บทที่ 6

การจัดแฟ้มข้อมูลแบบสุ่ม

ภาควิชาวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ มหาวิทยาลัยสงขลานครินทร์

ความหมาย

- **Relative file Organization** (การจัดแฟ้มข้อมูลแบบสัมพัทธ์) เป็นการจัดแฟ้มข้อมูลเพื่อการเข้าถึงข้อมูลแบบสุ่ม ทำให้สามารถเข้าถึงระเบียนที่ต้องการได้รวดเร็วโดยตรง เพราะค่าของคีย์มีความ สัมพัทธ์กับที่อยู่ของตำแหน่งของข้อมูลบนสื่อบันทึกข้อมูล
- **Random file Organization** (การจัดระเบียนแฟ้มข้อมูลแบบสุ่ม) เป็นวิธีการจัดระเบียนข้อมูล ที่สามารถเข้าถึงข้อมูลระเบียนหนึ่งระเบียนใดได้โดยตรง ไม่ต้องอ่านข้อมูลตั้งแต่ต้น อย่างเช่นแฟ้มลำดับ

ลักษณะข้อมูลที่เก็บในแฟ้ม

- ระเบียนต่างๆ ในแฟ้มไม่จำเป็นต้องเก็บเรียงตามลำดับของ Primary key ในแฟ้ม อาจมีที่ว่างปะปนอยู่เป็นระยะๆ

Key Value	Data	Physical Location
cat		1
dog		2
		3
bat		
cow		n-2
rat		n-1
hen		n

การสร้าง/การเรียกใช้ข้อมูลในแฟ้ม

- การสร้างและเรียกใช้ข้อมูลในแฟ้มข้อมูลชนิดนี้ เราสามารถหาตำแหน่งที่เก็บค่าระเบียนจากค่าของ primary key
- ตำแหน่งที่หาได้อยู่ในสภาพสุ่ม ไม่ได้ต่อเนื่องกันตลอดอย่างแฟ้มลำดับ (Sequential file)
- การหาตำแหน่งของระเบียน ทำได้โดยสร้างความสัมพันธ์ระหว่างค่าของ primary key กับตำแหน่งทางกายภาพของแฟ้มข้อมูล เขียนเป็นฟังก์ชันได้เป็น

Record (key value) ----------> Address

การสร้าง/การเรียกใช้ข้อมูลในแฟ้ม (ต่อ)

- ทุกครั้งที่ต้องการจะบันทึกระเบียนไว้ในแฟ้ม จะต้องนำค่า Primary Key มาแปลงให้เป็นตำแหน่งโดยผ่านฟังก์ชัน R แล้วนำข้อมูลไปเก็บในตำแหน่งที่หาได้
- ทุกครั้งที่ต้องการเรียกใช้หรืออ่านระเบียนหนึ่งระเบียนใดสามารถทำได้โดยผ่านฟังก์ชัน R เช่นเดียวกัน ค่า Primary Key จะกลายเป็นตำแหน่งที่เก็บข้อมูล ซึ่งจะพบข้อมูลระเบียนที่ต้องการในตำแหน่งนั้น

Interactive Processing

- การเข้าถึงข้อมูลระเบียนต่างๆในแฟ้ม ไม่จำเป็นต้องเข้าถึงแบบเรียงลำดับ สามารถเข้าถึงได้โดยตรง ฉะนั้นแฟ้มสุ่มจะต้องเก็บอยู่ในสื่ออุปกรณ์ที่เป็นแบบ Direct Access Storage Device (DASD) เท่านั้น
- สามารถประมวลผลได้ทั้งแบบ On-line และ Off-line
- แฟ้มสุ่มเหมาะสำหรับงานแบบ Interactive Processing เนื่องจากเข้าถึงระเบียนในแฟ้มสุ่ม สามารถเข้าถึงได้โดยตรงและรวดเร็ว
- จะเข้าถึงแฟ้มสุ่มตามลำดับทางกายภาพ ผลเป็นอย่างไร

