ระบบแนะนำการคัดเลือกดัชนี Bitmap Join Index สำหรับโครงสร้างแบบ Star Schema

Bitmap Join Index Selection Advisory System for Star Schema

หทัย แก้วกรณ์

Hathai Kaewkorn

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญา
วิทยาศาสตรมหาบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์
มหาวิทยาลัยสงขลานครินทร์

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Computer Science
Prince of Songkla University
2557

ชื่อวิทยานิพนธ์ ระบบแนะนำการคัดเลือกดัชนี Bitmap Join Index สำหรับโครงสร้างแบบ Star Schema

ผู้เขียน นางสาวหทัย แก้วกรณ์

สาขาวิชา วิทยาการคอมพิวเตอร์

อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก

.....

(ผู้ช่วยศาสตราจารย์ ดร.ศิริรัตน์ วณิชโยบล)

คณะกรรมการสอบ

.....ประธานกรรมการ

(ดร.นพมาศ ปักเข็ม)

.....กรรมการ

(ผู้ช่วยศาสตราจารย์ ดร.ศิริรัตน์ วณิชโยบล)

.....กรรมการ

(ผู้ช่วยศาสตราจารย์ ดร.ลัดดา ปรีชาวีรกุล)

.....กรรมการ

(ผู้ช่วยศาสตราจารย์ ดร.วิภาดา เวทย์ประสิทธิ์)

บัณฑิตวิทยาลัย มหาวิทยาลัยสงขลานครินทร์ อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษา ตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์

.....

(รองศาสตราจารย์ ดร.ธีระพล ศรีชนะ)

คณบดีบัณฑิตวิทยาลัย

ขอรับรองว่า ผลงานวิจัยนี้มาจากการศึกษาวิจัยของนักศึกษาเอง และได้แสดงความขอบคุณบุคคลที่มีส่วนช่วยเหลือแล้ว

ลงชื่อ.....
(ผู้ช่วยศาสตราจารย์ ดร.ศิริรัตน์ วณิชโยบล)
อาจารย์ที่ปรึกษาวิทยานิพนธ์

ลงชื่อ.....
(นางสาวหทัย แก้วกรณ์)
นักศึกษา

ข้าพเจ้าขอรับรองว่า ผลงานวิจัยนี้ไม่เคยเป็นส่วนหนึ่งในการอนุมัติปริญญาในระดับใดมาก่อน และไม่ได้ถูกใช้ในการยื่นขออนุมัติปริญญาในขณะนี้

ลงชื่อ.....
(นางสาวหทัย แก้วกรณ์)
นักศึกษา

ชื่อวิทยานิพนธ์ ระบบแนะนำการคัดเลือกดัชนี Bitmap Join Index สำหรับโครงสร้างแบบ Star Schema
ผู้เขียน นางสาวหทัย แก้วกรณ์
สาขาวิชา วิทยาการคอมพิวเตอร์
ปีการศึกษา 2556

บทคัดย่อ

คลังข้อมูลจัดเก็บข้อมูลเป็นจำนวนมาก และคำสั่งสอบถามส่วนใหญ่เป็นแบบทันทีทันใด มีเงื่อนไขการสอบถามที่ซับซ้อน และใช้การดำเนินการจับคู่ข้อมูลระหว่างตารางมาก ดังนั้นจึงทำให้ใช้เวลาในการสอบถามนาน จึงมีการใช้เทคนิคการสร้างดัชนีเข้ามาช่วยในการเพิ่มความเร็วในการสอบถาม เช่น Join Index และ Bitmap Join Index เป็นต้น แต่ปัญหาที่สำคัญคือ ระบบมีพื้นที่ในการเก็บดัชนีที่จำกัด ทำให้ไม่สามารถสร้างดัชนีสำหรับทุกคำสั่งสอบถามที่เป็นไปได้ ดังนั้นจึงเกิดปัญหาในการเลือกดัชนีว่าควรสร้างดัชนีอย่างไรให้มีประสิทธิภาพด้านเวลาในการสอบถามดีที่สุด ภายใต้พื้นที่ในการจัดเก็บดัชนีที่จำกัด

งานวิจัยนี้นำเสนอระบบแนะนำการคัดเลือกดัชนี Bitmap Join Index ของคลังข้อมูลที่มีโครงสร้างแบบ Star Schema โดยการทำงานของระบบแบ่งออกเป็น 4 ขั้นตอน คือ การสกัดแอททริบิวต์ที่จะนำมาสร้างดัชนี การลดจำนวนแอททริบิวต์ที่จะนำมาสร้างดัชนี การประเมินทรัพยากรที่ใช้ในการสร้างดัชนี และการเลือกดัชนีที่เหมาะสมที่สุดสำหรับทรัพยากรที่จำกัด โดยเพิ่มประสิทธิภาพของขั้นตอนการสกัดแอททริบิวต์ด้วยการกรองแอททริบิวต์ที่มีโอกาสถูกคัดเลือกน้อยออกไปตั้งแต่ขั้นตอนแรกๆ มีการคำนวณค่าใช้จ่ายในการจับคู่ข้อมูลเพื่อใช้ในการพิจารณาความเหมาะสมในการสร้างดัชนีของแอททริบิวต์ และนำวิธีการแก้ปัญหาแบบพลวัตมาประยุกต์ใช้ในขั้นตอนการคัดเลือกดัชนี ผลการทดลองพบว่าดัชนีที่ถูกเลือกจากระบบดังกล่าวช่วยเพิ่มประสิทธิภาพในการสอบถามข้อมูล และลดค่าใช้จ่ายที่ใช้ในการจับคู่ข้อมูลได้เป็นอย่างดี

Thesis Title Bitmap Join Index Selection Advisory System for Star Schema
Author Miss Hathai Kaewkorn
Major Program Computer Science
Academic Year 2013

ABSTRACT

Data warehouse stores large volume of data. Queries in data warehouse are ad hoc, complex, and use lots of join operations, leading to increase query response time. Several techniques are used to solve this problem such as Join Index, Bitmap Join Index, etc. A major issue is the limited space capacity for storing indices because all attributes cannot be indexed. Therefore, the research issue is which attributes are properly to be indexed for reducing query response time under space constraint.

This thesis presents a Bitmap Join Index selection advisory system for star schema structure to increase efficiency in query processing time under limited space to store indices. The system is divided into four phases: candidate extraction, candidate attribute reduction, resource evaluation, and index selection. In candidate extraction and candidate attribute reduction phase, only appropriate attributes based on frequently used are selected. In resource evaluation phase, cost of tables' joining is considered. Finally, in index selection phase, the dynamic programming approach is applied to find the optimal indices. The experiment results show that the proposed system can significantly improve query processing time by reducing cost of join operations in advance.

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จได้ด้วยความช่วยเหลือและสนับสนุนจากบุคคลหลายฝ่าย ผู้วิจัยรู้สึกซาบซึ้งและขอกราบขอบพระคุณอย่างสูง คือ

ผู้ช่วยศาสตราจารย์ ดร.ศิริรัตน์ วณิชโยบล อาจารย์ที่ปรึกษาวิทยานิพนธ์ ที่กรุณาให้คำปรึกษาแนะนำ และช่วยเหลือในการแก้ปัญหาต่างๆ ให้แก่ผู้วิจัยเสมอมา พร้อมทั้งตรวจทานและแก้ไขวิทยานิพนธ์ให้แก่ผู้วิจัย

ดร.นพมาศ ปักเข็ม ประธานกรรมการสอบวิทยานิพนธ์ ที่กรุณาช่วยตรวจทานและแก้ไขวิทยานิพนธ์ให้มีความสมบูรณ์

ผู้ช่วยศาสตราจารย์ ดร.ลัดดา ปรีชาวีรกุล กรรมการในการสอบวิทยานิพนธ์ที่กรุณาให้ความรู้และข้อเสนอแนะในการทำวิจัย รวมทั้งตรวจทานแก้ไขวิทยานิพนธ์

ผู้ช่วยศาสตราจารย์ ดร.วิภาดา เวทย์ประสิทธิ์ กรรมการในการสอบวิทยานิพนธ์ที่กรุณาให้ความรู้และข้อเสนอแนะในการทำวิจัย รวมทั้งตรวจทานแก้ไขวิทยานิพนธ์

อาจารย์ภาควิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ มหาวิทยาลัยสงขลานครินทร์ทุกท่านที่ให้ความรู้ทางด้านวิชาการ ซึ่งสามารถนำความรู้นี้มาใช้ในการทำวิทยานิพนธ์

เจ้าหน้าที่ภาควิชาวิทยาการคอมพิวเตอร์ และเจ้าหน้าที่บัณฑิตวิทยาลัยทุกท่านที่ให้ความช่วยเหลือ และอำนวยความสะดวกเกี่ยวกับเอกสารต่างๆ

เพื่อนๆ พี่ๆ และน้องๆ ภาควิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ ที่ให้คำปรึกษา และช่วยเหลือในการทำวิทยานิพนธ์

คุณพ่อ คุณแม่ และสมาชิกในครอบครัวทุกคน ที่ให้การสนับสนุนคอยเป็นห่วงสุขภาพ รวมทั้งให้การสนับสนุนในการทำวิทยานิพนธ์แก่ผู้วิจัยมาโดยตลอด

ผู้วิจัยขอขอบคุณทุกท่านเป็นอย่างสูง ณ โอกาสนี้

หทัย แก้วกรณ์

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

หน้า

รายการตาราง

รายการตาราง (ต่อ)

รายการภาพประกอบ

รายการภาพประกอบ (ต่อ)

บทที่ 1

บทนำ

บทนี้จะกล่าวถึง ที่มาและความสำคัญของการวิจัย เอกสารงานวิจัยที่เกี่ยวข้อง วัตถุประสงค์ของการวิจัย ขอบเขตของการวิจัย วิธีการดำเนินการวิจัย ระยะเวลาและแผนการดำเนินการ สถานที่ดำเนินการวิจัย เครื่องมือและอุปกรณ์ที่ใช้ และประโยชน์ที่คาดว่าจะได้รับ

1.1 ที่มาและความสำคัญของการวิจัย

คลังข้อมูล (Data Warehouse) เป็นฐานข้อมูลขนาดใหญ่ซึ่งมีการเก็บรวบรวมข้อมูลที่มีความเกี่ยวข้องกัน ซึ่งอาจนำมาจากฐานข้อมูลดำเนินการ (Operational Database) ต่างๆ ขององค์กรตั้งแต่อดีต (Historical Data) จนถึงปัจจุบัน เพื่อใช้เป็นแหล่งข้อมูลส่วนกลางสำหรับองค์กร มีวัตถุประสงค์หลักในการนำมาวิเคราะห์เพื่อสนับสนุนการตัดสินใจ และการวางแผนการดำเนินงานขององค์กร (Chaudhuri and Dayal, 1997) สำหรับข้อมูลที่จะนำเข้าสู่คลังข้อมูลซึ่งได้มาจากแหล่งต่างๆ นั้นจะมีการคัดเลือกเฉพาะข้อมูลที่มีประโยชน์ในการนำมาวิเคราะห์ธุรกิจ การวิจัยตลาด และสนับสนุนการตัดสินใจ จากนั้นจะต้องมีการนำข้อมูลเหล่านั้นมาสังเคราะห์ และแปลงให้เป็นมาตรฐานเดียวกัน

ลักษณะโครงสร้างของคลังข้อมูลส่วนใหญ่เป็นแบบ Star Schema (O'Neil, 1997) ซึ่งประกอบด้วยตาราง Fact ที่เป็นตารางหลักเชื่อมโยงกับตาราง Dimension หลายตารางด้วยแอททริบิวต์ที่เป็นคีย์หลักของตาราง Dimension และในจำนวนตาราง Dimension ทั้งหมดต้องมีตารางที่เก็บข้อมูลเกี่ยวกับเวลา (Time Dimension) เนื่องจากการนำข้อมูลมาใช้ในการตัดสินใจมักจะมีการเปรียบเทียบข้อมูลตามช่วงเวลาต่างๆ นอกจากนั้นคำสั่งสอบถามข้อมูลสำหรับคลังข้อมูลส่วนใหญ่เป็นคำสั่งสอบถามที่มีความซับซ้อน (Complex Query) ซึ่งต้องใช้การจับคู่ข้อมูล (Join) ระหว่างตารางตั้งแต่ 2 ตารางขึ้นไป และลักษณะการสอบถามเป็นแบบทันทีทันใด (Ad Hoc Query) ไม่สามารถคาดเดาได้ว่าผู้ใช้จะสอบถามอะไรบ้าง ซึ่งจะประกอบด้วยเงื่อนไขการสอบถามข้อมูลที่ซับซ้อนขึ้นอยู่กับความต้องการของผู้ใช้ ดังนั้นเมื่อข้อมูลในคลังข้อมูลมีปริมาณมาก จะทำให้ใช้เวลาในการสอบถามข้อมูลมาก ดังนั้นจึงจำเป็นต้องมีการเพิ่มประสิทธิภาพในการเข้าถึงข้อมูล เพื่อให้สามารถสอบถามได้รวดเร็ว

การเพิ่มประสิทธิภาพการสอบถามข้อมูลในคลังข้อมูลมีหลายวิธี เช่น 1) การสร้าง Materialized View หรือ Summary Table และ 2) การสร้างดัชนี (Index) เป็นต้น สำหรับ Materialized View คือ การสร้างตารางข้อมูลผลลัพธ์ตามคำสั่งสอบถามที่มีโอกาสถูกสอบถามบ่อยเอาไว้ล่วงหน้า ข้อดีคือ ช่วยเพิ่มความรวดเร็วในการค้นหาข้อมูลได้เป็นอย่างดีถ้าคำสั่งที่

ผู้ใช้ที่สอบถามตรงกับที่สร้างไว้ เพราะไม่จำเป็นต้องอ่านข้อมูลจากตารางจริง ข้อเสียคือ เปลืองพื้นที่ในการเก็บข้อมูลมาก ไม่สามารถกำหนดได้ว่าควรสร้าง Materialized View สำหรับคำสั่งสอบถามใดบ้าง ดังนั้นจึงไม่เหมาะกับการสอบถามแบบทันทีทันใด และเมื่อใดที่มีการปรับปรุงข้อมูลในตารางจริง ก็ต้องมีการสร้าง Materialized View ขึ้นมาใหม่ทั้งหมด (Bizarro and Madeira, 2001) ส่วนการสร้างดัชนี คือ วิธีการในการเพิ่มความรวดเร็วในการเข้าถึงข้อมูลโดยไม่ต้องมีการเพิ่มอุปกรณ์เข้ามาในระบบ ข้อดีคือ ช่วยให้ประหยัดค่าใช้จ่ายได้มาก และคุณสมบัติหนึ่งที่สำคัญของคลังข้อมูล คือข้อมูลในคลังข้อมูลจะไม่มีการเปลี่ยนแปลง คำสั่งส่วนใหญ่จะเป็นคำสั่งสอบถาม (SELECT) และคำสั่งเพิ่มข้อมูล (INSERT) ซึ่งในการเพิ่มข้อมูลจะดำเนินการครั้งละมากๆ ในช่วงเวลาที่แน่นอน (O'Neil and Quass, 1997) ดังนั้นการสร้างดัชนีจึงเป็นตัวเลือกที่ดีในการเพิ่มประสิทธิภาพการสอบถามของคลังข้อมูล แต่ยังคงมีข้อเสียคือ หากมีการสร้างดัชนีจำนวนมาก แม้ว่าจะช่วยลดเวลาในการสอบถามได้ดีแต่ต้องการใช้พื้นที่เก็บดัชนีเป็นจำนวนมาก ซึ่งอาจมีบางส่วนที่ไม่ได้ถูกเรียกใช้เลย ส่วนการสร้างดัชนีจำนวนน้อยเกินไป ก็จะทำให้ใช้เวลาในการเข้าถึงข้อมูลนาน

ในการสร้างดัชนีนั้นสามารถแบ่งได้เป็น 2 กลุ่ม คือกลุ่มที่สร้างดัชนีบนตารางเดียว และกลุ่มที่สร้างดัชนีบนหลายตาราง Bitmap Join Index เป็นดัชนีบนหลายตารางชนิดหนึ่งที่เป็นที่นิยมใช้ เนื่องจากมีความเหมาะสมสำหรับคลังข้อมูล (Wu and Buchmann, 1997) ในบทความนี้ ผู้วิจัยจึงได้เสนอแนวคิด และพัฒนาเทคนิคในการเลือกสร้างดัชนีสำหรับ Bitmap Join Index โดยคำนึงถึงทรัพยากรที่ใช้ในการสร้างดัชนีแต่ละตัว ว่าควรสร้างดัชนีสำหรับแอท-ทริบิวต์ใดบ้างเพื่อให้เหมาะสมกับทรัพยากรที่จำกัด และยังคงมีประสิทธิภาพในด้านการสอบถามข้อมูล

1.2 เอกสารงานวิจัยที่เกี่ยวข้อง

งานวิจัยที่ผ่านมา ได้มีการนำเสนอวิธีการคัดเลือกดัชนีสำหรับ Bitmap Join Index โดยใช้วิธีการที่หลากหลาย เพื่อให้มีประสิทธิภาพในด้านเวลาที่ใช้ในการสอบถามข้อมูลและพื้นที่ที่ใช้สำหรับจัดเก็บดัชนี ดังนี้

Automatic Selection of Bitmap Join Indexes in Data Warehouses

งานวิจัยนี้ (Aouiche *et al*., 2005) ได้นำเสนอวิธีการในการเลือกดัชนี โดยแบ่งการทำงานเป็น 2 ขั้นตอน ในขั้นตอนแรกเป็นการนำเอาอัลกอริทึม Close ซึ่งเป็นเทคนิคในการหากลุ่มข้อมูลที่ปรากฏด้วยกันบ่อยมาใช้ในการคัดเลือก Candidate Index ที่สกัดได้จากชุดคำสั่งสอบถามในอดีต หลังจากนั้นมีการสร้าง Cost Model ขึ้นมา เพื่อประเมินค่าใช้จ่ายในการสร้างดัชนี ซึ่งประกอบไปด้วยค่าใช้จ่ายในการเก็บดัชนี ค่าใช้จ่ายเมื่อมีการเพิ่มข้อมูลใน

ตารางที่นำมาสร้างดัชนี และค่าใช้จ่ายในการเข้าถึงข้อมูล จากนั้นใช้เทคนิคการแก้ปัญหาแบบ Greedy ในการคัดเลือกดัชนีที่มีความคุ้มค่ามากที่สุด สำหรับทรัพยากรที่จำกัดของระบบ เช่น จำนวนดัชนีต่อ 1 ตาราง หรือพื้นที่ที่ผู้ดูแลระบบกำหนดให้ใช้ในการสร้างดัชนี

A Data Mining Approach for Selecting Bitmap Join Indices

งานวิจัยนี้ (Bellatreche *et al*., 2007) ได้พัฒนาวิธีการสำหรับการแก้ปัญหาการเลือกดัชนี โดยมีการเพิ่มการพิจารณาปัจจัยที่เกี่ยวข้องกับการสร้างดัชนีที่นอกเหนือจากความถี่ในการสอบถามสำหรับขั้นตอนการเลือก Candidate Index เช่น จำนวนและขนาดของแถวข้อมูลในตาราง Dimension และขนาดของ Disk Page เพื่อเพิ่มประสิทธิภาพในการหากลุ่มข้อมูลที่ปรากฏร่วมกันบ่อย จากนั้นใช้เทคนิคการแก้ปัญหาแบบ Greedy ในการเลือกแอททริ-บิวต์ที่จะนำไปสร้างดัชนี โดยพิจารณาเลือกจากชุดดัชนีที่มีค่าใช้จ่ายในการเข้าถึงข้อมูลที่น้อยที่สุดภายใต้ทรัพยากรที่จำกัด

Improving Star Join Queries Performance: A Maximal Frequent Pattern Based Approach for Automatic Selection of Indexes in Relational Data Warehouse

งานวิจัยนี้ (Ziani and Ouinten, 2011) ได้ปรับปรุงขั้นตอนในการสร้าง Candidate Index โดยใช้วิธีการ Maximal Frequent Itemset ในการสร้างกลุ่มข้อมูลที่ปรากฏบ่อย ซึ่งมีแนวคิด คือเลือกเฉพาะชุดแอททริบิวต์ที่ไม่เป็นเซตย่อยของชุดแอททริบิวต์อื่น เพื่อลดจำนวน Candidate Index ที่จะนำมาสร้างดัชนีแต่ในงานวิจัยนี้ยังไม่ได้มีการพิจารณาการสร้างดัชนีภายใต้ทรัพยากรที่จำกัด

A Study on the Selection of Bitmap Join Index using Data Mining Techniques

งานวิจัยนี้ (Hyoung and Jae, 2012) ได้นำเสนอวิธีเลือกดัชนีโดยแบ่งการทำงานเป็น 2 ขั้นตอนหลักคือ การหากลุ่มข้อมูลที่ปรากฏร่วมกันบ่อย โดยมีการพัฒนาอัลกอริทึม EFP-Treeเพื่อลดจำนวน Candidate Index ซึ่งได้ปรับปรุงจากเทคนิค FP-Growth ของการทำเหมืองข้อมูล และกำจัดกลุ่มข้อมูลที่มีความเกี่ยวข้องกับแอททริบิวต์ที่เป็นคีย์หลักออกไปเพื่อปรับปรุงประสิทธิภาพการประมวลผลข้อมูล และพัฒนาอัลกอริทึม sBJI เพื่อใช้ในการคัดเลือกดัชนีจาก Candidate Index ที่ได้จากขั้นตอนแรก โดยใช้เทคนิคการแก้ปัญหาแบบ Greedy ในการเลือกสร้างดัชนีภายใต้เงื่อนไขของพื้นที่ที่จำกัด

1.3 วัตถุประสงค์ของการวิจัย

เพื่อวิเคราะห์ ออกแบบ และพัฒนาเทคนิคในการเลือกสร้างดัชนีสำหรับ Bitmap Join Indexในการแก้ปัญหาการเลือกดัชนีของคลังข้อมูล

1.4 ขอบเขตของการวิจัย

1. วิเคราะห์เทคนิคในการเลือกสร้างดัชนี Bitmap Join Index เพื่อให้เหมาะสมกับข้อจำกัดของระบบ และมีประสิทธิภาพในการสอบถาม
2. ออกแบบวิธีการในการเลือกสร้างดัชนี Bitmap Join Index ที่มีประสิทธิภาพด้านการสอบถาม
3. พัฒนาและทดสอบวิธีการในการเลือกสร้างดัชนี Bitmap Join Index ที่มีประสิทธิภาพด้านการสอบถาม

1.5 วิธีการดำเนินการวิจัย

1. ศึกษางานวิจัยและเอกสารที่เกี่ยวข้องดังนี้
    1) คลังข้อมูล
    2) กระบวนการสอบถามข้อมูล
    3) การเพิ่มประสิทธิภาพการสอบถามข้อมูล
    4) ค่าใช้จ่ายในการสร้างดัชนี
    5) วิธีการแก้ปัญหาแบบพลวัต
2. ศึกษาเทคโนโลยีและเครื่องมือสำหรับงานวิจัย
    1) ศึกษาแนวคิดในการแก้ปัญหาการเลือกดัชนี
    2) ศึกษาเทคโนโลยีการสร้างดัชนีสำหรับการเพิ่มประสิทธิภาพในการสอบถามข้อมูล
    3) ศึกษาเครื่องมือที่ใช้ในการทดสอบประสิทธิภาพของขั้นตอนวิธี ได้แก่ ระบบจัดการฐานข้อมูล Oracle 12g
    4) ศึกษาข้อมูลที่ใช้ในการทดสอบขั้นตอนวิธี ซึ่งเป็นตัวอย่างข้อมูลมีใช้โครงสร้างแบบ Star Schema ได้แก่ SSB Benchmark (O'neil *et al*., 2009)
3. วิเคราะห์และออกแบบขั้นตอนวิธี
4. พัฒนาและทดสอบขั้นตอนวิธีที่ได้ออกแบบไว้
    1) จัดเตรียมข้อมูลที่ใช้ในการทดสอบขั้นตอนวิธี
    2) พัฒนาโปรแกรมเพื่อใช้ในการคัดเลือกดัชนีโดยใช้ตัวแปลภาษาซี (C Compiler) ตามขั้นตอนวิธีที่ได้ออกแบบไว้

3) ทดสอบและประเมินประสิทธิภาพในการเลือกดัชนีในด้านเวลาที่ใช้ในการสอบถาม และด้านพื้นที่ที่ใช้ในการจัดเก็บดัชนี โดยใช้คำสั่งสอบถามตัวอย่างจาก SSB Benchmark

5. เขียนบทความเผยแพร่งานวิจัย
6. จัดทำเอกสารวิทยานิพนธ์

1.6 ระยะเวลาดำเนินการและแผนการดำเนินการ

ระยะเวลาดำเนินการ พฤษภาคม พ.ศ.2556 - มีนาคม พ.ศ.2557

ตาราง 1-1 แผนการดำเนินงาน

กิจกรรม/ขั้นตอนการดำเนินงาน	เดือน										
	พ.ศ. 2556								พ.ศ. 2557		
	5	6	7	8	9	10	11	12	1	2	3
1. ศึกษางานวิจัยและเอกสารที่เกี่ยวข้อง											
2. ศึกษาเทคโนโลยีและเครื่องมือสำหรับงานวิจัย											
3. วิเคราะห์และออกแบบกลไกการทำงาน											
4. พัฒนาและทดสอบกลไกการทำงาน											
5. เขียนบทความเผยแพร่งานวิจัย											
6. จัดทำเอกสารวิทยานิพนธ์											

1.7 สถานที่ดำเนินการวิจัย

ห้องปฏิบัติการวิจัยเทคโนโลยีระบบสารสนเทศและการประยุกต์ (*i*STAR) ภาควิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ มหาวิทยาลัยสงขลานครินทร์ วิทยาเขตหาดใหญ่

1.8 เครื่องมือและอุปกรณ์ที่ใช้

ด้านฮาร์ดแวร์

เครื่องคอมพิวเตอร์ส่วนบุคคลจำนวน 1 เครื่อง ซึ่งมีรายละเอียดดังนี้

1. หน่วยประมวลผลกลาง Intel core i7
2. หน่วยความจำ 4 GB
3. ฮาร์ดดิสก์ 1024 GB

ด้านซอฟต์แวร์

1. ระบบปฏิบัติการ Windows 7 64 bit
2. ตัวแปลภาษาซี (C Compiler)
3. ระบบจัดการฐานข้อมูลออราเคิล (Oracle 12g)
4. โปรแกรมจัดทำเอกสารประกอบการวิจัย

1.9 ประโยชน์ที่คาดว่าจะได้รับ

ได้เทคนิคในการเลือกสร้างดัชนี Bitmap Join Index ซึ่งสามารถนำไปใช้แก้ปัญหาในการเลือกดัชนีของคลังข้อมูล

บทที่ 2

ทฤษฎีที่เกี่ยวข้อง

บทนี้กล่าวถึง ทฤษฎีทั้งหมดที่เกี่ยวข้องในการวิจัย ได้แก่ คลังข้อมูล กระบวนการสอบถามข้อมูล การเพิ่มประสิทธิภาพการสอบถาม ค่าใช้จ่ายในการสร้างดัชนี และวิธีการแก้ปัญหาแบบพลวัต

2.1 คลังข้อมูล

คลังข้อมูล เป็นแหล่งข้อมูลขนาดใหญ่ขององค์กรที่มีการออกแบบโครงสร้างตามจุดมุ่งหมายของงาน เพื่อช่วยสนับสนุนการตัดสินใจของผู้บริหาร หรือผู้ที่ทำงานเกี่ยวกับการวิเคราะห์ข้อมูลให้เป็นไปอย่างรวดเร็ว ถูกต้อง และมีประสิทธิภาพยิ่งขึ้น โดยสนับสนุนการประมวลผลแบบ OLAP (On-Line Analytical Processing) ซึ่งฟังก์ชันการทำงานและความต้องการทางประสิทธิภาพนั้นแตกต่างอย่างสิ้นเชิงกับการประมวลผลแบบ OLTP (On-Line Transaction Processing) ที่ใช้ในฐานข้อมูลดำเนินการ (Chaudhuri and Dayal, 1997; Kimball and Ross, 2004)

ลักษณะงานของ OLTP มักจะเกี่ยวข้องกับงานด้านการจัดการเอกสาร เช่น การบันทึกข้อมูลสั่งซื้อ งานธุรกรรมของธนาคาร ซึ่งเป็นงานประจำวัน มีการทำงานแบบซ้ำๆ โดยงานประเภทนี้มักจะต้องการข้อมูลที่เป็นรายละเอียด ข้อมูลที่มีการอัพเดตเสมอ และการอ่านหรืออัพเดตข้อมูลโดยปกติแล้วจะเข้าถึงจากคีย์หลัก (Primary Key) ของข้อมูลนั้น ส่วนลักษณะงานของ OLAP มักจะต้องการข้อมูลสรุป (Summary Data) เช่น รายงานยอดขายประจำเดือน รายงานสรุปงบกำไรขาดทุน รายงานสินค้ายอดนิยมประจำเดือน เป็นต้น ซึ่งข้อมูลสรุปนี้อาจเป็นข้อมูลในอดีต หรือข้อมูลปัจจุบันก็ได้ และจะให้ความสำคัญกับข้อมูลสรุปมากกว่ารายละเอียดอื่นๆ ของข้อมูล

คลังข้อมูลเป็นแหล่งข้อมูลกลางที่เก็บรวบรวมข้อมูลจากฐานข้อมูลเชิงปฏิบัติการอื่นๆ ในองค์กรขนาดใหญ่ขนาดของคลังข้อมูลย่อมมีขนาดใหญ่ระดับกิกะไบท์ (Gigabytes) ถึงเทระไบท์ (Terabytes) ลักษณะของการสอบถามข้อมูลส่วนใหญ่จะเป็นแบบทันทีทันใด มักจะเป็นการสอบถามข้อมูลจากหลายตาราง และมีเงื่อนไขในการสอบถามที่ซับซ้อน ในการเข้าถึงข้อมูลแต่ละครั้งต้องเข้าถึงเป็นระดับหลายล้านเรคอร์ด (Records) ทำให้ใช้เวลาในการสอบถามข้อมูลนาน ดังนั้นการเพิ่มความรวดเร็วในการเข้าถึงข้อมูลจึงเป็นปัญหาที่สำคัญ

2.1.1 คุณสมบัติของคลังข้อมูล

คุณสมบัติที่สำคัญของคลังข้อมูลประกอบด้วย 1) การแบ่งโครงสร้างตามเนื้อหา 2) การรวมเป็นหนึ่ง 3) ความเสถียรของข้อมูลและ 4) ความสัมพันธ์กับเวลา ซึ่งสามารถอธิบายรายละเอียดได้ดังนี้ (Inmon, 2002)

1) การแบ่งโครงสร้างตามเนื้อหา (Subject Oriented) หมายถึง คลังข้อมูลจะถูกออกแบบมาโดยคำนึงถึงเนื้อหาที่สนใจ และจัดเก็บเฉพาะข้อมูลที่จำเป็นต่อกระบวนการตัดสินใจเท่านั้น ไม่ได้เน้นกระบวนการทำงานเหมือนฐานข้อมูลดำเนินการ

2) การรวมเป็นหนึ่ง (Integrated) หมายถึง คลังข้อมูลเป็นการรวบรวมข้อมูลจากแหล่งต่างๆ ทั้งภายในและภายนอกองค์กร จากหลายฐานข้อมูลดำเนินการที่มีคุณสมบัติแตกต่างกันให้มารวมอยู่ที่เดียวกัน ดังนั้นจะต้องทำให้ข้อมูลมีความสอดคล้องกัน หรือมีมาตรฐานเดียวกัน เช่น ข้อมูลชนิดเดียวกันควรมีรูปแบบการเก็บข้อมูลเหมือนกัน เป็นต้น

3) ความเสถียรของข้อมูล (Nonvolatile) หมายถึง ข้อมูลในคลังข้อมูลจะไม่มีการแก้ไข เปลี่ยนแปลง หลังจากที่นำเข้าสู่คลังข้อมูลแล้ว ผู้ใช้สามารถทำได้เพียงเข้าถึงข้อมูลและเพิ่มข้อมูลเท่านั้น

4) ความสัมพันธ์กับเวลา (Time Variant) หมายถึง การเก็บข้อมูลในคลังข้อมูลจะเป็นการเก็บข้อมูลย้อนหลังไปหลายๆ ปี โดยมีการกำหนดช่วงเวลาเอาไว้ เพราะกระบวนการตัดสินใจด้านการบริหารนั้น จำเป็นต้องมีข้อมูลเปรียบเทียบด้านเวลาเพื่อนำมาวิเคราะห์ เปรียบเทียบ และหาแนวโน้มของข้อมูล

