บทที่ ๓

การเปรียบเทียบระบบจัดการฐานข้อมูลไอเอ็มเอสโททอลและดีบีทีจีทางทฤษฎี

ระบบจัดการฐานข้อมูลได้เข้ามามีบทบาทในวงการคอมพิวเตอร์มานานกว่า ๑๐ ปี โดยมีบริษัทต่าง ๆ ทั้งที่เป็นบริษัทผู้ผลิตคอมพิวเตอร์และบริษัทอิสระได้ผลิตโปรแกรมสำเร็จรูประบบจัดการฐานข้อมูลออกมาเป็นจำนวนมาก โดยมีจุดมุ่งหมายหลักที่เหมือนกันคือโปรแกรมดังกล่าวเป็นตัวกลางระหว่างระบบฐานข้อมูลและผู้ใช้ข้อมูลในระบบฐานข้อมูลนั้น กล่าวคือโปรแกรมนี้จะเป็นผู้จัดการข้อมูลในระบบฐานข้อมูลตั้งแต่การสร้าง การดูแลให้เป็นปัจจุบัน และการดึงข้อมูลไปใช้ ตัวอย่างโปรแกรมระบบจัดการฐานข้อมูลที่เป็นที่รู้จักกันดี เช่น อะดาบัส(ADABAS) ดาต้าคอมดีบี(DATACOM/DB) ดีแอลวัน(DL/1) ดีเอ็มเอสทู(DMS-II) ดีเอ็มเอสหนึ่งพัน(DMS-1000) ไอเอ็มเอสและโททอล เป็นต้น

ในการศึกษาเปรียบเทียบระบบจัดการฐานข้อมูลต่าง ๆ เพื่อทราบข้อดีข้อเสีย และเป็นแนวทางในการตัดสินใจที่จะเลือกใช้ระบบจัดการฐานข้อมูลที่เหมาะสมกับงาน ในวิทยานิพนธ์ฉบับนี้ไม่สามารถเปรียบเทียบทุกระบบที่มีอยู่เนื่องจากขาดเอกสารประกอบการศึกษาและเวลาอันจำกัด ดังนั้นจึงจะกล่าวเฉพาะระบบจัดการฐานข้อมูลบางระบบที่มีใช้ในประเทศไทยคือ ไอเอ็มเอส และโททอลเท่านั้น

ในการเปรียบเทียบไอเอ็มเอสและโททอลทางทฤษฎีในวิทยานิพนธ์ฉบับนี้จะใช้ดีบีทีจีเป็นบรรทัดฐาน เพราะดีบีทีจีถูกสร้างขึ้นเพื่อกำหนดลักษณะทั่ว ๆ ไปของระบบจัดการฐานข้อมูลที่ควรจะเป็น การเปรียบเทียบดังกล่าวจะเปรียบเทียบตามหัวข้อต่อไปนี้

๑. <u>โครงสร้างข้อมูล</u> คือลักษณะของโครงสร้างข้อมูลที่มองโดยผู้ใช้ข้อมูลนั้นโดยไม่คำนึงถึงรายละเอียดในการเก็บข้อมูลทางกายภาพว่าจะเก็บในลักษณะใด และจำเป็นที่ผู้ใช้จะต้องเข้าใจลักษณะโครงสร้างข้อมูลของฐานข้อมูลเพราะจะเป็นปัจจัยให้เข้าใจความสามารถของระบบฐานข้อมูลนั้นได้ดีขึ้น ซึ่งระบบจัดการฐานข้อมูลต่าง ๆ สร้างโครงสร้างข้อมูลที่มีประสิทธิภาพต่างจากโครงสร้างข้อมูลในภาษาโคบอล

โครงสร้างข้อมูลแสดงส่วนประกอบในระดับต่าง ๆ ของฐานข้อมูล คือไอเทม(Item) กรุ๊ป(Group) ความสัมพันธ์กลุ่ม(Group relation) เอนทรี(entry) แฟ้มข้อมูล(file)และระบบฐานข้อมูล(data base) ซึ่งคำจำกัดความของโครงสร้างข้อมูลจะแสดงในส่วนที่เรียกว่าสกีม่า ซึ่งในหลาย ๆ ระบบในแต่ละสกีม่าจะมีซับสกีม่าได้หลาย ๆ ตัว รายละเอียดของโครงสร้างข้อมูลแสดงในหัวข้อ ๓.๒