ตัวอย่าง

- แฟ้มข้อมูลนักศึกษา ใช้รหัสนักศึกษาเป็น Primary Key
- แฟ้มข้อมูลบุคลากร ใช้รหัสบุคลากรเป็น Primary Key
- สามารถเพิ่ม/ลบ ข้อมูล ได้อย่างรวดเร็ว โดยใช้รหัสนักศึกษา หรือรหัสบุคลากรคำนวณหาตำแหน่งที่เก็บ
- สามารถปรับปรุงค่าข้อมูลแล้วบันทึกข้อมูลกลับลงในแฟ้มเดิมได้

ข้อดี

- สามารถเข้าถึงข้อมูลระเบียนต่างๆ ได้โดยตรง ทำให้มีความรวดเร็วในการปฏิบัติการ เมื่อเทียบกับการจัดเก็บแฟ้มแบบลำดับ
- การปรับปรุงค่าข้อมูลในระเบียน สามารถบันทึกกลับที่เดิมได้ โดยไม่กระทบกับระเบียนอื่นๆในแฟ้มข้อมูล

Transform Primary Key to Address

3 เทคนิคกับการแปลงส่ง primary key ไปเป็นตำแหน่งที่เก็บข้อมูลในแฟ้มสุ่ม

Record (key value) ----------> Address

- การแปลงส่งโดยตรง: Direct Mapping
- การค้นหาในพจนานุกรม : Dictionary Lookup
- การคำนวณตำแหน่ง : Address Calculation

Direct Mapping

Key Value -------> Relative Address

- ตำแหน่งของระเบียนมีค่าเท่ากับ primary key ไม่ต้องคำนวณ
- วิธีนี้โดยทั่วไปเรียกว่า Relation Addressing เพราะว่าตำแหน่งที่ได้นั้นเป็น Relative Address ของแฟ้มข้อมูล
- ถ้าแฟ้มข้อมูลเตรียมเนื้อที่สำหรับ N ระเบียน
- Relative Address จะเป็น 1,2,3,4,....., N-1, N

Relative Addressing

- 1 ระเบียน ต่อ 1 ตำแหน่ง primary key จะต้องเป็นตัวเลขเรียงจากน้อยไปหามาก และขนาดระเบียนต้องมีขนาดคงที่
- ตัวอย่าง ระเบียนในแฟ้มสุ่มชุดหนึ่งมี 1000 ระเบียน ค่า PK 000 - 999
- Dense Key , Nondense Key

Relative Address	Primary Key	Data	Physical Location
0	000		1
1	001		2
2	002		
3	003		3
	:		4
n-2	998		n-1
n-1	999		n

เมื่อไรจึงจะใช้ Relative Addressing

- ค่าของ Primary Key ต้องเป็นตัวเลขที่ต่อเนื่องกันตลอด
- ค่าของ Key อาจขาดช่วงได้ แต่ไม่ควรเกิน 30% ของช่วง Key สมควรใช้ Relative Addressing
- ข้อมูลที่บันทึกอาจจะไม่มีทุกตำแหน่ง แต่ต้องเตรียมที่สำหรับเก็บข้อมูลของทุกตำแหน่ง หรือทุก Primary Key
- การใช้ Relative Addressing จะมีประสิทธิภาพมาก ถ้าจำนวนค่าที่เป็นไปได้ของ Key ใกล้เคียงกับจำนวน Primary Key จริง

ข้อดี-ข้อเสีย สำหรับวิธี Relative Addressing

- **ข้อดี**
  - การแปลงส่งฟังก์ชัน R ทำได้ง่าย เพราะค่าคีย์ของระเบียนก็คือ ตำแหน่งสัมพัทธ์ของระเบียนในแฟ้มข้อมูล
  - สามารถประมวลผลได้ดีทั้งแบบลำดับและแบบสุ่ม
- **ข้อเสีย**
  - ไม่สามารถใช้กับ primary key ที่เป็น ตัวอักษรได้
  - ไม่เหมาะกับแฟ้มที่มีช่วงของ primary key กว้าง และมีจำนวนระเบียนที่เก็บจริงๆ น้อย