2.1.2 สถาปัตยกรรมของคลังข้อมูล

สถาปัตยกรรมของคลังข้อมูล ประกอบด้วยส่วนที่สำคัญ ได้แก่ แหล่งข้อมูล หน่วยเก็บข้อมูล หน่วยเก็บข้อมูลข้อเท็จจริงเพื่อการอธิบายข้อมูล OLAP Server และเครื่องมือสำหรับผู้ใช้ ดังแสดงในภาพประกอบ 2-1 ซึ่งมีรายละเอียดดังนี้ (Chaudhuri and Dayal, 1997)

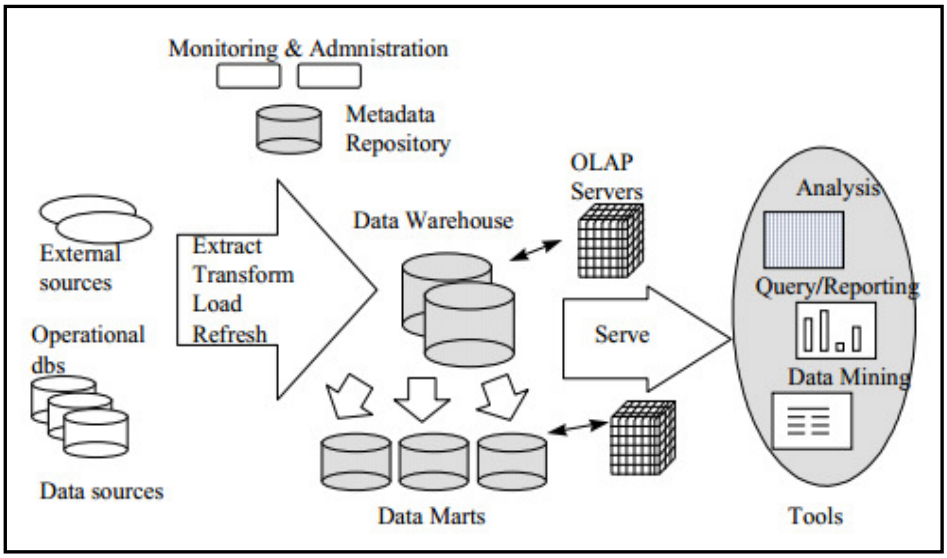
1. แหล่งข้อมูล (Data Source) เป็นส่วนที่เกี่ยวข้องกับเครื่องมือที่ใช้ในการนำข้อมูลจากแหล่งข้อมูลหลายๆ แหล่ง เช่น ฐานข้อมูลดำเนินการ และแหล่งข้อมูลภายนอก (External Sources) เพื่อใช้ในการเตรียมข้อมูลเข้าสู่คลังข้อมูล

2. หน่วยเก็บข้อมูล (Data Storage) ในคลังข้อมูลหลักอาจประกอบด้วยคลังข้อมูลย่อย (Data Marts) แบ่งเป็นหลายๆ ส่วน โดยมีเครื่องเซิร์ฟเวอร์ที่ใช้ในการเก็บและจัดการข้อมูล

3. หน่วยเก็บข้อมูลข้อเท็จจริงเพื่อการอธิบายข้อมูล (Metadata Repository) เป็นส่วนที่ใช้ในการจัดเก็บข้อมูลที่ใช้ในการอธิบายข้อมูล และควบคุมคลังข้อมูล ใช้สำหรับการดำเนินการต่างๆ กับข้อมูล ตามกระบวนการฐานข้อมูล เช่น รูปแบบของข้อมูล ข้อจำกัดของข้อมูล ข้อมูลแต่ละตัวมาจากแหล่งใด ถูกนำเข้าด้วยวิธีการใด เป็นต้น

4. OLAP Server ทำหน้าที่ในการจัดการข้อมูล และการเพิ่มประสิทธิภาพในการประมวลผล เช่น การทำดัชนี การทำ Materialize View เป็นต้น

5. เครื่องมือสำหรับผู้ใช้ (Front-End Tools) เป็นส่วนที่ประกอบด้วยเครื่องมือในการวิเคราะห์ข้อมูล เครื่องมือสำหรับการสอบถามข้อมูลและการรายงานเครื่องมือสำหรับการทำเหมืองข้อมูล เป็นต้น

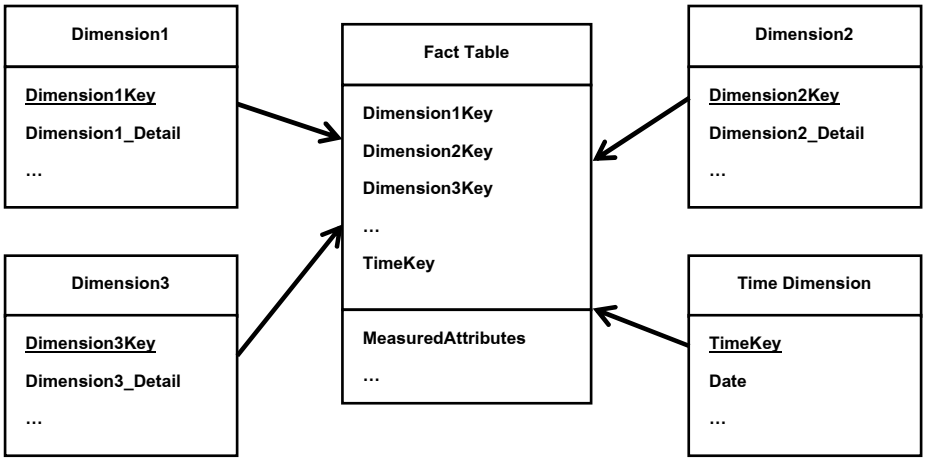


ภาพประกอบ 2-1 สถาปัตยกรรมของคลังข้อมูล (Chaudhuri and Dayal, 1997)

2.1.3 โครงสร้างของคลังข้อมูล

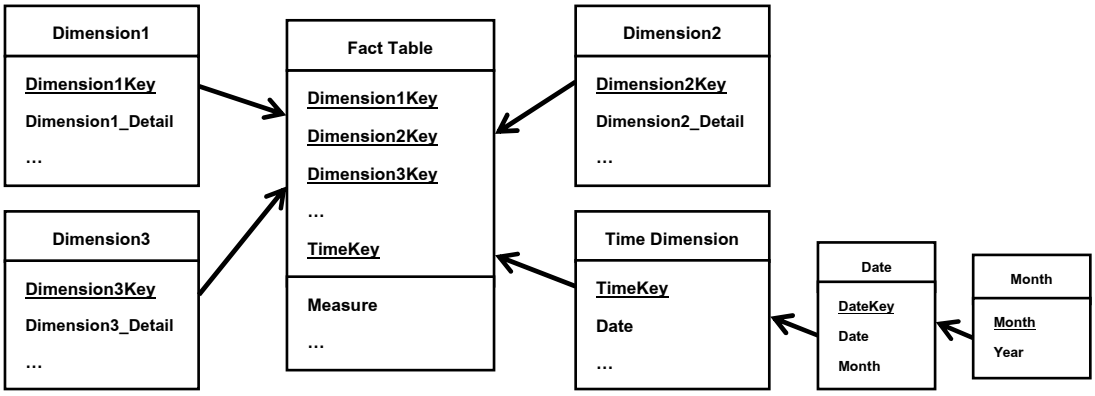
คลังข้อมูลส่วนใหญ่จะใช้โครงสร้างแบบ Star Schema (Gupta *et al*., 1997) ประกอบไปด้วยตาราง Fact ซึ่งเป็นตารางหลัก และตาราง Dimension หลายๆ ตาราง ซึ่งจะเก็บรายละเอียดเพื่ออธิบายข้อมูลในตาราง Fact และจะต้องมี Time Dimension เสมอเนื่องจากการสอบถามข้อมูลในคลังข้อมูลนั้นจะมีการเปรียบเทียบผลลัพธ์ในแต่ละช่วงเวลา ซึ่งสามารถนำไปใช้ประโยชน์ในการวิเคราะห์แนวโน้มขององค์กรในอนาคตได้ สำหรับแต่ละแอททริบิวต์ของตาราง Fact จะประกอบไปด้วยคีย์นอก (Foreign Key) ที่เชื่อมโยงกับคีย์หลักของตาราง Dimension และแอททริบิวต์ที่เกี่ยวข้องกับตัวเลขเชิงปริมาณของค่าต่างๆ (Measured Attributes) ส่วนตาราง Dimension ประกอบไปด้วยแอททริบิวต์ต่างๆ ที่สอดคล้องกับ Dimension นั้น ดังภาพประกอบ 2-2

การเพิ่มข้อมูลในคลังข้อมูลส่วนใหญ่จะมีการเพิ่มข้อมูลเข้าสู่ตาราง Fact ดังนั้นตาราง Fact จึงมีขนาดใหญ่มาก ส่วนตาราง Dimension จะมีการเพิ่มข้อมูลน้อยมากดังนั้นจึงมีขนาดเล็กและค่อนข้างคงที่



ภาพประกอบ 2-2 โครงสร้างแบบ Star Schema

โครงสร้างอีกประเภทหนึ่งคือ Snowflake Schema จะมีการเพิ่มรายละเอียดมากกว่าโครงสร้างแบบ Star Schema โดยการทำรูปแบบบรรทัดฐาน (Normalization) บนตาราง Dimension ดังแสดงในภาพประกอบ 2-3 ซึ่งมีข้อดีคือ ช่วยขจัดความซ้ำซ้อนของข้อมูล ทำให้ใช้พื้นที่ในการเก็บข้อมูลลดลง ข้อเสียคือ เมื่อต้องการสอบถามข้อมูลก็ต้องเสียเวลาในการจับคู่ข้อมูลเพิ่มมากขึ้น



ภาพประกอบ 2-3 โครงสร้างแบบ Snowflake Schema

2.1.4 ลักษณะการสอบถามข้อมูลในคลังข้อมูล

ในคลังข้อมูลประกอบด้วยข้อมูลเป็นจำนวนมากในระดับหลายล้านเรคอร์ดซึ่งการสอบถามข้อมูลมีลักษณะแบบทันทีทันใด ขึ้นอยู่กับความต้องการของผู้ใช้ โดยส่วนใหญ่เป็นการสอบถามเพื่อให้ได้ข้อมูลสรุป ซึ่งจะต้องมีการใช้ข้อมูลจากหลายตาราง และมีเงื่อนไขในการ

สอบถามที่ซับซ้อน จากลักษณะดังที่กล่าวมานั้นทำให้ใช้เวลาในการประมวลผลการสอบถามข้อมูลนาน ดังนั้นจึงจำเป็นต้องมีการใช้เทคนิคต่างๆ เข้ามาช่วยเพิ่มประสิทธิภาพในการสอบถามข้อมูล โดยวิธีการที่นิยมใช้ เช่น การประมวลผลการสอบถามแบบขนาน หรือ Parallel Query Processing เป็นการเพิ่มอุปกรณ์ฮาร์ดแวร์เข้ามาในระบบเพื่อช่วยในการลดเวลาในการสอบถาม การสร้างผลลัพธ์ของคำสั่งสอบถามเอาไว้ล่วงหน้า หรือ Pre-computed Structure เป็นการเตรียมผลลัพธ์ในของคำสั่งสอบถามที่เคยถูกเรียกใช้มาแล้ว เมื่อคำสั่งนั้นถูกเรียกใช้อีกครั้งก็จะให้ผลลัพธ์ได้อย่างรวดเร็วเนื่องจากไม่ต้องมีการเข้าถึงข้อมูลในตารางจริง เช่น การสร้าง Materialized View เป็นต้น ซึ่งจะยกตัวอย่างลักษณะของโครงสร้างคลังข้อมูล โดยใช้โครงสร้างคลังข้อมูลการสั่งซื้อสินค้า ซึ่งแสดงดังภาพประกอบ 2-4 และลักษณะการสอบถามข้อมูลของคลังข้อมูล แสดงได้ดังภาพประกอบ 2-5

Customer Table

cid	name	gender	state
1	Brian	male	California
2	Carol	female	California
3	Charles	male	New York
4	Daniel	male	Pennsylvania
5	Donna	female	Arizona

Order Table (Fact)

id	cid	pid	datekey	unit
1	1	1	19920327	10
2	2	1	19920327	20
3	3	2	19920328	30
4	4	2	19920328	40
5	5	3	19920328	50
6	1	3	19920401	60
7	2	4	19920401	70
8	3	4	19920402	80
9	4	5	19920402	90
10	5	5	19920403	100

Time Table

datekey	date	month	year
19920327	27	03	1992
19920328	28	03	1992
19920401	01	04	1992
19920402	02	04	1992
19920403	03	04	1992

Product Table

pid	pname	price	psupp	state
1	can	10	S1	New York
2	box	20	S1	New York
3	paper	30	S2	California
4	pencil	40	S2	California
5	pen	50	S3	Arizona

ภาพประกอบ 2-4 ตัวอย่างโครงสร้างของคลังข้อมูลการสั่งซื้อ

จากภาพประกอบ 2-4 ตัวอย่างโครงสร้างคลังข้อมูลการสั่งซื้อสินค้า ประกอบด้วย ตาราง Fact จำนวน 1 ตาราง ได้แก่ ตาราง Order และตาราง Dimension จำนวน 3 ตาราง ได้แก่ ตาราง Customer ตาราง Product และตาราง Time โดยที่ในตาราง Fact จะประกอบไปด้วยคีย์นอกซึ่งเชื่อมโยงกับคีย์หลักของตาราง Dimension ได้แก่ แอททริบิวต์ cid, pid และ datekey

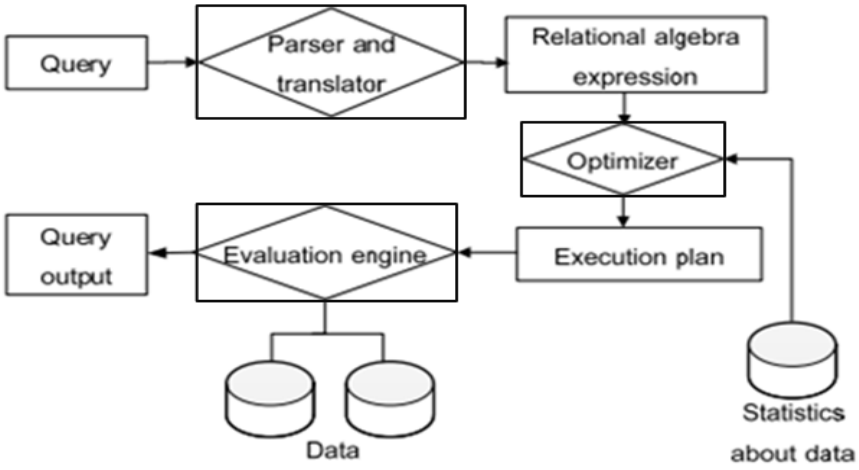
```
SELECT count(*)
FROM Order O,Customer C,Time T
WHERE O.cid=C.cid
        AND O.datekey=T.datekey
        AND C.state="California"
        AND C.gender="male"
        AND T.month="04"
        AND T.year="1992"
```

ภาพประกอบ 2-5 ตัวอย่างการสอบถามข้อมูลที่มีการจับคู่ข้อมูลระหว่างตาราง

ดังที่กล่าวไปแล้วข้างต้นว่าโครงสร้างของคลังข้อมูลส่วนใหญ่ ใช้โครงสร้างแบบ Star Schema ซึ่งการสอบถามข้อมูลจะต้องมีการจับคู่ข้อมูลระหว่างตาราง Fact และตาราง Dimension เช่น จากคำสั่งสอบถามข้อมูลในภาพประกอบ 2-5 หากผู้ใช้ต้องการทราบว่ามีลูกค้า (Customer) ที่มาจากรัฐ (State) “California” ที่เป็นเพศชาย (Male) และมีการสั่งสินค้าในเดือนเมษายน ปี 1992 มีจำนวนทั้งหมดกี่รายการ ซึ่งไม่สามารถหาคำตอบได้จากตาราง Order เพียงตารางเดียว ดังนั้นจึงต้องมีการจับคู่ข้อมูลระหว่างตาราง Order ตาราง Customer และ ตาราง Time เพื่อให้ได้ผลลัพธ์ที่ต้องการ

2.2 กระบวนการสอบถามข้อมูล (Query Processing)

กระบวนการสอบถามข้อมูล (Silberschatz*et al.*, 2011) เป็นกระบวนการที่เกี่ยวข้องกับการหาผลลัพธ์ของการสอบถามข้อมูล เพื่อให้ได้คำตอบที่ตรงกับความต้องการของผู้ใช้ โดยจะมีการเลือกวิธีการในการประมวลผลที่รวดเร็วและประหยัดค่าใช้จ่ายที่สุด ซึ่งประกอบด้วย 3 ขั้นตอนหลักได้แก่ 1) Parsing & Translation 2) Query Optimization และ 3) Evaluation ดังภาพประกอบ 2-6



ภาพประกอบ 2-6 กระบวนการสอบถามข้อมูล (Silberschatz et al., 2011)

สำหรับรายละเอียดการทำงานของแต่ละขั้นตอนสามารถอธิบายได้ ดังนี้

2.2.1 Parsing& Translation

เป็นขั้นตอนในการแปลงคำสั่งสอบถามให้อยู่ในรูปที่ตัวประมวลผลสอบถาม (Query Processor) สามารถเข้าใจได้ โดยมี Parser ทำหน้าที่ในการตรวจสอบไวยากรณ์ (Syntax) ของคำสั่งสอบถามข้อมูลในรูปคำสั่ง SQL ความสมบูรณ์ของความสัมพันธ์ (Relation) ระหว่างตาราง แอททริบิวต์ต่างๆ และตรวจสอบความยินยอมในการเข้าถึงข้อมูล ส่วน Translator ทำหน้าที่แปลงคำสั่งสอบถามข้อมูลซึ่งเป็นภาษาระดับสูง (High-level language) ให้อยู่ในรูป Parse Tree เพื่อใช้แทนคำสั่งสอบถาม จากนั้นก็จะแปลงให้อยู่ในรูปพีชคณิตเชิงสัมพันธ์ (Relational Algebra) ซึ่งเป็นรูปแบบภาษาการสอบถามที่มีการกำหนดขั้นตอนการทำงานอย่างชัดเจน (Procedural Query Language)

จากตัวอย่างคำสั่งสอบถาม (SQL Query) ในภาพประกอบ 2-7 สามารถแปลงให้อยู่ในรูปพีชคณิตเชิงสัมพันธ์ในรูปแบบต่างๆ ได้ดังภาพประกอบ 2-8

SQL Query

SELECT C.name

FROM Order O, Customer C

WHEREO.cid=C.cid AND C.state=”New York”;

ภาพประกอบ 2-7 ตัวอย่างคำสั่งสอบถาม

แบบที่ 1 $\prod_{C.name}(\sigma_{C.state="New\ York"}(O \bowtie C))$

แบบที่ 2 $\prod_{C.name}(\sigma_{O.cid=C.cid}((\sigma_{C.state="New\ York"}C)\times O))$

แบบที่ 3 $\prod_{C.name}((\sigma_{C.state="New\ York"}C) \bowtie O)$

ภาพประกอบ 2-8 Relational Algebra

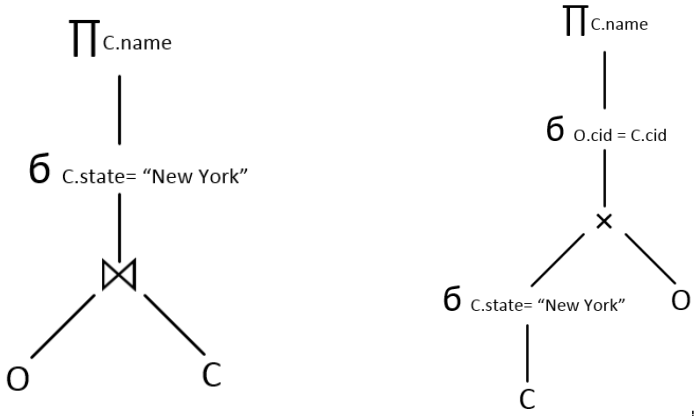
จากภาพประกอบ 2-8 การสร้างคำสั่งสอบถามในรูปแบบพีชคณิตเชิงสัมพันธ์ สามารถเขียนได้หลายรูปแบบ ซึ่งจะให้ค่าผลลัพธ์ในการสอบถามที่เหมือนกัน

2.2.2 Query Optimizer

เนื่องจากคำสั่งสอบถามในรูปแบบพีชคณิตเชิงสัมพันธ์ สามารถสร้างได้หลายแบบโดยให้ค่าผลลัพธ์เดียวกัน ดังนั้น Optimizer จะทำหน้าที่เลือกวิธีการเข้าถึงข้อมูลที่ใช้ทรัพยากรน้อยที่สุดหรือประหยัดเวลาที่สุด โดยวิธีการที่นิยมใช้คือ การแปลงคำสั่งสอบถามในรูปแบบพีชคณิตเชิงสัมพันธ์ให้อยู่ในรูปแบบโครงสร้างข้อมูลแบบกราฟ ที่เรียกว่า Operator Graph ซึ่งแสดงดังภาพประกอบ 2-9 จากนั้นคำนวณค่าใช้จ่ายในการประมวลผลจากกราฟที่ได้

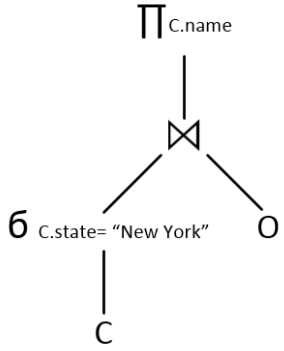
2.2.3 Evaluation

ใช้สำหรับการประมวลผลคำสั่งจาก Query Execution Plan หรือ Query Evaluation Plan ที่ใช้เวลาการประมวลผลน้อยที่สุด เพื่อให้ได้ผลลัพธ์ที่ต้องการ



(ก) Operator Graph แบบที่ 1 (ข) Operator Graph แบบที่ 2

ภาพประกอบ 2-9 ตัวอย่าง Operator Graph



(ค) Operator Graph แบบที่ 3

ภาพประกอบ 2-9 ตัวอย่าง Operator Graph (ต่อ)

2.3 การเพิ่มประสิทธิภาพการสอบถาม (Query Optimization)

วิธีการในการเพิ่มประสิทธิภาพการสอบถามสามารถแบ่งได้เป็น 2 กลุ่มใหญ่ๆ ได้แก่ วิธีการที่ไม่ต้องใช้ Pre-computed Structure และ วิธีการที่ใช้ Pre-computed Structure ซึ่งสามารถอธิบายรายละเอียดได้ดังนี้

2.3.1 วิธีการที่ไม่ต้องใช้ Pre-computed Structure

เป็นวิธีการในการเพิ่มประสิทธิภาพการสอบถามโดยไม่ต้องมีการสร้างผลลัพธ์หรือมีการคำนวณเอาไว้ล่วงหน้า (Wu and Buchman, 1997) ตัวอย่างวิธีการในกลุ่มนี้ ได้แก่ Hash Join, Nested Loop Join, Sort Merge Join เป็นต้น ซึ่งวิธีการในกลุ่มนี้เหมาะสำหรับการสอบถามแบบทันทีทันใด เนื่องจากไม่ต้องมีการคาดเดาว่าผู้ใช้ต้องการค้นหาอะไร

ข้อดี ของวิธีการที่ไม่ต้องใช้ Pre-computed Structure

- ไม่เสียพื้นที่ในการจัดเก็บผลลัพธ์ที่ต้องการคำนวณไว้ล่วงหน้า
- ไม่เสียค่าใช้จ่ายในการบำรุงรักษาตารางผลลัพธ์ที่คำนวณไว้แล้ว ในกรณีที่มีการเพิ่มข้อมูลในตารางจริง
- เหมาะสำหรับการสอบถามแบบทันทีทันใด
- ในกรณีที่มีการจับคู่ข้อมูลระหว่างตารางที่มีความซับซ้อนมากๆ จะช่วยลดเวลาในการสอบถามได้เป็นอย่างดี

ข้อเสีย ของวิธีการที่ไม่ต้องใช้ Pre-computed Structure

- ใช้เวลาในการสอบถามมากกว่าวิธีการที่ใช้ Pre-computed Structure เนื่องจากต้องใช้เวลาคำนวณในระหว่างการสอบถามมาก

2.3.2 วิธีการที่ใช้ Pre-computed Structure

เป็นวิธีการในการเพิ่มประสิทธิภาพการสอบถามซึ่งมีการให้ผลลัพธ์บางส่วนหรือทั้งหมดโดยไม่ต้องผ่านการอ่านฐานข้อมูล (Base Data) ก่อน แต่ต้องมีการปรับปรุงโครงสร้างใหม่เมื่อมีการเปลี่ยนแปลงข้อมูลในฐานข้อมูล ตัวอย่างวิธีการในกลุ่มนี้ ได้แก่ 1) Materialized View (หรือ Summary Tables) และ 2) การสร้างดัชนี เช่น Bitmap Index, B-Tree, Join Index และ Bitmap Join Index เป็นต้น

1) Materialized View เป็นการสร้างตารางเก็บผลลัพธ์ไว้ล่วงหน้า ตามคำสั่งสอบถามที่ถูกเรียกใช้บ่อยเมื่อผู้ใช้เรียกใช้คำสั่งเดิมซ้ำอีกก็ไม่ต้องไปค้นหาข้อมูลจากตารางจริง หากผู้ใช้สอบถามตรงกับคำสั่งที่สร้างไว้ก็จะช่วยลดเวลาในการเข้าถึงข้อมูลได้เป็นอย่างดี

ข้อดีของ Materialized View

- ให้ผลลัพธ์ในการค้นหาได้เร็วมาก เนื่องจากมีการเก็บตารางผลลัพธ์ไว้ก่อนแล้ว ไม่จำเป็นต้องค้นหาข้อมูลในตารางจริง

ข้อเสียของ Materialized View

- เปลืองพื้นที่ในการเก็บข้อมูลมาก เพราะหากต้องการความรวดเร็วในการสอบถาม ก็ต้องสร้างตารางผลลัพธ์ไว้สำหรับทุก ๆ การสอบถามที่เป็นไปได้
- เสียค่าใช้จ่ายในการบำรุงรักษามาก ในกรณีที่มีการเพิ่มข้อมูลในตารางจริง ก็ต้องมีการปรับปรุงตาราง Materialized View ใหม่
- ไม่สามารถสร้าง Materialized View สำหรับทุกคำสั่งสอบถามที่เป็นไปได้ เนื่องจากพื้นที่ในการจัดเก็บข้อมูลมีอยู่อย่างจำกัด
- ไม่เหมาะสำหรับการสอบถามแบบทันทีทันใด

2) การสร้างดัชนี เป็นการเพิ่มประสิทธิภาพการสอบถามซึ่งจะช่วยลดเวลาในการเข้าถึงข้อมูลได้เป็นอย่างดี โดยการเลือกแอททริบิวต์ใดแอททริบิวต์หนึ่งในตารางมาสร้างดัชนี และในแต่ละตารางสามารถมีได้หลายดัชนี สำหรับการเข้าถึงข้อมูลนั้นก็จะเข้าถึงได้จากค่าข้อมูลของดัชนี (Key) ซึ่งเป็นค่าของแอททริบิวต์ที่ใช้สร้างดัชนี เพื่อชี้ไปยังแถวข้อมูลที่ต้องการในตาราง ดังนั้นเมื่อมีการสอบถามข้อมูลจึงไม่จำเป็นต้องอ่านข้อมูลทั้งตาราง และการสร้างดัชนียังเหมาะสมกับลักษณะการสอบถามแบบทันทีทันใด เนื่องจากโอกาสที่แอททริบิวต์ที่สร้างดัชนีจะตรงกับแอททริบิวต์ที่ถูกสอบถามนั้นมีมาก

สำหรับวิธีการสร้างดัชนีนั้น สามารถแบ่งได้เป็น การสร้างดัชนีบนตารางเดียว และ การสร้างดัชนีบนหลายตาราง ซึ่งสามารถอธิบายได้ดังนี้

2.1) การสร้างดัชนีบนตารางเดียว เป็นการสร้างดัชนีบนแอททริบิวต์ที่อยู่ในตารางเพียงตารางเดียว ตัวอย่าง เช่น B-Tree Index และ Bitmap Index เป็นต้น ซึ่งปัจจัยที่

จะนำมาพิจารณาในการเลือกสร้างดัชนีชนิดต่างๆ ได้แก่ จำนวนคาร์ดินอลิตี้ (Cardinality) ของแอททริบิวต์ที่จะนำมาสร้างดัชนี ลักษณะการสอบถามข้อมูล เป็นต้น โดยสามารถอธิบายรายละเอียดของดัชนีแต่ละชนิดได้ ดังนี้

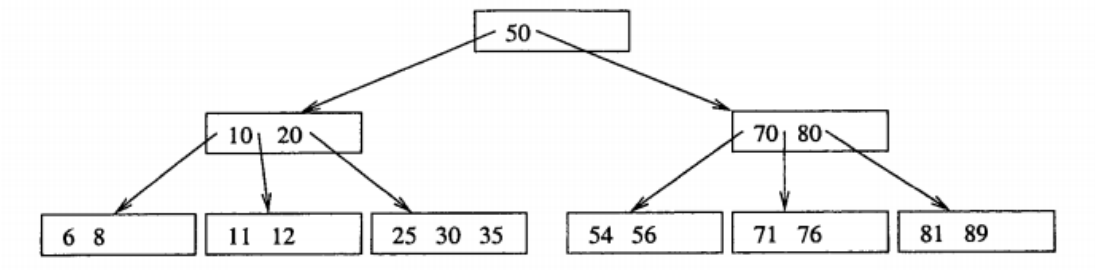
2.1.1) B-tree Index เป็นดัชนีที่ถูกเรียกใช้โดยอัตโนมัติสำหรับฐานข้อมูลเชิงสัมพันธ์ส่วนใหญ่ (Vanichayobon and Gruenwald, 1999) มีลักษณะเป็นต้นไม้ ซึ่งประกอบด้วยโหนดที่อยู่ลำดับบนสุดของต้นไม้ (Top Level) เรียกว่า โหนดราก (Root Node) ส่วนโหนดที่อยู่ลำดับล่างสุดของต้นไม้ เรียกว่า โหนดใบ (Leaf Node) ส่วนโหนดในระดับอื่นๆ เรียกว่า โหนดกิ่ง (Branches Node) ซึ่งในแต่ละคีย์ของโหนดรากและโหนดกิ่งนั้น จะประกอบไปด้วยตัวชี้ (Pointer) ที่ชี้ไปยังโหนดในระดับถัดไป ดัชนีชนิดนี้จะให้ประสิทธิภาพที่ดีเมื่อจำนวนผลลัพธ์ของการสอบถามมีจำนวนน้อย ตัวอย่างของการสร้างดัชนีแบบ B-tree แสดงได้ดังภาพประกอบ 2-10

ข้อดีของ B-tree Index

- เหมาะสำหรับข้อมูลที่มีคาร์ดินอลิตี้สูง เช่น ชื่อ, นามสกุล เป็นต้น
- เหมาะกับการสอบถามข้อมูลที่มีลักษณะเป็นช่วงค่าข้อมูล

ข้อเสียของ B-tree Index

- หากเป็นคอลัมน์ที่มีคาร์ดินอลิตี้ต่ำ เช่น เพศ จะช่วยลดจำนวนการเข้าถึงข้อมูลได้เพียงเล็กน้อย
- การดึงข้อมูลผลลัพธ์นั้นจะมีการเรียงลำดับตามค่าข้อมูลที่นำมาสร้างดัชนีซึ่งไม่ได้เรียงตามลำดับแถวข้อมูลของตารางข้อมูลจริง



ภาพประกอบ 2-10 ตัวอย่างดัชนีแบบ B-Tree (Ooi and Tan, 2002)

2.1.2) ดัชนีบิตแมป (Bitmap Index) เป็นดัชนีที่ช่วยเพิ่มความรวดเร็วในการค้นหาข้อมูล และสนับสนุนการดำเนินการระดับบิต (เช่น AND OR NOT XOR เป็นต้น) ระหว่างบิตแมปเวกเตอร์ (Bitmap Vector) ที่ลงรหัส ซึ่งเป็นประโยชน์ต่อการสอบถามข้อมูลที่มี