๒. คำจำกัดความข้อมูล ต่างจากโครงสร้างข้อมูลคือ จะบรรยายลักษณะ และฟอร์ม (format) ของข้อมูลตลอดจนความสัมพันธ์ระหว่างข้อมูลในระดับต่าง ๆ ของโครงสร้างข้อมูลที่เรียกว่าสกีม่า โดยใช้ภาษาดีดีแอลในการให้คำจำกัดความข้อมูลดังกล่าว ภาษาดีดีแอลนี้เคยกล่าวถึงในหัวข้อ ๒.๖.๑ และจะกล่าวในรายละเอียดทั้ง ๓ ระบบจัดการฐานข้อมูลในหัวข้อ ๓.๓

๓. คำสั่งในการโต้ตอบกับระบบฐานข้อมูล เป็นกระบวนการเลือกข้อมูลตามความต้องการของผู้ใช้ด้วยคำสั่งง่าย ๆ ซึ่งผู้ใช้ไม่จำเป็นจะต้องทราบขั้นตอนในการทำงานของระบบฐานข้อมูลที่จะดึงข้อมูลดังกล่าวออกมา การทำงานส่วนมากจะเป็นระบบการเข้าถึงข้อมูลตามลำดับ(sequencial search) และผลมักแสดงทางกระดาษพิมพ์(printout) หรือจอภาพในฟอร์มที่ออกแบบไว้แล้ว

๔. การทำให้เป็นปัจจุบัน เป็นกระบวนการเปลี่ยนข้อความบางส่วนในระบบฐานข้อมูล ซึ่งไม่รวมถึงการสร้างโครงสร้างข้อมูลใหม่

การทำให้เป็นปัจจุบัน มีการทำงานคล้ายกับคำสั่งในการโต้ตอบกับระบบฐานข้อมูล คือจะต้องมีการดึงข้อมูลที่ไม่ต้องการออกมาจากระบบฐานข้อมูลก่อน แต่การที่จะใส่ข้อมูลทดแทนจะยุ่งยากกว่า ในระบบจัดการฐานข้อมูลบางระบบจะสามารถทำคำสั่งในการโต้ตอบกับระบบฐานข้อมูลและการทำให้เป็นปัจจุบันได้ในเวลาเดียวกัน

๕. คำสั่งในการสร้างระบบฐานข้อมูล สิ่งสำคัญของคำสั่งในการสร้างระบบฐานข้อมูลก็คือคำจำกัดความข้อมูลเพื่อทราบฟอร์มของข้อมูลว่าจะต้องการให้มีลักษณะเป็นอย่างไร

ในบางระบบจะใช้คำสั่งการทำให้เป็นปัจจุบันในการสร้าง

๖. ความคล่องตัวของโปรแกรม เป็นคำสั่งที่ผู้เขียนโปรแกรมจะเรียกใช้ประกอบกับคำสั่งการทำงานในภาษาคอมพิวเตอร์ต่าง ๆ เช่นโคบอล พีแอลวัน เป็นต้น โดยคำสั่งต่าง ๆ จากความคล่องตัวของโปรแกรมจะช่วยโปรแกรมทำงานในการเข้าสู่การถ่ายข้อมูลจากระบบฐานข้อมูลทางกายภาพมาสู่โปรแกรมทำงาน และยังมีคำสั่งอื่น ๆ ที่ทำหน้าที่คล้ายคำสั่งในการควบคุม เช่น OPEN

CLOSE และ HOLD เป็นต้น

๗. คำสั่งในการบริหารข้อมูล เป็นคำสั่งที่รับผิดชอบโดยเฉพาะในระบบฐานข้อมูลซึ่งส่วนสำคัญก็กล่าวในคำจำกัดความข้อมูลและคำสั่งการสร้างระบบฐานข้อมูล แต่ยังมีคำสั่งอื่น ๆ ที่เรียกว่าเป็นคำสั่งด้านการบริหารข้อมูล ดังเช่นการเตรียมการรักษาความลับและความปลอดภัยของข้อมูล การสร้างโครงสร้างข้อมูลใหม่เมื่อมีการเพิ่มระเบียนหรือไอเทมใหม่ ๆ เข้ามาในฐานข้อมูลเดิมสำหรับคำสั่งในการบริหารข้อมูลบางครั้งจะถูกทำขึ้นในโปรแกรมทำงาน

