บทที่ 4

การทดลองและผลการทดลอง

จากแนวทางการวิเคราะห์ และผลของการคำนวณในบทที่ 3 เพื่อประเมินสมรรถนะของฟังก์ชันการแทรก (Insert) และฟังก์ชันการสอบถาม (Query) ของดัชนีวายทรี (Y tree) เมื่อใช้งานบนแฟลชไดร์ฟ (Flash Drive) ในบทนี้ได้นำเสนอแนวทางการพัฒนาดัชนีวายทรี ซึ่งเป็นต้นแบบของดัชนีวายทรี สำหรับนำไปใช้ทำการทดลองเพื่อวัดสมรรถนะ ของฟังก์ชันเมื่อใช้งานกับกับอุปกรณ์แฟลชไดร์ฟ โดยใช้ข้อกำหนดเดียวกันกับการคำนวณทุกประการ

4.1 การเตรียมข้อมูลสำหรับการพัฒนาดัชนีวายทรี

ในการทดลองเพื่อวัดสมรรถนะของฟังก์ชันการแทรก และฟังก์ชันการสอบถามนี้ จำเป็นจะต้องกำหนดค่าของ คุณสมบัติดัชนีวายทรี และแฟลชไดร์ฟให้ตรง หรือมีความใกล้เคียงกับสมการที่ได้เตรียมไว้ให้มากที่สุด

4.1.1 ข้อมูลที่ใช้ในการทดลองคือ ข้อมูลที่ได้จากการสุ่ม (Random) จำนวนทั้งหมด 1,000,000 แถว (Row) โดยแต่ละแถวประกอบด้วยตัวเลข 2 ชุด ดังนี้ ชุดที่ 1 เป็นตัวเลขไม่เกิน 2 หลัก ที่อยู่ระหว่าง 1 ถึง 99 และชุดที่ 2 เป็นตัวเลข 3 หลักที่อยู่ระหว่าง 100 ถึง 999 โดยระหว่างตัวเลขทั้ง 2 ชุด ของแต่ละแถวจะต้องเว้นวรรคจำนวน 1 วรรค ซึ่งแสดงตัวอย่างดังในรูปที่ 4.1

```
1.  1 100
2.  2 201
3.  5 310
4.  10 350
5.  55 888
```

รูปที่ 4.1 ตัวอย่างข้อมูลที่ใช้ในการทดลอง

4.1.2 แฟลชไดร์ฟ เลือกใช้แฟลชไดรฟ์แบบแนนด์ (NAND Flash Drive) โดยมีรายละเอียดดังนี้

1) General

- Brand Samsung
- Model MMDOE56G5MXP-0VB00
- Highlights Samsung MMDOE56G5MXP-0VB00 Solid State Disk Drive (SSD), 256GB Capacity, SATA II 3.0Gb/s Interface, 2.5" Form Factor.

2) Memory

- Capacity 256GB
- Bytes per Sector 512 Bytes
- Interface Serial ATA 3.0Gb/s (SATA)
- NAND Flash Components Multi-Level Cell (MLC) NAND Flash Memory

3) Performance

- Host Transfer Rate 300 MB/s
- Sequential Read Sector 220 MB/sec
- Sequential Write Sector 185 MB/sec

4) Power & Reliability

- Allowable voltage 5V + 5%
- Allowable noise/ripple 100mV p-p or less
- Active Power 0.26W
- Idle/ Standby/ Sleep 0.15W
- MTBF 1,000,000 Hours

5) Environment

- Temperature Operating : 0ºC to 70ºC
- Non-Operating : -55ºC to 95ºC

- Shock 1500G, duration 0.5ms, Half Sine Wave
- Vibration 20G Peak, 10~2000Hz,(15mins/Axis)x3 Axis
- Humidity 5% to 95%, non-condensing

