บทที่ 2

คลังข้อมูลและการค้นหาข้อมูล

สำหรับบทนี้ก็จะเป็นการกล่าวถึงคุณลักษณะของคลังข้อมูล ความแตกต่างระหว่างฐานข้อมูลดำเนินการและคลังข้อมูล แม่แบบแผนผังคลังข้อมูล ประโยชน์ของการสร้างคลังข้อมูล และวิธีการค้นหาข้อมูล

2.1 คลังข้อมูล (Data Warehouse)

คลังข้อมูลเป็นที่เก็บรวบรวมข้อมูล เพื่อใช้ในการสนับสนุนการตัดสินใจขององค์กรเป็นหัวใจสำคัญ โดยมีลักษณะดังต่อไปนี้ [13, 14, 15, 20]

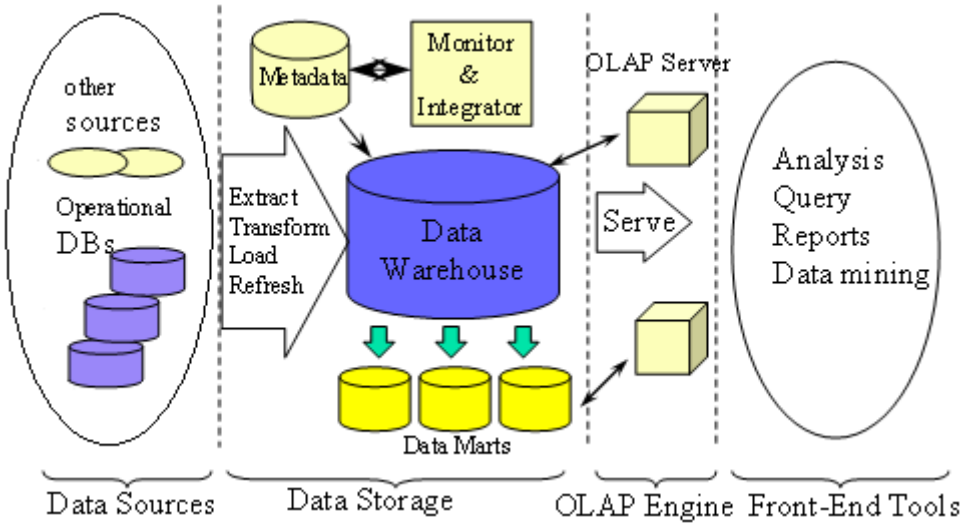
- การกำหนดทิศทางหรือมุ่งเน้นไปที่หัวข้อ (Subject-Oriented)
- การรวมเข้าด้วยกันเป็นหนึ่งเดียว (Integrated)
- มีเวลาเข้ามาเกี่ยวข้อง (Time-Variant)
- ไม่เปลี่ยนแปลงได้ง่ายหรือข้อมูลมีความเสถียร (Nonvolatile)

2.1.1 การกำหนดทิศทางหรือมุ่งเน้นไปที่หัวข้อ

เป็นการเก็บรวบรวมข้อมูลที่มีการกำหนดทิศทางหรือมุ่งเน้นไปที่หัวข้อหลัก ซึ่งเป็นข้อมูลขององค์กร มิใช่เป็นแอพลิเคชัน เช่น ลูกค้า ผู้จำหน่าย สินค้า การขาย เป็นต้น โดยมีจุดความสนใจอยู่ที่การสร้างและวิเคราะห์ข้อมูลเพื่อใช้ในการตัดสินใจ ไม่ใช่การดำเนินงานเป็นรายวันหรือเป็นการประมวลผลทรานแซคชัน และจะตัดข้อมูลที่ไม่มีประโยชน์ในการสนับสนุนการตัดสินใจออกไป ดังนั้นข้อมูลที่ถูกจัดเก็บและแสดงออกมาจึงอยู่ในรูปแบบที่ง่ายและกะทัดรัดภายใต้หัวข้อเรื่องที่ผู้ใช้สนใจ

2.1.2 การรวมเข้าด้วยกันเป็นหนึ่งเดียว

คลังข้อมูลถูกสร้างขึ้นจากการเก็บรวบรวมข้อมูลจากหลาย ๆ แหล่งข้อมูลที่แตกต่างกัน เช่น ฐานข้อมูลเชิงความสัมพันธ์ ไฟล์ธรรมดา และรายการทรานแซคชันออนไลน์ (On-Line Transaction Records) มารวมเข้าไว้ด้วยกันในที่เดียวกัน ซึ่งข้อมูลแต่ละแหล่งก็ย่อมมีความแตกต่างกัน หรือไม่สอดคล้องกันในเรื่องการตั้งชื่อ โครงสร้างการเข้ารหัส หน่วยการวัดแอทริบิวต์ และอื่น ๆ เช่น หน่วยเงินตรา ภาษี ค่าอาหารเช้า เป็นต้น ดังนั้นเมื่อมีการย้ายข้อมูลไปยังคลังข้อมูล ก็ต้องมีเทคนิควิธีการล้างทำความสะอาดข้อมูล และแปลงข้อมูล เพื่อให้เกิดความสอดคล้องตรงกันและเป็นไปในทิศทางเดียวกัน ดูภาพประกอบ 2-1



ภาพประกอบ 2-1 สถาปัตยกรรมคลังข้อมูล (Architecture of Data Warehouses) [18]

2.1.3 มีเวลาเข้ามาเกี่ยวข้อง

ทุก ๆ โครงสร้างหลักในคลังข้อมูล จะต้องประกอบด้วยเวลาเป็นปัจจัยสำคัญเสมอ อาจจะเป็นแบบชัดแจ้งหรือไม่ชัดแจ้งก็ได้ ทั้งนี้เพราะคลังข้อมูลเป็นการจัดเตรียมข้อมูลที่เป็นประวัติเอาไว้ ซึ่งเป็นสิ่งจำเป็นสำหรับการวิเคราะห์แนวโน้มของธุรกิจ สามารถแสดงได้หลายรูปแบบที่แตกต่างกันเพื่อให้เข้าใจได้ง่ายขึ้น โดยทั่วไปข้อมูลประวัติมักจะเป็นช่วงเวลา 5-10 ปีที่ผ่านมา ซึ่งจะตรงกันข้ามกับระบบฐานข้อมูลดำเนินการ (Operational Database System) ที่จะเป็นการเก็บรวบรวมข้อมูลที่เป็นปัจจุบัน ณ ขณะนั้น หรือมีการเก็บข้อมูลไว้ไม่เกิน 60 – 90 วัน ดังนั้นระบบฐานข้อมูลดำเนินการอาจจะมีเวลาเป็นองค์ประกอบด้วยหรือไม่ก็ได้

2.1.4 ไม่เปลี่ยนแปลงได้ง่ายหรือข้อมูลมีความเสถียร

การดำเนินการกับคลังข้อมูล มีเพียงแค่ 2 อย่างเท่านั้น คือ การเริ่มต้นโหลดข้อมูลและการเข้าถึงข้อมูล ดังนั้นจึงไม่จำเป็นต้องมีกลไกในการควบคุมการประมวลผลรายการข้อมูล การกู้คืน และการเกิดขึ้นพร้อมกัน เช่น การอ่านและการเขียน