ความซับซ้อน โดยดัชนีบิตแมปแต่ละชนิดจะมีรูปแบบการลงรหัสที่แตกต่างกัน ตัวอย่างการสร้างดัชนีบิตแมป สามารถแสดงได้ดังภาพประกอบ 2-11

ข้อดีของดัชนีบิตแมป

- การแทนค่าข้อมูลง่ายกว่าดัชนีชนิดอื่นๆ
- สามารถเพิ่มประสิทธิภาพการสอบถามข้อมูลได้ดีเมื่อนำมาใช้กับข้อมูลที่มีคาร์ดินอลิตี้ต่ำ
- สามารถปรับปรุงประสิทธิภาพการสอบถามของคำสั่งที่มีความซับซ้อนได้เพียงการใช้การดำเนินการทางตรรกะ ซึ่งมีค่าใช้จ่ายต่ำ เช่น OR AND NOT XOR เป็นต้น

ข้อเสียของดัชนีบิตแมป

- ไม่เหมาะสำหรับข้อมูลที่มีคาร์ดินอลิตี้สูง
- ไม่เหมาะสำหรับคำสั่งสอบถามที่มีลักษณะเป็นช่วงค่าข้อมูล
- เมื่อมีการเพิ่มข้อมูลในตารางจริง ก็จำเป็นต้องสร้างดัชนีขึ้นมาใหม่

Customer Table

cid	name	gender	state
1	Brian	male	California
2	Carol	female	California
3	Charles	male	New York
4	Daniel	male	Pennsylvania
5	Donna	female	Arizona

Bitmap Index on Customer.gender

male	female
1	0
0	1
1	0
1	0
0	1

ภาพประกอบ 2-11 ตัวอย่างดัชนีบิตแมป

จากภาพประกอบ 2-11 เป็นการสร้างดัชนีบิตแมป ในตาราง Customer บนแอททริบิวต์เพศ โดยใช้โครงสร้างคลังข้อมูลจากภาพประกอบ 2-4 เนื่องจากแอททริบิวต์เพศมีจำนวนคาร์ดินอลิตี้เท่ากับ 2 นั่นคือค่า Male และ Female ดังนั้นจะใช้จำนวนบิตแมปเวกเตอร์ (Bitmap Vector) ในการสร้างดัชนีเท่ากับ 2

2.2) การสร้างดัชนีบนหลายตาราง

2.2.1) Join Index เป็นการสร้างดัชนีโดยการรวมตารางตั้งแต่ 2 ตารางขึ้นไปที่มีข้อมูลของแอททริบิวต์ที่นำมาจับคู่ข้อมูลในแต่ละแถวเหมือนกันโดยเลือกเฉพาะแอททริบิวต์ที่เป็นคีย์หลักของแต่ละตาราง มาสร้างตารางดัชนีเพื่ออ้างถึงแถวข้อมูลในตารางจริง (Valduriez, 1987) ดังตัวอย่างในภาพประกอบ 2-12

Join Index for Customer and Product
on attribute state

cid	pid
1	3
1	4
2	3
2	4
3	1
3	2
5	5

ภาพประกอบ 2-12 ตัวอย่างดัชนีแบบ Join Index

จากภาพประกอบ 2-12 เป็นการสร้างดัชนีแบบ Join Index ซึ่งมีการจับคู่ข้อมูลระหว่างตาราง Customer และตาราง Product บนแอททริบิวต์ State โดยใช้โครงสร้างคลังข้อมูลจากภาพประกอบ 2-4

ข้อดีของ Join Index

- เพิ่มความรวดเร็วในการเข้าถึงข้อมูล เนื่องจากไม่ต้องมีการค้นหาข้อมูลจากตารางจริงก่อน หากมีคำสั่งสอบถามที่ต้องการข้อมูลจากการจับคู่ข้อมูล ก็สามารถไปค้นหาข้อมูลจากตารางดัชนีที่สร้างตามแอททริบิวต์ที่นำมาจับคู่กันได้ทันที
- เหมาะสำหรับคำสั่งสอบถามที่มีจำนวนแถวผลลัพธ์ (Selectivity) ของการสอบถามน้อย ๆ และจำนวนแถวที่ซ้ำกันมีปริมาณน้อย

ข้อเสียของ Join Index

- ต้องมีการอ่านข้อมูลจากทั้งสองตาราง
- หากผลลัพธ์ของการสอบถามมีจำนวนแถวมากก็ต้องมีการค้นหาข้อมูลจากตารางดัชนีหลาย ๆ แถวก็จะใช้เวลานาน

2.2.2) Bitmap Join index เป็นการสร้างดัชนีบิตแมปสำหรับการสอบถามข้อมูลจากตารางตั้งแต่ 2 ตารางขึ้นไป โดยการแทนค่าด้วยบิต 1 เมื่อค่าข้อมูลในแถวที่ i ตรงกับค่าของบิตแมปเวกเตอร์ i ส่วนที่เหลือจะถูกแทนด้วยบิต 0 และหากมีเงื่อนไขในการสอบถามมากกว่า 1 เงื่อนไขสามารถใช้การดำเนินการระดับบิตได้ ตัวอย่างของ Bitmap Join Index สามารถแสดงได้ดังภาพประกอบ 2-13

Bitmap Join Index for Order and Customer
on Customer.state

California	New York	Pennylvania	Arizona
1	0	0	0
1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1
1	0	0	0
1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

ภาพประกอบ 2-13 ตัวอย่างดัชนีแบบ Bitmap Join Index

จากภาพประกอบ 2-13 เป็นการสร้างดัชนี Bitmap Join Index บนตาราง Order ซึ่งจับคู่ข้อมูลกับตาราง Customer บนแอททริบิวต์ State โดยใช้โครงสร้างคลังข้อมูลจากภาพประกอบ 2-4

ข้อดีของ Bitmap Join Index

- เหมาะกับข้อมูลที่มีคาร์ดินอลิตี้ต่ำ
- การสอบถามข้อมูลมีความรวดเร็ว
- สามารถใช้กับข้อมูลที่มีคำสั่งสอบถามที่ซับซ้อนได้เป็นอย่างดี

ข้อเสียของ Bitmap Join Index

- ไม่เหมาะสำหรับข้อมูลที่มีคาร์ดินอลิตี้สูง
- ยังใช้พื้นที่ได้ไม่เต็มประสิทธิภาพ เนื่องจากมีการแทนค่าด้วยบิต 1 เมื่อข้อมูลในแถวที่ i ตรงกับค่าของบิตแมปเวกเตอร์ i ส่วนที่เหลือจะถูกแทนด้วยบิต 0 ดังนั้น หากข้อมูลมีการกระจายสูง ก็จะมีบิต 0 ในปริมาณที่มาก

2.4 ค่าใช้จ่ายในการสร้างดัชนี

ในการสร้างดัชนีนั้น สิ่งที่ต้องคำนึงถึงมากที่สุด นอกเหนือจากประสิทธิภาพที่ต้องการ คือ ต้นทุนค่าใช้จ่ายที่ใช้ในการสร้างดัชนี เนื่องจากในบางครั้งแต่ละระบบมีข้อจำกัดด้านทรัพยากรที่แตกต่างกัน ดังนั้น จึงไม่สามารถสร้างดัชนีสำหรับทุกๆ การสอบถามที่เป็นไปได้ด้วยเงื่อนไขที่มีอยู่อย่างจำกัด ดังนั้น จึงมีการคิดค้นวิธีการในการสร้างดัชนีที่ทำให้เกิดค่าใช้จ่ายน้อยที่สุด และยังคงประสิทธิภาพในด้านการสอบถามข้อมูล โดยมีการศึกษาและวิเคราะห์ปัจจัยที่มีผลต่อค่าใช้จ่ายของการสร้างดัชนี

ตาราง 2-1 สัญลักษณ์ที่ใช้สำหรับคำนวณค่าใช้จ่ายในการสร้างดัชนี

สัญลักษณ์	คำอธิบาย
$\|X\|$	จำนวนแถวข้อมูลในตาราง X
$\|C_x\|$	จำนวนคาร์ดินอลิตี้ของแอททริบิวต์ X
P_x	จำนวน Pages ที่ใช้ในการเก็บข้อมูลของตาราง X
S_p	ขนาดของ Disk Page (bytes)
S	ขนาดของพื้นที่ที่ใช้ในการเก็บข้อมูล (bytes)

1) ขนาดของพื้นที่ที่ใช้ในการเก็บข้อมูลของดัชนี Bitmap Join Index สามารถคำนวณได้จากปัจจัยต่างๆ ดังต่อไปนี้ (Aouiche, 2005)

-จำนวนคาร์ดินอลิตี้ของแต่ละแอททริบิวต์

-จำนวนแถวข้อมูลของตาราง Fact (F)

สามารถแสดงได้ดังสมการ 2.1

$$S = \frac{|C_x||F|}{8 S_p} \quad \text{Pages} \tag{2.1}$$

2) ค่าใช้จ่ายในการบำรุงรักษา

- ค่าใช้จ่ายเมื่อมีการเพิ่มข้อมูลในตาราง Fact

เมื่อกำหนดให้สร้าง Bitmap Join Index บนแอททริบิวต์ A ที่มาจาก Dimension T ในขั้นแรกจะต้องมีการอ่านข้อมูลในตาราง T ทั้งหมด (P_T) เพื่อหาว่าข้อมูลในแถวใดบ้างที่สามารถ จับคู่ข้อมูลกับข้อมูลในตาราง Fact ได้ และจะต้องมีการอัพเดตข้อมูลในตารางดัชนีทั้งหมด ซึ่งอย่างมากที่สุด ข้อมูลในตารางดัชนีทั้งหมดจะต้องถูกอ่านใหม่

ดังนั้นค่าใช้จ่ายทั้งหมด สามารถคำนวณได้จากสมการ 2.2

$$Cost_{maintenance} = P_T + \frac{|C_A||F|}{8 S_p} \quad \text{Pages} \tag{2.2}$$

- ค่าใช้จ่ายเมื่อมีการเพิ่มข้อมูลในตาราง Dimension

เมื่อกำหนดให้มีการเพิ่มข้อมูลในตาราง T จะเกิดกรณีที่เป็นไปได้อยู่ 2 กรณีนั่นคือ กรณีแรก ขอบเขตของแอททริบิวต์ A ไม่เพิ่มขึ้นส่วนอีกกรณีหนึ่งคือ ขอบเขตของแอททริ-

บิวต์ A จะเพิ่มขึ้น ในกรณีแรกจะต้องมีการอ่านข้อมูลในตาราง Fact (P_F) เพื่อค้นหาว่ามีข้อมูลในแถวใดบ้างที่สามารถจับคู่ข้อมูลกับข้อมูลในตาราง Dimension T ที่เพิ่มเข้ามาใหม่ได้ หลังจากนั้นก็จะต้องอัพเดตข้อมูลในตารางดัชนี สำหรับอีกกรณีหนึ่งที่ขอบเขตของแอททริบิวต์ A เพิ่มขึ้น ก็จะต้องมีการเพิ่มค่าใช้จ่ายในการสร้างตารางดัชนีใหม่

ดังนั้นค่าใช้จ่ายทั้งหมด สามารถคำนวณได้จากสมการ 2.3

$$Cost_{maintenance} = P_F + (1+\xi)\frac{|C_A||F|}{8S_p} \quad (2.3)$$

(เมื่อ ξ มีค่าเท่ากับ 1 เมื่อขอบเขตของแอททริบิวต์ A เพิ่มขึ้นและมีค่าเป็น 0 สำหรับกรณีอื่น ๆ

2.5 วิธีการแก้ปัญหาแบบพลวัต (Dynamic Programming)

เป็นวิธีการสำหรับแก้ปัญหาที่เป็นการตัดสินใจแบบหลายขั้นตอน ซึ่งในปัญหาย่อยๆนั้น อาจจะถูกเรียกใช้หลายๆ ครั้ง จึงนำเทคนิคนี้เข้ามาใช้โดยการเก็บผลลัพธ์ของปัญหาย่อยเอาไว้ในตาราง เมื่อถูกเรียกใช้ซ้ำอีกก็ไม่จำเป็นต้องมีการคำนวณใหม่ ตัวอย่างปัญหา เช่น ปัญหาการหาเส้นทางที่สั้นที่สุด (All-Pairs Shortest-Paths Problem) ปัญหาการเลือกสิ่งของที่เหมาะสมที่สุด (Knapsack Problem) เป็นต้น

2.5.1 ปัญหาการเลือกสิ่งของที่เหมาะสมที่สุด (The Knapsack Problem)

เป็นวิธีการในการหาชุดข้อมูลที่มูลค่ารวมสูงที่สุดและเหมาะสมกับพื้นที่ในการเก็บข้อมูลที่จำกัด (W) เมื่อมีข้อมูลที่เป็นตัวเลือกทั้งหมด n จำนวน ซึ่งคุณสมบัติของข้อมูลแต่ละตัวประกอบด้วยค่าน้ำหนักw_1, .., w_n และมูลค่าของข้อมูล v_1, .., v_n เมื่อต้องการพิจารณาข้อมูลลำดับที่ 1 ถึง i โดยที่ $1 \le i \le n$ พื้นที่ในการเก็บข้อมูลเท่ากับ j โดยที่ $1 \le j \le W$ และกำหนดให้ $F(i,j)$ คือค่าที่สูงที่สุดสำหรับข้อมูลดังกล่าว

ในการหาค่ามูลค่าสูงสุดของชุดข้อมูลซึ่งเป็นเซตย่อยของข้อมูลลำดับที่ 1 ถึง i ซึ่งใช้ขนาดพื้นที่ในการเก็บข้อมูลเท่ากับ j นั้นสามารถแบ่งได้เป็น 2 กรณี ดังนี้

1) เมื่อขนาดของ j น้อยกว่า w_i (ไม่สามารถใส่ข้อมูล i เพิ่มลงไปได้) ดังนั้นค่าของ $F(i,j)$ จะเท่ากับมูลค่าของชุดข้อมูลเมื่อไม่รวมข้อมูลลำดับที่ i ที่มีพื้นที่ในการเก็บข้อมูลเท่ากับ j นั่นคือ $F(i\text{-}1,j)$

2) เมื่อขนาดของ j มากกว่าหรือเท่ากับ w_i (สามารถใส่ข้อมูล i เพิ่มลงไปได้) จะเลือกข้อมูลที่มีมูลค่าสูงสุดระหว่าง มูลค่าของชุดข้อมูลเมื่อไม่รวมข้อมูลลำดับที่ i *นั่นคือค่าของ* $F(i\text{-}1,j)$ และมูลค่าของข้อมูลลำดับที่ i บวกกับมูลค่าของชุดข้อมูลเมื่อรวมข้อมูลลำดับที่ i ที่มีพื้นที่ในการเก็บข้อมูลเท่ากับ $j\text{-}w_i$ นั่นคือ $v_i + F(i\text{-}1, j\text{-}w_i)$

ดังนั้นสามารถแสดงการทำงานซึ่งเป็นความสัมพันธ์แบบเวียนเกิด (Recurrence Relation) ได้ดังสมการ 2.4

$$F(i,j)=\begin{cases}F(i-1,j) & \text{if } j-w_i<0\\ \max\left[F(i-1,j),v_i+F(i-1,j-w_i)\right] & \text{if } j-w_i\geq 0\end{cases} \quad (2.4)$$

ลักษณะตารางเก็บผลลัพธ์ของวิธีการแก้ปัญหาแบบพลวัตของ Knapsack Problem แสดงดังภาพประกอบ 2-14

		0	j-w_i	j	W
	0	0	0	0	0
w_i, v_i	i-1	0	$F(i\text{-}1, j\text{-}w_i)$	$F(i\text{-}1,j)$	
	i	0		$F(i,j)$	
	n	0			**goal**

ภาพประกอบ 2-14 ตารางผลลัพธ์ของ Knapsack Problem ด้วย
วิธีการแก้ปัญหาแบบพลวัต (Levitin, 2007)

ตัวอย่างการทำงานของวิธีการแก้ปัญหาแบบพลวัตสำหรับ Knapsack Problem เมื่อกำหนดให้พื้นที่ที่ใช้ในการเก็บข้อมูล เท่ากับ 5 ($W = 5$) และข้อมูลที่ต้องการคัดเลือกมีจำนวนเท่ากับ 4 ($n = 4$) ดังตาราง 2-2

ตาราง 2- 2 ตัวอย่างข้อมูลที่ใช้ในการอธิบายวิธีการแก้ปัญหา Knapsack Problem

item	weight	value
1	3	5
2	2	4
3	1	3
4	4	2

		j / i	0	1	2	3	4	5
w_i	v_i	0	0	0	0	0	0	0
3	5	1	0					
2	4	2	0					
1	3	3	0					
4	2	4	0					

(ก) กำหนดค่าเริ่มต้นให้$F(0,0..C)=0$ และ$F(0..n,0)=0$

		j / i	0	1	2	3	4	5
w_i	v_i	0	0	0	0	0	0	0
3	5	1	0	**0**				
2	4	2	0	**0**				
1	3	3	0	**3**				
4	2	4	0	**3**				

(ข) พิจารณา j=1 จะได้ F(1,1)=0, F(2,1)=0, F(3,1)=3, F(4,1)=3

		j / i	0	1	2	3	4	5
w_i	v_i	0	0	0	0	0	0	0
3	5	1	0	0	**0**			
2	4	2	0	0	**4**			
1	3	3	0	3	**4**			
4	2	4	0	3	**4**			

(ค) พิจารณา j=2จะได้ F(1,2)=0, F(2,2)=4, F(3,2)=4, F(4,2)=4

		j / i	0	1	2	3	4	5
w_i	v_i	0	0	0	0	0	0	0
3	5	1	0	0	0	**5**		
2	4	2	0	0	4	**5**		
1	3	3	0	3	4	**7**		
4	2	4	0	3	4	**7**		

(ง)พิจารณา j=3 จะได้ F(1,3)=5, F(2,3)=5, F(3,3)=7, F(4,3)=7

		j / i	0	1	2	3	4	5
w_i	v_i	0	0	0	0	0	0	0
3	5	1	0	0	0	5	**5**	
2	4	2	0	0	4	5	**5**	
1	3	3	0	3	4	7	**8**	
4	2	4	0	3	4	7	**8**	

(จ) พิจารณา j=4 จะได้ F(1,4)=5, F(2,4)=5, F(3,4)=8, F(4,4)=8

		j / i	0	1	2	3	4	5
w_i	v_i	0	0	0	0	0	0	0
3	5	1	0	0	0	5	5	**5**
2	4	2	0	0	4	5	5	**9**
1	3	3	0	3	4	7	8	**9**
4	2	4	0	3	4	7	8	**9**

(ฉ) พิจารณา j=5 จะได้ F(1,5)=5, F(2,5)=9, F(3,5)=9, F(4,5)=9

ภาพประกอบ 2-15 การคำนวณหาผลลัพธ์ของ Knapsack Problem

เมื่อดำเนินการจนครบรอบการทำงาน จะได้ตารางผลลัพธ์ ดังแสดงในภาพประกอบ 2-15(ฉ) ซึ่งได้ค่า $F(4,5) = 9$ เป็นมูลค่าสูงที่สุดของชุดข้อมูลที่ถูกเลือกภายใต้พื้นที่จัดเก็บข้อมูลเท่ากับ 5 จากนั้นสามารถหาชุดข้อมูลที่ถูกเลือกได้จากการดำเนินการแก้ปัญหาแบบย้อนรอย (Backtracking) โดยเริ่มจากให้ตัวแปร $i = n = 4$ และ $j = W = 5$ จากนั้นเปรียบเทียบค่า $F(i,j)$ กับ $F(i\text{-}1,j)$ หรือไม่ ถ้า $F(i,j) = F(i\text{-}1,j)$ ก็จะลดค่าของตัวแปร i ลงครั้งละ 1 และจะไม่เก็บค่าข้อมูลลำดับที่ i เป็นสมาชิกของเซตคำตอบแต่ถ้า $F(i,j) \neq F(i\text{-}1,j)$ ก็จะเก็บค่าข้อมูลลำดับที่ i เป็นสมาชิกของเซตคำตอบจากนั้นให้ $j = j\text{-}w_i$ (เลื่อนไปยังตำแหน่งที่เก็บมูลค่าสูงสุดเมื่อขนาดของพื้นที่เท่ากับ $j\text{-}w_i$) และลดค่าของตัวแปร i ลงครั้งละ 1 เมื่อค่า i หรือ j มีค่าน้อยกว่าหรือเท่ากับ 0 ก็จะหยุดการทำงาน

จากภาพประกอบ 2-15(ฉ)

- $F(4,5) = F(3,5)$ ดังนั้นข้อมูลลำดับที่ 4 ไม่เป็นสมาชิกของเซตคำตอบ
- $F(3,5) = F(2,5)$ ดังนั้นข้อมูลลำดับที่ 3 ไม่เป็นสมาชิกของเซตคำตอบ
- $F(2,5) \neq F(1,5)$ ดังนั้นข้อมูลลำดับที่ 2 เป็นสมาชิกของเซตคำตอบ

จากนั้นให้ $j = 5\text{-}2 = 3$ และลดค่า i ลง 1 ดังนั้นตำแหน่งที่จะพิจารณาถัดไปคือ $F(1,3)$

- $F(1,3) \neq F(0,3)$ ดังนั้นข้อมูลลำดับที่ 1 เป็นสมาชิกของเซตคำตอบ

จากนั้นให้ $j = 3\text{-}3 = 0$ และลดค่า i ลง 1 จะเห็นได้ว่าค่า i หรือ j มีค่าน้อยกว่าหรือเท่ากับ 0 จึงออกจากการทำงาน

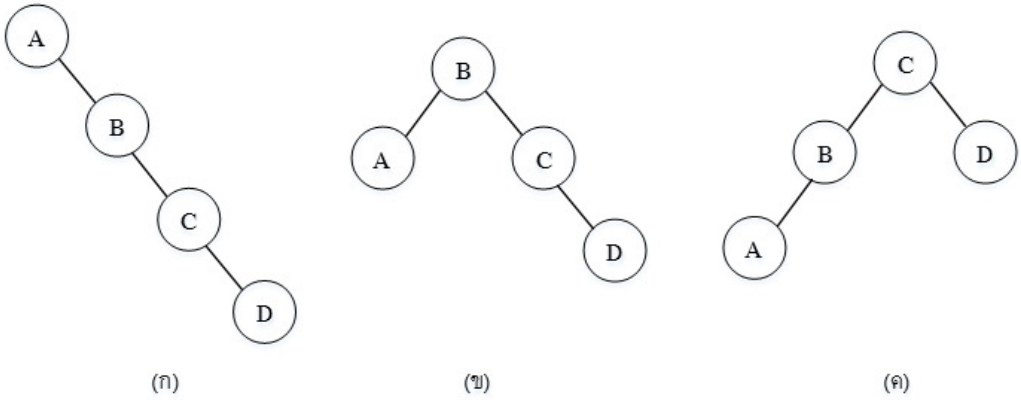
- ดังนั้นเซตคำตอบที่ได้คือข้อมูลลำดับที่ {1, 2} ซึ่งทำให้มูลค่าสูงสุดเท่ากับ 9 และใช้พื้นที่รวมเท่ากับ 5

2.5.2 Optimal Binary Search Tree

เป็นการหาว่าต้องสร้างต้นไม้ค้นหา (Binary Search Tree) อย่างไรจึงจะประหยัดค่าใช้จ่ายในการเปรียบเทียบที่สุด เนื่องจากวิธีการสร้างต้นไม้สามารถสร้างได้หลายวิธี ซึ่งจำนวนวิธีที่เป็นไปได้ของต้นไม้คำนวณได้จากสมการ 2.2 โดยที่ n คือจำนวนคีย์ของต้นไม้ (Levitin, 2007)

$$C(n) = \frac{1}{n+1}\left(\frac{2n}{n}\right) \qquad for\ \ n > 0, c(0) = 1 \qquad (2.5)$$

วิธีการนี้จะพิจารณาจากความน่าจะเป็นที่แต่ละคีย์จะถูกค้นหา โดยให้แต่ละโหนดของต้นไม้แทนค่าคีย์ที่ต้องการค้นหา ดังภาพ 2-16



ภาพประกอบ 2-16 ตัวอย่างต้นไม้ค้นหา 3 กรณีจากทั้งหมด 14 กรณี
เมื่อจำนวนคีย์เท่ากับ 4 คือ A, B, C และ D

เมื่อกำหนดให้ความน่าจะเป็นที่คีย์ A, B, C และ D จะถูกค้นหาเท่ากับ 0.1, 0.2, 0.4 และ 0.3 ตามลำดับ จะสามารถคำนวณค่าใช้จ่ายในการเปรียบเทียบสำหรับการค้นหาข้อมูลจากต้นไม้ทั้ง 3 แบบจากผลรวมของความน่าจะเป็นของแต่ละคีย์คูณกับระดับ (Level) ของคีย์นั้นๆ ซึ่งสามารถแสดงได้ดังนี้

- กรณี (ก), ค่าใช้จ่าย = (0.1×1) + (0.2×2) + (0.4×3) + (0.3×4) = 2.9
- กรณี (ข), ค่าใช้จ่าย = (0.1×2) + (0.2×1) + (0.4×2) + (0.3×3) = 2.1
- กรณี (ค), ค่าใช้จ่าย = (0.1×3) + (0.2×2) + (0.4×1) + (0.3×2) = 1.7

จะเห็นได้ว่า หากสร้างต้นไม้ค้นหาแบบกรณี (ค) ก็จะใช้ค่าใช้จ่ายในการเปรียบเทียบน้อยที่สุด เนื่องจากเมื่อคีย์ข้อมูลมีจำนวนมาก จึงมีการนำวิธีแก้ปัญหาแบบพลวัตมาประยุกต์ใช้กับปัญหานี้ เริ่มจากกำหนดให้ $a_1, .., a_n$ เป็นค่าของคีย์ที่เรียงลำดับจากน้อยไปมาก ให้ $p_1, .., p_n$ แทนความน่าจะเป็นที่แต่ละคีย์จะถูกค้นหา $C(i,j)$ แทนค่าใช้จ่ายที่น้อยที่สุดของต้นไม้ซึ่งประกอบด้วยคีย์ $a_i, .., a_j$ สามารถคำนวณได้จากสมการ 2.6 โดยที่ $1 \le i \le j \le n$

ค่าของ $C(i,j)$ จะเท่ากับค่าที่น้อยที่สุดของต้นไม้ย่อยที่สร้างขึ้นจากคีย์ $a_i, .. , a_j$ เมื่อต้นไม้แต่ละต้นมีการเลือกคีย์จากจำนวนคีย์ทั้งหมดที่แตกต่างกันให้เป็นโหนดราก (Root) ซึ่งแทนด้วย a_k คีย์ที่อยู่ทางด้านซ้ายของโหนดราก คือ $a_i, .., a_{k-1}$ และคีย์ที่อยู่ทางด้านขวาของโหนดราก คือ $a_{k+1}, .., a_j$

$$C(i,j) = \min_{i \le k \le j} \left\{ C(i,k-1) + C(k+1,j) \right\} + \sum_{s=i}^{j} P_s \qquad (2.6)$$

จากนั้นสร้างตารางเพื่อเก็บผลลัพธ์ในการคำนวณ (Main Table) และตารางเพื่อเก็บค่าคีย์ที่เป็นโหนดรากของแต่ละต้นไม้ย่อย (Root Table) โดยกำหนดค่าเริ่มต้นให้ตำแหน่งในเส้นทแยงมุมของตารางมีค่าเท่ากับ 0 และ $C(i,i)$ โดยที่ $1 \le i \le n$ มีค่าเท่ากับ i ดังแสดงในภาพประกอบ 2-17 (ก)

	0	1	2	3	4
1	0	0.1			
2		0	0.2		
3			0	0.4	
4				0	0.3
5					0

	0	1	2	3	4
1		1			
2			2		
3				3	
4					4
5					

(ก) กำหนดค่าเริ่มต้นให้กับตาราง

	0	1	2	3	4
1	0	0.1	0.4	1.1	**1.7**
2		0	0.2	0.8	1.4
3			0	0.4	1.0
4				0	0.3
5					0

	0	1	2	3	4
1		1	2	3	3
2			2	3	3
3				3	3
4					4
5					

(ข) ผลลัพธ์ทั้งหมดที่ได้จากการคำนวณ

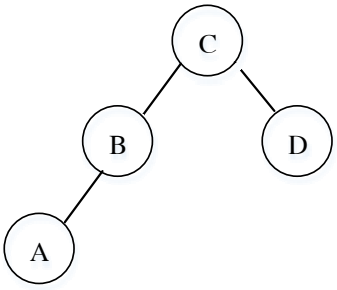
ภาพประกอบ 2-17 ตารางเก็บผลลัพธ์สำหรับต้นไม้ค้นหาที่มีค่าใช้จ่ายในการเปรียบเทียบน้อยที่สุด

ตัวอย่างการหาค่า $C(i,j)$

$$C(1,2) = \min \begin{cases} k = 1 : C(1,0) + C(2,2) + \sum_{s=1}^{2} P_s = 0 + 0.2 + 0.3 = 0.5 \\ k = 2 : C(1,1) + C(3,2) + \sum_{s=1}^{2} P_s = 0.1 + 0 + 0.3 = 0.4 \end{cases}$$

$= 0.4$

เมื่อคำนวณเสร็จสิ้นแล้วจะได้ค่าในตำแหน่ง $C(1,n)$ เป็นผลลัพธ์ที่ดีที่สุดเมื่อต้นไม้ประกอบด้วยคีย์ทั้งหมด n คีย์ จากภาพประกอบ 2-17(ข) ผลลัพธ์ที่ได้ คือ 1.7 เป็นค่าใช้จ่ายที่น้อยในการเปรียบเทียบเพื่อค้นหาข้อมูล และสามารถคำนวณหาต้นไม้ผลลัพธ์ได้จากการตรวจสอบ ตารางที่ใช้เก็บค่าคีย์ที่เป็นโหนดรากของแต่ละต้นไม้ย่อย เริ่มจาก $R(1,4) = 3$ ก็จะให้ค่าคีย์ลำดับที่ 3 เป็นโหนดราก นั่นคือคีย์ C จากนั้นพิจารณาต้นไม้ย่อยทางซ้ายของ C จะประกอบด้วยคีย์ A และ B ดังนั้นอ่านค่า $R(1,4) = 2$ ก็จะให้ค่าคีย์ลำดับที่ 2 เป็นโหนดราก นั่นคือ B ดังนั้นจะเหลือคีย์ A เป็นต้นไม้ย่อยทางซ้ายของ B ส่วนต้นไม้ย่อยทางขวาของ C เหลือเพียงค่าเดียวคือ D ดังนั้นผลลัพธ์ที่ได้แสดงดังภาพประกอบ 2-18



ภาพประกอบ 2-18 Optimal Binary Search Tree

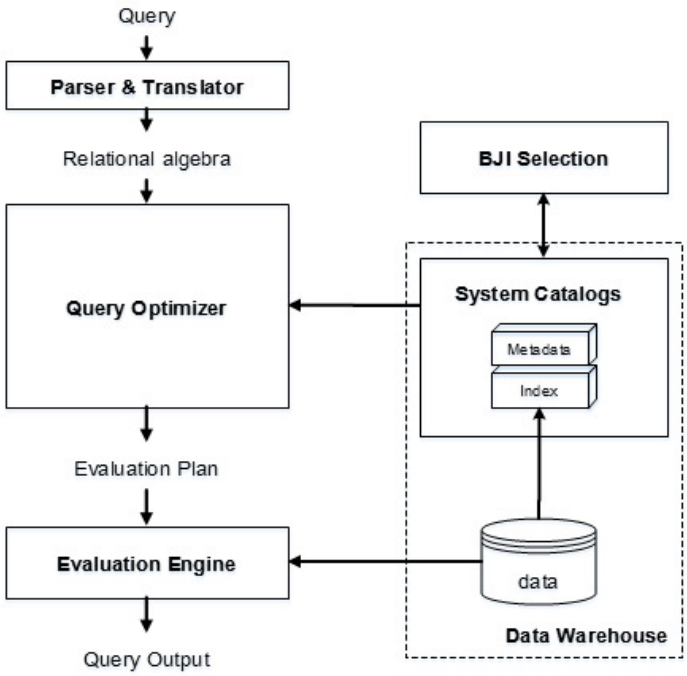
บทที่ 3

ขั้นตอนการทำงานของระบบแนะนำการคัดเลือกดัชนี Bitmap Join Index สำหรับโครงสร้างแบบ Star Schema