6) Form Factor

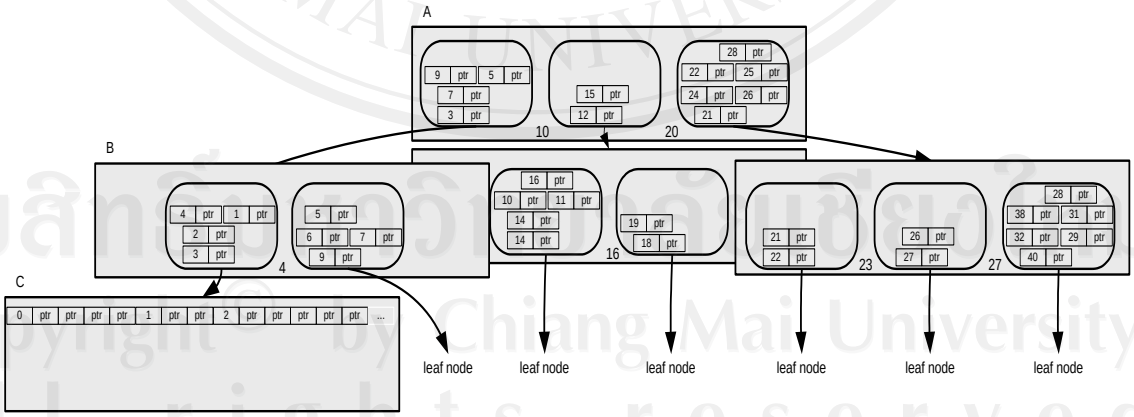
- Form Factor 2.5"

4.2 ดัชนีวายทรี

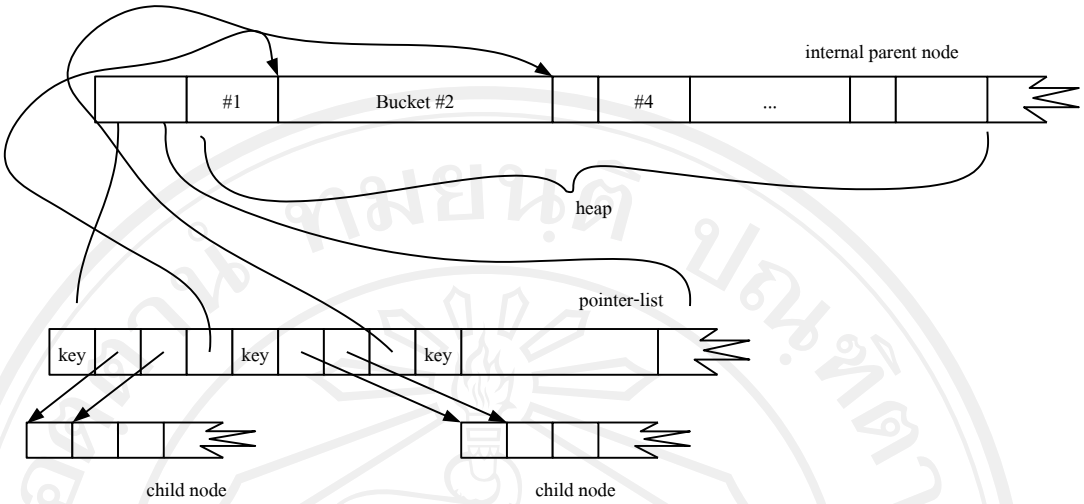
จากที่เคยกล่าวถึงดัชนีวายทรีแล้วในส่วนก่อนหน้า ทั้งโครงสร้างของดัชนีวายทรี โหนดใบ (Leaf Node) โหนดภายใน (Internal Node) และพอยเตอร์ (Pointer) แล้วนั้น ในบทนี้จะนำมากล่าวอีกครั้งแต่อยู่ในรูปแบบของโครงสร้างที่พร้อมนำไปพัฒนาโปรแกรมด้วยภาษาจาวา (JAVA) ซึ่งมีโครงสร้างของดัชนีวายที และองค์ประกอบที่เกี่ยวข้องดังนี้