Dictionary Lookup

- ในการแปลงส่งฟังก์ชั่น R(Key Value) ให้เป็น Address ก็คือวิธีการค้นหาในพจนานุกรม
- นิยมใช้เมื่อ Primary Key เป็นตัวอักขระ ขนาดใหญ่ หรือช่วงกว้างมาก
- Dictionary Lookup

  = Indexed file

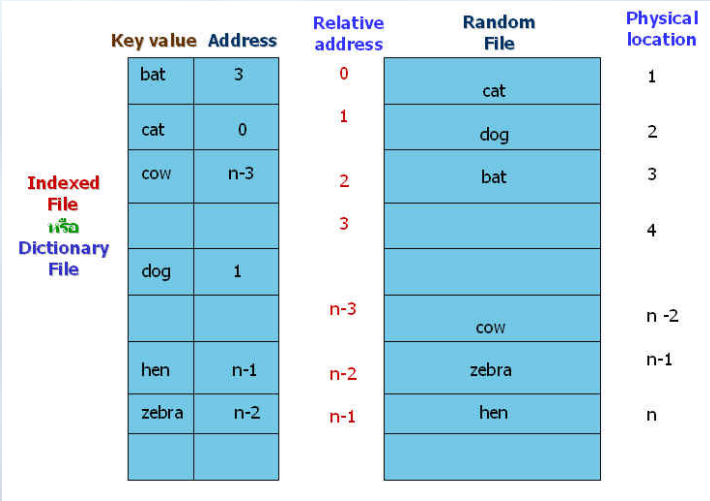
  = Indexed Nonsequential file

Dictionary Lookup

<u>แนวคิด</u>

- สร้างตารางหรือแฟ้มพจนานุกรม เก็บ Primary Key และ ตำแหน่ง (Relative Address) ทุกๆ ระเบียนในแฟ้มสุ่มจะแปลงได้ตำแหน่งที่ไม่ซ้ำกัน

Indexed File และ Random File



การเข้าถึงข้อมูล

- การเรียกใช้ระเบียนหนึ่งๆ จะต้องใช้ค่า Key ของระเบียนนั้นๆ ไปหา Address ที่คู่กับ Key นั้นในแฟ้มพจนานุกรม และใช้ Address นั้นในการเข้าถึงระเบียนที่ต้องการ
- ตัวอย่าง
  - ต้องการข้อมูลของ hen ก็นำ hen ไปหาใน Dictionary File ซึ่งจะได้ค่า Relative Address เป็น **n-1** จากนั้น ไปอ่านข้อมูลของ hen ใน Random File จากตำแหน่ง **n-1**

โครงสร้างของแฟ้มพจนานุกรม (Dictionary Structure)

- แฟ้มพจนานุกรม หรือแฟ้มดัชนี ประกอบ 2 ส่วน คือ
  - Key Value (Primary Key)
  - Address (Relative Address)
- โครงสร้างของแฟ้มเก็บได้ทั้ง 2 แบบ
  - เป็น Linear อาจสร้างเป็น ตาราง (Table/array) หรือ linked list ก็ได้ ถ้าเป็น linked list ไม่สามารถค้นแบบ Binary Search Tree ได้- เป็น nonlinear