โดยปกติการเข้าถึงคลังข้อมูล มักจะเป็นการอ่านเพียงอย่างเดียว ส่วนการปรับปรุงข้อมูลให้ทันสมัยนั้นมักจะเป็นการเพิ่มข้อมูลเข้าไปในคลังข้อมูล จะไม่มีการลบหรือแก้ไขข้อมูลในคลังข้อมูล ดังนั้นข้อมูลในคลังข้อมูลจึงเป็นข้อมูลที่เสถียร

ประโยชน์ของการสร้างคลังข้อมูล คือ เพื่อรองรับ OLAP และการทำเหมืองข้อมูล (Data Mining) ทั้งนี้เพื่อใช้ในการวิเคราะห์ข้อมูลและสนับสนุนการตัดสินใจ โดยจะมีการ

เก็บรวบรวมสารสนเทศจากหลายแหล่งข้อมูลที่แตกต่างกันเข้าไว้ด้วยกันในคลังข้อมูล เพื่อให้ผู้ใช้สามารถทำการสอบถามข้อมูลได้โดยตรง ทำการวิเคราะห์ และตัดสินใจได้ [16]

2.2 ฐานข้อมูลดำเนินการและคลังข้อมูล (Operational Databases vs. Data Warehouses)

2.2.1 ฐานข้อมูลดำเนินการ (Operational Databases)

งานหลัก [16] คือ OLTP (On-Line Transaction Processing) โดยจะมีการดำเนินงานแบบวันต่อวัน เช่น การซื้อ คลังสินค้า ธนาคาร โรงงาน เงินเดือน การลงทะเบียน การบัญชี และอื่น ๆ ผู้ใช้งานคือ ผู้ออกแบบระบบ ผู้บริหารระบบ

2.2.2 คลังข้อมูล (Data Warehouses)

งานหลัก [16] คือ OLAP (On-Line Analytical Processing) เพื่อใช้ในการวิเคราะห์ข้อมูลและสนับสนุนการตัดสินใจ โดยจะมีการเก็บรวบรวมสารสนเทศจากหลายแหล่งข้อมูลที่แตกต่างกันเข้าไว้ด้วยกันในคลังข้อมูล เพื่อให้ผู้ใช้สามารถทำการสอบถามข้อมูลได้โดยตรง เพื่อนำไปใช้ในการวิเคราะห์ และสนับสนุนการตัดสินใจ ผู้ใช้งาน คือ ผู้ตัดสินใจ ผู้บริหาร

ฐานข้อมูลดำเนินการและคลังข้อมูล สามารถสรุปเป็นตารางเปรียบเทียบได้ดังตาราง 2-1 [14, 16, 24]

ตาราง 2-1 ตารางเปรียบเทียบฐานข้อมูลดำเนินการและคลังข้อมูล

ลักษณะ/หัวข้อ	ฐานข้อมูลดำเนินการ	คลังข้อมูล
ผู้ใช้	- ผู้ออกแบบระบบ ผู้บริหารระบบ เสมียนนำเข้าข้อมูล ผู้ชำนาญการทางด้านเทคโนโลยีสารสนเทศ	- ผู้ตัดสินใจ ผู้รู้ ผู้บริหาร
ฟังก์ชันหน้าที่การทำงาน	- ดำเนินการรายวัน - การประมวลผลทรานแซคชันแบบออนไลน์ (Transaction Driven)	- สนับสนุนการตัดสินใจ - กระบวนการวิเคราะห์ข้อมูลแบบออนไลน์ (Analytical Driven)
อายุของข้อมูล	- ปัจจุบัน และ/หรือ 60-90 วัน	- ประวัติ นาน 5-10 ปี และปัจจุบัน
การออกแบบฐานข้อมูล	- มุ่งเน้นที่แอพลิเคชัน	- มุ่งเน้นที่หัวข้อเรื่อง

ตาราง 2-1 ตารางเปรียบเทียบฐานข้อมูลดำเนินการและคลังข้อมูล (ต่อ)

ลักษณะ/หัวข้อ	ฐานข้อมูลดำเนินการ	คลังข้อมูล
ขนาดฐานข้อมูล	- เมกะไบต์ – กิกะไบต์	- กิกะไบต์ – เทอราไบต์
แม่แบบจำลอง (Model)	- แผนภาพอี-อาร์ (E-R Diagram)	- แผนภาพหลายมิติ
ข้อมูล	- ปัจจุบัน ปรับปรุงข้อมูลให้ทันสมัย - ตารางเชิงสัมพันธ์ (Normalized) - หน่วยเล็กที่สุด - ข้อมูลแยกออกจากกัน - พลวัต (Dynamic)	- เก็บเป็นประวัติ - หลายมิติ (Multidimensional) - รวบรวมเข้าด้วยกัน เพื่อทำให้ข้อมูลมีน้ำหนักขึ้น - คงที่ (Static)
ลักษณะการสอบถาม	- ไม่เน้น Group By	- เน้น Group By
การใช้งาน	- ทำซ้ำ งานประจำ - การสอบถามข้อมูลแบบสำเร็จรูป มีการเตรียมไว้ให้แก่ผู้ใช้ล่วงหน้า (Canned Queries)	- การสอบถามข้อมูลแบบทันทีทันใด ไม่ทราบล่วงหน้ามาก่อนว่าจะถามอะไร (Ad-hoc Queries) และคำถามหลากหลายรูปแบบ
การเข้าถึง	- อ่าน/เขียน - ทรานแซคชันแบบง่าย ๆ และส่วนมากมักจะเกี่ยวข้อง 1 – 3 ตาราง	- ส่วนมากเป็นการอ่าน - การสอบถามข้อมูลที่ซับซ้อนเกี่ยวข้องกับหลาย ๆ ตาราง - เน้นการสแกนข้อมูลมาก
การดำเนินการเขียนข้อมูล	- ปรับปรุงข้อมูลให้ทันสมัย แทรก ลบ	- แทรก
การปรับปรุงข้อมูลให้ทันสมัย	- เวลาจริง (Real-Time)	- เป็นช่วงเวลา/ระยะเวลา (Periodically)
กลไกควบคุมการทำงาน	- จำเป็นมากที่จะต้องมีกลไกควบคุมการกู้กลับคืน การเกิดขึ้นพร้อมกัน	- ไม่จำเป็นต้องมีกลไกควบคุมการกู้กลับคืน การเกิดขึ้นพร้อมกัน
เมตริกซ์วัดปริมาณงาน	- ทรานแซคชัน (Transaction Throughput)	- การสอบถามข้อมูล (Query Throughput)

คลังข้อมูล จะต้องมีการเตรียมข้อมูลสำหรับกระบวนการวิเคราะห์ข้อมูลที่เป็นแบบหลายมิติ ดังนั้นตัวแบบสำหรับการสร้างคลังข้อมูลควรจะต้องมีความสามารถและเหมาะสมในการแสดงแบบหลายมิติได้