๘. โครงสร้างทางกายภาพ ดังการแสดงการเก็บข้อมูลระดับต่าง ๆ ทางกายภาพและควบคุมการจัดเก็บข้อมูลทางกายภาพ ซึ่งคำสั่งนี้ถูกควบคุมโดยไอโอคอนโทร(I/O Control) และในโครงสร้างทางกายภาพได้รวมถึงเทคนิคต่าง ๆ ในการเข้าถึงข้อมูลดังเช่น Index Sequential และยังรวมถึงการเรียงข้อมูล (sort) และแฟ้มข้อมูลเพื่อให้มีประสิทธิภาพในการเข้าถึงข้อมูล

๙. ส่วนประกอบในการทำงานร่วมกับระบบจัดการฐานข้อมูล ซึ่งประกอบด้วยส่วนที่เป็นเครื่องจักรและส่วนที่เป็นโปรแกรมที่เรียกว่าระบบคำสั่งการดำเนินงาน ซึ่งในส่วนนี้จะกล่าวเฉพาะส่วนที่เกี่ยวข้องกับระบบจัดการฐานข้อมูลเท่านั้น

ตั้งแต่ข้อ ๓ ถึงข้อ ๗ เป็นส่วนประกอบในดีเอ็มแอล ซึ่งเคยกล่าวในข้อ๒.๖.๒ส่วนรายละเอียดจะแสดงในหัวข้อ ๓.๔ โดยจะไม่แยกเป็นหัวข้อย่อย

๓.๑ ความเป็นมาและลักษณะทั่ว ๆ ไปของระบบจัดการฐานข้อมูลทั้ง ๓

๓.๑.๑ ดีบีทีจี เป็นระบบจัดการฐานข้อมูลที่ถูกสร้างขึ้นเพื่อเป็นแนวทางความคิดว่าในระบบจัดการฐานข้อมูลนั้นควรจะมีส่วนประกอบอะไรบ้าง กลุ่มผู้สร้างมีชื่อกลุ่มว่าโคดาซิล เป็นคณะบุคคลจากบริษัทผู้สร้างคอมพิวเตอร์ บริษัทผู้ใช้ และคณะอาจารย์มหาวิทยาลัยในสหรัฐอเมริกา ซึ่งเป็นที่รู้จักกันในฐานะที่เป็นผู้นำในการปรับปรุงภาษาโคบอลในระหว่างปี ค.ศ. ๑๙๕๙ - ๑๙๖๑

จุดมุ่งหมายของดีบีทีจี คือ

๑. จัดโครงสร้างข้อมูลให้เหมาะสมในการใช้งานแต่ละงานโดยข้อมูลบางส่วนอาจจะมีผู้ใช้ร่วมกัน ซึ่งจะช่วยไม่ให้เกิดการซ้ำซ้อนของข้อมูล

๒. อนุญาตให้ผู้ใช้มากกว่า ๑ ราย ทำการดึงหรือแก้ไขข้อมูลบางส่วนในระบบฐานข้อมูลได้ในเวลาเดียวกัน

๓. จัดเตรียมและอนุญาตให้มีวิธีการเข้าถึงข้อมูลได้หลาย ๆ วิธี ต่อระบบฐานข้อมูลทั้งหมดหรือเพียงบางส่วนก็ได้

๔. จัดเตรียมการป้องกันระบบฐานข้อมูลจากผู้ไม่มีสิทธิในการเข้าถึงข้อมูลไปเรียกใช้ข้อมูลบางส่วน และจากผลกระทบของโปรแกรม

๕. จัดเตรียมระบบกลางที่มีความสามารถที่จะควบคุมการเก็บข้อมูลทางกายภาพ

๖. อนุญาตให้มีโครงสร้างข้อมูลได้หลาย ๆ ลักษณะ เช่น โครงสร้างแบบต้นไม้แบบร่างแห เป็นต้น