4.2.1 โครงสร้างดัชนีวายทรี

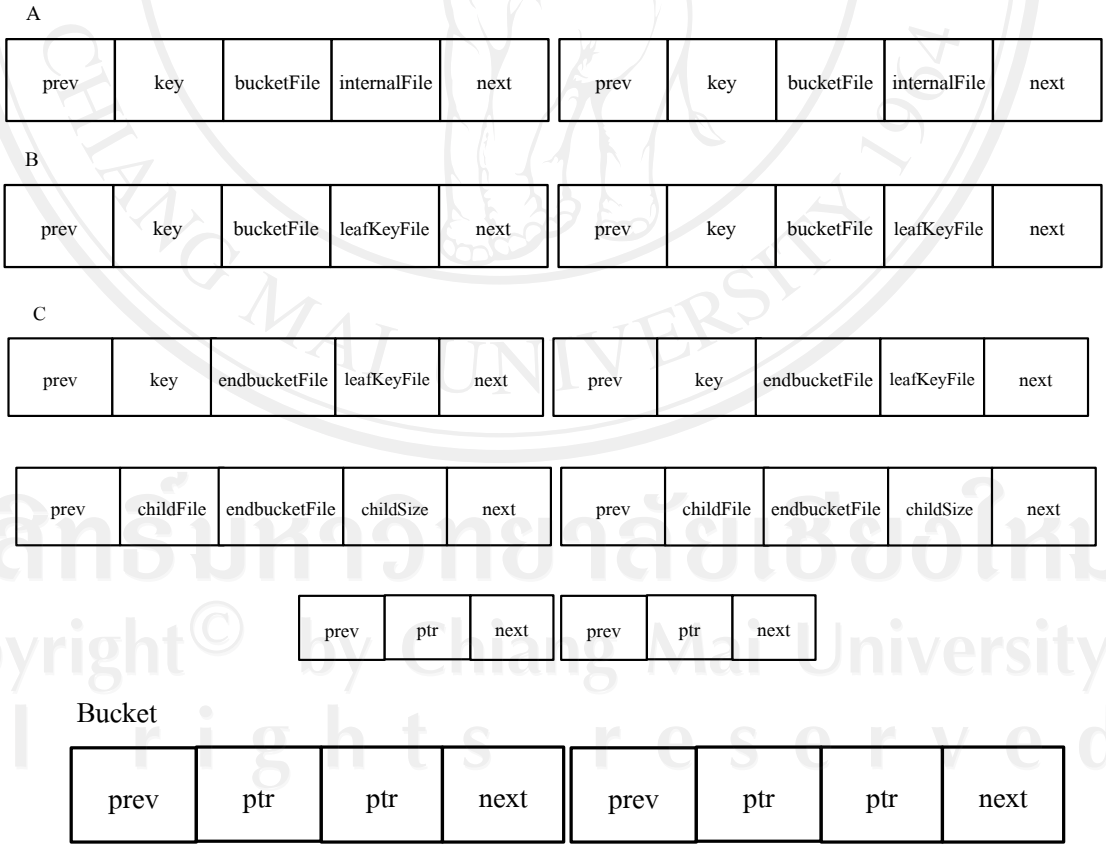
จากเอกสารการนำเสนอดัชนีวายทรีซึ่งแสดงดังรูปที่ 4.2 โดยในส่วนก่อนหน้านี้ได้อธิบายแล้วถึงโครงสร้าง โหนดใบ โหนดภายใน ความสูงของดัชนีวายทรี และพอยเตอร์ต่างๆ ประกอบกับในเอกสารยังได้แนะนำการปรับปรุงโครงสร้างของดัชนีวายทรี เพื่อรองรับการเก็บข้อมูลปริมาณมาก ดังแสดงในรูปที่ 4.3 และเพื่อให้การพัฒนาโปรแกรมได้ง่ายขึ้นจึงได้นำรูปที่ 4.2 และ 4.3 มารวมกัน เป็นดังรูปที่ 4.4