บทนี้กล่าวถึง ขั้นตอนการทำงานของระบบแนะนำการคัดเลือกดัชนี Bitmap Join Index สำหรับโครงสร้างแบบ Star Schema เพื่อปรับปรุงประสิทธิภาพด้านเวลาในการสอบถามข้อมูลของคลังข้อมูล โดยมีเงื่อนไขการใช้ทรัพยากรในการสร้างดัชนีที่จำกัด

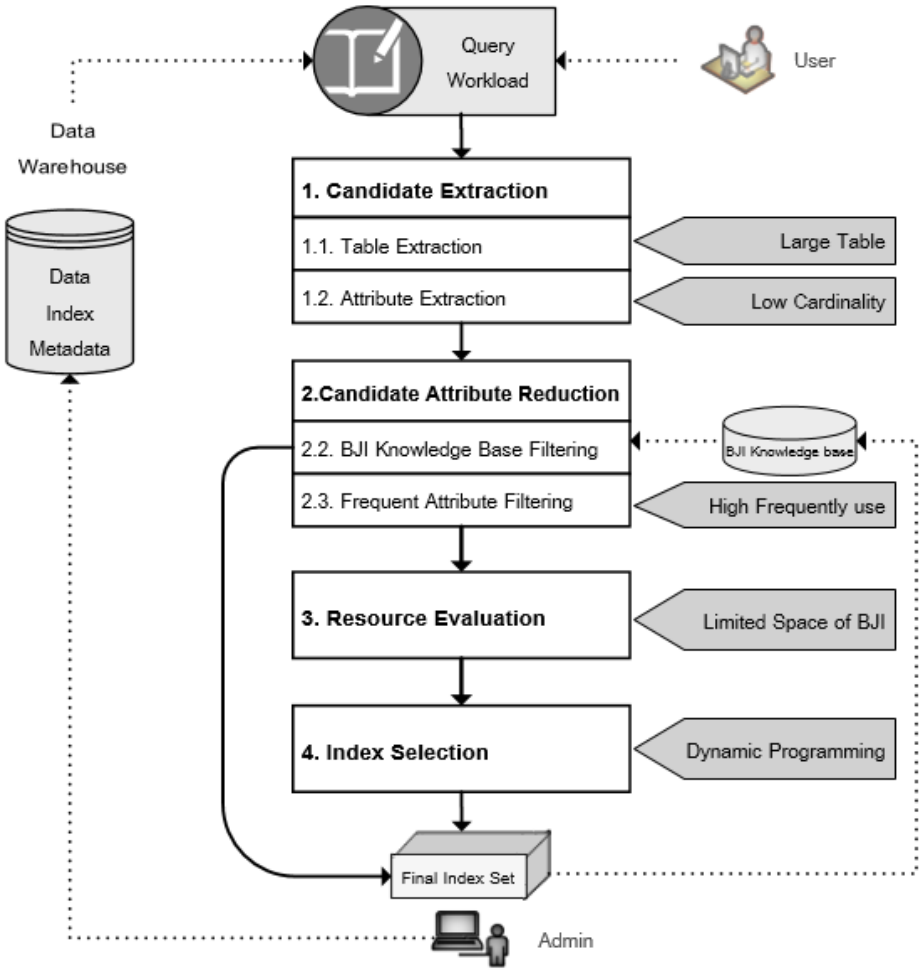
3.1 ขั้นตอนการทำงานของระบบ

ระบบแนะนำการคัดเลือกดัชนี Bitmap Join Index ช่วยให้ผู้ใช้คลังข้อมูลสามารถเลือกสร้างดัชนีที่ช่วยลดเวลาการสอบถามได้อย่างมีประสิทธิภาพภายใต้การใช้ทรัพยากรที่จำกัด เป็นส่วนหนึ่งของวิธีการปรับปรุงประสิทธิภาพการสอบถาม (Query Optimizer) ในกระบวนการสอบถามข้อมูล ดังภาพประกอบ 3-1



ภาพประกอบ 3-1 การทำงานของ BJI Selection ในกระบวนการสอบถาม

ระบบแนะนำการคัดเลือกดัชนี Bitmap Join Index สำหรับโครงสร้างแบบ Star Schema (BJI Selection) ประกอบด้วย 4 ขั้นตอน ได้แก่ ขั้นตอนที่ 1 การสกัดแอททริบิวต์บนตาราง Dimension ที่จะนำมาสร้างดัชนี Bitmap Join Index แบ่งได้เป็น 1.1) การสกัดตารางที่มีขนาดใหญ่ (Table Extraction) และ 1.2) การสกัดแอททริบิวต์ที่มีคาร์ดินอลิตี้ต่ำ(Attribute Extraction) ขั้นตอนที่ 2 การลดจำนวน Candidate Attribute แบ่งได้เป็น 2.1) การกรองแอท-ทริบิวต์ที่ไม่ปรากฏใน BJI Knowledge Base (BJI Knowledge Base Filtering) และ 2.2) การกรองแอททริบิวต์ที่ถูกสอบถามบ่อย (Frequent Attribute Filtering) ขั้นตอนที่ 3 การประเมินทรัพยากรที่ใช้ในการสร้างดัชนีสำหรับแอททริบิวต์แต่ละตัว (Resource Evaluation) และขั้นตอนที่ 4 การเลือกดัชนีที่เหมาะสมที่สุดสำหรับทรัพยากรที่จำกัด (Index Selection) ดังภาพประกอบ 3-2 โดยสามารถอธิบายรายละเอียดของแต่ละขั้นตอนการทำงานได้ดังนี้



ภาพประกอบ 3-2 โครงสร้างของ BJI Selection

3.1.1 ขั้นตอนที่ 1 การสกัดแอททริบิวต์บนตาราง Dimension ที่จะนำมาสร้างดัชนี Bitmap Join Index (Candidate Extraction)

เป็นขั้นตอนในการสกัดแอททริบิวต์ทั้งหมดที่จะนำมาสร้างดัชนีจากชุดคำสั่งสอบถามข้อมูลในอดีต (Workload) ของคลังข้อมูล แบ่งการทำงานเป็น 2 ขั้นตอนย่อย ได้แก่ การสกัดตารางที่มีขนาดใหญ่ (Table Extraction) และการสกัดแอททริบิวต์ที่มีคาร์ดินอลิตี้ต่ำ (Attribute Extraction)

1.1) การสกัดตารางที่มีขนาดใหญ่ เป็นขั้นตอนในการคัดเลือกเฉพาะตาราง Dimension ที่มีขนาดใหญ่ เนื่องจากกลุ่มของแอททริบิวต์ที่ต้องการคัดเลือกคือแอททริบิวต์ที่อยู่ในตารางที่มีขนาดใหญ่ หากเลือกสร้างดัชนีบนแอททริบิวต์เหล่านี้ก็จะช่วยลดเวลาที่ใช้ในระหว่างการ จับคู่ข้อมูลระหว่างตารางได้มาก ส่วนตารางที่มีขนาดเล็กจะใช้เวลาในการจับคู่ข้อมูลระหว่างตารางน้อยมากจึงไม่จำเป็นต้องลดเวลาในการประมวลผล

โดยขนาดของตารางจะพิจารณาจากจำนวนบล็อค (Block) ที่ใช้ในการจัดเก็บข้อมูลของตาราง Z (b_z) ซึ่งสามารถคำนวณได้จาก จำนวนของแถวข้อมูลในตาราง ($|T_z|$) คูณกับความยาวของแถวข้อมูล (L_z) หารด้วยขนาดของบล็อคที่ใช้ในการเก็บข้อมูล (B) ดังสมการ 3.1

$$b_z = \left\lceil \frac{|T_z| \times L_z}{B} \right\rceil \tag{3.1}$$

เมื่อกำหนดให้ T แทนเซตของตาราง Dimension ทั้งหมด และ T_β แทนเซตของตาราง Dimension ที่ถูกคัดเลือกซึ่งมีขนาดใหญ่กว่าหรือเท่ากับค่าที่กำหนด ซึ่งแทนด้วย β โดยสามารถคำนวณได้จากค่าเฉลี่ยของขนาดของตารางทั้งหมด สามารถแสดงในรูปแบบเซตได้ดังนี้

$T = \{ D_1, D_2, \ldots, D_m \}$

$T_\beta = \{ D_i \in T \mid D_i$ มีขนาดใหญ่กว่าหรือเท่ากับ $\beta \}$

1.2) การสกัดแอททริบิวต์ที่มีคาร์ดินอลิตี้ต่ำ โดยคัดเลือกเฉพาะแอททริบิวต์จากชุดคำสั่งสอบถามข้อมูลที่เป็นการสอบถามที่มีการจับคู่ข้อมูลระหว่างตาราง ที่ปรากฏอยู่หลังเงื่อนไข WHERE, ORDER BY และ GROUP BY และเป็นแอททริบิวต์ที่อยู่ในตารางที่สกัดได้จากขั้นตอน Table Extraction (T_β) จากนั้นจะคัดเลือกแอททริบิวต์ที่มีความเหมาะสมสำหรับการนำมาสร้างดัชนี Bitmap Join Index นั่นคือแอททริบิวต์ที่มีคาร์ดินอลิตี้ต่ำ ตามคุณสมบัติของดัชนีบิตแมปทั่วไป โดยคัดเฉพาะแอททริบิวต์ที่มีคาร์ดินอลิตี้ไม่เกินค่าสูงสุดที่ผู้ดูแลระบบกำหนด

เมื่อกำหนดให้ U แทนเซตของแอททริบิวต์ทั้งหมดที่ปรากฏในชุดคำสั่งสอบถามข้อมูลในอดีต และ A_1 แทนเซตของแอททริบิวต์ทั้งหมดที่จะนำมาสร้างดัชนี Bitmap Join Index (Candidate Attribute) และ A_α แทนเซตของ Candidate Attribute ที่มีคาร์ดินอลิตี้น้อยกว่าหรือเท่ากับค่าที่กำหนด ซึ่งแทนด้วย α โดยทั่วไปจะกำหนดให้แอททริบิวต์ที่มีคาร์-ดินอลิตี้ต่ำคือแอททริบิวต์ที่มีจำนวนคาร์ดินอลิตี้น้อยกว่าหรือเท่ากับ 64 สามารถแสดงในรูปแบบเซตได้ดังนี้

A_1 = { x $\in$ U | x เป็นแอททริบิวต์ที่ปรากฏอยู่หลังเงื่อนไข WHERE, ORDER BY และ GROUP BY และ x เป็นแอททริบิวต์ที่อยู่ในตารางที่เป็นสมาชิกของ T_β }

จะได้ A_α = { x $\in$ U | x มีคาร์ดินอลิตี้น้อยกว่าหรือเท่ากับ α }

3.1.2 ขั้นตอนที่ 2 การลดจำนวน Candidate Attribute (Candidate Attribute Reduction)

เป็นขั้นตอนในการลดจำนวน Candidate Attribute ที่จะนำมาสร้างดัชนี เพื่อช่วยให้ขั้นตอนในการเลือกดัชนีมีประสิทธิภาพมากขึ้น โดยแบ่งการทำงานเป็น 2 ขั้นตอนย่อย ได้แก่ การกรองแอททริบิวต์ที่ไม่ปรากฏใน BJI Knowledge Base (BJI Knowledge Base Filtering) และการกรองแอททริบิวต์ที่มีความถี่ในการสอบถามบ่อย (Frequent Attribute Filtering)

2.1) การกรองแอททริบิวต์ที่ไม่ปรากฏใน BJI Knowledge Base เป็นขั้นตอนในการลดจำนวน Candidate Attribute โดยการกรองแอททริบิวต์ที่เป็นสมาชิกของเซตของแอททริบิวต์ที่อยู่ใน BJI Knowledge Base ออกไปจากเซตของ Candidate Attribute จากนั้นจะเก็บแอททริบิวต์ที่ไม่ผ่านเงื่อนไขการกรอง (คือ เป็นสมาชิกของ BJI Knowledge Base) ให้อยู่ในชุดดัชนีที่ถูกเลือก (Final Index Set) ทันทีโดยไม่ต้องผ่านขั้นตอนอื่นๆ ทำให้ลดเวลาในกระบวนการคัดเลือกดัชนีได้เนื่องจากจำนวนของ Candidate Attribute ลดลง

BJI Knowledge Base เป็นชุดข้อมูลที่ได้มาจากการเก็บข้อมูลแอททริบิวต์ที่มักปรากฏเสมอจากขั้นตอนการคัดเลือกดัชนีในแต่ละครั้ง โดยจะมีการอัพเดตข้อมูล BJI Knowledge Base ทุกครั้งเมื่อมีการดำเนินการคัดเลือกดัชนีจากระบบ สำหรับข้อมูลที่เก็บใน BJI Knowledge Base จะประกอบด้วยค่าของแอททริบิวต์ และพื้นที่ที่ใช้ในการสร้างดัชนีของแอททริบิวต์นั้นๆ

เมื่อกำหนดให้ K แทนเซตของแอททริบิวต์ที่ปรากฏใน Knowledge Base, A_2 แทนเซตของ Candidate Attribute ที่ไม่รวมแอททริบิวต์ที่ปรากฏใน BJI Knowledge Base และ I แทนเซตของแอททริบิวต์ที่ถูกเลือกมาสร้างดัชนี (Final Index Set) สามารถแสดงในรูปแบบเซตได้ดังนี้

K = { x $\in$ U | x เป็นแอททริบิวต์ที่ปรากฏใน BJI Knowledge Base }

I = { x $\in$ U | x $\in A_\alpha$ $\cap$ x $\in$ K }

ดังนั้น $A_2 = A_\alpha - I$

2.2) การกรองแอททริบิวต์ที่ถูกสอบถามบ่อย เป็นขั้นตอนในการนำ Candidate Attribute ที่ผ่านเงื่อนไขการกรองจากขั้นตอน BJI Knowledge Base Filtering (A_2) มาคัดเลือกแอททริบิวต์ที่มีการสอบถามบ่อย โดยการคำนวณค่าสนับสนุนของแอททริบิวต์แต่ละตัว (sA_i) ซึ่งได้มาจาก จำนวนความถี่ของแต่ละแอททริบิวต์ที่ปรากฏในชุดคำสั่งสอบถามข้อมูลในอดีต หารด้วยจำนวนคำสั่งสอบถามทั้งหมด (f_i/nQ) จากนั้นเลือกเฉพาะแอททริบิวต์ที่มีค่าสนับสนุนมากกว่าหรือเท่ากับขีดแบ่งค่าสนับสนุน (Support Threshold) ซึ่งจะถูกกำหนดโดยผู้ดูแลระบบ ด้วยสมมติฐานที่ว่าแอททริบิวต์ที่มีการสอบถามในอดีตสูงอาจมีโอกาสถูกสอบถามซ้ำมากกว่าแอททริบิวต์ที่มีการสอบถามในอดีตต่ำ

เมื่อกำหนดให้ A_δ แทนเซตของ Candidate Attribute ที่มีค่าสนับสนุนในการสอบถามมากกว่าหรือเท่ากับขีดแบ่งค่าสนับสนุนที่กำหนด ซึ่งแทนด้วย δ โดยที่ขีดแบ่งค่าสนับสนุน สามารถเปลี่ยนแปลงได้ตามความต้องการของผู้ดูแลระบบ เช่น เมื่อมีพื้นที่ที่ใช้ในการจัดเก็บดัชนีมาก ก็ควรกำหนดให้ขีดแบ่งค่าสนับสนุนมีค่าต่ำ เพื่อให้ได้จำนวนแอททริบิวต์ที่นำไปสร้างดัชนีเพิ่มมากขึ้น แต่หากมีพื้นที่ที่ใช้ในการจัดเก็บดัชนีน้อย ก็ควรกำหนดให้ขีดแบ่งค่าสนับสนุนมีค่าสูง เพื่อลดจำนวนแอททริบิวต์ที่จะนำไปสร้างดัชนี สามารถแสดงในรูปแบบเซตได้ดังนี้

A_δ = { x $\in A_2$ | x มีค่าสนับสนุนในการสอบถามมากกว่าหรือเท่ากับ δ }

3.1.3 ขั้นตอนที่ 3 การประเมินทรัพยากรที่ใช้ในการสร้างดัชนีสำหรับแอททริบิวต์แต่ละตัว (Resource Evaluation)

เมื่อได้ Candidate Attribute ที่ต้องการแล้ว จากนั้นจะดำเนินการประเมินทรัพยากรที่ใช้ในการสร้างดัชนี Bitmap Join Index ของแอททริบิวต์แต่ละตัว เช่น พื้นที่ที่ใช้ในการสร้างดัชนี และค่าใช้จ่ายที่ใช้ในการ Join ระหว่างตาราง Fact และ Dimension เพื่อนำไปใช้ในขั้นตอน Index Selection ซึ่งแบ่งการทำงานออกเป็นขั้นตอนดังนี้

3.1) สร้าง Candidate Attribute Table (CAT) ซึ่งใช้เก็บคุณสมบัติทั้งหมดของแอททริบิวต์ที่จะนำไปใช้ในขั้นตอน Index Selection ได้แก่ ทรัพยากรที่ใช้ในการสร้างดัชนี และค่าความเหมาะสมในการสร้างดัชนีของแต่ละแอททริบิวต์

3.2) คำนวณทรัพยากรที่ใช้ในการสร้างดัชนี (S) Bitmap Join Index ของแอททริบิวต์แต่ละตัว ในที่นี้ได้แก่ พื้นที่ที่ใช้ในการสร้างดัชนี ซึ่งสามารถหาได้จากจำนวนแถวใน

ตาราง Fact ($|F|$) คูณกับขนาดของพื้นที่ที่ใช้ในการลงรหัสของแต่ละแอททริบิวต์ ($Area(C)$) ดังสมการ 3.2 สำหรับงานวิจัยนี้ใช้การลงรหัสดัชนีบิตแมปแบบพื้นฐานจะใช้จำนวนบิตแมปเวกเตอร์เท่ากับ C หากต้องการลดขนาดของพื้นที่ที่ใช้ในการสร้างดัชนีนั้น สามารถเลือกวิธีการลงรหัสดัชนีบิตแมปแบบอื่นๆ ได้ เช่น ดัชนีบิตแมปแบบเข้ารหัส (Encoded Bitmap Index) ซึ่งใช้จำนวนบิตแมปเวกเตอร์เพียง $\lceil \log_2 C \rceil$ (Wu and Buchmann, 1998) ดัชนีบิตแมปแบบคู่กัน (Dual Bitmap Index) ซึ่งใช้จำนวนบิตแมปเวกเตอร์เท่ากับ $\left\lceil \sqrt{2C + 0.25} + 0.5 \right\rceil$ (Wattanakitrungroj, 2006) เป็นต้น

$$S = |F| \times Area(C) \tag{3.2}$$

โดยที่ผลลัพธ์ของฟังก์ชัน $Area(C)$ นั้นแปรผันตามชนิดของบิตแมปที่จะนำมาใช้ในการลงรหัส

3.3) คำนวณหาค่าใช้จ่ายที่ใช้ในการจับคู่ข้อมูลระหว่างตาราง Fact และ Dimension สำหรับงานวิจัยนี้จะพิจารณาค่าใช้จ่ายในการจับคู่ข้อมูล โดยใช้วิธีการ Sort-Merge Join (Silberschatz *et al*., 2011) ซึ่งจะคิดค่าใช้จ่ายในการเรียงลำดับข้อมูลในตาราง Dimension และตาราง Fact ที่จะนำดำเนินการจับคู่ข้อมูลกัน ($Cost_{sort}$) ดังสมการ 3.3 บวกกับค่าใช้จ่ายในการรวมข้อมูล (Merge) ระหว่าง 2 ตาราง ($Cost_{merge}$) ดังสมการ 3.4 จะได้ค่าใช้จ่ายทั้งหมดที่ใช้ในการจับคู่ข้อมูล ($TotalCost$) ดังสมการ 3.5 กำหนดให้ M คือ จำนวนบล็อคบัฟเฟอร์ (Block Buffer) ของหน่วยความจำหลัก b_{D_x} คือจำนวนบล็อกที่ใช้ในการเก็บข้อมูลแอททริบิวต์ที่จะนำมาใช้ในการจับคู่ข้อมูล (x) ของตาราง Dimension และ b_F คือจำนวนบล็อกที่ใช้ในการเก็บข้อมูลของแอททริบิวต์ที่จะนำมาใช้ในการจับคู่ข้อมูลในตาราง Fact

$$Cost_{sort} = b_{D_x}(2\left\lceil \log_{M-1}(b_{D_x}/M) \right\rceil + 1) + b_F(2\left\lceil \log_{M-1}(b_F/M) \right\rceil + 1) \tag{3.3}$$

$$Cost_{merge} = b_{D_x} + b_F \tag{3.4}$$

$$TotalCost = Cost_{sort} + Cost_{merge} \tag{3.5}$$

3.4) คำนวณหาค่าความเหมาะสมในการสร้างดัชนี (F_i) ของแต่ละ Candidate Attribute (A_i) ซึ่งสามารถคำนวณได้จากค่าสนับสนุนของแต่ละแอททริบิวต์ (sA_i) คูณกับค่าใช้จ่ายทั้งหมดที่ใช้ในการจับคู่ข้อมูลระหว่างตาราง ($TotalCost_i$) ดังแสดงในสมการ 3.6

$$F_i = s A_i \times TotalCost_i \tag{3.6}$$

ซึ่งแนวคิดในการคำนวณค่าความเหมาะสมคือ หากเลือกสร้างดัชนีบนแอททริ-บิวต์ที่มีค่าใช้จ่ายในการจับคู่ข้อมูลสูง และมีโอกาสในการถูกสอบถามบ่อย จะทำให้สามารถลดเวลาในการจับคู่ข้อมูลที่จะเกิดขึ้นในระหว่างการประมวลผลการสอบถามได้มาก ดังนั้นงานวิจัยนี้จึงใช้ค่าความเหมาะสมของแต่ละแอททริบิวต์เป็นปัจจัยในการพิจารณาสำหรับขั้นตอนในการเลือกดัชนีภายใต้พื้นที่การจัดเก็บดัชนีที่จำกัด

3.1.4 ขั้นตอนที่ 4 การเลือกดัชนีที่เหมาะสมที่สุดสำหรับพื้นที่การจัดเก็บดัชนีที่จำกัด (Index Selection)

เป็นขั้นตอนในการเลือกชุดดัชนีที่เหมาะสมที่สุด ภายใต้ขนาดของพื้นที่ทั้งหมดที่จะใช้ในการจัดเก็บดัชนี โดยข้อมูลที่นำมาใช้ในการประมวลผลได้แก่ ขนาดของพื้นที่ที่ใช้ในการจัดเก็บดัชนีแต่ละตัว และค่าความเหมาะสมในการสร้างดัชนีของแต่ละแอททริบิวต์ โดยใช้อัลกอริทึม Index Selection ดังภาพประกอบ 3-3

สำหรับการทำงานของอัลกอริทึม Index Selection สามารถอธิบายการทำงานได้ดังนี้

ข้อมูลนำเข้าของอัลกอริทึม ประกอบด้วย 1) ขนาดของพื้นที่ทั้งหมด (S) ที่กำหนดไว้เพื่อจัดเก็บดัชนี 2) พื้นที่จัดเก็บดัชนีของแอททริบิวต์แต่ละตัว(S_i) 3) ค่าความเหมาะสมในการสร้างดัชนีของแต่ละแอททริบิวต์ (F_i) และ 4) จำนวน Candidate Attribute ทั้งหมด (n) ส่วนข้อมูลส่งออกจะได้ชุดดัชนี (I) ที่ถูกเลือกภายใต้พื้นที่ที่จำกัด โดยการประมวลผลสามารถแบ่งได้เป็น 3 Steps (ดูภาพประกอบ 3-3) ดังนี้

Step 1 กำหนดค่าเริ่มต้นให้กับตารางขนาด 2 มิติที่ใช้ในการคำนวณซึ่งแต่ละแถวของตารางแทนด้วยค่าของ Candidate Attribute ตั้งแต่ 0 ถึง n ในแต่ละคอลัมน์ของตารางแทนด้วยขนาดของพื้นที่ตั้งแต่ 0 ถึง S และค่าที่เก็บในตารางได้แก่มูลค่าที่สูงที่สุดของเมื่อมีข้อมูลแอททริบิวต์ที่ 1 ถึง แอททริบิวต์ที่ i สำหรับพื้นที่ขนาด j *(V[i][j])*

1.1 กำหนดค่าเริ่มต้นให้กับทุกค่าที่อยู่ในแถวที่ 0 ของตารางผลลัพธ์เท่ากับ 0 ($V[0][0 \ldots S] = 0$)

1.2 กำหนดค่าเริ่มต้นให้กับทุกค่าที่อยู่ในคอลัมน์ที่ 0 ของตารางผลลัพธ์เท่ากับ 0 ($V[0 \ldots n][0] = 0$)

1.3 กำหนดค่าเริ่มต้นให้กับค่าตำแหน่งที่เหลือของตารางผลลัพธ์เท่ากับ -1 ($V[1 \ldots n][1 \ldots S] = -1$)

1.4 กำหนดค่าเริ่มต้นให้กับตัวแปร $i = n$ และ $j = S$

Algorithm* : *Index Selection

Input : *- Capacity of index, S*

- Space usage of attribute i, S_i

- Fitness value of attribute i, F_i

- Amount of all candidate attribute, n

Output : *-Final Index Set under limited of S (I)*

Step 1. Initialize values in table *V*[0..*n*][0..*S*]

1.1 Set all values in row 0 to 0

1.2 Set all values in column 0 to 0

1.3 Set all values in table (*V*[1..*n*][1..*S*]) to -1

1.4 Set value of variables *i* to *n* and *j* to *S*

Step 2. BJISelection(*i*, *j*)

2.1 if(*V*[*i*][*j*]<0)

if (*j*<*S*[*i*])

V[*i*][*j*]= BJISelection (*i*-1, *j*);

else

V[*i*][*j*]=max[BJISelection (*i*-1, *j*),

F[*i*]+ BJISelection (*i*-1, *j*- *S*[*i*])];

2.2 Store *V*[*i*][*j*] as the largest value of coordinate (*i,j*)

Step 3. FindIndexSet(*i*, *j*)

3.1 For (*i*=*n*, *j*=*S* ; *j*>0 && *i*>0 , *i*--)

if(*V*[*i*][*j*]!= *V*[*i*-1][*j*])

Store attribute *i* in Final Index Set

j=*j*- *S*[*i*]

3.2 Return Final Index Set (*I*)

ภาพประกอบ 3-3 อัลกอริทึม Index Selection

Step 2 เป็นขั้นตอนในการหามูลค่าที่สูงที่สุดที่ตำแหน่ง (*i, j*) โดยอัลกอริทึมนี้จะคำนวณค่าและมีการบันทึกไว้ เมื่อมีการเรียกใช้ซ้ำจึงไม่ต้องมีการคำนวณใหม่

2.1 ตรวจสอบเงื่อนไขว่าค่าที่เก็บอยู่ในตำแหน่ง (*i, j*) มีค่าน้อยกว่า 0 หรือไม่หากเงื่อนไขเป็นจริง หมายความว่าค่าที่ตำแหน่งนั้นยังไม่เคยถูกคำนวณมาก่อน (มีค่า

เท่ากับ -1) ก็จะเริ่มคำนวณมูลค่าที่สูงที่สุดที่เป็นไปได้ที่ตำแหน่งนั้นๆ โดยดำเนินการตรวจสอบเงื่อนไขว่าค่า *j* น้อยกว่า S_i ซึ่งสามารถให้ผลลัพธ์ได้ดังนี้

- หากเงื่อนไขเป็นจริง (*j* น้อยกว่า S_i) นั่นคือขนาดของพื้นที่ที่ใช้ในการจัดเก็บดัชนีไม่เพียงพอสำหรับแอททริบิวต์ที่ *i* ก็จะให้มูลค่าของตำแหน่งนั้นเท่ากับมูลค่าสูงสุดที่ไม่รวมแอททริบิวต์ที่ *i* จะได้ว่า *V[i][j]* = BJISelection (*i*-1, *j*)
- หากเงื่อนไขเป็นเท็จ (*j* มากกว่าหรือเท่ากับ S_i) ก็จะให้มูลค่าของตำแหน่งนั้นเท่ากับค่าสูงสุดระหว่าง มูลค่าสูงสุดที่ไม่รวมแอททริบิวต์ที่ *i* นั่นคือ BJISelection (*i*-1, *j*) กับมูลค่าของแอททริบิวต์ที่ i บวกกับมูลค่าสูงสุดที่ไม่รวมแอททริบิวต์ที่ *i* เมื่อมีขนาดของพื้นที่ที่ใช้ในการจัดเก็บดัชนีเท่ากับ *j-S[i]* นั่นคือ *F[i]*+ BJISelection (*i*-1, *j-S[i]*)

2.2 บันทึกค่า V[i][j] ที่ได้จากขั้นตอนข้างต้น

Step 3 เป็นขั้นตอนในการหาชุดดัชนีที่เป็นคำตอบ โดยใช้ตารางเก็บผลลัพธ์ที่ได้จาก Step 2 สามารถอธิบายการทำงานได้ ดังนี้

3.1 ลักษณะการทำงานเป็นการวนรอบ โดยกำหนดค่าเริ่มต้นให้กับตัวแปร *i = n* และ *j = S* โดยมีเงื่อนไขในการทำงานของลูปคือ *j > 0* และ *i > 0* และในการทำงานแต่ละรอบจะมีการลดค่า *i* ลงครั้งละ 1 ในขณะที่เงื่อนไขของลูปเป็นจริงจะมีการดำเนินการตามขั้นตอนดังนี้

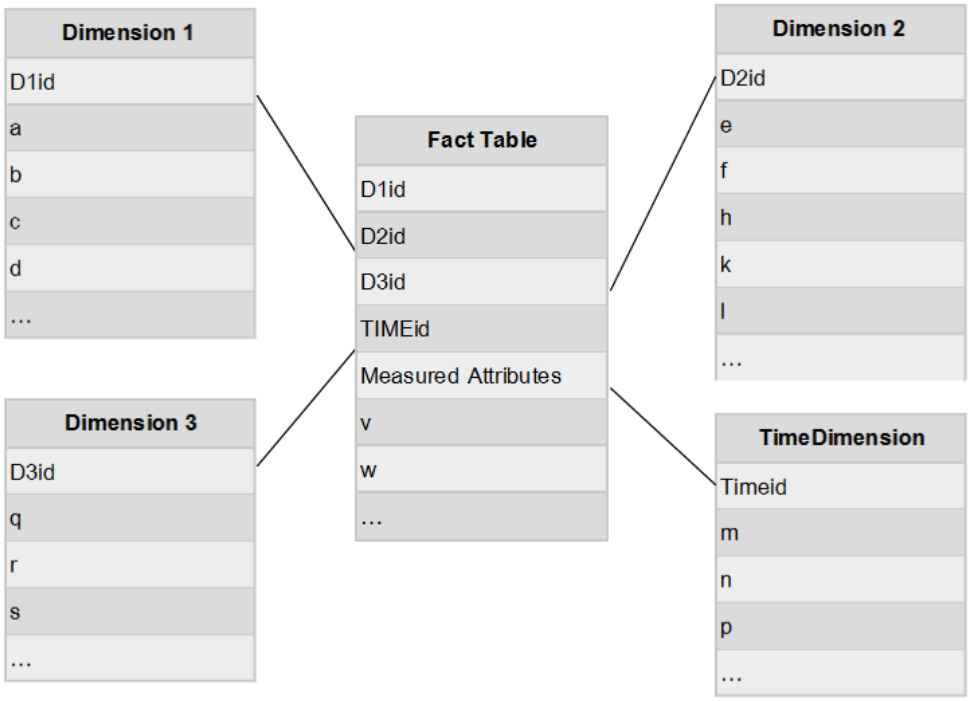
- เปรียบเทียบค่า *V[i][j]* กับค่าในตำแหน่งก่อนหน้านั่นคือค่าของ *V[i-1][j]* ว่ามีค่าเท่ากันหรือไม่ หากมีค่าไม่เท่ากัน ก็จะบันทึกค่าของแอททริบิวต์ที่ *i* ในชุดดัชนีที่ถูกเลือก (Final Index Set) จากนั้นก็ให้ *j* เท่ากับ *j-S[i]* เพื่อไปหาแอททริบิวต์ถัดไปที่ใช้พื้นที่ในการจัดเก็บดัชนีไม่เกิน *j-S[i]*
- เมื่อดำเนินการจนเงื่อนไขของการวนรอบเป็นเท็จ *j <= 0* หรือ *i <= 0* ก็จะออกจากลูป