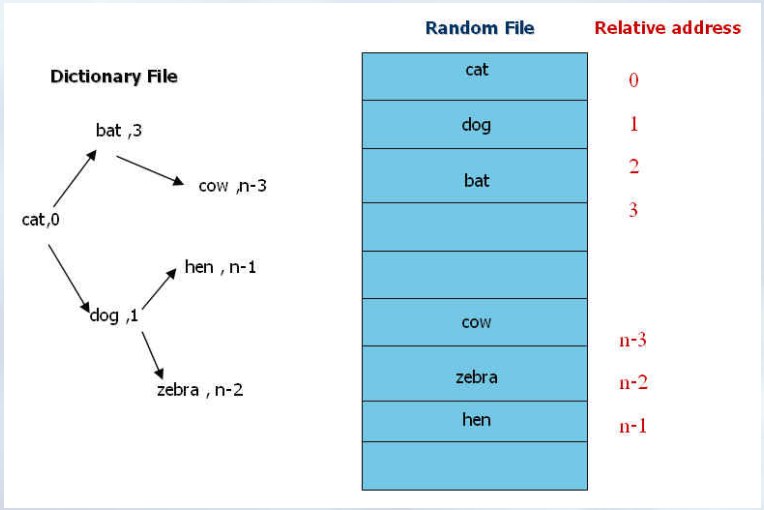
เก็บแบบ Linear หรือ Table

- ข้อมูลเรียงตามลำดับของ Key
  - ค้นหาข้อมูลได้เร็ว สามารถค้นแบบ Binary Search
  - การแทรกหรือลบข้อมูล เสียเวลามาก
- ข้อมูลไม่เรียงตามลำดับของ Key
  - ต้องค้นแบบเรียงลำดับ ทำให้หาได้ช้า
  - การแทรกหรือลบข้อมูล เสียเวลาน้อย และไม่ยุ่งยากเหมือนแบบแรก

เก็บแบบ Nonlinear

- Binary Search Tree
- M-Way Search Tree
- B-Tree
- โครงสร้างข้อมูลแบบนี้ค้นหาข้อมูลได้เร็ว แต่ซับซ้อนกว่า Linear
- การแทรกหรือลบข้อมูล ทำได้ง่ายกว่าแบบ Linear แต่มีความซับซ้อนมากกว่า

เก็บแบบ Binary Search Tree



ข้อมูลควรเก็บในหน่วยความจำทั้งหมด

- ในการใช้งาน Dictionary File หรือ Indexed File ควรจะต้องให้อยู่ในหน่วยความจำทั้งหมด
- ในกรณีที่ไม่สามารถเก็บได้ทั้งหมดสามารถแบ่งข้อมูล ออกเป็นส่วนๆ ก็ได้ แต่จะมีความซับซ้อนมากขึ้น

โครงสร้างของแฟ้มสุ่ม

- กำหนดเลขที่ตำแหน่งให้กับทุกๆ key ที่เป็นไปได้
- สร้างพจนานุกรมให้สมบูรณ์ในช่วงต้น
- **ข้อดี** เมื่อมีระเบียนใหม่สามารถพิจารณาที่เก็บได้ง่าย
- **ข้อเสีย** ถ้าแฟ้มเต็มตั้งแต่ตอนแรกแล้ว จะไม่สามารถเก็บข้อมูลใหม่ได้อีกต่อไป
- ควรสร้างให้มีที่ว่างพอที่เพิ่มระเบียนใหม่ต่อไปได้ และควรเก็บในตำแหน่งว่างจากบนก่อน

ข้อดี-ข้อเสีย ของวิธีการค้นหาแบบพจนานุกรม

ข้อดี

- ตำแหน่งของระเบียนสามารถหาได้โดยไม่ต้องผ่านการคำนวณ เพราะระบุไว้ให้ชัดเจนแล้วในแฟ้มพจนานุกรม
- primary key ที่ใช้สามารถเป็นไปตามความต้องการของผู้ใช้
- เมื่อทำ Reorganization เลขที่ตำแหน่งที่คู่กับ key ในแฟ้มพจนานุกรมเปลี่ยนค่าไป แต่ key ไม่ต้องเปลี่ยนตาม

ข้อเสีย

- เปลืองเนื้อที่สำหรับทำ Dictionary File หรือ Indexed File