รูปที่ 4.2 โครงสร้างของดัชนีวายทรีตามเอกสาร



รูปที่ 4.3 การปรับโครงสร้างของโหนดภายในสำหรับโหนดขนาดใหญ่



รูปที่ 4.4 โครงสร้างดัชนีวายทรีสำหรับการพัฒนาโปรแกรม

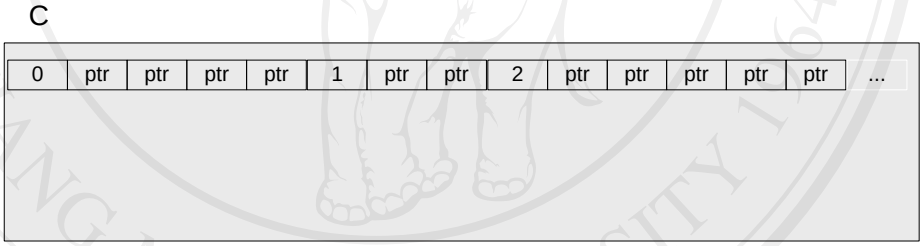
4.2.2 ลักษณะของโหนดใบ และ โหนดภายใน

จากเอกสารของดัชนีวายทรีกล่าวไว้ว่าโหนดของดัชนีวายทรีประกอบด้วยโหนด 2 ประเภท และในรูปที่ 4.4 ก็เช่นกัน กล่าวคือ

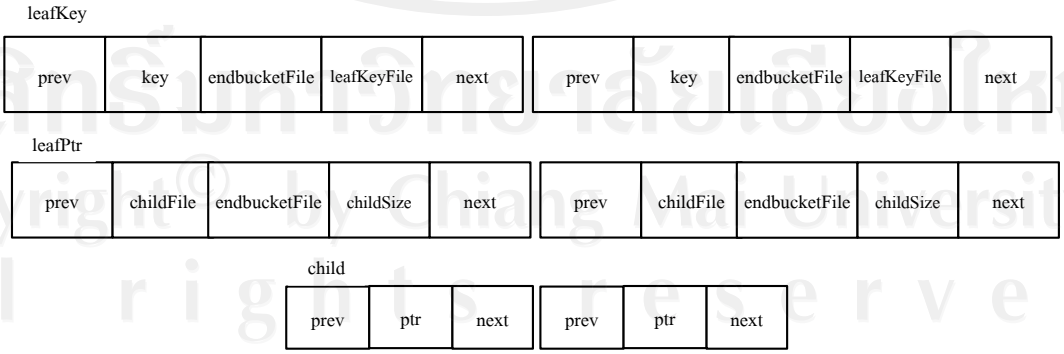
1) โหนดใบ จากโครงสร้างเดิมในรูปที่ 4.2 ลักษณะของโหนดใบเป็นดังรูปที่ 4.5 และเมื่อนำมาเขียนใหม่ให้ง่ายต่อการพัฒนาโปรแกรมดังรูปที่ 4.6 และได้กระจายออกเป็น 3 โหนดดังนี้

1.1 leafKey เป็นโหนดที่ใช้สำหรับเก็บค่าคีย์ของโหนดภายใน ซึ่งประกอบด้วย

- prev ใช้สำหรับเก็บตำแหน่งของโหนดก่อนหน้า
- key ใช้สำหรับเก็บค่าคีย์ของดัชนีวายทรี
- endBucket ใช้สำหรับเก็บตำแหน่งบัคเก็ตของโหนดภายใน
- leafPtrFill ใช้สำหรับเก็บตำแหน่งของ ptr-list
- next ใช้สำหรับเก็บตำแหน่งของโหนดถัดไป