3.2 ส่งค่าชุดดัชนีที่ถูกเลือกทั้งหมดเป็นผลลัพธ์ของการทำงาน

เมื่อดำเนินการจนจบขั้นตอนของระบบแนะนำการคัดเลือกดัชนีแล้ว จะนำชุดดัชนีที่เป็นผลลัพธ์ไปเก็บไว้ เพื่อใช้ในการคำนวณว่าแอททริบิวต์ใดที่ควรจะเก็บอยู่ใน BJI Knowledge Base

3.2 ตัวอย่างการทำงานของระบบ

เพื่อให้เข้าใจกระบวนการทำงานของระบบการคัดเลือกดัชนี Bitmap Join Index สำหรับโครงสร้างแบบ Schema จะแสดงขั้นตอนการทำงานโดยใช้ตัวอย่างคลังข้อมูลที่มีโครงสร้างของระบบ ดังภาพประกอบ 3-4



ภาพประกอบ 3-4 ตัวอย่าง Star Schema

1) การสกัดแอททริบิวต์บนตาราง Dimension ที่จะนำมาสร้างดัชนี Bitmap Join Index

ในขั้นตอนนี้จะมีการแบ่งการทำงานเป็น 2 ส่วน ได้แก่ การสกัดตารางที่มีขนาดใหญ่ และการสกัดแอททริบิวต์ที่มีคาร์ดินอลิตี้ต่ำ

1.1) การสกัดตารางที่มีขนาดใหญ่ จากตาราง Dimension ทั้งหมดโดยพิจารณาจากจำนวน Block ที่ใช้ในการจัดเก็บข้อมูล

เมื่อกำหนดให้ตาราง Dimension1, Dimension2, Dimension3, TimeDimension และ Fact มีจำนวนแถวข้อมูล เท่ากับ 10,000, 20,000, 500, 40,000 และ 1,000,000 ตามลำดับ และกำหนดให้ขนาดของแถวของตาราง Dimension1, Dimension2, Dimension3 และ TimeDimension มีค่าเท่ากับ 24, 12, 12, 36 และ 12 ตามลำดับ และขนาดของบล็อคข้อมูล เท่ากับ 4,096 ไบต์ (Bytes) สามารถคำนวณหาจำนวนบล็อคที่ใช้ในการเก็บข้อมูล (b_D) ได้ดังตาราง 3-1

ตาราง 3-1 การคำนวณหาขนาดของตาราง Dimensions

ตาราง (D)	จำนวนแถว ($\|R_D\|$)	ขนาดของแถว (L_D)	จำนวน Block ที่ใช้ในการเก็บข้อมูล (b_D)
Dimension1 (D_1)	10,000	24	$\left\lceil \frac{10,000 \times 24}{4,096} \right\rceil = 59$
Dimension2 (D_2)	20,000	12	$\left\lceil \frac{20,000 \times 12}{4,096} \right\rceil = 59$
Dimension3 (D_3)	500	12	$\left\lceil \frac{500 \times 12}{4,096} \right\rceil = 2$
TimeDimension(D_T)	40,000	36	$\left\lceil \frac{40,000 \times 12}{4,096} \right\rceil = 88$

จากตาราง 3-1 เมื่อคำนวณหาค่าเฉลี่ยของขนาดของตารางทั้งหมด จะได้จำนวนบล็อคขั้นต่ำที่ใช้ในการเก็บข้อมูลเท่ากับ 52 (β=52) จะเห็นได้ว่า แอททริบิวต์ที่อยู่ในตาราง Dimension3 ซึ่งใช้จำนวน 2 บล็อคในการเก็บข้อมูล จะไม่ถูกนำมาพิจารณาเป็น Candidate Attribute เนื่องจากตารางมีขนาดเล็กจึงใช้เวลาในการจับคู่ข้อมูลน้อย ดังนั้นตารางทั้งหมดที่จะนำมาพิจารณาสามารถเขียนแทนด้วยเซตแบบแจกแจงสมาชิกได้ ดังนี้

$T = \{D_1, D_2, D_3, D_T\}$

$T_\beta = \{D_1, D_2, D_T\}$

1.2) การสกัดแอททริบิวต์ที่มีคาร์ดินอลิตี้ต่ำ

ในขั้นตอนนี้จะคัดเลือกเฉพาะแอททริบิวต์ที่ปรากฏในคำสั่งสอบถามของคลังข้อมูลที่มีการสอบถามข้อมูลโดยมีการจับคู่ข้อมูลระหว่างตารางจากชุดคำสั่งสอบถามในอดีต ซึ่งมีลักษณะดังภาพประกอบ 3-5 (กำหนดให้ F แทน ตาราง Fact, D_1 แทน ตาราง Dimension1, D_2 แทน ตาราง Dimension2, D_3 แทน ตาราง Dimension3 และ D_T แทนตาราง TimeDimension)

Q1 : **SELECT** * **FROM** F, D_1, D_2
WHERE AND F.D_1id=D_1.D_1id AND F.D_2id=D_2.D_2id
AND D_1.*a*=1 AND D_1.*c*=2 AND D_1.*d*=3 AND D_2.*e*=4 AND D_2.*k*=5 AND F.*w*=5

Q2 : **SELECT** *, sum(*v*) **FROM** F, D_1, D_2, D_T
WHERE AND F.D_1id=D_1.D_1id AND F.D_2id=D_2.D_2id AND F.$D_T$id= D_T.$D_T$id
AND D_1.*b*=1 AND D_1.*c*=3 AND D_2.*g*=4 AND D_T.*m*=5 **GROUP BY** D_2.*k*

Q3 : **SELECT** * **FROM** F, D_1, D_2, D_T
WHERE AND F.D_1id=D_1.D_1id AND F.D_2id=D_2.D_2id AND F.$D_T$id= D_T.$D_T$id
AND D_1.*b*=1 AND D_1.*c*=2 AND D_1.*d*=3 AND D_2.*h*=4 AND D_T.*m*=5 AND F.*w*=5
ORDER BY D_2.*l*

Q4 : **SELECT** * **FROM** F, D_1, D_2, D_T, D_3
WHERE AND F.D_1id=D_1.D_1id AND F.D_2id=D_2.D_2id AND F.D_3id=D_3.D_3id AND
F.$D_T$id= D_T.$D_T$id AND D_1.*a*=1 AND D_1.*b*=2 AND D_2.*e*=3 AND D_2.*f*=4 AND D_T.*m*=5
AND D_3.*r*=6 AND D_3.*s*=7 **ORDER BY** D_2.*l*

Q5 : **SELECT** * **FROM** F, D_1, D_2, D_3
WHERE AND F.D_1id=D_1.D_1id AND F.D_2id=D_2.D_2id AND F.D_3id=D_3.D_3id
AND D_1.*b*=2 AND D_2.*e*=3 AND D_2.*f*=4 AND D_3.*r*=5 AND D_3.*s*=6 **ORDER BY** D_2.*l*

ภาพประกอบ 3-5 ตัวอย่างชุดคำสั่งสอบถามของคลังข้อมูล

เมื่อดำเนินการสกัดแอททริบิวต์ทั้งหมดบนตาราง Dimension ที่ปรากฏในชุดคำสั่งสอบถามหลังเงื่อนไข WHERE, ORDER BY และ GROUP BY และเป็นแอททริบิวต์ที่อยู่ในตารางที่สกัดได้จากขั้นตอนที่ 1.1) ของ T_β ได้ผลลัพธ์ดังตาราง 3-2 จะเห็นได้ว่าแอททริบิวต์ {*v, w*} ซึ่งเป็นแอททริบิวต์ที่อยู่ในตาราง Fact จะไม่ถูกคัดเลือกเป็น Candidate Attribute และแอททริบิวต์ {*r, s*} ซึ่งอยู่ในตาราง D_3 จะไม่ถูกคัดเลือกเป็น Candidate Attribute สามารถเขียนแทนด้วยเซตแบบแจกแจงสมาชิกได้ ดังนี้

U = {*a, c, d, e, k, v, w, b, g, m, r, s, h, l, f*}

A_1 = {*a, c, d, e, k, b, g, m, h, l, f*}

ตาราง 3-2 แอททริบิวต์ทั้งหมดที่สกัดได้จากขั้นตอนการสกัดแอททริบิวต์จาก Workload

คำสั่งสอบถาม	แอททริบิวต์ทั้งหมดที่สกัดได้จาก Workload
1	*a, c, d, e, k*
2	*b, c, g, k, m*
3	*b, c, d, h, l, m*
4	*a, b, e, f, l, m*
5	*b, e, f, l*

ตาราง 3-3 การสกัดแอททริบิวต์ที่มีคาร์ดินอลิตี้ต่ำ

แอททริบิวต์	คาร์ดินอลิตี้
a	12
c	5
d	24
e	53
k	**1000**
b	31
g	5
m	5
h	64
l	**366**
f	40

(ก) สกัดจำนวนคาร์แอททริบิวต์(A_1)ทั้งหมด

→

แอททริบิวต์	คาร์ดินอลิตี้
a	12
c	5
d	24
e	53
b	31
g	5
m	5
h	64
f	40

(ข) ผลลัพธ์ในการสกัดแอททริบิวต์ที่มีคาร์ดินอลิตี้น้อยกว่าหรือเท่ากับ 64 (α=64)

นำแอททริบิวต์ทั้งหมดที่ได้จากขั้นตอนการสกัดแอททริบิวต์ (A_1) จากตาราง 3-2 มาสกัดจำนวนคาร์ดินอลิตี้จาก Metadata ของคลังข้อมูล จะได้ผลลัพธ์ตาราง 3.3 (ก) และเมื่อกำหนดคาร์ดินอลิตี้สูงสุดที่ต้องการไม่เกิน 64 (α=64) จะได้ผลลัพธ์ดังตาราง 3-3 (ข) จะเห็นได้ว่าแอททริบิวต์ *{k, l}* มีจำนวนคาร์ดินอลิตี้สูงกว่าที่กำหนดจึงถูกคัดออกไป ดังนั้นจำนวน Candidate Attribute ทั้งหมดที่ได้จากขั้นตอนนี้แสดงดังตาราง 3-4 สามารถเขียนแทนด้วยเซตแบบแจกแจงสมาชิกได้ดังนี้

A_α = {*a, c, d, e, b, g, m, h, f*}

ตาราง 3-4 แอททริบิวต์ทั้งหมดที่ได้จากขั้นตอนการสกัดแอททริบิวต์ที่มีคาร์ดินอลิตี้ต่ำ

คำสั่งสอบถาม	แอททริบิวต์ทั้งหมดที่สกัดได้จาก Workload
1	*a, c, d, e*
2	*b, c, g, m*
3	*b, c, d, h, m*
4	*a, b, e, f, m*
5	*b, e, f*

ตาราง 3-5 ตัวอย่าง BJI Knowledge Base

แอททริบิวต์	พื้นที่ที่ใช้(S_i)
m	**1**
p	2
q	1
r	3

ตาราง 3-6 แอททริบิวต์ทั้งหมดที่ได้จากขั้นตอน BJI Knowledge Base Filtering

คำสั่งสอบถาม	แอททริบิวต์
1	*a, c, d, e*
2	*b, c, g*
3	*b, c, d, h*
4	*a, b, e, f*
5	*b, e, f*

2) การลดจำนวน Candidate Attribute

2.1) การกรองแอททริบิวต์ที่ไม่ปรากฏใน BJI Knowledge Base

การกรองแอททริบิวต์ที่ปรากฏอยู่ใน BJI Knowledge Base (ตาราง 3-4) ออกจาก Candidate Attribute และเก็บแอททริบิวต์นั้นเป็นเซตคำตอบของชุดดัชนีที่ถูกเลือก (I) ทันที

จากตาราง 3-5 จะพบว่าแอททริบิวต์ *m* นั้นเป็นสมาชิกของ BJI Knowledge Base ดังนั้นจะตัดแอททริบิวต์ *m* ซึ่งไม่จำเป็นต้องนำไปคำนวณในขั้นตอนถัดไป

ออกจาก Candidate Attribute ก็จะช่วยลดเวลาที่ใช้ในการคำนวณได้ ดังนั้นจะได้แอททริบิวต์ที่นำมาผ่านขั้นตอนการคัดเลือกดังตาราง 3-6

จากขั้นตอนข้างต้นสามารถเขียนแทนด้วยเซตแบบแจกแจงสมาชิกได้ ดังนี้

$K = \{m, p, q, r\}$

$I = \{ x \in U \mid x \in A \cap x \in K \}$ ดังนั้น จะได้ $I = \{m\}$

$A_2 = A_\alpha - I$ ดังนั้น จะได้ $A_2 = \{a, c, d, e, b, g, h, f\}$

2.2) การกรองแอททริบิวต์ที่ถูกสอบถามบ่อย

กรองแอททริบิวต์ที่ถูกสอบถามบ่อยจาก Candidate Attribute ทั้งหมดที่ได้จากขั้นตอน 2.1) เมื่อกำหนดให้ขีดแบ่งค่าสนับสนุนเท่ากับ 40 % (δ =40% หรือ 0.4) จากตัวอย่างในตาราง 3-7 (ก) จะได้ว่าแอททริบิวต์ที่ไม่ผ่านเงื่อนไขที่กำหนด ได้แก่ แอททริบิวต์ {*g, h*} จึงไม่ถูกคัดเลือกมาสร้างดัชนี ดังนั้นจะได้ผลลัพธ์ดังตาราง 3-7 (ข) ซึ่งจะเป็น Candidate Attribute ที่จะนำไปสร้างดัชนี สามารถเขียนแทนด้วยเซตแบบแจกแจงสมาชิกได้ ดังนี้

$A_\delta = \{a, c, d, e, b, f\}$

เมื่อจบขั้นตอนที่ 2 แล้วจะได้จำนวน Candidate Attribute ทั้งหมด ซึ่งมีคุณสมบัติที่จะนำไปใช้ในขั้นตอนการคำนวณทรัพยากรที่ใช้ในการสร้างดัชนีแต่ละตัว คือ จำนวนคาร์ดินอลิตี้ของแต่ละแอททริบิวต์ (จากขั้นตอน 1.2) และค่าสนับสนุนในการสอบถามของแอททริบิวต์แต่ละตัว (จากขั้นตอน 2.2) ดังแสดงในตาราง 3-8

ตาราง 3-7 การสกัดแอททริบิวต์ที่มีค่าสนับสนุนในการสอบถามมากกว่าหรือเท่ากับขีดแบ่งค่าสนับสนุนที่กำหนด

แอททริบิวต์	ค่าสนับสนุน
a	0.4
c	0.6
d	0.4
e	0.6
b	0.8
g	**0.2**
h	**0.2**
f	0.4

(ก) สกัดค่าสนับสนุนของแต่ละแอททริบิวต์ (A_α)

→

แอททริบิวต์	ค่าสนับสนุน
a	0.4
c	0.6
d	0.4
e	0.6
b	0.8
f	0.4

(ข) ผลลัพธ์ในการสกัดแอททริบิวต์ที่มีค่าสนับสนุนในการสอบถามมากกว่าหรือเท่ากับขีดแบ่งค่าสนับสนุน 40% (δ =40% หรือ 0.4)

ตาราง 3-8 คุณสมบัติของ Candidate Attribute ทั้งหมด

ตาราง	แอททริบิวต์ (A_i)	คาร์ดินอลิตี้ (C_i)	ค่าสนับสนุน(sA_i)
D_1	*a*	12	0.4
D_1	*c*	5	0.6
D_1	*d*	24	0.4
D_2	*e*	53	0.6
D_1	*b*	31	0.8
D_2	*f*	40	0.4

จากตาราง 3-8 แอททริบิวต์ {*a, b, c, d*} อยู่ในตาราง Dimension1 (D_1) และแอททริบิวต์ {*e, f*} อยู่ในตาราง Dimension2 (D_2) สามารถประเมินทรัพยากรที่ใช้ในการสร้างดัชนี และค่าใช้จ่ายที่ใช้ในการจับคู่ข้อมูลได้ดังขั้นตอนถัดไป

จากขั้นตอนในการสกัด และการกรอง Candidate Attribute ที่จะนำมาสร้างดัชนีนั้น เพื่อให้สามารถเข้าใจได้ง่าย จะสรุปเป็นแผนภาพเวนน์-ออยเลอร์ ได้ดังภาพประกอบ 3-6 โดยที่

U แทนเซตของแอททริบิวต์ทั้งหมดที่ปรากฏในชุดคำสั่งสอบถามข้อมูลในอดีต

A_1 แทนเซตของแอททริบิวต์ทั้งหมดที่จะนำมาสร้างดัชนี Bitmap Join Index

A_α แทนเซตของ Candidate Attribute ที่มีคาร์ดินอลิตี้น้อยกว่าหรือเท่ากับค่าที่กำหนด (α)

A_2 แทนเซตของ Candidate Attribute ที่ไม่รวมแอททริบิวต์ที่ปรากฏใน BJI Knowledge Base

A_δ แทนเซตของ Candidate Attribute ที่มีค่าสนับสนุนในการสอบถามมากกว่าหรือเท่ากับขีดแบ่งค่าสนับสนุนที่กำหนด (δ)

เซตของผลลัพธ์ที่ได้จากขั้นตอนทั้งหมด แสดงได้ดังนี้

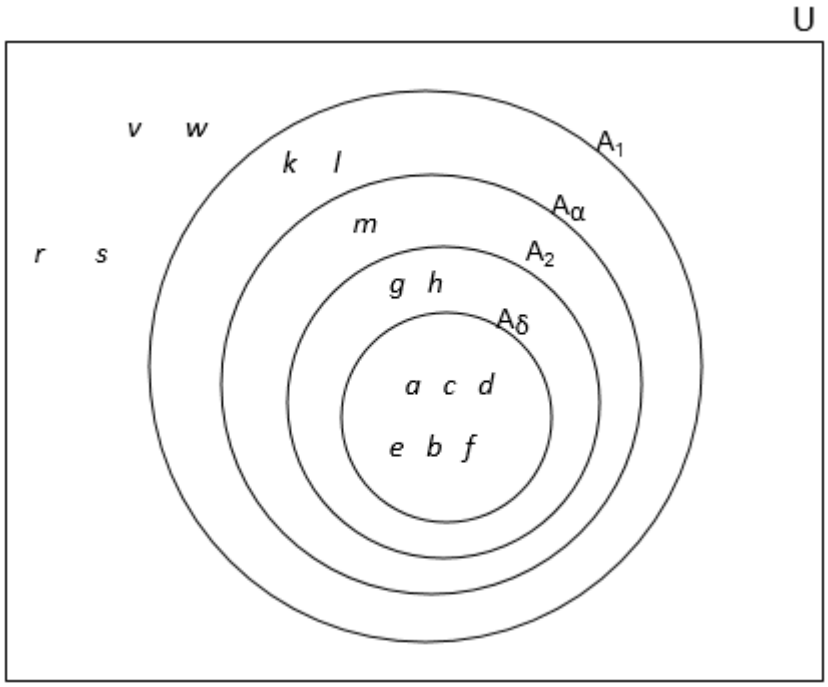
U = {*a, c, d, e, k, v, w, b, g, m, r, s, h, l, f*}

A_1 = {a, c, d, e, k, b, g, m, h, l, f}

A_α = {a, c, d, e, b, g, m, h, f}

A_2 = {a, c, d, e, b, g, h, f}

A_δ = {a, c, d, e, b, f}



ภาพประกอบ 3-6 แผนภาพเวนน์-ออยเลอร์ของเซตคำตอบที่ได้จากขั้นตอนการสกัดและการกรอง Candidate Attribute

3) การประเมินทรัพยากรที่ใช้ในการสร้างดัชนีสำหรับแอททริบิวต์แต่ละตัว

3.1) สร้าง Candidate Attribute Table (CAT)

เป็นตารางที่ใช้เก็บคุณสมบัติของแอททริบิวต์แต่ละตัว ได้แก่ พื้นที่ที่ใช้ในการสร้างดัชนี (S_i) และค่าความเหมาะสมในการสร้างดัชนีของแต่ละแอททริบิวต์ (F_i) เพื่อนำไปใช้ในขั้นตอนการเลือกดัชนี (Index Selection)

3.2) คำนวณพื้นที่ที่ใช้ในการสร้างดัชนี Bitmap Join Index ของแอททริ-บิวต์แต่ละตัว

พื้นที่ที่ใช้ในการสร้างดัชนี Bitmap Join Index สามารถคำนวณได้จากสมการ 3.2 จากนั้นนำผลลัพธ์ที่ได้ทั้งหมดไปเก็บใน CAT เพื่อใช้เป็นข้อมูลนำเข้าในขั้นตอนการเลือกดัชนี

3.3) คำนวณหาค่าใช้จ่ายที่ใช้ในการจับคู่ข้อมูลระหว่างตาราง Fact และ Dimension

การคำนวณค่าใช้จ่ายที่ใช้ในการจับคู่ข้อมูลระหว่างตาราง Fact และ Dimension ของแต่ละแอททริบิวต์ สามารถดำเนินการตามขั้นตอน ดังนี้

3.3.1) คำนวณจำนวนบล็อคใช้ในการเก็บข้อมูลของแอททริบิวต์ที่จะนำมา จับคู่ข้อมูลซึ่งจะนำไปใช้ในขั้นตอนการคำนวนค่าใช้จ่ายในการจับคู่ข้อมูล (จากสมการ 3.1) กำหนดให้ขนาดของแอททริบิวต์ *a, c, d, e, b, f* เท่ากับ 2, 4, 6, 4, 8, 8 bytes ตามลำดับ แสดงตัวอย่างดังตาราง 3-9

ตาราง 3-9 การคำนวณหาจำนวนบล็อคที่ใช้ในการเก็บข้อมูลของแต่ละแอททริบิวต์

แอททริบิวต์ (A_i)	จำนวนแถว ($\|T_{Ai}\|$)	ขนาดของแอท-ทริบิวต์ (L_{Ai})	จำนวน Block ที่ใช้ในการเก็บข้อมูล (b_{Ai})
a	10,000	2	$\left\lceil \frac{10,000 \times 2}{4,096} \right\rceil = 5$
c	10,000	4	$\left\lceil \frac{10,000 \times 4}{4,096} \right\rceil = 10$
d	10,000	6	$\left\lceil \frac{10,000 \times 6}{4,096} \right\rceil = 15$
e	20,000	4	$\left\lceil \frac{20,000 \times 4}{4,096} \right\rceil = 20$
b	10,000	8	$\left\lceil \frac{10,000 \times 8}{4,096} \right\rceil = 20$
f	20,000	8	$\left\lceil \frac{20,000 \times 8}{4,096} \right\rceil = 40$

3.3.2) คำนวณค่าใช้จ่ายในการจับคู่ข้อมูล ของแต่ละแอททริบิวต์ โดยใช้สมการ 3-3-3.5 จากนั้นนำผลลัพธ์ที่ได้จากขั้นตอนนี้ (*TotalCost*) ซึ่งเป็นการประเมินค่าใช้จ่ายในการจับคู่ข้อมูล โดยใช้วิธีการ Sort-Merge Join ไปคำนวณหาค่าความเหมาะสมในการสร้างดัชนีของแต่ละแอททริบิวต์

3.4) คำนวณหาค่าความเหมาะสมในการสร้างดัชนีของแต่ละแอททริบิวต์

ค่าความเหมาะสมในการสร้างดัชนีสามารถคำนวณได้จากสมการ 3.6 จากนั้นจะดำเนินการแมปปิง (Mapping) จากค่าจำนวนจริงที่อยู่ในช่วงข้อมูลที่กว้างมากให้เป็น

จำนวนเต็มที่อยู่ในช่วงข้อมูลที่แคบลง เพื่อลดเวลาในการคำนวณของขั้นตอนการเลือกดัชนี ซึ่งผลลัพธ์ที่ได้ทั้งหมดจะนำไปเก็บใน CAT เพื่อใช้เป็นข้อมูลนำเข้าในขั้นตอนการเลือกดัชนี ตัวอย่างการคำนวณหาค่าความเหมาะสมแสดงในตาราง 3-10

ตาราง 3-10 การคำนวณหาค่าความเหมาะสมในการสร้างดัชนีของแต่ละแอททริบิวต์

แอททริบิวต์ (A_i)	ค่าสนับสนุน (sA_i)	ค่าใช้จ่ายในการ Join ($TotalCost_i$)	$sA_i \times TotalCost_i$ (F_i)
a	0.4	8	3
c	0.6	7	4
d	0.4	13	5
e	0.6	9	6
b	0.8	9	7
f	0.4	20	8

ตาราง 3-11 Candidate Attribute Table

แอททริบิวต์	พื้นที่ที่ใช้(S_i)	ค่าความเหมาะสม(F_i)
a	2	3
c	1	4
d	3	5
e	6	6
b	4	7
f	5	8

ตาราง 3-11 คือ CAT ที่ใช้สำหรับเก็บข้อมูลพื้นที่ที่ใช้ในการสร้างดัชนี Bitmap Join Index ของแอททริบิวต์ *i* (S_i) และค่าความเหมาะสมของแอททริบิวต์ *i* (F_i)

4) การเลือกดัชนีที่เหมาะสมที่สุดสำหรับทรัพยากรที่จำกัด

นำข้อมูลจาก CAT มาผ่านขั้นตอนการคัดเลือกดัชนีโดยใช้การแก้ปัญหาแบบพลวัต เพื่อให้ได้ชุดดัชนีที่เหมาะสมที่สุด

จากตัวอย่างกำหนดให้พื้นที่ในการจัดเก็บดัชนีทั้งหมด(S) เท่ากับ 7 และจำนวน Candidate Attribute (n) เท่ากับ 6 ซึ่งจากขั้นตอนที่ 1 จะได้ชุดดัชนีที่ถูกคัดเลือกแล้ว 1

แอททริบิวต์ คือ $I = \{m\}$ ซึ่งใช้พื้นที่ในการจัดเก็บดัชนีเท่ากับ 1 (จากตาราง 3-5) ดังนั้นในขั้นตอนนี้จึงคำนวณโดยใช้พื้นที่ในการจัดเก็บดัชนีทั้งหมด เท่ากับ 7-1 เท่ากับ 6 จากนั้นดำเนินการคัดเลือกดัชนีโดยใช้อัลกอริทึม BJI Selection เมื่อดำเนินการเสร็จเรียบร้อยแล้วจะได้ผลลัพธ์ดังตาราง 3-12 ซึ่งแสดงมูลค่าความเหมาะสมที่สูงที่สุดที่เป็นไปได้ ซึ่งเท่ากับ 12 จากนั้นในการหาชุดดัชนีที่ถูกเลือกได้ดังตาราง 3-1

ตาราง 3-12 ผลลัพธ์จากการทำงานของฟังก์ชัน BJISelection

			i \ j	0	1	2	3	4	5	6
Attribute	S_i	F_i	0	0	0	0	0	0	0	0
a	2	3	1	0	0	3	3	3	3	3
c	1	4	2	0	4	4	4	7	7	7
d	3	5	3	0	4	4	7	9	9	12
e	6	6	4	0	4	4	7	9	9	12
b	4	7	5	0	4	4	7	9	11	12
f	5	8	6	0	4	4	7	9	11	12

ตาราง 3-13 ผลลัพธ์จากการทำงานของฟังก์ชัน FindIndexSet

			i \ j	0	1	2	3	4	5	6	
Attribute	S_i	F_i	0	0	0	0	0	0	0	0	**i=0,j=2**
(*a*)	2	3	1	0	0	3	3	3	3	3	i=1, j=2
(*c*)	1	4	2	0	4	4	4	7	7	7	i=2, j=3
(*d*)	3	5	3	0	4	4	7	9	9	12	i=3, j=6
e	6	6	4	0	4	4	7	9	9	12	i=4, j=6
b	4	7	5	0	4	4	7	9	11	12	i=5, j=6
f	5	8	6	0	4	4	7	9	11	12	i=6, j=6

ตัวอย่างการทำงานของขั้นตอนที่ 4 สามารถแสดงได้ดังตาราง 3-13 เริ่มต้นโดยให้ค่าตัวแปร $i = 6$ และตัวแปร $j = 6$

i = 6 และ j = 6, เปรียบเทียบค่าที่ตำแหน่ง (6, 6) กับค่าที่ตำแหน่ง (5, 6) พบว่ามีค่าเท่ากันก็จะลดค่า i ลง 1

i = 5 และ j = 6, เปรียบเทียบค่าที่ตำแหน่ง (5, 6) กับค่าที่ตำแหน่ง (4, 6) พบว่ามีค่าเท่ากันก็จะลดค่า i ลง 1

i = 4 และ j = 6, เปรียบเทียบค่าที่ตำแหน่ง (4, 6) กับค่าที่ตำแหน่ง (3, 6) พบว่ามีค่าเท่ากันก็จะลดค่า i ลง 1

i = 3 และ j = 6, เปรียบเทียบค่าที่ตำแหน่ง (3, 6) กับค่าที่ตำแหน่ง (2, 6) จะเห็นว่าค่าที่ตำแหน่ง (3, 6) = 12 และค่าที่ตำแหน่ง (2, 6) = 7 ซึ่งไม่เท่ากัน ดังนั้นจึงเก็บแอททริบิวต์ d ในชุดดัชนีที่เป็นคำตอบ จากนั้นให้ค่า $j = j\text{-}S_3$(6-3) ซึ่งเท่ากับ 3 และลดค่า i ลง 1

i = 2 และ j = 3, เปรียบเทียบค่าที่ตำแหน่ง (2, 3) กับค่าที่ตำแหน่ง (1, 3) จะเห็นว่าค่าที่ตำแหน่ง (2, 3) = 4 กับค่าที่ตำแหน่ง (1, 3) = 3 ซึ่งไม่เท่ากัน ดังนั้นจึงเก็บแอททริบิวต์ c ในชุดดัชนีที่เป็นคำตอบ จากนั้นให้ค่า $j = j\text{-}S_2$(3-1) ซึ่งเท่ากับ 2 และลดค่า i ลง 1

i = 1 และ j = 2, เปรียบเทียบค่าที่ตำแหน่ง (1, 2) กับค่าที่ตำแหน่ง (0, 2) จะเห็นว่าค่าที่ตำแหน่ง (1, 2) = 3 กับค่าที่ตำแหน่ง (0, 2) = 0 ซึ่งไม่เท่ากัน ดังนั้นจึงเก็บแอททริบิวต์ a ในชุดดัชนีที่เป็นคำตอบ จากนั้นให้ค่า $j = j\text{-}S_1$(2-1) ซึ่งเท่ากับ 1 และลดค่า i ลง 1

i = 0 และ j = 1, ไม่เป็นไปตามเงื่อนไข j>0 && i>0 จึงออกจากการทำงาน และคืนค่าชุดดัชนี {d, c, a} เป็นคำตอบสำหรับขั้นตอนการเลือกดัชนี

จากนั้นรวมชุดดัชนีที่ได้จากขั้นตอน 2.1) (BJI Knowledge Base Filtering) จะได้ผลลัพธ์ทั้งหมด สามารถเขียนแทนด้วยเซตแบบแจกแจงสมาชิกได้ดังนี้

$$I = \{m\} \cup \{d, c, a\} = \{m, d, c, a\}$$

ดังนั้นสำหรับพื้นที่ในการจัดเก็บดัชนีเท่ากับ 7 จะได้ชุดดัชนี {m, d, c, a} เป็นคำตอบที่เหมาะสมที่สุดในการนำไปสร้างดัชนี Bitmap Join Index สำหรับคลังข้อมูลเพื่อช่วยลดเวลาในการสอบถามข้อมูล

3.3 ข้อเด่นของระบบการคัดเลือกดัชนี Bitmap Join Index สำหรับโครงสร้างแบบ Star Schema

1) ในขั้นตอนการสกัดเลือก Candidate Attribute มีการสกัดแอททริบิวต์ที่มีความเหมาะสมในการสร้างดัชนีน้อยออกไปตั้งแต่ขั้นตอนแรกๆ ดังนั้นจะช่วยลดเวลาในการประมวลผลในขั้นตอนอื่นได้

2) ในการเลือกดัชนีที่จะนำมาสร้างนั้น ได้เลือกสร้างดัชนีบนแอททริบิวต์เดียว ดังนั้นจะลดเวลาในขั้นตอนการสกัดแอททริบิวต์ที่ปรากฏบ่อย และจะได้จำนวนชุดแอททริบิวต์มากกว่าวิธีการที่ใช้ Frequent Itemset