ในการสร้างตัวแบบหลายมิตินั้นจะต้องมี 2 อย่างเข้ามาเกี่ยวข้อง คือ ตาราง Fact และตาราง Dimensions โดยตาราง Fact จะเป็นตารางที่มีขนาดใหญ่มาก คีย์หลักของตารางจะเป็นคีย์ผสม และจะต้องมีเวลาเข้ามาเป็นส่วนหนึ่งของคีย์ด้วยเสมอ และมี Foreign Key เป็นตัวเชื่อมไปยังตาราง Dimension ซึ่งตาราง Dimension จะมีขนาดข้อมูลเล็กมากเมื่อเปรียบเทียบกับตาราง Fact โดยตาราง Dimension จะเก็บข้อมูลรายละเอียดของหัวข้อเรื่องนั้น ๆ เอาไว้ เช่น ตาราง Dimension สินค้า (product) ก็จะเก็บข้อมูลรายละเอียดของสินค้าเอาไว้ เช่น รหัสสินค้า ชื่อสินค้า สี ขนาด ยี่ห้อ และรุ่น เป็นต้น

2.3 การสร้างคลังข้อมูลเชิงแนวคิด (Conceptual Modeling of Data Warehouses)

ในการออกแบบฐานข้อมูลเชิงความสัมพันธ์จะมีการใช้แผนภาพอี-อาร์ (Entity-Relationship Diagrams) กันอย่างแพร่หลาย ซึ่งจะประกอบด้วย เซตของเอนติตี้หรือออบเจกต์และความสัมพันธ์ระหว่างเอนติตี้ ตัวแบบนี้เหมาะสำหรับการประมวลผลทรานแซคชันแบบออนไลน์ (On-Line Transaction Processing) แต่ไม่เหมาะสำหรับระบบสนับสนุนการตัดสินใจที่ต้องการประสิทธิภาพสูงในการประมวลผลการสอบถามและการโหลดข้อมูล สำหรับคลังข้อมูลจะต้องการความกะทัดรัด จึงมุ่งเน้นไปที่หัวข้อเรื่องเพื่อให้การวิเคราะห์ข้อมูลแบบออนไลน์สะดวกขึ้น ดังนั้นตัวแบบที่มีชื่อเสียงและใช้กันมากที่สุด คือ ตัวแบบหลายมิติ (Multidimensional Model) ซึ่งมีปัจจัยที่ต้องพิจารณา คือ มิติ (Dimensions) และตัววัด (Measures) โดยสามารถแบ่งออกเป็น 3 แบบด้วยกัน คือ [13, 14, 16]

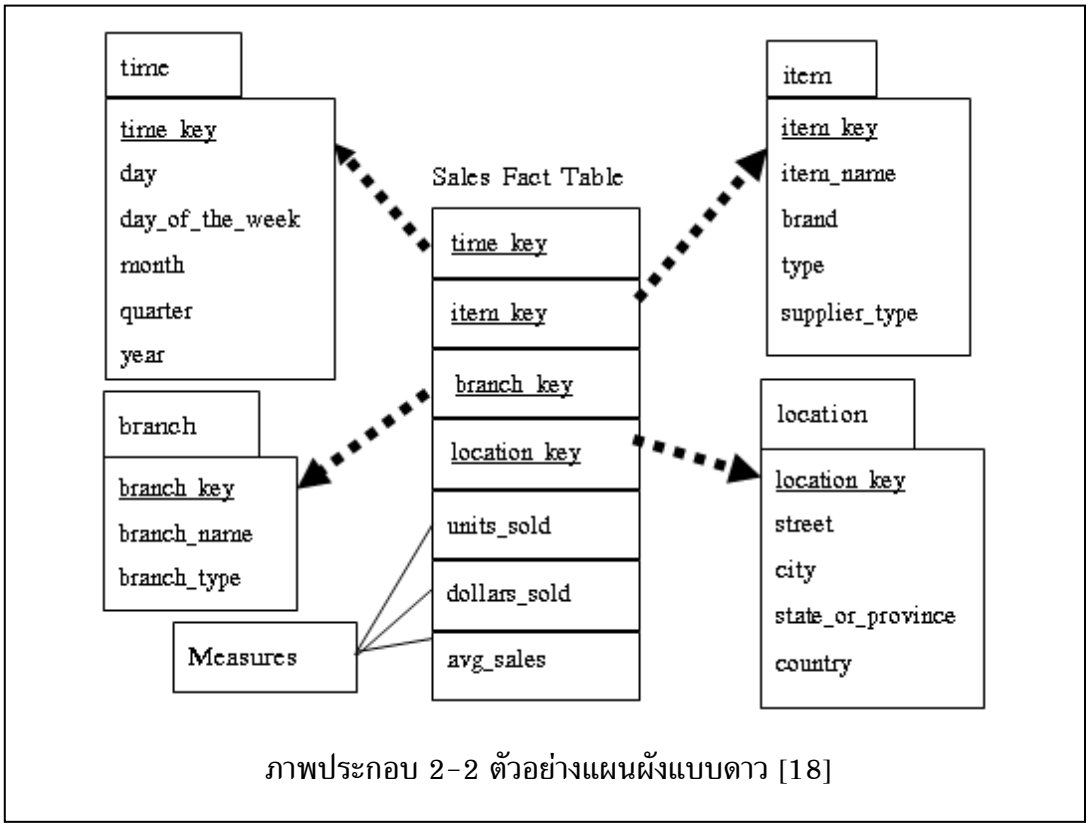
2.3.1 แผนผังแบบดาว (Star Schema)

แผนผังแบบดาว ประกอบด้วยตาราง Fact อยู่ตรงกลาง แล้วล้อมรอบด้วยกลุ่มของตาราง Dimensions และมีการเชื่อมต่อกันอยู่ระหว่างตาราง Fact และ Dimensions โดยมี Foreign Key เป็นตัวเชื่อม แผนผังแบบดาวเป็นแผนผังที่มีการใช้กันมากที่สุด ดังภาพประกอบ 2-2

การวัดสิ่งที่สนใจของ OLAP จะถูกเก็บรักษาไว้ในตาราง Fact เช่น จำนวนดอลลาร์ (dollars_sold) จำนวนหน่วยในตารางการขาย (units_sold)

สำหรับแต่ละมิติของตัวแบบหลายมิติ (Multidimensional Model) ก็คือตาราง Fact 1 ตาราง (sales) และตาราง Dimension หลายตาราง เช่น จากภาพประกอบ 2-2 ตารางรายการสินค้า (item), ตารางตำแหน่งที่ตั้ง (location), ตารางสาขา (branch) และตารางเวลา (time) โดยแต่ละตารางก็จะประกอบด้วยแอทริบิวต์ที่แตกต่างกัน ตัวอย่างเช่น

ตาราง Dimension ตำแหน่งที่ตั้ง ประกอบด้วยแอทริบิวต์รหัสตำแหน่งที่ตั้ง (location_key), ถนน (street), เมือง (city), จังหวัด (state_or_province) และประเทศ (country)