รูปที่ 4.5 โหนดใบของดัชนีวายทรี



รูปที่ 4.6 โหลดใบเพื่อการพัฒนาโปรแกรม

1.2 leafPtr เป็นโหนดที่ใช้สำหรับเก็บ ptr-list ซึ่งประกอบด้วย

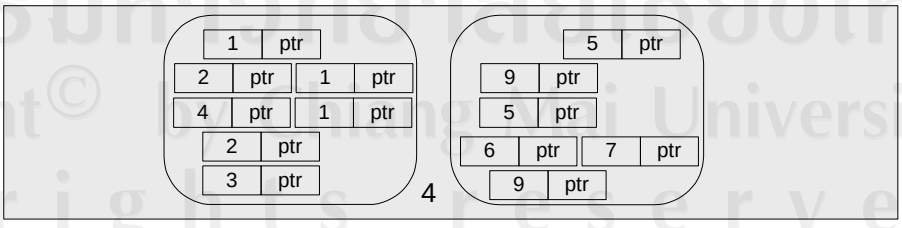
- prev ใช้สำหรับเก็บตำแหน่งของโหนดก่อนหน้า
- childFile ใช้สำหรับเก็บตำแหน่งของโหนดลูก (Child Node)
- childSize ใช้สำหรับเก็บจำนวนของโหนดลูกในแต่ละค่าคีย์ของโหนดใบนั้นๆ
- next ใช้สำหรับเก็บตำแหน่งของโหนดถัดไป

1.3 child เป็นโหนดที่ใช้สำหรับเก็บตำแหน่งของข้อมูลจริงที่อยู่ในแฟลชไดร์ฟ ซึ่งประกอบด้วย

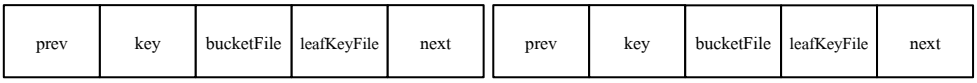
- prev ใช้สำหรับเก็บตำแหน่งของโหนดก่อนหน้า
- pt ใช้สำหรับเก็บตำแหน่งข้อมูลจริงในแฟลชไดร์ฟ
- next ใช้สำหรับเก็บตำแหน่งของโหนดถัดไป

2) โหนดภายใน จากโครงสร้างเดิมในรูปที่ 4.2 ลักษณะของโหนดภายในเป็นดังรูปที่ 4.7 และเมื่อนำมาเขียนใหม่ให้ง่ายต่อการพัฒนาโปรแกรมแสดงดังรูปที่ 4.8 ซึ่งโหนดภายในนี้ประกอบด้วย

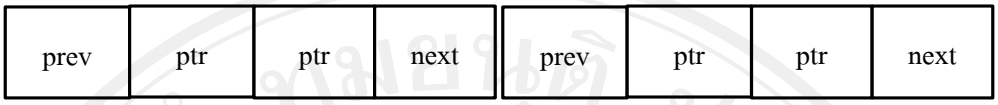
- prev ใช้สำหรับเก็บตำแหน่งของโหนดก่อนหน้า
- key ใช้สำหรับเก็บค่าคีย์ของโหนดภายใน
- bucketFile ใช้สำหรับเก็บตำแหน่งบัคเก็ตของโหนดภายใน
- leafKeyFile ใช้สำหรับเก็บตำแหน่งของโหนดใบ
- next ใช้สำหรับเก็บตำแหน่งของโหนดถัดไป



รูปที่ 4.7 โหนดภายในของดัชนีวายทรี



รูปที่ 4.8 โหนดภายในเพื่อการพัฒนาโปรแกรม



รูปที่ 4.9 บัคเก็ตเพื่อการพัฒนาโปรแกรม

และนอกจากนี้ยังปรากฏมีโหนดอีกประเภทที่มีการใช้งานร่วมกันระหว่างโหนดใบ และโหนดภายในนั้นคือบัคเก็ต โดยมีลักษณะดังรูปที่ 4.9 ซึ่งบัคเก็ตนี้ประกอบด้วย

- prev ใช้สำหรับเก็บตำแหน่งของโหนดก่อนหน้า
- key ใช้สำหรับเก็บค่าคีย์ของบัคเก็ต
- ptr ใช้สำหรับเก็บค่าของตำแหน่งข้อมูลจริงในแฟลชไดร์ฟ
- next ใช้สำหรับเก็บตำแหน่งของโหนดถัดไป