3) การสร้างดัชนี Bitmap Join Index บนแอททริบิวต์เดียว (มีการจับคู่ข้อมูลระหว่าง Fact และ Dimension บนแอททริบิวต์เดียว) จะมีความยืดหยุ่นในการสอบถามมากกว่าการสร้างดัชนีบนแอททริบิวต์ร่วม (มีการจับคู่ข้อมูลระหว่าง Fact และ Dimension บนหลายแอททริบิวต์) เนื่องจากโอกาสที่จะสร้างดัชนีได้ตรงกับคำสั่งสอบถามมีมากกว่า โดยใช้การดำเนินการทางตรรกะเข้ามาช่วยในการสอบถามข้อมูล เช่น กรณีที่สร้างดัชนีบนแอททริบิวต์เดียว ได้แก่ {*a, b, c, d, f*} และกรณีที่สร้างดัชนีบนแอททริบิวต์ร่วม {*ab, ac, adf*} หากผู้ใช้มีการสอบถามตรงกับแอททริบิวต์ {*a, b, ab, ac, adf*} กรณีที่สร้างดัชนีบนแอททริบิวต์ร่วมก็จะให้ผลลัพธ์ได้ดีกว่า แต่หากผู้ใช้มีการสอบถามนอกเหนือจากชุดดัชนีที่สร้างได้ เช่น {*a, c, acd, af, ad*} กรณีที่สร้างดัชนีบนแอททริบิวต์เดียวก็จะให้ผลลัพธ์ได้ดีกว่า เนื่องจากสามารถนำดัชนีแต่ละชุดมาดำเนินการ AND OR เพื่อเพิ่มประสิทธิภาพในการสอบถามได้

4) มีการเก็บข้อมูลที่เป็นประโยชน์เพื่อนำมาใช้ในขั้นตอนการสกัด Candidate Attribute เพื่อช่วยลดเวลาในการคำนวณที่ไม่จำเป็น นั่นคือ การสร้าง BJI Knowledge Base ดังนั้น แอททริบิวต์ที่มักจะถูกเลือกเสมอก็ไม่จำเป็นต้องนำมาพิจารณา

5) ในการเลือกดัชนีได้คำนึงถึงปัจจัยค่าใช้จ่ายที่ใช้ในการจับคู่ข้อมูลระหว่างตาราง ในกรณีที่ไม่มีสร้างดัชนี เนื่องจากหากเลือกสร้างดัชนีบนแอททริบิวต์ที่มีค่าใช้จ่ายในการจับคู่ข้อมูลที่สูงจะช่วยลดเวลาในระหว่างการประมวลผลได้มาก

6) นำเทคนิคการแก้ปัญหาแบบพลวัตมาประยุกต์ใช้ในการคัดเลือกดัชนี ซึ่งมีความยืดหยุ่นเมื่อต้องการปรับลดขนาดของพื้นที่ที่ใช้ในการจัดเก็บ เนื่องจากมีการเก็บผลลัพธ์ของปัญหาย่อยเอาไว้แล้ว

3.4 ข้อด้อยของระบบการคัดเลือกดัชนี Bitmap Join Index สำหรับโครงสร้างแบบ Star Schema

1) ในการสอบถามข้อมูลที่มีเงื่อนไขการสอบถามที่ซับซ้อนมากๆ จะต้องใช้เวลาในการดำเนินการทางตรรกะ เช่น AND OR NOT XOR มาก เนื่องจากสร้างดัชนีบนแอททริบิวต์เดียว

2) ในการพิจารณาการเลือกสร้างดัชนีนั้นใช้แนวโน้มจากข้อมูลคำสั่งสอบถามที่ผ่านมาในอดีต ดังนั้นดัชนีที่คัดเลือกได้จึงขึ้นอยู่กับคำสั่งสอบถามในอดีตนั้น ซึ่งในบางครั้งอาจไม่ตรงกับความต้องการของผู้ใช้เสมอไป

3) การประเมินค่าใช้จ่ายในการจับคู่ข้อมูลขึ้นอยู่กับเทคนิคการจับคู่ข้อมูลแบบ Sort-Merge Join เพียงอย่างเดียว ซึ่งในระบบจัดการฐานข้อมูลต่าง ๆ อาจมีการใช้การเทคนิค

การจับคู่ข้อมูลแบบอื่นด้วย เช่น Hash Join เป็นต้น ซึ่งเทคนิคแต่ละชนิดก็จะมีการคิดค่าใช้จ่ายแตกต่างกันไป

บทที่ 4

ผลการทดลองและวิจารณ์

บทนี้กล่าวถึงข้อมูลที่ใช้ทดสอบขั้นตอนวิธี เครื่องมือที่ใช้ในการทดสอบ และผลการทดลองเพื่อวัดประสิทธิภาพการทำงานในด้านพื้นที่ที่ใช้ในการสร้างดัชนี และเวลาในการสอบถามข้อมูลของระบบแนะนำการคัดเลือกดัชนี Bitmap Join Index สำหรับโครงสร้างแบบ Star Schema

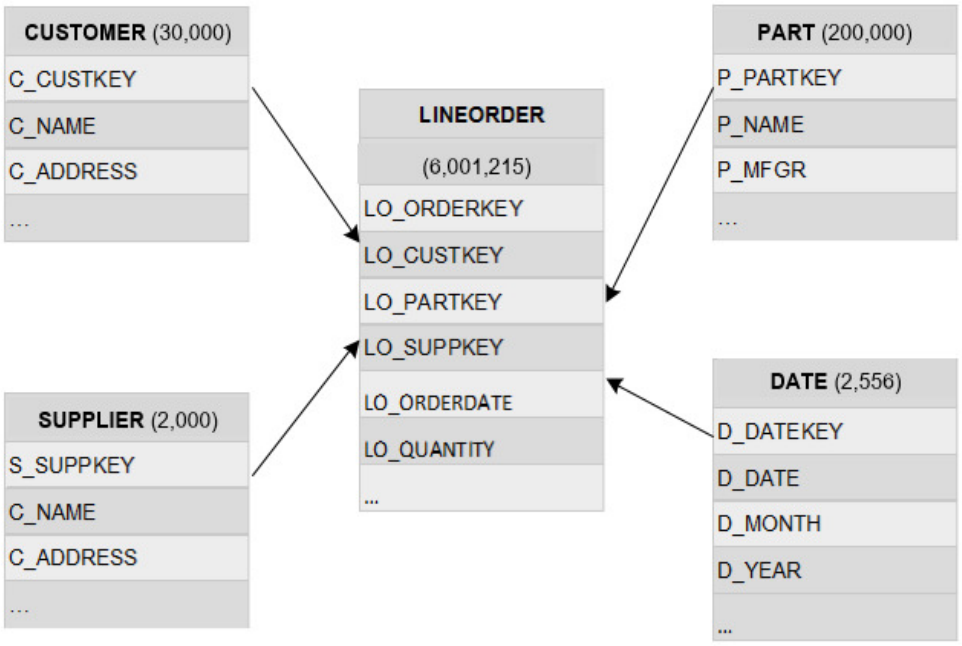
4.1 ข้อมูลที่ใช้ทดสอบขั้นตอนวิธี

ข้อมูลที่ใช้สำหรับการทดสอบเป็นชุดข้อมูลมาตรฐานจาก SSB Benchmark (Star Schema Benchmark, 2009) ซึ่งประกอบด้วยตาราง Fact 1 ตาราง ได้แก่ ตาราง LINEORDER ซึ่งมีจำนวนเรคอร์ด เท่ากับ 6,001,215 เรคอร์ด และตาราง Dimension ทั้งหมด 4 ตารางได้แก่ ตาราง CUSTOMER ซึ่งประกอบด้วย 8 แอททริบิวต์ และจำนวนเรคอร์ดเท่ากับ 30,000 เรคอร์ด ตาราง SUPPLIER ซึ่งประกอบด้วย 7 แอททริบิวต์ และจำนวนเรคอร์ดเท่ากับ 2,000 เรคอร์ด ตาราง PART ซึ่งประกอบด้วย 9 แอททริบิวต์ และจำนวนเรคอร์ดเท่ากับ 200,000 เรคอร์ด และ ตาราง DATE ซึ่งประกอบด้วย 17 แอททริบิวต์ และจำนวน เรคอร์ด เท่ากับ 2,556 เรคอร์ด ดังแสดงในภาพประกอบ 4.1

สำหรับชุดคำสั่งสอบถามที่ใช้ในการทดลองจะคัดเลือกจากคำสั่งสอบถามตัวอย่างของ SSB Benchmark โดยชุดการสอบถามที่ 1 ประกอบด้วย Q1(Q3.2) Q2(Q3.4) Q3(Q4.1) Q4(Q4.2) และ Q5(4.3) ชุดการสอบถามที่ 2 ประกอบด้วย Q1(Q2.1) Q2(Q3.1) Q3(Q3.2) Q4(Q3.4) และ Q5(Q4.2) ซึ่งรายละเอียดของแต่ละชุดคำสั่งสอบถามแสดงดังภาคผนวก ก.

4.2 เครื่องมือที่ใช้ในการทดสอบ

ในการประมวลผลโปรแกรมสำหรับการเลือกดัชนี และการทดสอบประสิทธิภาพในการประมวลผลการสอบถามข้อมูล จะใช้เครื่องคอมพิวเตอร์จำนวน 1 เครื่อง ซึ่งมีสมรรถนะดังนี้ หน่วยประมวลผลกลาง Intel Core i7 หน่วยความจำขนาด 4 GB และฮาร์ดดิสก์ขนาด 1024 GB ติดตั้งระบบปฏิบัติการ Microsoft Windows 7 และใช้ระบบจัดการฐานข้อมูล Oracle Database 12g สำหรับการจัดการข้อมูลที่ใช้ทดสอบและวัดประสิทธิภาพในการสอบถามของดัชนี



ภาพประกอบ 4-1 ตัวอย่างโครงสร้างของ SSB

4.3 กระบวนการทดลองและผลการทดลอง

4.3.1 กระบวนการทดลอง

1. เตรียมชุดคำสั่งสอบถามที่จะใช้ในการทดลอง โดยคัดเลือกชุดคำสั่งสอบถาม 2 ชุด แต่ละชุดประกอบด้วย 5 คำสั่งสอบถามจากคำสั่งสอบถามตัวอย่างจาก SSB Benchmark ทั้งหมด

2. ประมวลผลโปรแกรมการคัดเลือกดัชนีจากชุดคำสั่งสอบถามที่เตรียมไว้ ซึ่งจะได้ชุดดัชนีที่มีความเหมาะสมกับทรัพยากรที่กำหนด

3. นำชุดดัชนีที่ถูกคัดเลือกมาทดสอบกับข้อมูลจาก SSB Benchmark ที่เตรียมไว้โดยใช้ระบบการจัดการฐานข้อมูล Oracle Database 12g โดยเวลาที่ได้ในการสอบถามข้อมูลจากระบบจัดการฐานข้อมูลจะใช้เป็นตัววัดประสิทธิภาพของการสอบถาม

4. แบ่งการทดลองออกเป็น 3 กลุ่ม เพื่อใช้ในการเปรียบเทียบ ได้แก่ กลุ่มที่ไม่มีการสร้างดัชนีใดๆ (No Index) กลุ่มที่มีการสร้างดัชนีบนทุกแอททริบิวต์ที่ปรากฏในชุดคำสั่งสอบถาม (Full Index) และ กลุ่มที่สร้างดัชนีที่ได้จากระบบการคัดเลือกดัชนี Bitmap Join Index (BJIS)

5. ในขั้นตอนการสอบถามข้อมูล แต่ละชุดการทดลองจะทดสอบด้วยการใช้คำสั่งสอบถามทั้ง 5 คำสั่งและแต่ละคำสั่งจะถูกสอบถามซ้ำทั้งหมด 5 รอบ จากนั้นหาค่าเฉลี่ยของเวลาที่ใช้ในการประมวลผล และบันทึกเวลาในการสอบถามข้อมูล

4.3.2 ผลการทดลอง

บันทึกผลการทดลอง และเปรียบเทียบผลการทดลองโดยวัดจากประสิทธิภาพด้านเวลาในการสอบถามข้อมูล และประสิทธิภาพด้านการใช้พื้นที่ในการสอบถามข้อมูล

การวัดประสิทธิภาพด้านเวลาในการสอบถามข้อมูล สามารถพิจารณาได้จากการนำดัชนีที่ถูกคัดเลือกจากระบบเข้ามาใช้ในการสอบถามแล้วทำให้การสอบถามรวดเร็วยิ่งขึ้น นั่นหมายความว่าสามารถเลือกสร้างดัชนีที่ได้อย่างถูกต้องตรงกับความต้องการ หากดัชนีที่ถูกเลือกนั้นไม่ได้ช่วยเพิ่มประสิทธิภาพในการสอบถาม นั่นหมายความว่าระบบไม่สามารถคัดเลือกดัชนีได้อย่างเหมาะสม

การวัดประสิทธิภาพด้านการใช้พื้นที่ในการจัดเก็บดัชนี สามารถพิจารณาได้จากขนาดของพื้นที่ที่ใช้ในการจัดเก็บดัชนีที่ถูกคัดเลือกจากระบบคัดเลือกดัชนี เปรียบเทียบกับขนาดของพื้นที่ที่ใช้เมื่อไม่เลือกสร้างดัชนีใดๆ และเมื่อสร้างดัชนีสำหรับทุกแอททริบิวต์ ว่าสามารถใช้พื้นที่ได้อย่างคุ้มค่าหรือไม่

การวัดประสิทธิภาพด้านการลดค่าใช้จ่ายในการประมวลผลของการดำเนินการสอบถามข้อมูล สามารถพิจารณาได้จากจำนวนบล็อคข้อมูลที่ถูกอ่านในระหว่างการประมวลผลการสอบถาม

การวัดประสิทธิภาพในแง่ Space-Time Trade-off ซึ่งในการสร้างดัชนีเพื่อช่วยในการสอบถามนั้น ยิ่งต้องการให้เวลาในการสอบถามเร็วขึ้นเท่าไร ก็จะต้องเสียพื้นที่ที่ใช้ในการจัดเก็บดัชนีมากขึ้นเท่านั้น ซึ่งดัชนีที่มีประสิทธิภาพ คือดัชนีที่สามารถช่วยลดเวลาในการสอบถามได้เป็นอย่างดีโดยใช้พื้นที่ในการเก็บดัชนีที่เหมาะสม

ผลการทดสอบประสิทธิภาพของระบบแนะนำการคัดเลือกดัชนี Bitmap Join Index สำหรับโครงสร้างแบบ Star Schema สามารถแสดงได้ดังนี้

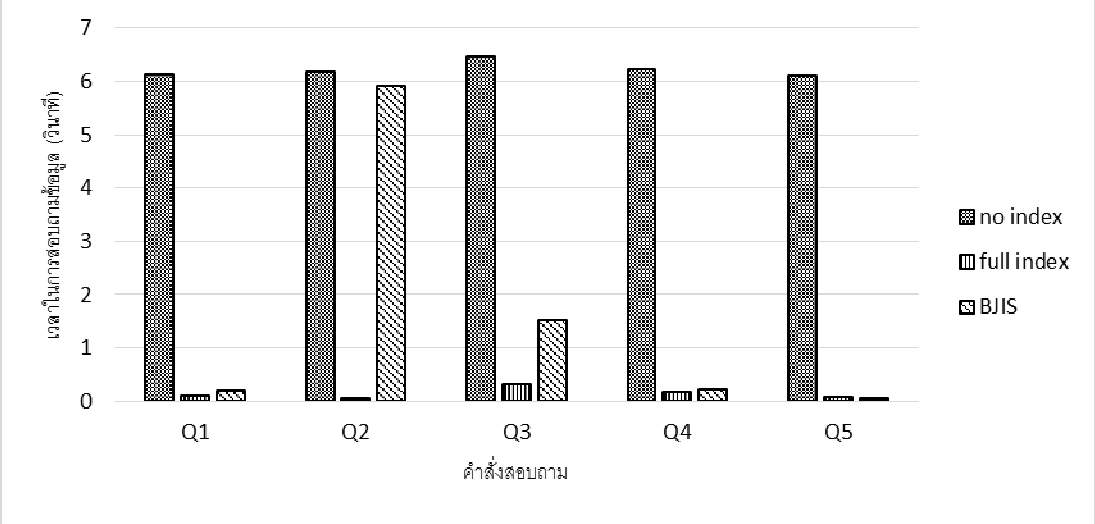
1. ผลการทดลองโดยเปรียบเทียบประสิทธิภาพด้านเวลาในการสอบถามข้อมูล ทั้งหมด 3 กลุ่ม ระหว่างกลุ่มที่ไม่มีการสร้างดัชนีใดๆ กลุ่มที่มีการสร้างดัชนีบนทุกแอท-ทริบิวต์ที่ปรากฏในชุดคำสั่งสอบถาม และกลุ่มที่สร้างดัชนีที่ได้จากระบบการคัดเลือกดัชนี Bitmap Join Index

สำหรับชุดการสอบถามที่ 1 และ 2 สามารถแสดงการเปรียบเทียบผลการทดลองได้ดังตาราง 4-1 และตาราง 4-2 ตามลำดับ และสามารถเขียนเป็นกราฟแสดงการเปรียบเทียบผลการทดลองได้ดังภาพประกอบ 4-2 และ 4-3 ตามลำดับ

ตาราง 4-1 เปรียบเทียบเวลาที่ใช้ในการสอบถามข้อมูลระหว่างกลุ่มที่ไม่มีการสร้างดัชนีใด ๆ กลุ่มที่มีการสร้างดัชนีบนทุกแอททริบิวต์ที่ปรากฏในชุดคำสั่งสอบถาม และกลุ่มที่สร้างดัชนีจากระบบคัดเลือกดัชนี Bitmap Join Index บนชุดการสอบถามที่ 1

คำสั่งสอบถาม	เวลาที่ใช้ในการสอบถาม (วินาที)		
	ไม่สร้างดัชนีใด ๆ (No Index)	สร้างดัชนีบนทุกแอททริบิวต์ (Full Index)	สร้างดัชนีที่เลือกโดยระบบคัดเลือกดัชนี (BJIS)
Q1	6.1308	0.1120	0.1870
Q2	6.1898	0.0440	5.9276
Q3	6.4804	0.3240	1.5256
Q4	6.2400	0.1622	0.2276
Q5	6.1160	0.0780	0.0620

สามารถนำข้อมูลผลการทดลองจากตาราง 4-1 มาเขียนในรูปแบบกราฟได้ดังภาพประกอบ 4-2

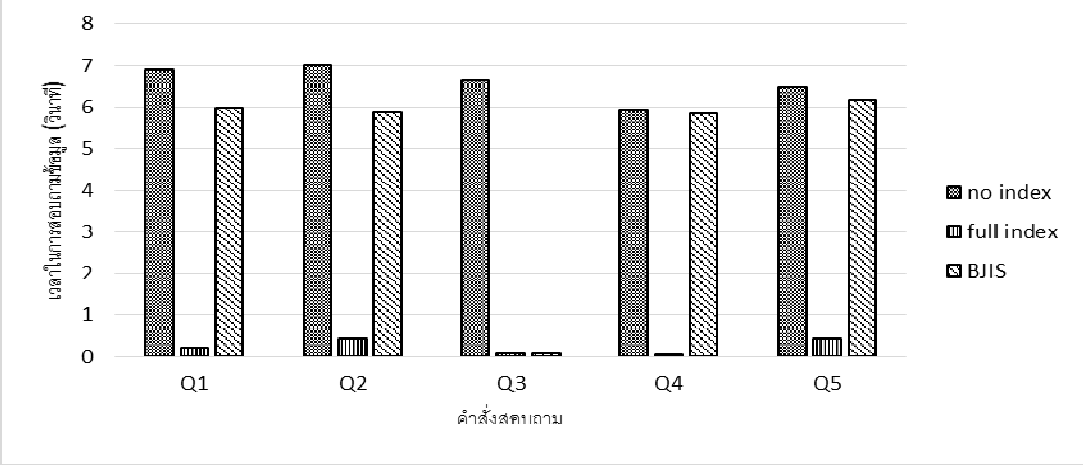


ภาพประกอบ 4-2 กราฟเปรียบเทียบเวลาที่ใช้ในการสอบถามข้อมูลระหว่างกลุ่มที่ไม่มีการสร้างดัชนีใด ๆ กลุ่มที่มีการสร้างดัชนีบนทุกแอททริบิวต์ที่ปรากฏในชุดคำสั่งสอบถาม และกลุ่มที่สร้างดัชนีจากระบบคัดเลือกดัชนี Bitmap Join Index บนชุดการสอบถามที่ 1

ตาราง 4-2 เปรียบเทียบเวลาที่ใช้ในการสอบถามข้อมูลระหว่างกลุ่มที่ไม่มีการสร้างดัชนีใด ๆ กลุ่มที่มีการสร้างดัชนีบนทุกแอททริบิวต์ที่ปรากฏในชุดคำสั่งสอบถาม และกลุ่มที่สร้างดัชนีจากระบบคัดเลือกดัชนี Bitmap Join Index บนชุดการสอบถามที่ 2

คำสั่งสอบถาม	เวลาที่ใช้ในการสอบถาม (วินาที)		
	ไม่สร้างดัชนีใด ๆ (No Index)	สร้างดัชนีบนทุกแอททริบิวต์ (Full Index)	สร้างดัชนีที่เลือกโดยระบบคัดเลือกดัชนี (BJIS)
Q1	6.8954	0.2292	5.9832
Q2	7.0138	0.4224	5.8914
Q3	6.6570	0.0840	0.0884
Q4	5.9362	0.0496	5.8448
Q5	6.4814	0.4432	6.1646

สามารถนำข้อมูลผลการทดลองจากตาราง 4-2 มาเขียนในรูปแบบกราฟได้ดังภาพประกอบ 4-3



ภาพประกอบ 4-3 กราฟเปรียบเทียบเวลาที่ใช้ในการสอบถามข้อมูลระหว่างกลุ่มที่ไม่มีการสร้างดัชนีใด ๆ กลุ่มที่มีการสร้างดัชนีบนทุกแอททริบิวต์ที่ปรากฏในชุดคำสั่งสอบถาม และกลุ่มที่สร้างดัชนีจากระบบคัดเลือกดัชนี Bitmap Join Index บนชุดการสอบถามที่ 2

จากผลการทดสอบประสิทธิภาพด้านเวลาที่ใช้ในการสอบถามข้อมูลบนชุดการสอบถามที่ 1 ได้ผลลัพธ์ดังนี้ กลุ่มที่การเลือกสร้างดัชนีโดยใช้ขั้นตอนวิธี BJIS และกลุ่มที่สร้างดัชนีสำหรับทุกแอททริบิวต์ ใช้เวลาในการสอบถามข้อมูลของทุกคำสั่งสอบถามน้อยกว่ากลุ่มที่ไม่มีการสร้างดัชนีใดๆ สำหรับกลุ่มที่มีการเลือกสร้างดัชนีที่เลือกโดยขั้นตอนวิธี BJIS นั้น คำสั่งสอบถาม Q1 Q3 Q4 และ Q5 ใช้เวลาในการสอบถามใกล้เคียงกับการสอบถามข้อมูลของกลุ่มที่มีสร้างดัชนีสำหรับทุกแอททริบิวต์ เนื่องจากดัชนีที่ถูกเลือกจากชุดคำสั่งสอบถามในอดีตนั้นตรงกับแอททริบิวต์ที่ถูกสอบถามในคำสั่งสอบถาม Q1 และคำสั่งสอบถาม Q2 ใช้เวลาในการสอบถามข้อมูลมากที่สุดซึ่งใกล้เคียงกับกลุ่มที่ไม่มีการสร้างดัชนีใดๆ เนื่องจากดัชนีที่ถูกเลือกจากชุดคำสั่งสอบถามในอดีตนั้นไม่ตรงกับแอททริบิวต์ที่ถูกสอบถามในคำสั่งสอบถาม Q2

สำหรับการทดสอบประสิทธิภาพด้านเวลาที่ใช้ในการสอบถามข้อมูลบนชุดการสอบถามที่ 2 ได้ผลลัพธ์ดังนี้ กลุ่มที่การเลือกสร้างดัชนีโดยใช้ขั้นตอนวิธี BJIS และกลุ่มที่สร้างดัชนีสำหรับทุกแอททริบิวต์ ใช้เวลาในการสอบถามข้อมูลของทุกคำสั่งสอบถามน้อยกว่ากลุ่มที่ไม่มีการสร้างดัชนีใดๆ สำหรับกลุ่มที่มีการเลือกสร้างดัชนีที่เลือกโดยขั้นตอนวิธี BJIS นั้น คำสั่งสอบถาม Q3 ใช้เวลาในการสอบถามใกล้เคียงกับการสอบถามข้อมูลของกลุ่มที่มีสร้างดัชนีสำหรับทุกแอททริบิวต์ คำสั่งสอบถาม Q1 Q2 Q4 Q5 ใช้เวลาในการสอบถามน้อยกว่ากลุ่มที่ไม่มีการสร้างดัชนีใดๆ เพียงเล็กน้อย เนื่องจากเมื่อพิจารณาชุดคำสั่งสอบถามในอดีตของชุดการสอบถามที่ 2 พบว่าแต่ละแอททริบิวต์มีการสอบถามซ้ำน้อย จึงทำให้แอททริบิวต์มีค่าสนับสนุนมากกว่าขีดแบ่งค่าสนับสนุนที่กำหนด ที่จะถูกเลือกเป็นชุดคำตอบมีจำนวนน้อยกว่าชุดการสอบถามที่ 1 ดังนั้นประสิทธิภาพในการสอบถามจึงด้อยกว่าชุดการสอบถามที่ 1

จากผลการทดลองของชุดการสอบถามทั้ง 2 ชุด สามารถสรุปได้ว่า ระบบคัดเลือกดัชนี Bitmap Join Index ช่วยให้สามารถเลือกดัชนีได้อย่างถูกต้อง เนื่องจากผลการทดสอบปรากฏว่า การสร้างดัชนีที่เลือกโดยใช้ขั้นตอนวิธี BJIS ช่วยลดเวลาในการสอบถามข้อมูลของคำสั่งสอบถามส่วนใหญ่ได้เป็นอย่างดี แต่ในบางคำสั่งสอบถามข้อมูล สามารถลดเวลาในการสอบถามได้เพียงเล็กน้อย เนื่องจากชุดดัชนีที่ถูกเลือกนั้นไม่ตรงกับแอททริบิวต์ที่ถูกสอบถาม

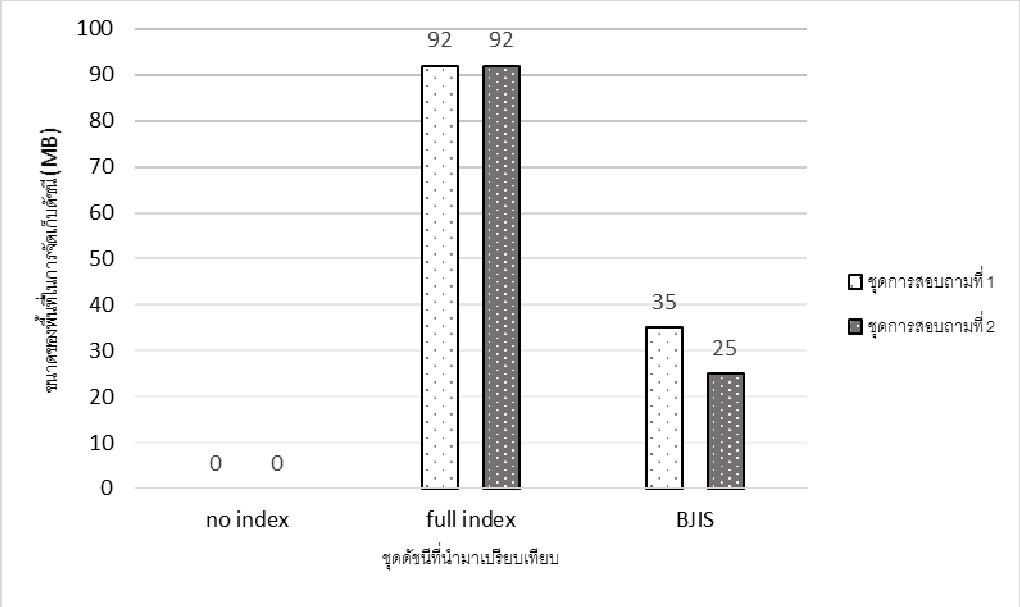
2. ผลการทดลองโดยเปรียบเทียบประสิทธิภาพด้านพื้นที่ที่ใช้ในการสร้างดัชนีทั้งหมด 3 กลุ่ม ระหว่างกลุ่มที่ไม่มีการสร้างดัชนีใดๆ กลุ่มที่มีการสร้างดัชนีบนทุกแอททริบิวต์ที่ปรากฏในชุดคำสั่งสอบถาม และกลุ่มที่สร้างดัชนีที่ได้จากระบบการคัดเลือกดัชนี Bitmap Join Index

สำหรับชุดการสอบถามที่ 1 และ 2 สามารถแสดงการเปรียบเทียบผลการทดลองได้ดังตาราง 4-3 และสามารถเขียนเป็นกราฟแสดงการเปรียบเทียบผลการทดลองได้ดังภาพประกอบ 4-4

ตาราง 4-3 เปรียบเทียบพื้นที่ที่ใช้ในการจัดเก็บดัชนีระหว่างกลุ่มที่ไม่มีการสร้างดัชนีใดๆ กลุ่มที่มีการสร้างดัชนีบนทุกแอททริบิวต์ที่ปรากฏในชุดคำสั่งสอบถาม และกลุ่มที่สร้างดัชนีจากระบบคัดเลือกดัชนี Bitmap Join Index

กลุ่มดัชนี	พื้นที่ที่ใช้ในการจัดเก็บดัชนี (MB)	
	ชุดการสอบถามที่ 1	ชุดการสอบถามที่ 2
ไม่สร้างดัชนีใดๆ (No Index)	0	0
สร้างดัชนีบนทุกแอททริบิวต์ (Full Index)	92	92
สร้างดัชนีที่เลือกโดยระบบคัดเลือกดัชนี (BJIS)	35	25

สามารถนำข้อมูลผลการทดลองจากตาราง 4-3 มาเขียนในรูปแบบกราฟได้ดังภาพประกอบ 4-4



ภาพประกอบ 4-4 กราฟแสดงการเปรียบเทียบพื้นที่ที่ใช้ในการจัดเก็บดัชนีระหว่างกลุ่มที่ไม่มีการสร้างดัชนีใดๆ กลุ่มที่มีการสร้างดัชนีบนทุกแอททริบิวต์ที่ปรากฏในชุดคำสั่งสอบถาม และกลุ่มที่สร้างดัชนีจากระบบคัดเลือกดัชนี Bitmap Join Index

จากผลการทดสอบประสิทธิภาพด้านพื้นที่ที่ใช้ในการจัดเก็บดัชนีบนชุดการสอบถามที่ 1 และชุดการสอบถามที่ 2 ได้ผลลัพธ์ดังนี้ กลุ่มที่ไม่มีการสร้างดัชนีใดๆ ไม่ต้องเสียพื้นที่ในการจัดเก็บดัชนี กลุ่มที่สร้างดัชนีสำหรับทุกแอททริบิวต์ ใช้พื้นที่ในจัดเก็บดัชนีมากที่สุด และกลุ่มที่การเลือกสร้างดัชนีโดยใช้ขั้นตอนวิธี BJIS ใช้พื้นที่ในการจัดเก็บดัชนีมากกว่ากลุ่มที่ไม่มีการสร้างดัชนีใดๆ แต่น้อยกว่ากลุ่มที่สร้างดัชนีสำหรับทุกแอททริบิวต์เนื่องจากมีการจำกัดพื้นที่ที่ใช้ในการจัดเก็บดัชนี ซึ่งในชุดการสอบถามที่ 1 มีชุดดัชนีที่ถูกเลือกมากกว่าชุดการสอบถามที่ 2 ดังนั้นในชุดการสอบถามที่ 1 จะใช้พื้นที่ในการจัดเก็บดัชนีมากกว่าชุดการสอบถามที่ 2

จากผลการทดลองของชุดการสอบถามทั้ง 2 ชุด สามารถสรุปได้ว่า ระบบคัดเลือกดัชนี Bitmap Join Index ช่วยให้สามารถเลือกดัชนีได้อย่างถูกต้องภายใต้พื้นที่การเก็บดัชนีที่จำกัด เนื่องจากในขณะที่มีการใช้พื้นที่ในการจัดเก็บดัชนีน้อยลง ยังสามารถเพิ่มประสิทธิภาพด้านเวลาในการสอบถามในบางคำสั่งสอบถามได้ใกล้เคียงกับกลุ่มที่สร้างดัชนีสำหรับทุกแอททริบิวต์