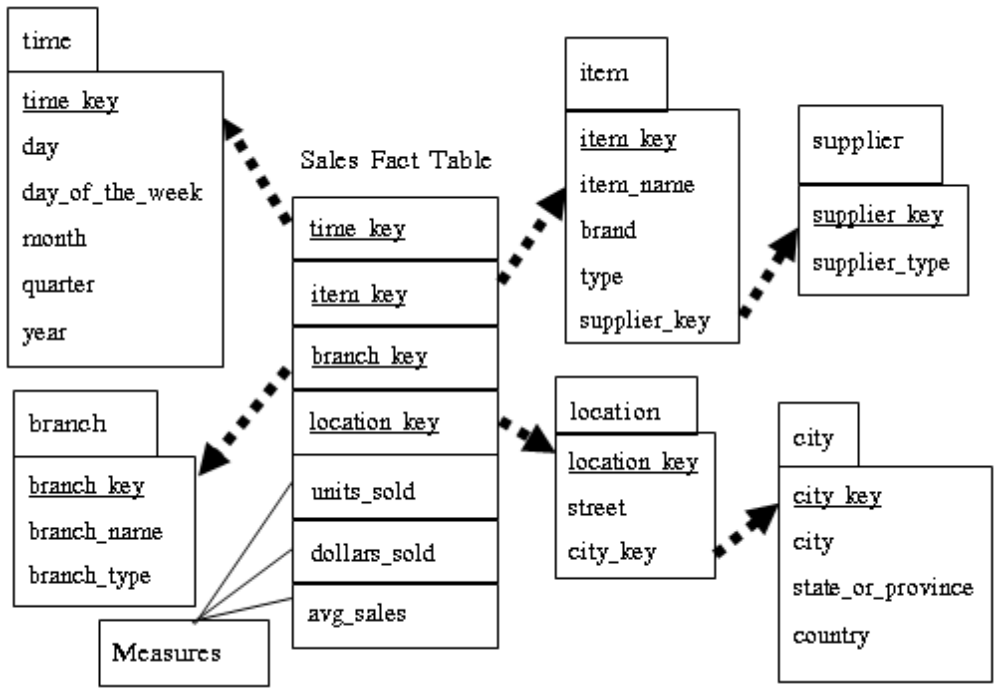


ภาพประกอบ 2-2 ตัวอย่างแผนผังแบบดาว [18]

2.3.2 แผนผังแบบผลึกหิมะ (Snowflake Schema)

เป็นการปรับแผนผังแบบดาวให้นอร์มัลไลซ์ โดยการปรับตาราง Dimension ให้เห็นเป็นลำดับขั้นอย่างชัดเจน โดยการแตกตารางนั้นขึ้นมาเพิ่มเติม ยกตัวอย่างเช่น จากภาพประกอบ 2-2 แผนผังแบบดาว จะเห็นได้ว่าตาราง Dimension ตำแหน่งที่ตั้ง (location) จะมีแอทริบิวต์เมือง (city), จังหวัด (state_or_province) และประเทศ (country) ที่ซ้ำซ้อนกันได้ ดังนั้นจึงต้องทำการนอร์มัลไลซ์ โดยการแตกแอทริบิวต์เหล่านี้ออกมาเป็นตาราง Dimension ลำดับขั้นใหม่ขึ้นมาอีกตารางหนึ่งชื่อว่า ตาราง Dimension เมือง (city) และมีการเชื่อมต่อตาราง Dimension ตำแหน่งที่ตั้ง ที่ไม่มีแอทริบิวต์เมือง จังหวัด และประเทศ แต่จะมีแอทริบิวต์รหัสเมืองเป็นตัวเชื่อมต่อไปยังตาราง Dimension เมืองแทน ซึ่งจะเห็นได้ว่าจะได้ตาราง Dimension ที่มีจำนวนตารางเพิ่มขึ้น แต่ขนาดของตารางจะเล็กลง รูปร่างจะคล้ายกับผลึกหิมะ ดังภาพประกอบ 2-3

ข้อดีของแผนผังแบบนี้ คือ [16, 22] การดูแลบำรุงรักษาจะง่ายและประหยัดพื้นที่ในการจัดเก็บข้อมูล แต่มีข้อเสีย คือ การตอบการสอบถามข้อมูลจะช้าลง ทำให้ประสิทธิภาพของระบบลดลง ดังนั้นแผนผังแบบนี้จึงได้รับความนิยมน้อยกว่าแผนผังแบบดาวในการออกแบบคลังข้อมูล

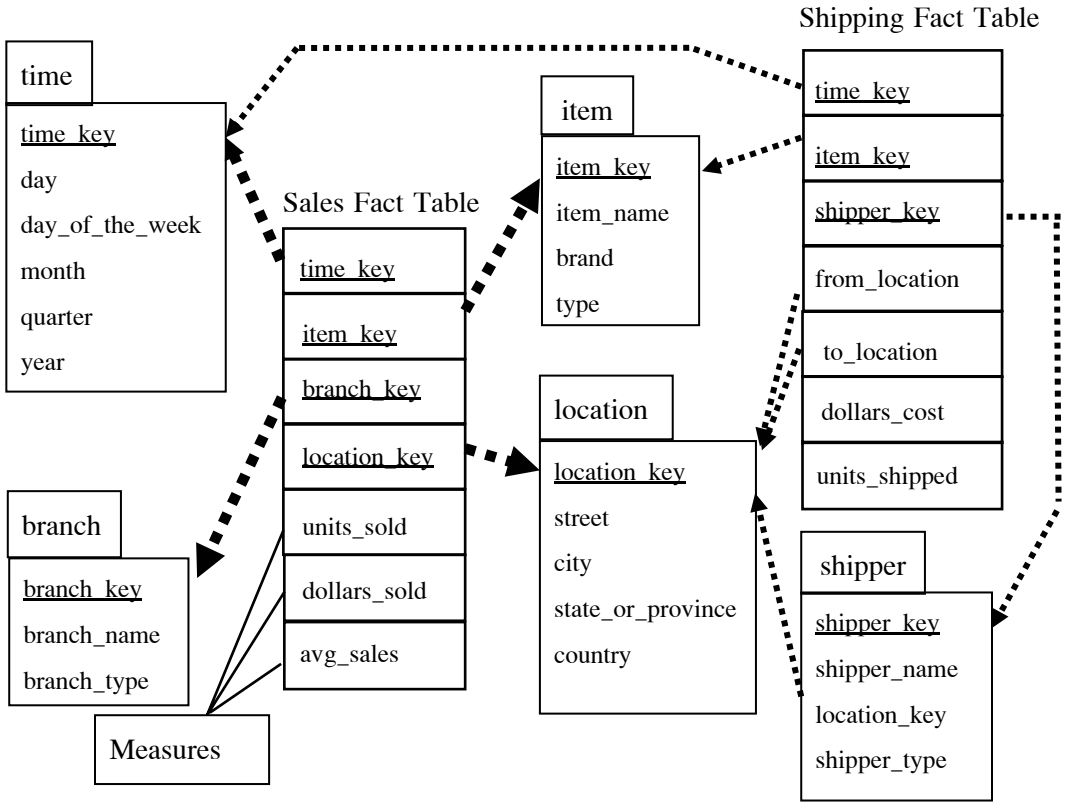


ภาพประกอบ 2-3 ตัวอย่างแผนผังแบบผลึกหิมะ [18]

2.3.3 แผนผังแบบกลุ่มกาแลคซี่ของ Fact (Fact Constellations)

ประกอบด้วยตาราง Fact หลาย ๆ ตาราง มีการใช้ตาราง Dimensions ร่วมกัน ภาพที่มองเห็นจะเป็นเหมือนการรวมตัวกันของดาว ดังนั้นจึงเรียกว่า แผนผังแบบกาแลคซี่ (Galaxy Schema) หรือกลุ่มดาว Fact (Fact Constellation) ดังภาพประกอบ 2-4

