 1 และ $p_{n1} \wedge p_{n2} \ldots\ldots\ldots p_{nm}$ สำหรับเงื่อนไขที่ n หลังจากนั้นนำจอยพรีดิเคตมาทำการเปรียบเทียบกันเป็นคู่ๆ (p_{1i} , p_{2j}) จนครบทุกตัว ซึ่งจะทำการเปรียบเทียบเพื่อหาจอยพรีดิเคตที่ใช้คอลัมน์ (Colunm)

เดียวกันถ้าเป็นการจอยพรีดิเคตที่มีการใช้คอลัมน์เดียวกันจะทำการเพิ่มลงไปในเคสคิวรีที่ทำการสร้างขึ้น โดยจะเป็นจอยพรีดิเคตใหม่และจอยพรีดิเคตใหม่จะต้องเป็นค่าขอบเขตครอบคลุมที่น้อยที่สุด ยกตัวอย่างเช่น l_discount < 0.05 กับ l_discount < 0.15 เมื่อทำการเปรียบเทียบแล้ว มีการใช้คอลัมน์ที่เหมือนกัน แต่จะทำการเลือก l_discount < 0.15 เนื่องจากตรงกับเหตุผลที่ได้กล่าวมาข้างต้น หลังจากนั้นในเงื่อนไขการเลือก (Select) จะทำการเพิ่มเคสคิวรีของแต่ละเงื่อนไขเข้าไปโดยประกอบไปด้วยจอยพรีดิเคตของแต่ละเงื่อนไข สุดท้ายผลลัพธ์ที่ได้จะเป็นการสอบถามใหม่ที่เป็นการสอบถามแบบเคสคิวรี

GenerateCandidateQuery

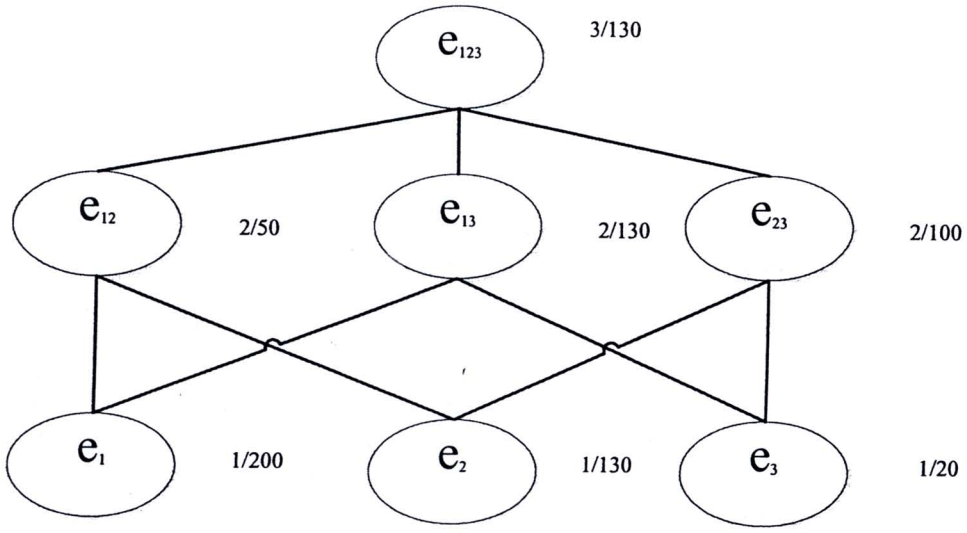
Input e_1,e_2: Conjunctive SPJ expressions with the same signature

Output : Conjunctive query Q that obtains cardinality of e_1 and e_2

1. Let e be an expression, initialized to the tables,join predicates in the signature
2. Let $p_{11} \wedge \ldots p_n$ be the selection predicates of e_1 and $p_{21} \wedge \ldots p_{2m}$ be the selection predicates of e_2
3. For each predicate pair (p_{1i} , p_{2j}) that is defined on the same column,let p = $(p_{1i} \vee p_{2j})$ // disjunction of predicates. Add p as conjuct to e
4. In the SELECT clause of e,add a CASE statement with one clause for each input expression. The predicate in the WHEN clause are $(p_{11} \wedge \ldots p_n)$ and $(p_{21} \wedge \ldots p_{2m})$ respectively. The value of the THEN clause is 1 and ELSE clause is 0
5. Let Q be a scalar aggregate query on expression e, with one SUM aggregate for each column of e.
6. Return Q