4.2.3 โครงสร้างข้อมูลของดัชนีวายทรี

จากลักษณะของโหนดต่างๆที่กล่าวมาแล้วข้างต้น เพื่อให้ง่ายต่อการพัฒนาประกอบกับเพื่อเพิ่มความสะดวกในการพัฒนาโปรแกรม และตัดความยุ่งยากของข้อผิดพลาดที่อาจเกิดขึ้น พร้อมกับแนวทางการนำเสนอนี้ ได้เลือกใช้การพัฒนาโปรแกรมด้วยภาษาจาวา (JAVA) ดังนั้น อาร์เรย์ลิสต์ (Array-List) จึงเป็นตัวเลือกที่เหมาะสมที่ใช้เป็นตัวเชื่อมโยงของโหนดต่างๆ ภายในดัชนีวายทรี โดยนำโหนดต่างๆนี้บรรจุภายในอาร์เรย์ลิสต์

เมื่อโครงสร้างข้อมูลทุกอย่างเก็บลงอาร์เรย์ลิสต์แล้วเพื่อให้การประเมินสมรรถนะตรงตามสมการที่ได้วิเคราะห์ไว้มากที่สุด ลักษณะของการเก็บข้อมูลลงในแฟลชไดร์ฟ จะเก็บเป็นไฟล์ข้อมูลโดยให้ หนึ่งอาร์เรย์ลิสต์คือหนึ่งไฟล์ข้อมูล และในแต่ละไฟล์ข้อมูลเชื่อมโยงกันด้วยพอยเตอร์ที่อยู่ในดัชนีวายทรี ซึ่งจะกล่าวต่อไปในส่วนถัดไป ส่วนของการตั้งชื่อไฟล์นั้นใช้เป็นตัวเลขของเวลาขณะนั้นนำมาตั้งเป็นชื่อไฟล์ เพื่อป้องกันการซ้ำของชื่อไฟล์ ส่วนในกรณีที่ต้องการปรับปรุง (Update) โหนดต่างไม่ว่าจะเป็นลบ หรือเพิ่มข้อมูลก็ตามให้ใช้วิธีการเขียนทับที่ไฟล์เดิม ซึ่งจะไม่ทำให้ข้อมูลเดิมนั้นหายแต่ประการใด เพราะว่าได้กระทำการอ่านข้อมูลของไฟล์นั้นๆ เก็บในดัชนีวายทรีแล้ว

4.2.4 พอยเตอร์ของดัชนีวายทรี

จากรูปโครงสร้างดัชนีวายทรีดังรูปที่ 4.2 จะเห็นได้ว่ามีเส้นเชื่อมโยงระหว่างโหนดต่างๆ ไม่ว่าจะเป็นโหนดภายในไปยังโหนดภายใน หรือโหนดภายในไปยังโหนดใบ ซึ่งเส้นเชื่อมโยงนี้เรียกว่าพอยเตอร์ แต่ในที่นี้มิได้หมายถึงพอยเตอร์ที่อยู่ในภาษาจาวา แต่จะหมายถึงชื่อไฟล์ข้อมูลของอาร์เรย์ลิสต์ที่ถูกบันทึกไว้ในหัวข้อก่อนหน้านี้ ซึ่งการกระทำแบบนี้จะช่วยให้เวลาของการวัดสมรรถนะเป็นเวลาของการเข้าถึงข้อมูลจริงๆ มิใช่เวลาที่ถูกอ่าน หรือเขียนจากบัฟเฟอร์ (Buffer) ของแฟลชไดร์ฟ เพราะว่าแฟลชไดร์บางรุ่นในปัจจุบันมีความจุสูงมาก เช่นแฟลชไดร์ฟที่เลือกมาใช้ในครั้งนี้มีความจุของบัฟเฟอร์ถึง 32 MB และเมื่อนำโครงสร้างนี้ไปพัฒนาโปรแกรมต่อยังปรากฏมีพอยเตอร์เพิ่มเติมอีกหลายแบบดังแสดงในรูปที่ 4.4 ซึ่งมีรายการดังต่อไปนี้