จากภาพประกอบ 2-4 จะเห็นได้ว่า แผนผังแบบกลุ่มกาแลคซี่ของ Fact ประกอบด้วยตาราง Fact 2 ตาราง คือ ตารางการขาย (sales) และการขนส่ง (shipping) โดยตารางการขายจะเป็นแผนผังแบบดาว ตารางการขนส่งประกอบด้วย 5 แอทริบิวต์ คือ รหัสเวลา (time_key), รหัสรายการ (item_key), รหัสการขนส่ง (shipper_key), ตำแหน่งที่ตั้งเริ่มต้น (from_location) และตำแหน่งที่ตั้งสิ้นสุด (to_location) และแอทริบิวต์ที่เป็นตัววัด 2 แอทริบิวต์ คือ ค่าใช้จ่ายเป็นดอลลาร์ (dollars_cost) และ จำนวนหน่วยที่ขนส่ง (units_shipped) ตาราง Fact ทั้ง 2 นี้จะมีการใช้ตาราง Dimension เวลา (time), รายการสินค้า (item) และตำแหน่งที่ตั้ง (location) ร่วมกัน



ภาพประกอบ 2-4 ตัวอย่างแผนผังแบบกลุ่มกาแลคซี่ของ Fact [18]

2.4 การค้นหาข้อมูล (Searching)

การค้นหาข้อมูลที่เราต้องการนั้น สามารถทำได้หลายวิธีด้วยกัน เช่น

1. การเข้าถึงแบบตามลำดับและใช้ลิสต์ หรือตาราง หรืออาร์เรย์
2. การเข้าถึงโดยตรงโดยใช้ค่าคีย์ ซึ่งเรียกว่า การแฮช (Hashing)
3. วิธีการทำดัชนี (Indexing Method)

แต่ในที่นี้จะขอกล่าวถึงเฉพาะวิธีที่ 2 และ 3 เท่านั้น เพราะเลือกใช้วิธีการทำดัชนีในการค้นหาข้อมูลในวิทยานิพนธ์นี้

2.4.1 แฮช (Hash)

แฮช คือ กระบวนการเข้าถึงเรคอร์ดโดยการเทียบค่าคีย์เข้ากับตำแหน่งในตาราง เรียกว่า การแฮชชิง (Hashing) ซึ่งจะต้องมี [12, 25]

- ฟังก์ชันที่ใช้ในการเทียบค่าคีย์เข้ากับตำแหน่ง เรียกว่า ฟังก์ชันแฮช (Hash Function) โดยใช้สัญลักษณ์ h
- อาร์เรย์ที่ใช้เก็บเรคอร์ด เรียกว่า ตารางการแฮช (Hash Table) ใช้สัญลักษณ์ HT โดยตำแหน่งในตารางแฮช จะเรียกว่า สล็อต (Slot) เช่น กำหนดให้เป็นจำนวน M สล็อต คือ สล็อตที่ 0 ถึง M-1 จะได้ว่า $0 \leq h(K) < M$ โดยที่ $HT[i] = K$

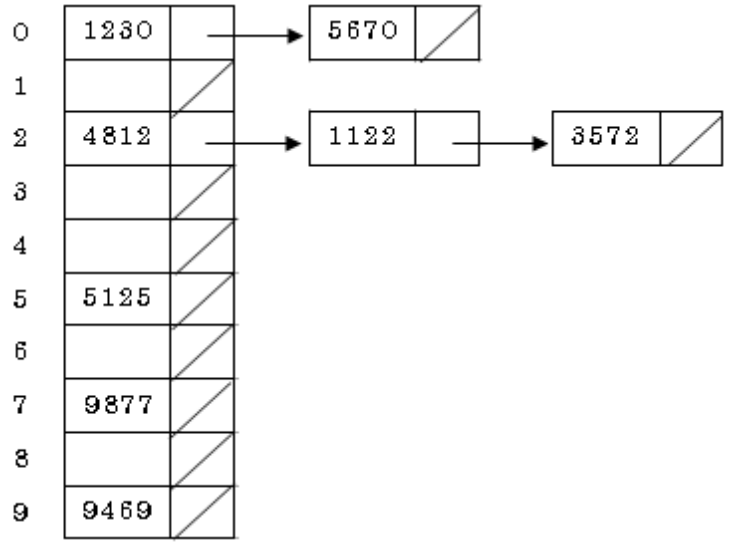
โดยปกติ ช่วงของค่าคีย์จะมีขนาดกว้างกว่าจำนวนสล็อตในตารางแฮช เช่น ค่าของคีย์ที่จะถูกเก็บอยู่ในช่วง 0 – 65,535 เป็นจำนวน 1,000 เรคอร์ด ดังนั้นเราอาจจะสร้างตารางแฮชขึ้นมา 65,536 สล็อต เพื่อรองรับค่าที่เป็นไปได้ของคีย์จำนวน 65,536 ค่า แต่ก็จะทำให้เกิดการเปลืองเนื้อที่สล็อต เพราะจะเป็นสล็อตว่างจำนวนมากที่ไม่ได้เก็บค่าคีย์เอาไว้ ดังนั้นจึงต้องมีการสร้างฟังก์ชันแฮชเพื่อสร้างตารางแฮชที่มีขนาดเล็กมาก ๆ ในการเก็บค่าคีย์เหล่านั้น ถ้าหากช่วงของคีย์ที่เป็นไปได้โตกว่าขนาดของตารางแฮช ก็อาจจะทำให้การเทียบค่าคีย์กับตำแหน่งสล็อตซ้ำกันได้ ทำให้เกิดปัญหาการชนกัน (Collision) ในตำแหน่งสล็อต นั่นคือ ในตำแหน่งสล็อตเดียวกัน มีค่าคีย์หลายค่า ซึ่งถ้าสร้างฟังก์ชันการแฮชไม่ดี ก็จะทำให้เกิดการชนกันสูงมาก แต่ถ้าสร้างฟังก์ชันการแฮชดี โอกาสการชนกันก็จะต่ำ [12, 25]

2.4.1.1 วิธีการแก้ปัญหาการชนกัน แบ่งออกได้เป็น 2 ชนิด คือ [12, 25]

- การแฮชแบบเปิด (Open Hashing) หรือแบบแยก (Separate Chaining) การชนกันจะถูกจัดเก็บไว้นอกตารางแฮช
- การแฮชแบบปิด (Closed Hashing) การชนกันจะถูกจัดเก็บไว้ในตารางแฮชแต่อยู่อีกสล็อตหนึ่ง

การแฮชแบบเปิด (Open Hashing)

การชนกันจะถูกจัดเก็บไว้นอกตารางแฮช คือ เก็บต่อกันไปเรื่อย ๆ [12, 25] ตัวอย่างดังภาพประกอบ 2-5



ภาพประกอบ 2-5 ตัวอย่างการแฮชแบบเปิด โดยตารางแฮชมี 10 สล็อต ใช้ฟังก์ชันแฮช h(K) = K mod 10 จัดเก็บเลข 8 ค่า