รูปที่ 2.4 แสดงขั้นตอนการสร้างเคสคิวรี

จากวิธีการสร้างข้างต้นที่ได้กล่าวมาเป็นวิธีการสร้างเคสคิวรี แต่ก่อนที่จะนำเงื่อนไขที่มีความสัมพันธ์กันมาทำเคสคิวรีนั้นจะมีขั้นตอนการเพิ่มประสิทธิภาพให้แก่การคำนวณจำนวนแถวแบบเอ็กแซกท์ โดยจะทำเลือกเงื่อนไขที่จะนำมาทำเคสคิวรี่โดยใช้ประสิทธิภาพมาเป็นตัวตัดสินใจว่าเงื่อนไขที่จะนำมาทำเคสคิวรีนั้นต้องมีประสิทธิภาพที่ดีที่สุดตัวอย่างแสดงดังรูปที่ 2.5 จากรูปมีเงื่อนไขที่ 1 เงื่อนไขที่ 2 และเงื่อนไขที่ 3 ที่จะนำมาทำเคสคิวรี ซึ่งประกอบไปด้วยตัวเลข 2 ชุดในแต่ละเงื่อนไข โดยตัวเลขชุดแรกเกิดจากจำนวนเงื่อนไขที่จะนำมาทำเคสคิวรี และตัวเลขชุดที่ 2 คือค่าประสิทธิภาพของเงื่อนไขนั้น โดยคำนวณจากการประมาณค่าโดยออพติไมเซอร์ (Optimizer) และจากรูปที่ 2.5 e_{12} จะหมายถึงเคสคิวรีที่เกิดจากเงื่อนไขที่ 1 และเงื่อนไขที่ 2 โดยการเลือกมีขั้นตอนดังแสดงในรูปที่ 2.6 โดยขั้นตอนจะเริ่มจากโหนดที่อยู่บนสุดก่อนทำการเปรียบเทียบกับทุกโหนดจนได้ค่าประสิทธิภาพที่ดีที่สุด โดยจากรูปที่ 2.5 จะทำการเลือกเงื่อนไขที่ 1 เงื่อนไขที่ 2 และเงื่อนไขที่ 3 นำมาทำเคสคิวรี เพราะเมื่อผ่านขั้นตอนวิธีในรูปที่ 2.6 แล้วประสิทธิภาพของโหนดบนสุดนั้นมากที่สุด ซึ่งในขั้นตอนวิธี (Algorithm) ในรูปที่ 2.6 ยังได้ทำการตรวจสอบว่าถ้าโหนดลูกของโหนดที่กำลังวัดประสิทธิภาพ ไม่มีโหนดลูกตัวไหนที่มีค่าประสิทธิภาพมากกว่าโหนดแม่ ก็จะทำเลือกโหนดแม่นั้นมาทำเคสคิวรีเลย



รูปที่ 2.5 แสดงการเลือกเงื่อนไขมาทำเคสคิวรรี

```
SelectCASEQueryForSignature

Input : Set of expression R belonging to a particular signature

Output : Candidate expression for signature

1.  Let u = candidate obtained by invoking
    GenerateCandidateQuery(R)// (least upper bound of the lattice)
2.  Do
3.      BetterNodeExists = false
4.      For each child expression e of u in the lattice
5.          GenerateCandidateQuery(e)
6.          If Benefit(e) > BestBenefit
7.              BestBenefit = Benefit(e); u =e;
8.              BetterNodeExists = true;
9.  While(BetterNodeExists)
10. Return u
```

รูปที่ 2.6 แสดงขั้นตอนวิธีการเลือกเคสคิวรี

นอกจากการทำเคสคิวรีที่ช่วยเพิ่มประสิทธิภาพในการคำนวณจำนวนแถวแบบเอ็กแซกท์แล้ว ในงานวิจัย [1] ยังทำการใช้ขั้นตอนวิธีที่มีชื่อว่า คอเวอร์ริงคิวรีออพติไมเซชัน(Covering Queries Optimization) ซึ่งเป้าหมายของขั้นตอนวิธีนี้คือ ต้องการหาเซ็ตย่อย (Subset) ของการสอบถามที่มีค่าประสิทธิภาพที่น้อยที่สุดที่ซึ่งครอบคลุมทุกเงื่อนไขความสัมพันธ์ที่มีอยู่ในเวิร์คโหลด โดยในที่นี้ได้กำหนดความสัมพันธ์ของเงื่อนไขทุกๆ เงื่อนไขคือ R_Q ขั้นตอนการทำงานแสดงอยู่ในรูปที่ 2.7 โดยขั้นตอนแรกอินพุตเริ่มต้นจะเป็นชุดความสัมพันธ์ของทุกๆ เงื่อนไขที่มีอยู่ในเวิร์คโหลด กำหนดตัวแปรขึ้นมา 2 ตัวคือ R และ S โดยกำหนดให้เป็นเซ็ตว่าง ต่อจากนั้นทำการเลือกเงื่อนไขมาหนึ่งเงื่อนไขจากเงื่อนไขทั้งหมด แล้วทำการวัดประสิทธิภาพ โดยกำหนดให้ Exprs(e) เป็นเซ็ตย่อยของ R_Q และ Cost(e) เป็นค่าประสิทธิภาพของเงื่อนไขนั้น นำไปแทนค่าใน