3. ผลการทดลองโดยเปรียบเทียบประสิทธิภาพด้านการลดค่าใช้จ่ายที่ใช้ในการสร้างดัชนี ทั้งหมด 3 กลุ่ม ระหว่างกลุ่มที่ไม่มีการสร้างดัชนีใดๆ กลุ่มที่มีการสร้างดัชนีบนทุกแอททริบิวต์ที่ปรากฏในชุดคำสั่งสอบถาม และกลุ่มที่สร้างดัชนีที่ได้จากระบบการคัดเลือกดัชนี Bitmap Join Index

สำหรับชุดการสอบถามที่ 1 และ 2 สามารถแสดงการเปรียบเทียบผลการทดลองได้ดังตาราง 4-4 และตาราง 4-5 ตามลำดับ

ตาราง 4-4 เปรียบเทียบค่าใช้จ่ายที่ใช้ในการสอบถามข้อมูลระหว่างกลุ่มที่ไม่มีการสร้างดัชนีใดๆ กลุ่มที่มีการสร้างดัชนีบนทุกแอททริบิวต์ที่ปรากฏในชุดคำสั่งสอบถาม และกลุ่มที่สร้างดัชนีจากระบบคัดเลือกดัชนี Bitmap Join Index บนชุดการสอบถามที่ 1

คำสั่งสอบถาม	จำนวนบล็อคข้อมูลที่ถูกอ่าน (Block)		
	ไม่สร้างดัชนีใดๆ (No Index)	สร้างดัชนีบนทุกแอททริบิวต์ (Full Index)	สร้างดัชนีที่เลือกโดยระบบคัดเลือกดัชนี (BJIS)
Q1	17,149	697	697
Q2	15,535	195	15,535
Q3	11,148	9,326	9,326
Q4	34,011	6,890	6,890
Q5	24,243	1,019	1,019

ตาราง 4-5 เปรียบเทียบค่าใช้จ่ายที่ใช้ในการสอบถามข้อมูลระหว่างกลุ่มที่ไม่มีการสร้างดัชนีใดๆ กลุ่มที่มีการสร้างดัชนีบนทุกแอททริบิวต์ที่ปรากฏในชุดคำสั่งสอบถาม และกลุ่มที่สร้างดัชนีจากระบบคัดเลือกดัชนี Bitmap Join Index บนชุดการสอบถามที่ 2

คำสั่งสอบถาม	จำนวนบล็อคข้อมูลที่ถูกอ่าน (Block)		
	ไม่สร้างดัชนีใดๆ (No Index)	สร้างดัชนีบนทุกแอททริบิวต์ (Full Index)	สร้างดัชนีที่เลือกโดยระบบคัดเลือกดัชนี (BJIS)
Q1	17,277	805	17,277
Q2	28,880	5,796	28,880
Q3	17,149	697	697
Q4	15,535	195	15,535
Q5	34,011	6,890	34,011

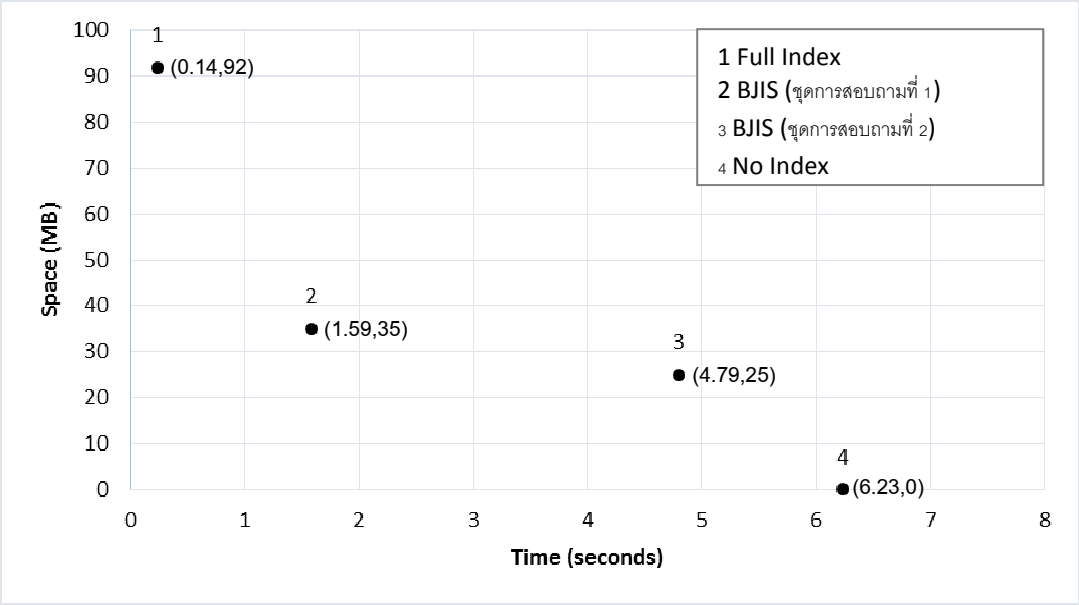
สำหรับผลการทดลองด้านการลดค่าใช้จ่ายในการสอบถามข้อมูล สำหรับชุดการสอบถามที่ 1 สามารถสรุปได้ดังนี้ สำหรับคำสั่งสอบถาม Q1 Q3 Q4 และ Q5 ในกลุ่มที่มีการสร้างดัชนีจากระบบคัดเลือกดัชนี สามารถลดค่าใช้จ่ายในการสอบถามข้อมูลเมื่อเทียบกับกลุ่มที่ไม่สร้างดัชนีใดๆ ได้โดยคิดเป็นเปอร์เซ็นต์ได้เท่ากับ 95.93%, 16.34%, 79.74% และ 95.80% ตามลำดับ ส่วนคำสั่งสอบถาม Q2 ไม่สามารถลดค่าใช้จ่ายในการสอบถามได้เลย

สำหรับผลการทดลองด้านการลดค่าใช้จ่ายในการสอบถามข้อมูล สำหรับชุดการสอบถามที่ 2 สามารถสรุปได้ดังนี้ สำหรับคำสั่งสอบถาม Q3 ในกลุ่มที่มีการสร้างดัชนีจากระบบคัดเลือกดัชนี สามารถลดค่าใช้จ่ายในการสอบถามข้อมูลเมื่อเทียบกับกลุ่มที่ไม่สร้างดัชนีใดๆ ได้โดยคิดเป็นเปอร์เซ็นต์ได้เท่ากับ 95.93% ส่วนคำสั่งสอบถาม Q1 Q2 Q4 และ Q5 ไม่สามารถลดค่าใช้จ่ายในการสอบถามได้เลย จึงทำให้ใช้เวลาในการสอบถามมาก

4. การวัดประสิทธิภาพในแง่ Space-Time Trade-off ของระบบการคัดเลือกดัชนีนั้นพบว่า หากต้องการให้ประสิทธิภาพด้านเวลาในการสอบถามเพิ่มมากขึ้น ก็จะต้องเสียพื้นที่ที่ใช้ในการจัดเก็บดัชนีเพิ่มขึ้นด้วย

สำหรับกลุ่มที่ไม่มีการสร้างดัชนีใดๆ จะใช้เวลาในการสอบถามข้อมูลมาก แต่ไม่ต้องเสียพื้นที่ที่ใช้ในการจัดเก็บดัชนี กลุ่มที่สร้างดัชนีสำหรับทุกแอททริบิวต์ จะใช้เวลาในการสอบถามข้อมูลน้อย แต่ต้องเสียพื้นที่ในการจัดเก็บดัชนีมาก และกลุ่มที่การเลือกสร้างดัชนีโดยใช้ขั้นตอนวิธี BJIS จะใช้เวลาในการสอบถามมากหรือน้อยขึ้นอยู่กับดัชนีที่ถูกเลือกว่าตรงกับคำสั่งสอบถามหรือไม่ และใช้พื้นที่ในการจัดเก็บดัชนีเท่าที่ถูกกำหนดโดยผู้ดูแลระบบ

จากการทดลองของชุดการสอบถามทั้ง 2 ชุด จะเห็นว่าดัชนีที่ถูกเลือกจากชุดการสอบถามที่ 1 ใช้เวลาในการสอบถามได้มีประสิทธิภาพกว่าดัชนีที่ถูกเลือกจากชุดการสอบถามที่ 2 ซึ่งพื้นที่ที่ใช้ในการจัดเก็บดัชนีนั้นแปรผกผันกับเวลาที่ใช้ในการสอบถาม ตามลักษณะของ Space-Time Trade-off ดังนั้นสามารถเขียนกราฟแสดงความสัมพันธ์ระหว่างพื้นที่ในการจัดเก็บดัชนีและเวลาที่ใช้ในการสอบถามของกลุ่มที่ไม่มีการสร้างดัชนีใดๆ กลุ่มที่มีการสร้างดัชนีบนทุกแอททริบิวต์ที่ปรากฏในชุดคำสั่งสอบถาม และกลุ่มที่สร้างดัชนีจากระบบคัดเลือกดัชนี Bitmap Join Index จากชุดการสอบถามที่ 1 และชุดการสอบถามที่ 2 ได้ดังภาพประกอบ 4-5



ภาพประกอบ 4-5 กราฟแสดงความสัมพันธ์ระหว่างพื้นที่ที่ใช้ในการจัดเก็บดัชนีและเวลาที่ใช้ในการสอบถาม ของกลุ่มที่ไม่มีการสร้างดัชนีใดๆ กลุ่มที่มีการสร้างดัชนีบนทุกแอททริ-บิวต์ที่ปรากฏในชุดคำสั่งสอบถาม และกลุ่มที่สร้างดัชนีจากระบบคัดเลือกดัชนี Bitmap Join Index จากชุดการสอบถามที่ 1 และชุดการสอบถามที่ 2

จากภาพประกอบ 4-5 จะเห็นได้ว่ากลุ่มสร้างดัชนีบนทุกแอททริบิวต์ที่ปรากฏในชุดคำสั่งสอบถามมีประสิทธิภาพด้านเวลาในการสอบถามมากที่สุด แต่มีประสิทธิภาพด้านพื้นที่ที่ใช้ในการจัดเก็บดัชนีน้อยที่สุด ในกลุ่มที่ไม่มีการสร้างดัชนีใดๆ มีประสิทธิภาพด้านเวลาในการสอบถามน้อยที่สุด แต่มีประสิทธิภาพด้านพื้นที่ที่ใช้ในการจัดเก็บดัชนีมากที่สุด และกลุ่มที่สร้างดัชนีจากระบบคัดเลือกดัชนี Bitmap Join Index จากชุดการสอบถามที่ 1 มีประสิทธิภาพด้านเวลาในการสอบถามมากกว่ากลุ่มที่สร้างดัชนีจากระบบคัดเลือกดัชนี Bitmap Join Index จากชุดการสอบถามที่ 2 แต่มีประสิทธิภาพด้านพื้นที่ที่ใช้ในการจัดเก็บดัชนีน้อยกว่า ดังนั้นการเลือกดัชนีที่มีประสิทธิภาพ ควรจะพิจารณาเลือกดัชนีที่ช่วยลดเวลาในการสอบถามได้เป็นอย่างดี ในขณะที่ใช้พื้นที่ในการจัดเก็บดัชนีน้อยที่สุด

จากผลการทดลองทั้งหมด สามารถสรุปได้ว่าการเลือกชุดคำสั่งสอบถามในอดีตที่จะนำมาใช้ในการคัดเลือกดัชนีนั้นมีความสำคัญเป็นอย่างมาก เนื่องจากชุดคำสั่งสอบถามในอดีตนั้นมีผลต่อชุดดัชนีที่จะถูกเลือก เช่น หากจำนวนแอททริบิวต์ที่ถูกสอบถามซ้ำในชุดคำสั่งสอบถามในอดีตมีจำนวนน้อยเกินไป ก็จะทำให้ไม่สามารถเลือกชุดดัชนีที่เหมาะสมกับพื้นที่ที่กำหนดไว้ได้

บทที่ 5

บทสรุป และข้อเสนอแนะ

งานวิทยานิพนธ์นี้ ได้นำเสนอระบบแนะนำการคัดเลือกดัชนี Bitmap Join Index สำหรับโครงสร้างแบบ Star Schema ซึ่งใช้ในการแก้ปัญหาการสอบถามในคลังข้อมูล โดยมีแนวคิดในการลดจำนวน Candidate Attribute ที่จะนำมาสร้างดัชนีในขั้นตอนแรกๆ เพื่อลดเวลาในการประมวลผลในขั้นตอนถัดไป จากนั้นมีการประเมินทรัพยากรที่ใช้ในการสร้างดัชนีของแต่ละแอททริบิวต์ และใช้เทคนิคการแก้ปัญหาแบบพลวัตในการเลือกดัชนีที่มีความเหมาะสมภายใต้ทรัพยากรที่จำกัด โดยขั้นตอนการทำงานของระบบแบ่งเป็น 4 ขั้นตอนหลัก คือ ขั้นตอนที่ 1 การสกัดแอททริบิวต์บนตาราง Dimension ที่จะนำมาสร้างดัชนี (Candidate Extraction) ขั้นตอนที่ 2 การลดจำนวน Candidate Attribute (Candidate Attribute Reduction) ขั้นตอนที่ 3 การประเมินทรัพยากรที่ใช้ในการสร้างดัชนีสำหรับแอททริบิวต์แต่ละตัว(Resource Evaluation) และขั้นตอนที่ 4 การเลือกดัชนีที่เหมาะสมที่สุดสำหรับทรัพยากรที่จำกัด (Index Selection) ซึ่งผลลัพธ์ที่ได้ คือระบบแนะนำการคัดเลือกดัชนี Bitmap Join Index ที่มีประสิทธิภาพ สามารถช่วยให้ผู้ใช้สามารถเลือกดัชนีที่ถูกต้องสำหรับเพิ่มประสิทธิภาพด้านเวลาที่ใช้ในการสอบถามในคลังข้อมูลได้เป็นอย่างดี ในขณะที่มีเงื่อนไขการใช้ทรัพยากรที่จำกัด

5.1 สรุปผลการวิจัย

การสอบถามข้อมูลในคลังข้อมูลที่มีขนาดใหญ่ ที่มีลักษณะการสอบถามแบบทันทีทันใด และมีเงื่อนไขการสอบถามที่ซับซ้อน จะทำให้ใช้เวลาในการประมวลผลมาก ดังนั้นการเพิ่มประสิทธิภาพในการสอบถามจึงมีความจำเป็นมากโดยวิธีการที่มีความนิยมใช้ ได้แก่ การสร้างดัชนี Bitmap Join Index เนื่องจากมีความเหมาะสมกับคลังข้อมูลโดยสามารถช่วยลดเวลาที่ใช้ในการสอบถามที่มีความซับซ้อนได้เป็นอย่างดี

ในการสร้างดัชนีนั้นมีปัจจัยที่ต้องคำนึงถึง นั่นคือ พื้นที่ที่ใช้ในการจัดเก็บดัชนี ซึ่งหากตารางข้อมูลมีขนาดใหญ่มาก ก็จะใช้พื้นที่ในการจัดเก็บดัชนีมากเช่นกัน ด้วยเหตุนี้จึงไม่สามารถสร้างดัชนีสำหรับทุกแอททริบิวต์ได้ภายใต้พื้นที่จัดเก็บดัชนีที่มีอยู่อย่างจำกัด ดังนั้นงานวิทยานิพนธ์นี้จึงได้นำเสนอระบบแนะนำการคัดเลือกดัชนี Bitmap Join Index สำหรับคลังข้อมูลที่มีโครงสร้างแบบ Star Schema โดยมีการพิจารณาปัจจัยที่เกี่ยวข้องเพื่อใช้ในการคัดเลือกดัชนีที่เหมาะสมที่สุดที่จะนำมาสร้างดัชนี ได้แก่ การเลือกแอททริบิวต์ที่มีโอกาสถูกสอบถามบ่อย และมีค่าใช้จ่ายในการจับคู่ข้อมูลสูง โดยมีแนวคิดที่ว่าหากสร้างดัชนีบนแอททริ-

บิวต์ที่มีคุณสมบัติดังกล่าวนั้น จะช่วยลดเวลาในการประมวลผลระหว่างการสอบถามได้มากที่สุด

ในการสรุปผลการวิจัย จะแบ่งสรุปเป็น 4 ประเด็นคือ 1) ประสิทธิภาพด้านเวลาในการสอบถามข้อมูล 2) ประสิทธิภาพด้านการใช้พื้นที่ในการจัดเก็บดัชนี 3) ประสิทธิภาพด้านการลดค่าใช้จ่ายในการสอบถามข้อมูล และ 4) ประสิทธิภาพด้านการประมวลผลของระบบ ซึ่งมีรายละเอียดดังนี้

5.1.1 ประสิทธิภาพด้านเวลาในการสอบถามข้อมูล

ระบบแนะนำการคัดเลือกดัชนี Bitmap Join Index สามารถเลือกสร้างดัชนีได้ตรงตามความต้องการของผู้ใช้ภายใต้พื้นที่จัดเก็บดัชนีที่จำกัด โดยดัชนีที่เลือกโดยระบบช่วยลดเวลาในการสอบถามได้เป็นอย่างดี เนื่องจากในการสอบถามข้อมูลสามารถค้นหาข้อมูลจากตารางดัชนีที่สร้างไว้ก่อนที่จะเข้าถึงข้อมูลในตารางจริง และสามารถดำเนินการทางตรรกะสำหรับการสอบถามที่มีหลายเงื่อนไขได้ ซึ่งในบางคำสั่งสอบถามสามารถให้ประสิทธิภาพได้ใกล้เคียงกับการสร้างดัชนีบนทุกแอททริบิวต์

5.1.2 ประสิทธิภาพด้านการใช้พื้นที่ในการจัดเก็บดัชนี

ระบบแนะนำการคัดเลือกดัชนี Bitmap Join Index สามารถเลือกสร้างดัชนีได้อย่างเหมาะสมภายใต้พื้นที่จัดเก็บดัชนีที่จำกัด โดยนำเทคนิคการแก้ปัญหาแบบพลวัตมาประยุกต์ใช้ ซึ่งมีความยืดหยุ่นเมื่อต้องการปรับลดขนาดของพื้นที่ที่ใช้ในการจัดเก็บดัชนี เนื่องจากมีการเก็บผลลัพธ์ของปัญหาย่อยเอาไว้เสมอ

5.1.3 ประสิทธิภาพด้านการลดค่าใช้จ่ายในการสอบถามข้อมูล

จากการวิเคราะห์ค่าใช้จ่ายในการสอบถามข้อมูล ซึ่งคำนวณจากจำนวนบล็อคข้อมูลที่ถูกอ่านระหว่างการประมวลผลการสอบถามนั้นพบว่า การสร้างดัชนีบนทุกแอททริบิวต์เพื่อเพิ่มประสิทธิภาพในการสอบถามข้อมูล สามารถลดค่าใช้จ่ายในการอ่านข้อมูลได้มากที่สุด ซึ่งในบางคำสั่งสอบถามระบบแนะนำการคัดเลือกดัชนีสามารถเลือกดัชนีได้ตรงกับความต้องการ โดยให้ผลลัพธ์ที่ดีเท่ากับการสร้างดัชนีบนทุกแอททริบิวต์ แต่ในบางคำสั่งสอบถามแอททริบิวต์ที่สอบถามไม่ตรงกับแอททริบิวต์ที่ถูกเลือก จึงช่วยลดค่าใช้จ่ายในการอ่านข้อมูลได้เพียงเล็กน้อย

5.1.4 ประสิทธิภาพด้านการประมวลผลของระบบ

ระบบแนะนำการคัดเลือกดัชนี Bitmap Join Index แบ่งการทำงานเป็น 4 ขั้นตอน ได้แก่ ขั้นตอนในการสกัดแอททริบิวต์จากตาราง Dimension ที่จะนำมาสร้างดัชนี ขั้นตอนการลดจำนวน Candidate Attribute ขั้นตอนการประเมินทรัพยากรที่ใช้ในการสร้างดัชนี และขั้นตอนการเลือกดัชนีสำหรับพื้นที่ที่จำกัด ซึ่งสามารถอธิบายได้ดังนี้

1) ในขั้นตอนการสกัดแอททริบิวต์จากตาราง Dimension ที่จะนำมาสร้างดัชนี มีการตัดตารางที่มีขนาดเล็กกว่าค่าที่กำหนดออกก่อน ซึ่งแอททริบิวต์ที่ปรากฏอยู่ในตาราง Dimension นั้นจะไม่ถูกนำมาพิจารณาในขั้นตอนถัดไปอีก ดังนั้นก็จะช่วยลดการประมวลผลในขั้นตอนอื่นๆ ได้

2) ในขั้นตอนการลดจำนวน Candidate Attribute นั้นใช้ข้อมูล BJI Knowledge Base เข้ามาช่วยลดจำนวน Candidate Attribute ดังนั้นจึงไม่จำเป็นต้องเสียเวลาในการคำนวณค่าความเหมาะสมของแอททริบิวต์ที่มักถูกเลือกมาสร้างดัชนีเสมอ และขั้นตอนการสกัดแอททริบิวต์ที่ถูกสอบถามบ่อยจะใช้เพียงการหาค่าสนับสนุนของกลุ่มข้อมูลที่มีขนาดเท่ากับ 1 เท่านั้น จึงใช้เวลาในการคำนวณน้อยกว่าการใช้ Frequent Itemset

ในการสร้างดัชนี Bitmap Join Index บนแอททริบิวต์เดียวนอกจากจะช่วยลดเวลาในการประมวลผลของระบบแนะนำการคัดเลือกดัชนีแล้ว ยังช่วยให้การสอบถามข้อมูลมีความยืดหยุ่นมากขึ้น เนื่องจากโอกาสที่จะสร้างดัชนีได้ตรงกับคำสั่งสอบถามมีมากกว่า

3) ในขั้นตอนการคำนวณทรัพยากรในการสร้างดัชนีนั้น การคำนวณพื้นที่ที่ใช้ในการจัดเก็บดัชนีสามารถปรับเปลี่ยนค่าของจำนวนบิตแมปเวกเตอร์ที่ใช้ได้ นั่นคือ หากผู้ใช้ต้องการลดขนาดของพื้นที่ที่ใช้ในการจัดเก็บดัชนี ก็สามารถเปลี่ยนรูปแบบในการลงรหัสจากดัชนีบิตแมปแบบพื้นฐาน เป็นดัชนีบิตแมปชนิดอื่น ๆ ได้

4) ในขั้นตอนการเลือกดัชนีสำหรับพื้นที่ที่จำกัด ใช้วิธีแก้ปัญหาแบบพลวัต ซึ่งสามารถคำนวณค่าใช้จ่ายของการหามูลค่าความเหมาะสมของชุดดัชนีที่ถูกเลือก รวมกับค่าใช้จ่ายในการหาเซตคำตอบของดัชนีที่ถูกเลือก จะได้ค่าใช้จ่ายในขั้นตอนการเลือกดัชนีทั้งหมด เท่ากับ $O(nS) + O(n+S)$ เมื่อ n คือจำนวนดัชนีทั้งหมดต้องการคัดเลือก และ S คือขนาดของพื้นที่ทั้งหมดที่ใช้ในการจัดเก็บดัชนี

5.1.5 สรุปลักษณะข้อมูลที่เหมาะสมกับระบบแนะนำการคัดเลือกดัชนี Bitmap Join Index สำหรับโครงสร้างแบบ Star Schema

จากผลการทดลองทั้งหมด สามารถสรุปได้ว่าประสิทธิภาพของระบบแนะนำการคัดเลือกดัชนีนั้นจะให้ประสิทธิภาพที่ดีเมื่อสามารถกำหนดขีดแบ่งค่าสนับสนุนที่เหมาะสมกับชุดคำสั่งสอบถามในอดีต และขนาดพื้นที่ที่กำหนดไว้เพื่อจัดเก็บดัชนี เนื่องจากหากกำหนด

ขีดแบ่งค่าสนับสนุนสูงเกินไป และมีขนาดของพื้นที่ที่กำหนดไว้เพื่อจัดเก็บดัชนีมาก แต่มีจำนวนแอททริบิวต์ในชุดคำสั่งสอบถามในอดีตนั้นถูกสอบถามซ้ำน้อย แอททริบิวต์ที่มีค่าสนับสนุนในการสอบถามน้อยกว่าขีดแบ่งค่าสนับสนุนที่กำหนดจะไม่มีโอกาสถูกเลือก ก็จะทำให้ใช้พื้นที่ได้ไม่เต็มประสิทธิภาพ แต่หากกำหนดขีดแบ่งค่าสนับสนุนต่ำเกินไปและมีขนาดของพื้นที่ที่กำหนดไว้เพื่อจัดเก็บดัชนีน้อย จำนวนแอททริบิวต์ที่มีค่าสนับสนุนสูงกว่าขีดแบ่งค่าสนับสนุนก็จะมีโอกาสถูกเลือกเข้ามามาก ก็ทำให้เสียเวลาในการประมวลผลในขั้นตอนการคัดเลือกดัชนี

ดังนั้นขีดแบ่งค่าสนับสนุนนั้นจะส่งผลต่อขนาดของพื้นที่ที่ใช้ในการจัดเก็บข้อมูล หากผู้ดูแลระบบกำหนดขีดแบ่งค่าสนับสนุน ไม่เหมาะสมกับพื้นที่ที่ใช้ในการจัดเก็บดัชนี ก็จะทำให้เพิ่มประสิทธิภาพในการสอบถามได้น้อย และใช้พื้นที่ได้ไม่เต็มประสิทธิภาพ

5.2 ข้อเสนอแนะและงานในอนาคต

1) ในการคัดเลือกกลุ่มข้อมูลที่ปรากฏบ่อย ควรจะมีการพิจารณาความแตกต่างระหว่างการสร้างดัชนี Bitmap Join Index บนแอททริบิวต์เดียว กับการสร้างดัชนีบนแอททริบิวต์ร่วม ว่ามีประสิทธิภาพในการประมวลผลแตกต่างกันอย่างไร

2) การประเมินค่าใช้จ่ายในการจับคู่ข้อมูล ควรมีปรับให้ระบบสามารถเลือกใช้เทคนิคสำหรับการจับคู่ข้อมูลในแบบอื่น ๆ ได้เนื่องจากในแต่ละระบบจัดการฐานข้อมูลอาจใช้เทคนิคในการจับคู่ข้อมูลที่แตกต่างกัน

3) ในการอัพเดต BJI Knowledge Base อาจเลือกเก็บแอททริบิวต์ที่เป็นฐานนิยมของกลุ่มของชุดดัชนีที่นำมาพิจารณา และอาจมีการพิจารณาว่าชุดดัชนีจำนวนเท่าไรที่เหมาะสมที่จะนำมาใช้สำหรับการคัดเลือกว่าแอททริบิวต์ใดควรจะถูกจัดเก็บไว้ใน BJI Knowledge Base เป็นต้น

4) มีการพิจารณาปัจจัยที่มีผลต่อค่าใช้จ่ายการสร้างดัชนีเพิ่มเติมเช่น หากพบว่ามีดัชนี 2 ตัวใช้พื้นที่ในการเก็บดัชนีเท่ากัน ควรจะเลือกดัชนีที่อยู่ในตารางเดียวกันกับดัชนีที่ถูกเลือกตัวอื่นๆ เพื่อลดเวลาในการอ่านข้อมูลของขั้นตอนการสร้างดัชนี

5) ควรมีการเลือกขีดแบ่งค่าสนับสนุนได้โดยอัตโนมัติตามขนาดของพื้นที่ที่ใช้สำหรับจัดเก็บดัชนี เพื่อให้ระบบสามารถเลือกดัชนีที่ช่วยเพิ่มประสิทธิภาพด้านเวลาในการสอบถาม และใช้พื้นที่ได้อย่างเต็มประสิทธิภาพ

บรรณานุกรม

An, H. G. and Koh, J. J. 2012. A Study on the Selection of Bitmap Join Index using Data Mining Techniques. Strategic Technology (IFOST), 7th International Forum on. IEEE, pp. 1-5.

Aouiche, K., Darmont, J., Boussaïd, O. and Bentayeb, F. 2005. Automatic Selection of Bitmap Join Indexes in Data Warehouses. Data Warehousing and Knowledge Discovery, pp. 64-73.

Bellatrecheet, L., Missaoui, R., Necir, H. and Drias, H. 2007. A Data Mining Approach for Selecting Bitmap Join Indices. JCSE 1.2, pp. 177-194.

Bellatrecheet, L., Missaoui, R., Necir, H. and Drias, H. 2007. Selection and Pruning Algorithms for Bitmap Index Selection Problem using Data Mining. Data Warehousing and Knowledge Discovery, pp. 221-230.

Bizarro, P. and Madeira, H. 2001. The Dimension-Join: A New Index for Data Warehouses. SBBD, pp. 259-273.

Chaudhuri, S. and Dayal, U. 1997. An Overview of Data Warehousing and OLAP Technology. ACM Sigmod record 26.1, pp. 65-74.

Gupta, H., Harinarayan, V., Rajaraman A. and Ullman, J. D. 1997. Index Selection for OLAP. Proceedings. 13th International Conference on. IEEE, pp. 208-219.

Inmon, W. H. 2005. Building the data warehouse., Wiley Computer Publishing.

Kimball, R. and Margy, R. 2004. The Data Warehouse Toolkit-The Complete Guide to Dimensional Modeling., John Wiley and Sons (Asia) Pte. Limited.

Levitin, A. 2007. Introduction to the design & analysis of algorithms., Pearson Addison-Wesley.

O'Neil, P. and Quass, D. 1997. Improved Query Performance with Variant Indexes. ACM Sigmod Record., Vol. 26, No. 2, pp.38-49.

O'Neil, P., O'Neil, E. and Chen, X. 2009. The Star Schema Benchmark. Revision 3. http://www.cs.umb.edu/?poneil/StarSchemaB.PDF (accessed 10/09/13).

Ooi, B. C. and Tan, K. L. 2002. B-trees: Bearing Fruits of All Kinds. Australian Computer Science Communications, Vol. 24, No. 2. Australian Computer Society, Inc.

Valduriez, P. 1987. Join indices. ACM Transactions on Database Systems (TODS), 12.2, pp. 218-246.

Vanichayobon, S. and Gruenwald, Le. 1999. Indexing Techniques for Data Warehouses' Queries. The University of Oklahoma, School of Computer Science, Technical Report (1999).

Wu, M. C. and Buchmann, A. P. 1997. Research Issues in Data Warehousing. Datenbanksysteme in Büro, Technik und Wissenschaft, pp. 61-82.

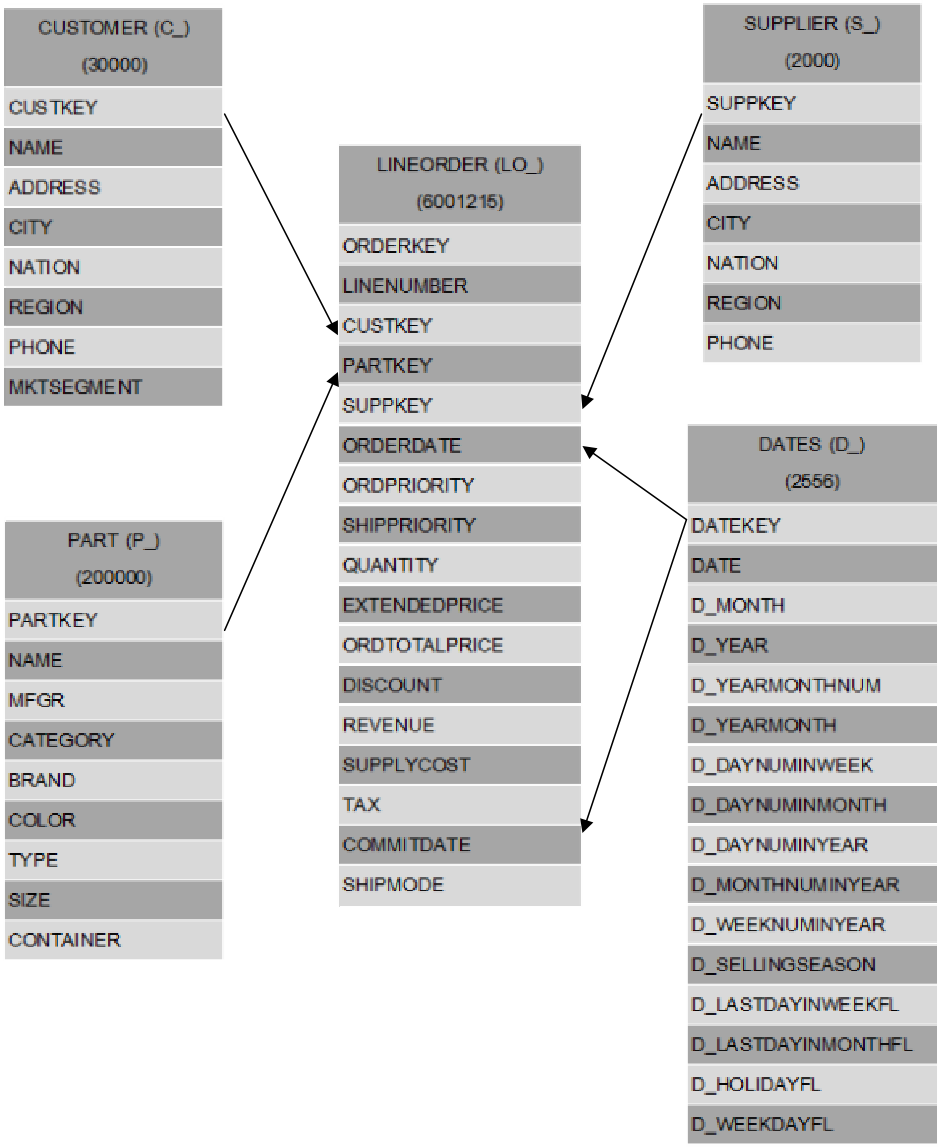
Ziani, B. and Ouinten, Y. 2011. Improving Star Join Queries Performance: A Maximal Frequent Pattern Based Approach for Automatic Selection of Indexes in Relational Data Warehouses. Internet Computing & Information Services (ICICIS), 2011 International Conference on. IEEE.