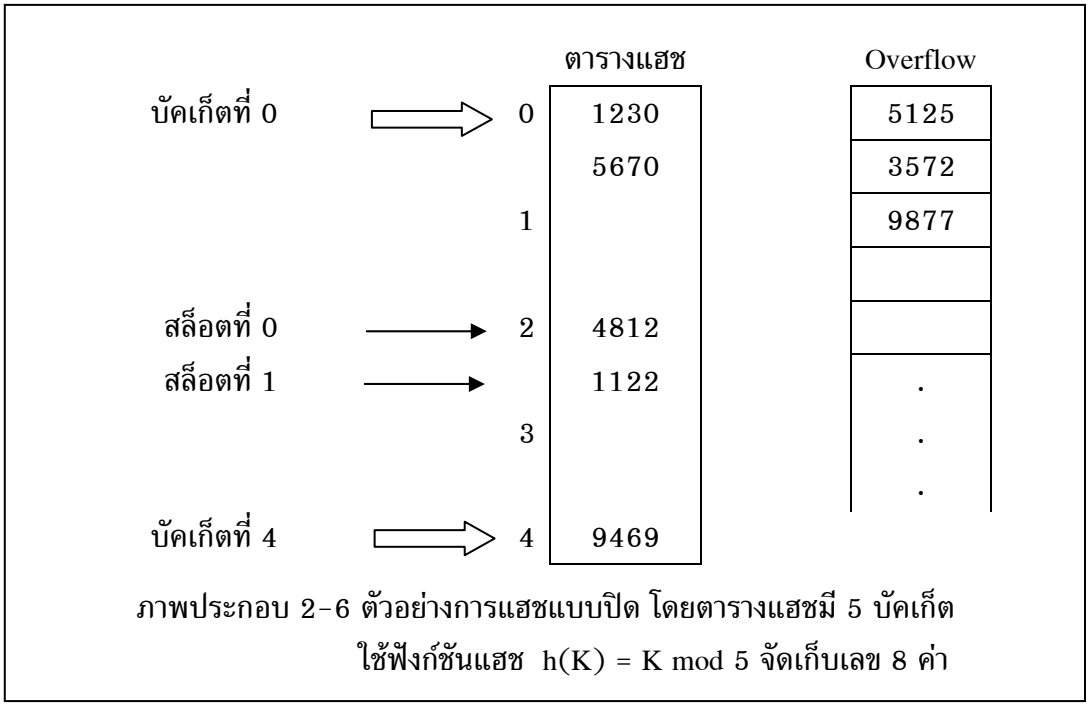
จากภาพประกอบ 2-5 ต้องการจัดเก็บเลขจำนวน 8 ค่า ได้แก่ 4812, 1230, 9469, 5670, 5125, 1122, 3572 และ 9877 ไว้ในสล็อตในตารางแฮช ซึ่งมีจำนวน 10 สล็อต โดยใช้ฟังก์ชัน h(K) = K mod 10 แต่สามารถจัดเก็บไว้ในตารางแฮชได้เพียง 5 ค่าเท่านั้น สำหรับเลขอีก 3 ค่าซ้ำกันกับสล็อตเดิม คือ ค่า 5670 จะต้องจัดเก็บไว้ในสล็อตที่ 0 แต่เนื่องจากได้มีการจัดเก็บค่า 1230 ไว้ในสล็อตที่ 0 แล้ว ดังนั้นจึงจัดเก็บค่า 5670 ไว้นอกตารางแฮชแทน ส่วนค่า 1122 และ 3572 ก็ทำนองเดียวกัน กล่าวคือ ค่า 1122 และ 3572 จะต้องจัดเก็บไว้ในสล็อตที่ 2 แต่เนื่องจากได้มีการจัดเก็บค่า 4812 ไว้ในสล็อตที่ 2 แล้ว ดังนั้นจึงจัดเก็บค่า 1122 และ 3572 ไว้นอกตารางแฮชแทน

การแฮชแบบเปิดเหมาะสำหรับตารางแฮชถูกจัดเก็บไว้ในหน่วยความจำหลักสร้างด้วยลิงก์ลิสต์ [12, 25]

การแฮชแบบปิด (Closed Hashing)

การแฮชแบบปิด จะจัดเก็บทุกเรคอร์ดไว้ในตารางแฮช โดยแต่ละเรคอร์ด i จะมีตำแหน่งที่อยู่เป็น $h(K_i)$ และถ้าตำแหน่งซ้ำกัน ก็จะจัดเก็บค่าใหม่ที่ซ้ำไว้ในสล็อตอื่น [12, 25] การแฮชแบบปิด แบ่งออกเป็น 2 ประเภทย่อย คือ

- **มีบัคเก็ต** (Bucket Hashing) จะมีการแบ่งสล็อต M สล็อตออกเป็น B บัคเก็ต (Bucket) ก็จะได้ $\frac{M}{B}$ สล็อตต่อบัคเก็ต ฟังก์ชันแฮชก็จะเทียบ (Map) ค่าไปยังบัคเก็ต เมื่อบัคเก็ตเต็มก็จะมีการจัดเก็บค่าเอาไว้ที่ Overflow Bucket ซึ่งทุกบัคเก็ตจะมีการใช้ Overflow เดียวกัน [25] ตัวอย่างดังภาพประกอบ 2-6



ภาพประกอบ 2-6 ตัวอย่างการแฮชแบบปิด โดยตารางแฮชมี 5 บัคเก็ต ใช้ฟังก์ชันแฮช h(K) = K mod 5 จัดเก็บเลข 8 ค่า

วิธีการค้นหาข้อมูล [25] โดยการใช้ฟังก์ชันแฮชเป็นตัวกำหนดบัคเก็ต ถ้าหาคีย์ไม่เจอและบัคเก็ตไม่เต็ม ก็จะหยุดการค้นหา แต่ถ้าหาคีย์ไม่เจอและบัคเก็ตเต็ม ก็จะไปค้นหาที่ Overflow Bucket เช่น จากภาพประกอบ 2-6 ตารางแฮชประกอบด้วย 5 บัคเก็ต คือ บัคเก็ตที่ 0 ถึง 4 โดยแต่ละบัคเก็ต มี 2 สล็อต คือ สล็อตที่ 0 และ 1 ในการจัดเก็บค่าข้อมูล 8 ค่า ได้แก่ 4812, 1230, 9469, 5670, 5125, 1122, 3572 และ 9877 ไว้ในตารางแฮช ใช้ฟังก์ชัน h(K) = K mod 5 จะสังเกตได้ว่า มีการจัดเก็บค่า 1230 และ 5670 ไว้ในบัคเก็ตที่ 0 สล็อตที่ 0 และ 1 ตามลำดับ สำหรับค่า 5125 จะจัดเก็บไว้ในตารางแฮช ส่วนที่เรียกว่า Overflow แทน เนื่องจากไม่มีสล็อตว่างของบัคเก็ตที่ 0 ให้จัดเก็บได้อีก สำหรับค่า 3572 และ 9877 ก็ทำนองเดียวกัน

- **ไม่มีบัคเก็ต** (No Bucketing) การแก้ปัญหาการชนกัน คือ สามารถที่จะใช้สล็อตใด ๆ ในตารางแฮชก็ได้ โดยสล็อตแรกจะเป็น Home Position ของคีย์ ถ้า Home Position ไม่ว่าง ก็จะไปยังสล็อตถัดไป จนกระทั่งเจอสล็อตว่าง หรือเมื่อไปจนกระทั่งถึงสล็อตสุดท้ายของตารางแล้ว ก็จะย้อนกลับไปยังสล็อตเริ่มต้นของตารางแฮชอีก ซึ่งมีหลายวิธี เช่น