- ระหว่างโหนดภายในไปยังโหนดภายใน
- ระหว่างโหนดภายในไปยังบัคเก็ต
- ระหว่างโหนดภายในไปยังโหนดใบ
- ระหว่างโหนดใบไปยังบัคเก็ต
- ระหว่างโหนดใบไปยังโหนดรายการพีทีอาร์ (ptr-list)
- ระหว่างโหนดรายการพีทีอาร์ไปยังโหนดลูก

4.3 การทำงานของดัชนีวายทรีบนแฟลชไดร์ฟ

กระบวนการอ่าน และเขียนของแฟลชไดร์ฟถือได้ว่ามีความสำคัญมากสำหรับงานวิจัยนี้ เพราะว่าการอ่าน และการเขียนแฟลชไดร์ฟแต่ละครั้งต้องใช้เวลา ยิ่งถ้าต้องเขียนข้อมูลมากจะใช้เวลามากเช่นกัน แต่ดัชนีวายทรีสามารถแก้ปัญหาเรื่องเวลาการเขียนข้อมูลที่ล่าช้า ได้ในระดับหนึ่งด้วยวิธีการปรับแก้ส่วนของการเขียนข้อมูลลงแฟลชไดร์ฟให้มีจำนวนครั้งน้อยที่สุด และเมื่อจำนวนครั้งของการเขียนข้อมูลน้อยลงแล้วเวลาที่ใช้ก็จะน้อยลงด้วยเช่นกัน ส่วนเวลาของการอ่านข้อมูลจากแฟลชไดร์ฟนั้นทั้งเรื่องเวลาของการอ่านข้อมูลจากแฟลชไดร์ฟ และกระบวนการอ่านข้อมูลของดัชนีวายทรีนั้นมีความรวดเร็วอยู่แล้วจึงไม่ต้องปรับแก้อะไร โดยมีขั้นตอนวิธี (Algorithm) การแทรกข้อมูล และการสอบถามข้อมูลมีดังนี้

4.3.1 ขั้นตอนวิธีการแทรกข้อมูลในดัชนีวายทรี และในแฟลชไดร์ฟ

ขั้นตอนวิธีการแทรกข้อมูลนี้ขั้นตอนวิธีการเดียวกับที่ได้นำเสนอไว้ในเอกสารดัชนีวายทรีมิได้เปลี่ยนแปลงอะไร เพราะขั้นตอนวิธีที่ดีอยู่แล้ว ดังแสดงในรูปที่ 4.10

Algorithm Insert (parameters *S*: set of (*key, ptr*) pairs of cardinality no greater than *d*, *N: Node* having fanout f_N)

1. **If** *N* is an internal node:
2. For each element *s* of *S*, add *s* into the first heap bucket b_i such that the associated key value K_i *s.key*; or, inset into the last heap bucket if there is no such K_i.
3. Choose the bucket b_j that has the most (*key, ptr*) pairs.
4. **If** the heap contains more than $(f_N - 1) * d$ pairs,
5. Remove *min* (*d, size*(b_j)) (*key, ptr*) pairs from b_j to create S_{new}, write *N* to disk, And recursively call *Insert* (S_{new}, node pointed to by P_j).
6. **Else** write *N* to disk.
7. **Else** *N* is a leaf node:
8. Simply add *S* to the set of (*key, ptr*) pairs in *N*, then write *N* to disk.

รูปที่ 4.10 ขั้นตอนวิธีการแทรกของดัชนีวายทรี

4.3.2 ขั้นตอนวิธีการสอบถามข้อมูลในดัชนีวายทรี และในแฟลชไดร์ฟ

ขั้นตอนวิธีการสอบถามจะใช้วิธีการเดียวกับการสอบถามของดัชนีบีพลัสทรี (B^+ tree) ด้วยวิธีการทราเวิร์สอินออเดอร์ (Inorder Traverse) แบบปกติทั่วไป ซึ่งเป็นวิธีการสอบถามข้อมูลที่มีความรวดเร็วในการสอบถามอยู่แล้ว และได้รับคำแนะนำจากเอกสารดัชนีวายทรีด้วย