สมการ |Exprs(e) – S| / Cost(e) ทำการเพิ่มและจัดเรียงลำดับใน S และนำเงื่อนไขที่เลือกมาเพิ่มเข้าไปใน R ทำซ้ำจนครบทุกเงื่อนไขที่มีอยู่ในเวิร์คโหลดหลังจากนั้นนำ R มาทำการประมวลผลเพื่อที่จะนำค่าจำนวนแถวแบบเอ็กแซกท์ โดยในรูปที่ 2.8 แสดงขั้นตอนวิธีการคำนวณจำนวนแถวแบบเอ็กแซกท์ทั้งหมด ซึ่งเป็นการรวมเทคนิคที่กล่าวมาข้างต้นทั้งหมดนำมารวมกัน โดยสุดท้ายผลลัพธ์ที่ได้จะเป็นแผนประมวลผลที่ได้จากการคำนวณจำนวนแถวแบบเอ็กแซกท์

CoveringQueriesOptimization

Input : Set of relevant expressions R_Q for a query Q

Output : $R \subseteq R_Q$ Such that executing all expressions in R gives exact cardinalities for all expression in R_Q

1. R = {}, S ={}
2. While (S != R_Q) Do
3. Pick $e \in (R_Q - R)$ With the largest value of
   | Exprs(e) –S| /Cost(e)
4. $R = R \cup \{e\}$; $S = S \cup exprs(e)$
5. End While
6. Return R

รูปที่ 2.7 แสดงขั้นตอนวิธีของครอเวอร์ริงคิวรีออพติไมเซชัน

ExactCardinalityQueryOptimization

Input : Workload of Queries, W

Output : Cardinality-optimal plan for each query in the workload

1. Let R_w be the set of relevant expressions for the workload
2. M = {} // set of candidates selected, one per signature
3. For each signature in R_w
4. Let C = Set of expressions in R_w belonging to that signature
5. c = *SelectCASEQueryForSignature* (c)
6. $M = M \cup \{c\}$
7. Let S = {}, R={}; $U = M \cup R_w$
8. While (S!= R_w)
9. Pick $e \in (U - R)$ with the largest value of | Exprs(e) –S| /Cost(e)
10. $R = R \cup \{e\}$; $S = S \cup exprs(e)$
11. End While
12. Execute each expression in S to obtain all relavant expression

รูปที่ 2.8 แสดงขั้นตอนวิธีการคำนวณจำนวนแถวแบบเอ็กแซกท์

2.2 การปรับปรุงประสิทธิภาพการสอบถาม

จากที่ได้นำเสนอวิธีการคำนวณจำนวนแถวแบบเอ็กแซกท์ในหัวข้อก่อนหน้านี้ สังเกตได้ว่าสิ่งที่สามารถทำให้การทำงานเร็วขึ้นส่วนหนึ่งมาจากการสอบถามข้อมูลที่เข้ามา ถ้าสามารถปรับปรุงประสิทธิภาพการสอบถามที่เข้ามาได้แล้ว ก็จะทำให้ลดการใช้เวลาในการประมวลผลขั้นตอนต่างๆ ลงไปได้ ในการปรับปรุงประสิทธิภาพการสอบถามได้มีเทคนิคต่างๆ มากมาย

2.2.1 การปรับปรุงประสิทธิภาพการสอบถามด้วยการเขียนการสอบถามใหม่

การเข้าถึงข้อมูลด้วยการเขียนการสอบถาม โดยทั่วไปการเขียนการสอบถามจะเขียนด้วยภาษาเอสคิวแอล ซึ่งการสอบถามดังกล่าวจะถูกประมวลผลด้วยระบบจัดการฐานข้อมูล เมื่อระบบได้รับการสอบถาม ระบบจะเริ่มตรวจสอบไวยากรณ์ว่าการสอบถามนั้น ถูกต้องตามรูปแบบของภาษาเอสคิวแอลหรือไม่เป็นอันดับแรก จากนั้น จึงเปลี่ยนแปลงการสอบถามด้วยตัวปรับปรุงการสอบถามต่อไป

จากข้อความข้างต้นจะเห็นได้ว่าตัวปรับปรุงการสอบถามเป็นส่วนหนึ่งที่สามารถเพิ่มประสิทธิภาพในการประมวลผลได้ ดังนั้นหากผู้ใช้เขียนการสอบถามที่ดี ตัวปรับปรุงการสอบถามก็สามารถทำงานได้อย่างมีประสิทธิภาพมากยิ่งขึ้น ซึ่งจะส่งผลดีในการสอบถามข้อมูลด้วย

ตัวปรับปรุงการสอบถามนั้นมีหลากหลายประเภท การแปลงการสอบถามเป็นตัวปรับปรุงตัวหนึ่งที่จะช่วยให้ประสิทธิภาพของการสอบถามข้อมูลดีขึ้น โดยเทคนิคการแปลงการสอบถามจะทำการนำการสอบถามเดิมมาทำการเขียนการสอบถามใหม่ ทำให้เกิดเป็นรูปแบบใหม่ที่มีการลดรูปหรือเปลี่ยนตำแหน่งในการสอบถาม เพื่อที่จะช่วยให้การประมวลผลของออพติไมเซอร์มีประสิทธิภาพที่ดีขึ้นจากเดิม ซึ่งในงานวิจัยนี้จะนำการปรับปรุงประสิทธิภาพการสอบถามแบบการแปลงการสอบถามมาใช้