สล็อตที่	ตารางแฮช	ลำดับที่การป้อนเข้า	คำอธิบาย
0	5125	⑤	h(5125) mod 11 = 10 แต่สล็อตที่ 10 ไม่ว่าง จัดเก็บค่า 9469 อยู่ จึงจัดเก็บ 5125 ไว้ในสล็อตที่ 0 แทน
1	1122	⑥	h(1122) mod 11 = 0 แต่สล็อตที่ 0 ไม่ว่าง จัดเก็บค่า 5125 อยู่ จึงจัดเก็บ 1122 ไว้ในสล็อตที่ 1 แทน
2	9877	⑧	h(9877) mod 11 = 10 แต่สล็อตที่ 10 ไม่ว่าง จัดเก็บค่า 9469 อยู่ และสล็อตที่ 0 และ 1 ก็ไม่ว่าง จึงจัดเก็บ 9877 ไว้ในสล็อตที่ 2 แทน
3			
4			
5	4812	①	h(4812) mod 11 = 5
6	5670	④	h(5670) mod 11 = 5 ชนกันกับ h(4812) mod 11 = 5 จึง จัดเก็บ 5670 ไว้ในสล็อตที่ 5 + 1 คือ 6 แทน
7			
8	3572	⑦	h(3572) mod 11 = 8
9	1230	②	h(1230) mod 11 = 9
10	9469	③	h(9469) mod 11 = 9 ชนกันกับ h(1230) mod 11 = 9 จึง จัดเก็บ 9469 ไว้ในสล็อตที่ 9 + 1 คือ 10 แทน

ภาพประกอบ 2-7 ตัวอย่างการแฮชแบบปิด โดยตารางแฮชมี 11 สล็อต ใช้ฟังก์ชันแฮช (h(K) + i) mod M จัดเก็บเลข 8 ค่า (M=11)

- ***Linear Probing*** โดยมีฟังก์ชันการแฮชอย่างง่าย ๆ ดังนี้

$$(h(K) + i) \bmod M$$

ตัวอย่างเช่น ต้องการจัดเก็บค่าข้อมูล 8 ค่า ได้แก่ 4812, 1230, 9469, 5670, 5125, 1122, 3572 และ 9877 ไว้ในตารางแฮช ดังภาพประกอบ 2-7

2.4.1.2 ข้อดีของการแฮช

การแฮชเหมาะสำหรับเซตเท่านั้น จะไม่นำการแฮชไปใช้กับเรคอร์ดที่มีค่าคีย์เดียวกันหลายเรคอร์ด หรือค่าที่ซ้ำซ้อนกัน การแฮชเหมาะที่สุดสำหรับการสอบถามข้อมูลที่เป็นค่าคงที่ เช่น การค้นหาเรคอร์ดที่มีค่าคีย์เท่ากับ K นอกจากนี้การแฮชเหมาะสำหรับการค้นหาในหน่วยความจำและบนดิสก์ [12, 25] ที่เป็นการค้นหาแบบค่าเท่ากัน

2.4.1.3 ข้อจำกัดของการแฮช

การแฮชไม่เหมาะสำหรับการค้นหาเรคอร์ดที่เป็นช่วง ค่าต่ำสุด ค่าสูงสุด หรือการค้นหาที่มีลำดับที่เข้ามาเกี่ยวข้อง [12, 25] และหาฟังก์ชันการแฮชที่สมบูรณ์แบบ (ไม่เกิดการชนกัน) ได้ยาก

2.4.2 ดัชนี (Index)

ดัชนี เป็นวิธีการที่ช่วยทำให้เราสามารถค้นหาข้อมูล หรือหาแหล่งที่ตั้งของข้อมูลที่ตรงกับเงื่อนไขที่เราต้องการได้อย่างรวดเร็วมาก [12] โดยดัชนีจะถูกเก็บบันทึกไว้ในไฟล์อีกชนิดหนึ่งที่ประกอบด้วยคีย์และตัวชี้ (Key/Pointer) ที่สามารถเข้าถึงเรคอร์ดข้อมูลได้โดยตรง [12, 26] ทุก ๆ ดัชนีจะมีคีย์ที่ใช้ในการค้นหาที่อาจจะประกอบด้วย 1 แอทริบิวต์หรือมากกว่าก็ได้ [26] แต่สูงสุดไม่ควรเกิน 5 แอทริบิวต์ ถ้ามากกว่านั้นจะทำให้การดูแลบำรุงรักษาแพง [27] รายการของดัชนีจะสั้นและเล็กกว่าเรคอร์ดข้อมูลจริงที่เก็บไว้มาก เป็นการลดการเข้าถึงอุปกรณ์ I/O ได้อย่างมาก เพราะถ้าไม่มีดัชนี ก็จะต้องสแกนตารางทั้งตาราง เพื่อหาแหล่งที่ตั้งของข้อมูลที่เราต้องการ ซึ่งเป็นวิธีที่ช้าและไม่มีประสิทธิภาพ [9] ดัชนีที่มีประสิทธิภาพ จะสามารถทำการประมวลผลการสอบถามข้อมูลที่ซับซ้อนได้อย่างรวดเร็วกว่าดัชนีที่มีประสิทธิภาพน้อยกว่า ซึ่งชนิดของดัชนีจะเป็นปัจจัยสำคัญกว่าจำนวนของดัชนีที่มีอยู่ในตาราง ต่อประสิทธิภาพของการประมวลผลการสอบถามข้อมูล [27]

2.4.2.1 ลักษณะดัชนีที่ดี

ดัชนีที่ดี คือ สามารถที่จะใช้ได้กับการสอบถามข้อมูลหลาย ๆ กรณี แทนที่จะเป็นแค่การสอบถามข้อมูลที่เกิดขึ้นเพียงกรณีเดียว [27] ทำให้ประหยัดพื้นที่และเวลาในการจัดเก็บดัชนี ใช้พื้นที่และเวลาน้อยในการค้นหาข้อมูล

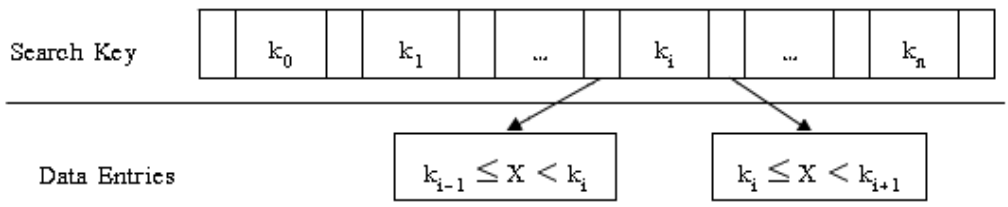
2.4.2.2 ประเภทของดัชนี

ดัชนีมีหลายแบบด้วยกัน เช่น ดัชนีแบบเชิงเส้น (Linear Indexing) ดัชนีแบบต้นไม้ (Tree Indexing) และดัชนีแบบบิตแมป ดัชนีแต่ละแบบก็เหมาะกับลักษณะการสอบถามข้อมูลที่แตกต่างกัน ในที่นี้จะขอกล่าวถึงเฉพาะดัชนีแบบต้นไม้ ส่วนดัชนีแบบบิตแมปจะกล่าวในบทต่อไป