ภาคผนวก

ภาคผนวก ก.

ก.1 การเตรียมข้อมูลเพื่อใช้ในการทดลอง

ข้อมูลที่นำมาใช้ในการวัดประสิทธิภาพของระบบ เป็นข้อมูลมาตรฐานจาก SSB Benchmark ซึ่งลักษณะของข้อมูลเป็นโครงสร้างแบบ Star Schema สามารถนำมาใช้ในการวัดประสิทธิภาพการสอบถามข้อมูลได้ โดยมีโครงสร้างดังนี้



ภาพประกอบ ก-1 โครงสร้างของ SSB

ข้อมูลทั้งหมดที่นำมาใช้ในการทดสอบ ได้มีการนำเข้าระบบจัดการฐานข้อมูล Oracle Database 12g สามารถแสดงรายละเอียดของแต่ละตาราง ดังต่อไปนี้

1. ข้อมูลในตาราง LINEORDER ซึ่งเป็น Fact Table ประกอบไปด้วยข้อมูลจำนวน 6,001,215 เรคอร์ด แต่ละเรคอร์ดประกอบด้วยแอททริบิวต์ต่างๆ ดังตาราง ก-1

ตาราง ก-1 รายละเอียดแอททริบิวต์ของตาราง LINEORDER

ชื่อแอททริบิวต์	ประเภทข้อมูล	ประเภทคีย์
LO_ORDERKEY	numeric	PRIMARY KEY
LO_LINENUMBER	numeric 1-7	PRIMARY KEY
LO_CUSTKEY	numeric identifier	FK to C_CUSTKEY
LO_PARTKEY	numeric identifier	FK to P_PARTKEY
LO_SUPPKEY	numeric identifier	FK to S_SUPPKEY
LO_ORDERDATE	numeric identifier	FK to D_DATEKEY
LO_ORDERPRIORITY	fixed text (15)	-
LO_SHIPPRIORITY	fixed text (1)	-
LO_QUANTITY	numeric 1-50	-
LO_EXTENDEDPRICE	numeric	-
LO_ORDTOTALPRICE	numeric	-
LO_DISCOUNT	numeric 0-10	-
LO_REVENUE	numeric	-
LO_SUPPLYCOST	numeric	-
LO_TAX	numeric 0-8	-
LO_COMMITDATE	numeric identifier	FK to D_DATEKEY
LO_SHIPMODE	fixed text (10)	-

ตัวอย่างข้อมูลที่จัดเก็บในระบบจัดการฐานข้อมูลของตาราง LINEORDER แสดงดังภาพประกอบ ก-2

LO_ORDERKEY	LO_LINENUMBER	LO_CUSTKEY	LO_PARTKEY	LO_SUPPKEY	LO_ORDERDATE	LO_ORDERPRIORITY	LO_SHIPPRIORITY	LO_QUANTITY	LO_EXTENDEDPRICE	LO_ORDTOTALPRICE
69	1	16898	115209	619	19940604	4-NOT SPECI	0	48	5876160	19768949
69	2	16898	104180	1160	19940604	4-NOT SPECI	0	32	3789376	19768949
69	3	16898	137267	1537	19940604	4-NOT SPECI	0	17	2217242	19768949
69	4	16898	37502	1197	19940604	4-NOT SPECI	0	3	431850	19768949
69	5	16898	92070	1323	19940604	4-NOT SPECI	0	42	4460694	19768949
69	6	16898	18504	841	19940604	4-NOT SPECI	0	23	3271750	19768949
70	1	12868	64128	1211	19931218	5-LOW	0	8	873696	11353442
70	2	12868	196156	1262	19931218	5-LOW	0	13	1627795	11353442
70	3	12868	179809	1589	19931218	5-LOW	0	1	188880	11353442
70	4	12868	45734	1057	19931218	5-LOW	0	11	1847703	11353442
70	5	12868	37131	1226	19931218	5-LOW	0	37	3952081	11353442
70	6	12868	55655	1872	19931218	5-LOW	0	19	3060235	11353442
71	1	676	61931	297	19980124	4-NOT SPECI	0	25	4732325	27699274
71	2	676	65916	1714	19980124	4-NOT SPECI	0	3	564573	27699274
71	3	676	34432	1720	19980124	4-NOT SPECI	0	45	6148935	27699274
71	4	676	96645	624	19980124	4-NOT SPECI	0	33	5417412	27699274
71	5	676	103255	930	19980124	4-NOT SPECI	0	39	4907175	27699274
71	6	676	195635	1472	19980124	4-NOT SPECI	0	34	5884142	27699274
96	1	21556	123076	1724	19940417	2-HIGH	0	23	2527861	6898990
96	2	21556	135390	348	19940417	2-HIGH	0	30	4276170	6898990

ภาพประกอบ ก-2 ตัวอย่างข้อมูลที่จัดเก็บในระบบฐานข้อมูลของตาราง LINEORDER

2. ข้อมูลในตาราง PART ซึ่งเป็น Dimension Table ประกอบไปด้วยข้อมูลจำนวน 200,000 เรคอร์ด แต่ละเรคอร์ดประกอบด้วยแอททริบิวต์ต่างๆ ดังตาราง ก-2

ตาราง ก-2 รายละเอียดแอททริบิวต์ของตาราง PART

ชื่อแอททริบิวต์	ประเภทข้อมูล	ประเภทคีย์	คาร์ดินอลิตี้
P_PARTKEY	numeric identifier	PRIMARY KEY	200,000
P_NAME	variable text (22)	-	8,372
P_MFGR	fixed text (6)	-	5
P_CATEGORY	fixed text (7)	-	25
P_BRAND	fixed text (9)	-	1,000
P_COLOR	variable text (11)	-	92
P_TYPE	variable text (25)	-	150
P_SIZE	numeric (38,0)	-	50
P_CONTAINER	fixed text (10)	-	40

ตัวอย่างข้อมูลที่จัดเก็บในระบบจัดการฐานข้อมูลของตาราง PART แสดงดังภาพประกอบ ก-3

P_PARTKEY	P_NAME	P_MFGR	P_CATEGORY	P_BRAND	P_COLOR	P_TYPE	P_SIZE	P_CONTAINER
84	peru frosted	MFGR#3	MFGR#31	MFGR#3140	salmon	SMALL ANODIZED NICKEL	26	JUMBO PACK
85	cornsilk floral	MFGR#5	MFGR#53	MFGR#5340	aquamarine	PROMO ANODIZED NICKEL	16	LG BAG
86	blanched khaki	MFGR#1	MFGR#15	MFGR#1529	green	STANDARD PLATED TIN	37	LG CASE
87	khaki sandy	MFGR#1	MFGR#11	MFGR#117	pale	LARGE PLATED STEEL	41	WRAP PACK
88	olive azure	MFGR#1	MFGR#12	MFGR#1230	blue	PROMO PLATED COPPER	16	SM CASE
89	khaki lawn	MFGR#2	MFGR#21	MFGR#2123	ghost	STANDARD BURNISHED STEEL	7	MED JAR
90	goldenrod violet	MFGR#4	MFGR#43	MFGR#431	ghost	ECONOMY POLISHED STEEL	49	JUMBO CAN
91	beige metallic	MFGR#2	MFGR#23	MFGR#238	dark	STANDARD BRUSHED TIN	32	JUMBO PKG
92	dodger turquoise	MFGR#5	MFGR#55	MFGR#5511	chiffon	STANDARD ANODIZED TIN	35	JUMBO PKG
93	yellow cornflower	MFGR#4	MFGR#42	MFGR#4228	blanched	LARGE ANODIZED TIN	2	WRAP DRUM
94	pink orange	MFGR#4	MFGR#45	MFGR#4539	azure	STANDARD POLISHED BRASS	32	SM BOX
95	chocolate wheat	MFGR#2	MFGR#25	MFGR#2517	coral	LARGE BRUSHED TIN	36	WRAP DRUM
96	light drab	MFGR#3	MFGR#33	MFGR#3323	steel	STANDARD BRUSHED STEEL	32	SM CASE
97	dodger aquamarine	MFGR#4	MFGR#45	MFGR#4522	metallic	MEDIUM POLISHED BRASS	49	WRAP CAN
98	goldenrod chartreuse	MFGR#4	MFGR#43	MFGR#4331	frosted	STANDARD ANODIZED BRASS	22	MED JAR
99	peru chiffon	MFGR#5	MFGR#54	MFGR#544	lemon	SMALL BURNISHED STEEL	11	JUMBO PKG
100	orange khaki	MFGR#3	MFGR#35	MFGR#3522	light	ECONOMY ANODIZED TIN	4	LG BAG
101	yellow turquoise	MFGR#4	MFGR#44	MFGR#4415	hot	LARGE ANODIZED STEEL	26	JUMBO JAR
102	orchid blanched	MFGR#5	MFGR#55	MFGR#551	firebrick	MEDIUM BURNISHED BRASS	17	SM DRUM
103	cream lime	MFGR#5	MFGR#52	MFGR#5236	navy	MEDIUM PLATED BRASS	45	WRAP DRUM

ภาพประกอบ ก-3 ตัวอย่างข้อมูลที่จัดเก็บในระบบฐานข้อมูลของตาราง PART

3. ข้อมูลในตาราง SUPPLIER ซึ่งเป็น Dimension Table ประกอบไปด้วยข้อมูลจำนวน 2,000 เรคอร์ด แต่ละเรคอร์ดประกอบด้วยแอททริบิวต์ต่างๆ ดังตาราง ก-3

ตาราง ก-3 รายละเอียดแอททริบิวต์ของตาราง SUPPLIER

ชื่อแอททริบิวต์	ประเภทข้อมูล	ประเภทคีย์	คาร์ดินอลิตี้
S_SUPPKEY	numeric identifier	PRIMARY KEY	2,000
S_NAME	fixed text (25)	-	2,000
S_ADDRESS	variable text (25)	-	2,000
S_CITY	variable text (10)	-	250
S_NATION	variable text (15)	-	25
S_REGION	variable text (12)	-	5
S_PHONE	fixed text (15)	-	2,000

ตัวอย่างข้อมูลที่จัดเก็บในระบบจัดการฐานข้อมูลของตาราง SUPPLIER แสดงดังภาพประกอบ ก-4

S_SUPPKEY	S_NAME	S_ADDRESS	S_CITY	S_NATION	S_REGION	S_PHONE
78	Supplier#000000078	StXc1Zo5rJl z avspI5Ys7	ETHIOPIA 0	ETHIOPIA	AFRICA	15-960-700-9191
79	Supplier#000000079	C,xXyty836oYhSwmLfdle	KENYA 7	KENYA	AFRICA	24-147-850-4166
80	Supplier#000000080	PhYej9mhZXAifY4IXnVIV	VIETNAM 7	VIETNAM	ASIA	31-267-172-7101
81	Supplier#000000081	9fFQOQJKBgF5Q0K	JAPAN 8	JAPAN	ASIA	22-165-277-3269
82	Supplier#000000082	5t7gqU1BlZW	CHINA 1	CHINA	ASIA	28-159-442-5305
83	Supplier#000000083	59h6GtoD,V0 M5s	KENYA 0	KENYA	AFRICA	24-817-154-4122
84	Supplier#000000084	oO2H4fI1kaBmgchJ	UNITED ST6	UNITED STATES	AMERICA	34-546-818-3802
85	Supplier#000000085	pGFzRay	GERMANY 8	GERMANY	EUROPE	17-745-585-8219
86	Supplier#000000086	iZLKKW	ROMANIA 6	ROMANIA	EUROPE	29-677-951-2353
87	Supplier#000000087	5ovT6anHSsD1T	UNITED ST7	UNITED STATES	AMERICA	34-869-884-7053
88	Supplier#000000088	uczJ2Tv	IRAQ 7	IRAQ	MIDDLE EAST	21-516-273-2566
89	Supplier#000000089	MJys1P8C	INDONESIA9	INDONESIA	ASIA	19-394-451-5404
90	Supplier#000000090	QbmU6KsZLYFXKv8le D9z0K0	FRANCE 8	FRANCE	EUROPE	16-603-491-1238
91	Supplier#000000091	35WVnU7GLNbQDcc2TARav	CANADA 2	CANADA	AMERICA	13-239-400-3677
92	Supplier#000000092	EWS4tXaiXFF	BRAZIL 3	BRAZIL	AMERICA	12-446-416-8471
93	Supplier#000000093	wNZNHIg37	MOZAMBIQU3	MOZAMBIQUE	AFRICA	26-359-388-5266
94	Supplier#000000094	Gg0BfcHqe5lhc3dT,s	EGYPT 5	EGYPT	MIDDLE EAST	14-953-499-8833
95	Supplier#000000095	CYrYJqL	ROMANIA 5	ROMANIA	EUROPE	29-923-255-2929
96	Supplier#000000096	gPuBX5iIBtNrCYv2EMZ	JAPAN 8	JAPAN	ASIA	22-422-845-1202
97	Supplier#000000097	f oaJRYkUDO	EGYPT 4	EGYPT	MIDDLE EAST	14-588-919-5638

ภาพประกอบ ก-4 ตัวอย่างข้อมูลที่จัดเก็บในระบบฐานข้อมูลของตาราง SUPPLIER

4. ข้อมูลในตาราง CUSTOMER ซึ่งเป็น Dimension Table ประกอบไปด้วยข้อมูลจำนวน 30,000 เรคอร์ด แต่ละเรคอร์ดประกอบด้วยแอททริบิวต์ต่างๆ ดังตาราง ก-4

ตาราง ก-4 รายละเอียดแอททริบิวต์ของตาราง CUSTOMER

ชื่อแอททริบิวต์	ประเภทข้อมูล	ประเภทคีย์	คาร์ดินอลิตี้
C_CUSTKEY	numeric identifier	PRIMARY KEY	30,000
C_NAME	variable text (25)	-	30,000
C_ADDRESS	variable text (25)	-	30,000
C_CITY	variable text (10)	-	250
C_NATION	variable text (15)	-	25
C_REGION	variable text (12)	-	5
C_PHONE	fixed text (15)	-	30,000
C_MKTSEGMENT	fixed text (10)	-	5

ตัวอย่างข้อมูลที่จัดเก็บในระบบจัดการฐานข้อมูลของตาราง CUSTOMER แสดงดังภาพประกอบ ก-5

C_CUSTKEY	C_NAME	C_ADDRESS	C_CITY	C_NATION	C_REGION	C_PHONE	C_MKTSEGMENT
76	Customer#000000076	FXzPoxHecjuQcCLk	ALGERIA 4	ALGERIA	AFRICA	10-349-718-3044	FURNITURE
77	Customer#000000077	XyqmVhNelkoWP	PERU 9	PERU	AMERICA	27-269-357-4674	BUILDING
78	Customer#000000078	kpcyQ11dABLX6YO9f	INDONESIA0	INDONESIA	ASIA	19-960-700-9191	FURNITURE
79	Customer#000000079	EVJjYMyG59w4S	MOROCCO 7	MOROCCO	AFRICA	25-147-850-4166	MACHINERY
80	Customer#000000080	h0wy3CS	ALGERIA 7	ALGERIA	AFRICA	10-267-172-7101	FURNITURE
81	Customer#000000081	9jUFbrThIIe	SAUDI ARA8	SAUDI ARABIA	MIDDLE EAST	30-165-277-3269	BUILDING
82	Customer#000000082	tJkXm2PQBWPRRYk	CHINA 1	CHINA	ASIA	28-159-442-5305	AUTOMOBILE
83	Customer#000000083	kDJ7dqVuBD91Y,hqjW2	RUSSIA 0	RUSSIA	EUROPE	32-817-154-4122	BUILDING
84	Customer#000000084	GB3sUmv RRXV DPzeOSbG	IRAQ 6	IRAQ	MIDDLE EAST	21-546-818-3802	FURNITURE
85	Customer#000000085	I9M GnvICGJ29rT	ETHIOPIA 8	ETHIOPIA	AFRICA	15-745-585-8219	FURNITURE
86	Customer#000000086	78Umklj3P783bfR	ALGERIA 6	ALGERIA	AFRICA	10-677-951-2353	HOUSEHOLD
87	Customer#000000087	KKkJj684aF	UNITED KI7	UNITED KINGDOM	EUROPE	33-869-884-7053	FURNITURE
88	Customer#000000088	wyGHpeRMt mxmd8fEel	MOZAMBIQU7	MOZAMBIQUE	AFRICA	26-516-273-2566	AUTOMOBILE
89	Customer#000000089	Oy90ruRha47cUlC	KENYA 9	KENYA	AFRICA	24-394-451-5404	FURNITURE
90	Customer#000000090	buosjU5uu2	MOZAMBIQU8	MOZAMBIQUE	AFRICA	26-603-491-1238	BUILDING
91	Customer#000000091	9Sce2m BjvDdjQkqMx8Unr	INDIA 2	INDIA	ASIA	18-239-400-3677	AUTOMOBILE
92	Customer#000000092	DPbrb7fGYfkrfAdm0gRJPPdA	BRAZIL 3	BRAZIL	AMERICA	12-446-416-8471	MACHINERY
93	Customer#000000093	nj3p Z	GERMANY 3	GERMANY	EUROPE	17-359-388-5266	MACHINERY
94	Customer#000000094	jL5dieRgyG9OhRN7HgXb	INDONESIA5	INDONESIA	ASIA	19-953-499-8833	HOUSEHOLD
95	Customer#000000095	n6,uwF4wcE76DVh1YsS	MOROCCO 5	MOROCCO	AFRICA	25-923-255-2929	MACHINERY

ภาพประกอบ ก-5 ตัวอย่างข้อมูลที่จัดเก็บในระบบฐานข้อมูลของตาราง CUSTOMER

5. ข้อมูลในตาราง DATES ซึ่งเป็น Dimension Table ประกอบไปด้วยข้อมูลจำนวน 30,000 เรคอร์ด แต่ละเรคอร์ดประกอบด้วยแอททริบิวต์ต่างๆ ดังตาราง ก-5

ตาราง ก-5 รายละเอียดแอททริบิวต์ของตาราง DATES

ชื่อแอททริบิวต์	ประเภทข้อมูล	ประเภทคีย์	คาร์ดินอลิตี้
D_DATEKEY	identifier	PRIMARY KEY	2,556
D_DATE	variable text (18)	-	2,556
D_DAYOFWEEK	variable text (9)	-	7
D_MONTH	variable text (9)	-	12
D_YEAR	numeric 1992-1998	-	7
D_YEARMONTHNUM	numeric (YYYYMM)	-	84
D_YEARMONTH	fixed text (7)	-	84
D_DAYNUMINWEEK	numeric 1-7	-	7
D_DAYNUMINMONTH	numeric 1-31	-	31
D_DAYNUMINYEAR	numeric 1-366	-	366
D_MONTHNUMINYEAR	numeric 1-12	-	12
D_WEEKNUMINYEAR	numeric 1-53	-	53
D_SELLINGSEASON	fixed text (12)	-	5
D_LASTDAYINWEEKFL	numeric	-	2
D_LASTDAYINMONTHFL	numeric	-	2
D_HOLIDAYFL	numeric	-	2
D_WEEKDAYFL	numeric	-	2

ตัวอย่างข้อมูลที่จัดเก็บในระบบจัดการฐานข้อมูลของตาราง CUSTOMER แสดงดังภาพประกอบ ก-6

D_DATEKEY	D_DATE	D_DAYOFWEEK	D_MONTH	D_YEAR	D_YEARMONTHNUM	D_YEARMONTH	D_DAYNUMINWEEK	D_DAYNUMINMONTH	D_DAYNUMINYEAR	D_MONTHNUMINYEAR	D_WEEKNUMINYEAR
19920327	27-Mar-92	Saturday	March	1992	199203	Mar-92	7	27	87	3	13
19920328	28-Mar-92	Sunday	March	1992	199203	Mar-92	1	28	88	3	13
19920329	29-Mar-92	Monday	March	1992	199203	Mar-92	2	29	89	3	13
19920330	30-Mar-92	Tuesday	March	1992	199203	Mar-92	3	30	90	3	13
19920331	31-Mar-92	Wednesday	March	1992	199203	Mar-92	4	31	91	3	14
19920401	1-Apr-92	Thursday	April	1992	199204	Apr-92	5	1	92	4	14
19920402	2-Apr-92	Friday	April	1992	199204	Apr-92	6	2	93	4	14
19920403	3-Apr-92	Saturday	April	1992	199204	Apr-92	7	3	94	4	14
19920404	4-Apr-92	Sunday	April	1992	199204	Apr-92	1	4	95	4	14
19920405	5-Apr-92	Monday	April	1992	199204	Apr-92	2	5	96	4	14
19920406	6-Apr-92	Tuesday	April	1992	199204	Apr-92	3	6	97	4	14
19920407	7-Apr-92	Wednesday	April	1992	199204	Apr-92	4	7	98	4	15
19920408	8-Apr-92	Thursday	April	1992	199204	Apr-92	5	8	99	4	15
19920409	9-Apr-92	Friday	April	1992	199204	Apr-92	6	9	100	4	15
19920410	10-Apr-92	Saturday	April	1992	199204	Apr-92	7	10	101	4	15
19920411	11-Apr-92	Sunday	April	1992	199204	Apr-92	1	11	102	4	15
19920412	12-Apr-92	Monday	April	1992	199204	Apr-92	2	12	103	4	15
19920413	13-Apr-92	Tuesday	April	1992	199204	Apr-92	3	13	104	4	15
19920414	14-Apr-92	Wednesday	April	1992	199204	Apr-92	4	14	105	4	16
19920415	15-Apr-92	Thursday	April	1992	199204	Apr-92	5	15	106	4	16

ภาพประกอบ ก-6 ตัวอย่างข้อมูลที่จัดเก็บในระบบฐานข้อมูลของตาราง DATES

ก.2 คำสั่งสอบถามที่ใช้ในการทดสอบระบบ

คำสั่งสอบถามที่ใช้ในการทดสอบระบบ จะใช้คำสั่งสอบถามตัวอย่างของ SSB Benchmark ซึ่งได้มีการปรับปรุงให้เหมาะสมกับโครงสร้างของคลังข้อมูลมาตรฐานนี้ ซึ่งคำสั่งสอบถามทั้งหมดสามารถแสดงได้ ดังต่อไปนี้

1) RDBMS Q1.1

```
SELECT sum(lo_extendedprice*lo_discount) as revenue
FROM lineorder, dates
WHERE  lo_orderdate = d_datekey
        and d_year = 1993
        and lo_discount between 1 and 3
        and lo_quantity < 25;
```

2) RDBMS Q1.2

```
SELECT sum(lo_extendedprice*lo_discount) as revenue
FROM lineorder, dates
WHERE  lo_orderdate = d_datekey
        and d_yearmonthnum = 199401
        and lo_discount between 4 and 6
        and lo_quantity between 26 and 35;
```

3) RDBMS Q1.3

```
SELECT sum(lo_extendedprice*lo_discount) as revenue
FROM lineorder, dates
WHERE  lo_orderdate = d_datekey
       and d_weeknuminyear = 6
       and d_year = 1994
       and lo_discount between 5 and 7
       and lo_quantity between 26 and 35;
```

4) RDBMS Q2.1

```
SELECT sum(lo_revenue) as lo_revenue, d_year, p_brand1
FROM lineorder, dates, part, supplier
WHERE  lo_orderdate = d_datekey
       and lo_partkey = p_partkey
       and lo_suppkey = s_suppkey
       and p_category = 'MFGR#12'
       and s_region = 'AMERICA'
GROUP BY d_year, p_brand
ORDER BY d_year, p_brand;
```

5) RDBMS Q2.2

```
SELECT sum(lo_revenue) as lo_revenue, d_year, p_brand1
FROM lineorder, dates, part, supplier
WHERE  lo_orderdate = d_datekey
       and lo_partkey = p_partkey
       and lo_suppkey = s_suppkey
       and p_brand1 between 'MFGR#2221' and 'MFGR#2228'
       and s_region = 'ASIA'
GROUP BY d_year, p_brand
ORDER BY d_year, p_brand;
```

6) RDBMS Q2.3

```
SELECT sum(lo_revenue) as lo_revenue, d_year, p_brand1
FROM lineorder, dates, part, supplier
WHERE  lo_orderdate = d_datekey
       and lo_partkey = p_partkey
       and lo_suppkey = s_suppkey
       and p_brand1 = 'MFGR#2239'
       and s_region = 'EUROPE'
GROUP BY d_year, p_brand1
ORDER BY d_year, p_brand1;
```

7) RDBMS Q3.1

```
SELECT c_nation, s_nation, d_year, sum(lo_revenue) as lo_revenue
FROM customer, lineorder, supplier, dates
WHERE  lo_custkey = c_custkey
       and lo_suppkey = s_suppkey
       and lo_orderdate = d_datekey
       and c_region = 'ASIA'
       and s_region = 'ASIA'
       and d_year >= 1992 and d_year <= 1997
GROUP BY c_nation, s_nation, d_year
ORDER BY d_year asc, lo_revenue desc;
```

8) RDBMS Q3.2

```
SELECT c_city, s_city, d_year, sum(lo_revenue) as lo_revenue
FROM customer, lineorder, supplier, dates
WHERE  lo_custkey = c_custkey
       and lo_suppkey = s_suppkey
       and lo_orderdate = d_datekey
       and c_nation = 'UNITED STATES'
       and s_nation = 'UNITED STATES'
```

```
        and d_year >= 1992 and d_year <= 1997
GROUP BY c_city, s_city, d_year
ORDER BY d_year asc, lo_revenue desc;
```

9) RDBMS Q3.3

```
SELECT c_city, s_city, d_year, sum(lo_revenue) as lo_revenue
FROM customer, lineorder, supplier, dates
WHERE  lo_custkey = c_custkey
        and lo_suppkey = s_suppkey
        and lo_orderdate = d_datekey
        and (c_city='UNITED KI1' or c_city='UNITED KI5')
        and (s_city='UNITED KI1' or s_city='UNITED KI5')
        and d_year >= 1992 and d_year <= 1997
GROUP BY c_city, s_city, d_year
ORDER BY d_year asc, lo_revenue desc;
```

10) RDBMS Q3.4

```
SELECT c_city, s_city, d_year, sum(lo_revenue) as lo_revenue
FROM customer, lineorder, supplier, dates
WHERE  lo_custkey = c_custkey
        and lo_suppkey = s_suppkey
        and lo_orderdate = d_datekey
        and (c_city='UNITED KI1' or c_city='UNITED KI5')
        and (s_city='UNITED KI1' or s_city='UNITED KI5')
        and d_yearmonth = 'Dec-97'
GROUP BY c_city, s_city, d_year
ORDER BY d_year asc, lo_revenue desc;
```

11) RDBMS Q4.1

```
SELECT d_year, c_nation, sum(lo_revenue - lo_supplycost) as profit
FROM dates, customer, supplier, part, lineorder
WHERE  lo_custkey = c_custkey
```

```
        and lo_suppkey = s_suppkey
        and lo_partkey = p_partkey
        and lo_orderdate = d_datekey
        and c_region = 'AMERICA'
        and s_region = 'AMERICA'
        and (p_mfgr = 'MFGR#1' or p_mfgr = 'MFGR#2')
GROUP BY d_year, c_nation
ORDER BY d_year, c_nation;
```

12) RDBMS Q4.2 (12)

```
SELECT d_year, s_nation, p_category, sum(lo_revenue -
lo_supplycost) as profit
FROM dates, customer, supplier, part, lineorder
WHERE  lo_custkey = c_custkey
        and lo_suppkey = s_suppkey
        and lo_partkey = p_partkey
        and lo_orderdate = d_datekey
        and c_region = 'AMERICA'
        and s_region = 'AMERICA'
        and (d_year = 1997 or d_year = 1998)
        and (p_mfgr = 'MFGR#1' or p_mfgr = 'MFGR#2')
GROUP BY d_year, s_nation, p_category
ORDER BY d_year, s_nation, p_category;
```

13) RDBMS Q4.3

```
SELECT d_year, s_city, p_brand, sum(lo_revenue - lo_supplycost)
as profit
FROM dates, customer, supplier, part, lineorder
WHERE  lo_custkey = c_custkey
        and lo_suppkey = s_suppkey
        and lo_partkey = p_partkey
        and lo_orderdate = d_datekey
```

```
        and c_region = 'AMERICA'
        and s_nation = 'UNITED STATES'
        and (d_year = 1997 or d_year = 1998)
        and p_category = 'MFGR#14'
GROUP BY d_year, s_city, p_brand
ORDER BY d_year, s_city, p_brand;
```

ก.3 การสร้างดัชนีเพื่อวัดประสิทธิภาพของระบบ

คำสั่งในการสร้างดัชนี Bitmap Join Index บนระบบจัดการฐานข้อมูล Oracle มีรูปแบบคำสั่งดังภาพประกอบ ก-7

```
CREATE BITMAP INDEX Index_Name
ON      Table1_Name(Table2_Name.Attribute_Name) [ASC | DESC]
FROM  Table1_Name, Table2_Name
WHERE Condition;
```

ภาพประกอบ ก-7 รูปแบบคำสั่งสำหรับสร้างดัชนี Bitmap Join Index

โดยที่

- *Index_name* คือ ชื่อของดัชนีที่ต้องการสร้าง
- *Table1_name* คือ ชื่อของตารางที่ต้องการสร้างดัชนี
- *Table2_name* คือ ชื่อของตารางที่นำมา Join
- *Attribute_name* คือ ชื่อของแอททริบิวต์ที่ต้องการสร้างดัชนี
- *Condition* คือ เงื่อนไขในการ Join

ตัวอย่าง เช่น เมื่อต้องการสร้างดัชนี Bitmap Join Index บนตาราง LINEORDER ซึ่ง Join กับตาราง SUPPLIER บนแอททริบิวต์ S_NATION สามารถเขียนคำสั่งได้ดังภาพประกอบ ก-8

```
CREATE BITMAP INDEX     S_NATION_BJIX
ON                      LINEORDER (SUPPLIER.S_NATION)
FROM                    LINEORDER, SUPPLIER
WHERE                   LINEORDER.LO_SUPPKEY = SUPPLIER.S_SUPPKEY;
```

ภาพประกอบ ก-8 ตัวอย่างการใช้คำสั่งในการสร้างดัชนี Bitmap Join Index

สำหรับรูปแบบคำสั่งที่ใช้ในการลบดัชนีที่สร้างขึ้นมาในระบบ สามารถแสดงดังภาพประกอบ ก-9

```
DROP INDEX Index_name;
```

ภาพประกอบ ก-9 รูปแบบคำสั่งสำหรับการลบดัชนี

ภาคผนวก ข.

ผลงานตีพิมพ์

เรื่อง	การคัดเลือกดัชนี Bitmap Join Index สำหรับคลังข้อมูลโดยมีพื้นที่จัดเก็บดัชนีที่จำกัด
งานประชุมวิชาการ	The 10th National Conference on Computing and Information Technology (NCCIT 2014)
สถานที่	จังหวัดภูเก็ต ประเทศไทย
วันที่	8-9 พฤษภาคม 2557