ตัวอย่างดัชนีแบบต้นไม้ เช่น 2-3 Tree, B-Tree, B+ Tree และ B* Tree ซึ่งในที่นี้จะกล่าวถึงเฉพาะ B+ Tree เท่านั้น

2.4.2.2.1 ดัชนีแบบ B+ Tree

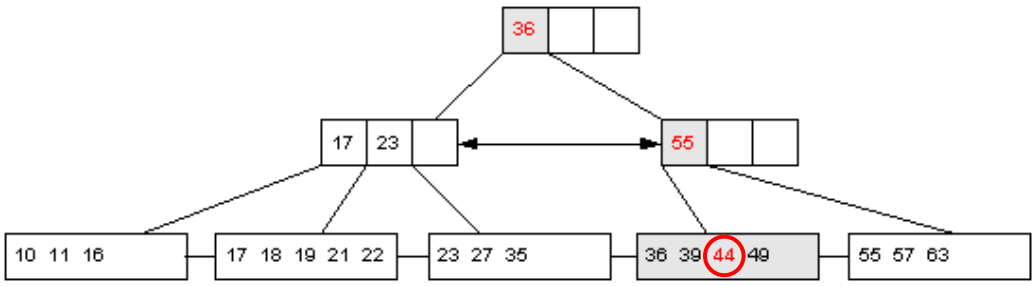
B+ Trees เป็นโครงสร้างแบบต้นไม้ที่มีหลายระดับประกอบด้วยรูทโหนดที่ชี้ไปยังระดับล่าง ระดับล่างสุดใน B+ Tree จะเป็นลีฟเพจหรือลีฟบล็อก [12] ซึ่งระดับนี้จะมีดัชนีของเรคอร์ดทุกเรคอร์ดในตารางที่ถูกอ้างอิงถึงอย่างหนาแน่นอย่างน้อยครึ่งหนึ่ง และจะมีการเชื่อมแต่ละโหนดเข้าด้วยกันด้วยลิงก์ลิสต์แบบคู่ (Doubly Linked List) [12] นั่นคือ B+ Tree ที่ลีฟโหนด (Leaf Nodes) จะจัดเก็บเรคอร์ดข้อมูลจริงไว้ และโหนดภายใน (Internal Node) จะจัดเก็บค่าคีย์เพื่อเป็นตัวนำทางในการค้นหา โดยจะมีตัวชี้ (Pointer) ชี้ไปยังโหนดลูก และระหว่างลีฟโหนดก็จะมีลิงก์ลิสต์แบบคู่เชื่อมอยู่ ดังนั้นโครงสร้างของโหนดภายในและลีฟโหนดจึงแตกต่างกัน [12] ดังภาพประกอบ 2-8



ภาพประกอบ 2-8 โครงสร้างดัชนีแบบต้นไม้ (Tree-Structured Indexing)

ในกรณีที่ระดับลีฟของดัชนีมีค่าซ้ำ ๆ กัน (Nonunique) ก็จะมีการจัดเก็บหมายเลขเรคอร์ดที่มีค่าซ้ำ ๆ กันเรียงตามลำดับหมายเลขเรคอร์ดจากน้อยไปมาก ซึ่งทำให้สามารถเพิ่มความเร็วในการดำเนินการลบและปรับปรุงข้อมูลให้ทันสมัย

ลักษณะที่สำคัญของดัชนีแบบ B+ Tree คือ ระยะทางระหว่างรูทและแต่ละลีฟโหนดจะเหมือนกัน การสแกนต้นไม้โดยไม่คำนึงถึงค่าคีย์เพื่อเข้าถึงลีฟเพจต่าง ๆ ในระดับเดียวกันจะใช้เวลาเท่ากัน ซึ่งการเข้าถึงต้นไม้แบบกว้างจะเร็วกว่าต้นไม้แบบลึก เพราะการท่องแต่ละระดับของต้นไม้จะต้องเข้าถึงเพจหรือบล็อกอื่น ๆ เพิ่มเติมด้วย ดังนั้นเวลาที่ใช้ในการค้นหาข้อมูล คือ เวลาที่ใช้ในการท่องต้นไม้จากรูทไปยังลีฟโหนด และการท่องไปยังหมายเลขเรคอร์ด [10] ตัวอย่างเช่น ต้องการค้นหาค่า 44 ดังภาพประกอบ 2-9



ภาพประกอบ 2-9 ตัวอย่างดัชนี B+ Tree ที่โหนดภายในมีลูกระหว่าง 2 ถึง 4 และลีฟโหนดจัดเก็บข้อมูล 3 ถึง 5 เรคอร์ด

ข้อดีของดัชนีแบบ B+ Tree

เหมาะสำหรับฐานข้อมูลขนาดใหญ่ (Large Scale Disk-Based Systems) ที่ต้องรองรับการแทรก (Insertion) การลบ (Deletion) และการสอบถามข้อมูลแบบช่วง เมื่อค้นหาเรคอร์ดแรกเจอ เรคอร์ดที่เหลือซึ่งอยู่ในช่วงที่จะค้นหาก็จะเจอด้วย เนื่องจากมีการเรียงตามลำดับเรคอร์ดอยู่ภายในโหนด [12] และเหมาะกับแอทริบิวต์ที่จะนำมาทำดัชนีที่มีค่าคาร์ดินอลิตี้สูงอีกด้วย

ข้อจำกัดของดัชนีแบบ B+ Tree

ดัชนีแบบ B+ Trees ไม่เหมาะสำหรับการสอบถามข้อมูลที่ซับซ้อนและแอทริบิวต์ที่มีคาร์ดินอลิตี้ต่ำ เพราะจะมีข้อจำกัดเรื่องการทำงานร่วมกัน (Cooperativity) ของเงื่อนไขต่าง ๆ ที่อยู่หลังอนุประโยค WHERE ซึ่งต้องสร้างเป็นคีย์ผสม (Compound Key) เช่น ถ้ามี n แอทริบิวต์ ก็อาจจะเลือก 1 แอทริบิวต์ หรือ 2 แอทริบิวต์ หรือ n แอทริบิวต์ใด ๆ มาผสมกัน เพื่อรองรับการสอบถามข้อมูลที่จะเกิดขึ้นได้ทุกกรณี ซึ่งอาจจะต้องสร้างดัชนีแบบ B+ Tree ที่เป็นคีย์ผสมทั้งหมด $C_1^n + C_2^n + \ldots + C_n^n = 2^n - 1$ กรณี ดังนั้นจะเห็นได้ว่าจะยุ่งยากและค่าใช้จ่ายจะสูงมาก [10]

ผู้จำหน่ายระบบจัดการฐานข้อมูลแบบเชิงความสัมพันธ์ที่มีชื่อเสียงที่ใช้ดัชนีแบบ B+ Tree เช่น ไอบีเอ็ม ออราเคิล ไซเบส และแทนเด็ม (Tandem) [28]

การสอบถามข้อมูลแบบซับซ้อนและแบบทันทีทันใดที่เกี่ยวข้องกับแอทริบิวต์ที่มีคาร์ดินอลิตี้ต่ำ มักจะเกิดขึ้นกับคลังข้อมูล ซึ่งดัชนีที่เหมาะสำหรับการสอบถามข้อมูลในลักษณะนี้คือ ดัชนีแบบบิตแมป ซึ่งจะอธิบายรายละเอียดในบทที่ 3