๗. ยอมให้ตัวโปรแกรมและข้อมูลเป็นอิสระต่อกัน

๘. แยกการบรรยายลักษณะของข้อมูลในระบบฐานข้อมูลจากบรรยายลักษณะของข้อมูลที่จะใช้ในโปรแกรม

๙. จัดเตรียมการบรรยายลักษณะของระบบฐานข้อมูลโดยไม่เกี่ยวข้องกับโปรแกรมทำงาน

๓.๑.๒ <u>ไอเอ็มเอส</u> ระบบจัดการฐานข้อมูลไอเอ็มเอสผลิตโดยบริษัทไอบีเอ็ม ตั้งอยู่เลขที่ ๑๑๓๓ Westchester Avenue, While Plains, New York 10604 ไอเอ็มเอสถูกนำออกสู่วงการคอมพิวเตอร์ครั้งแรกในปี ค.ศ. ๑๙๖๘ ด้วยความก้าวหน้าทางวิทยาการและการปรับปรุงเพื่อเพิ่มประสิทธิภาพ บริษัทไอบีเอ็มได้นำไอเอ็มเอสรุ่นต่าง ๆ ออกสู่วงการคอมพิวเตอร์เป็นระยะ ๆ จนถึงปัจจุบัน โดยใช้ชื่อเรียกต่าง ๆ กัน เช่น ไอเอ็มเอสวีเอสวัน(IMS/VS 1) ไอเอ็มเอสทู(IMS II) ดีแอลวัน(DL/1) ดีแอลวันเอนทรี(DL/1 Entry) แวนดีแอลวัน(VANDL - 1) เป็นต้น แต่ทั้งหมดนี้มีโครงสร้างของระบบฐานข้อมูลที่เหมือนกัน แตกต่างกันในรายละเอียดปลีกย่อย ดังเช่น ไอเอ็มเอส วีเอลวันใช้ได้กับโอเอส(OS-Operating System) หรือ โอเอส-วีเอสวัน(OS/VS 1) ในขณะที่ดีแอลวันต้องใช้กับโอเอสที่ชื่อดีโอเอสวีเอสวัน(DOS/VS 1) เป็นต้น

ในวิทยานิพนธ์ฉบับนี้จะใช้คำว่าไอเอ็มเอสโดยหมายถึงไอเอ็มเอสวีเอสวันซึ่งเป็นระบบที่มีใช้อยู่ในประเทศไทยในปัจจุบันนี้

ไอเอ็มเอสทำงานได้ทั้งการประมวลงานต่อเนื่อง(Batch System) และการประมวลงานในสาย(On-line System) และยังติดต่อกับภาษาโฮสท์(Host Language) คือโคบอล พีแอลวัน และแอสแซมเบอร์ แต่ไอเอ็มเอสมีข้อจำกัดคือใช้ได้กับเครื่องคอมพิวเตอร์ของบริษัทไอบีเอ็มเท่านั้นคือ ใช้ได้กับไอบีเอ็ม ๓๖๐(IBM/360) ไอบีเอ็ม ๓๗๐(IBM 370) ไอบีเอ็ม ๓๐๓ X(IBM/303X) และไอบีเอ็ม ๔๓XX(IBM 43XX)

ไอเอ็มเอสมีผู้ใช้ประมาณ ๑ พันรายทั่วโลก และด้วยเหตุผลที่ว่าในประเทศไทยมีผู้ใช้เครื่องคอมพิวเตอร์ของบริษัทไอบีเอ็มเป็นจำนวนมากจึงมีหลายหน่วยงานที่ใช้ไอเอ็มเอส เช่น สำนักงานสถิติแห่งชาติติดตั้งไอเอ็มเอสเพื่อใช้ในสำนักงานปฏิรูปที่ดินเพื่อเกษตรกรรม กรมวิชาการกระทรวงเกษตร บริษัทการบินไทยจำกัดติดตั้งไอเอ็มเอส เพื่อใช้ในงานบัญชีและการควบคุมสินค้าคงคลัง บริษัทปูนซีเมนต์ไทย ติดตั้งดีแอลวัน เพื่อใช้ในการควบคุมสินค้าคงคลัง การจัดการผลิต และการให้บริการแก่ลูกค้า

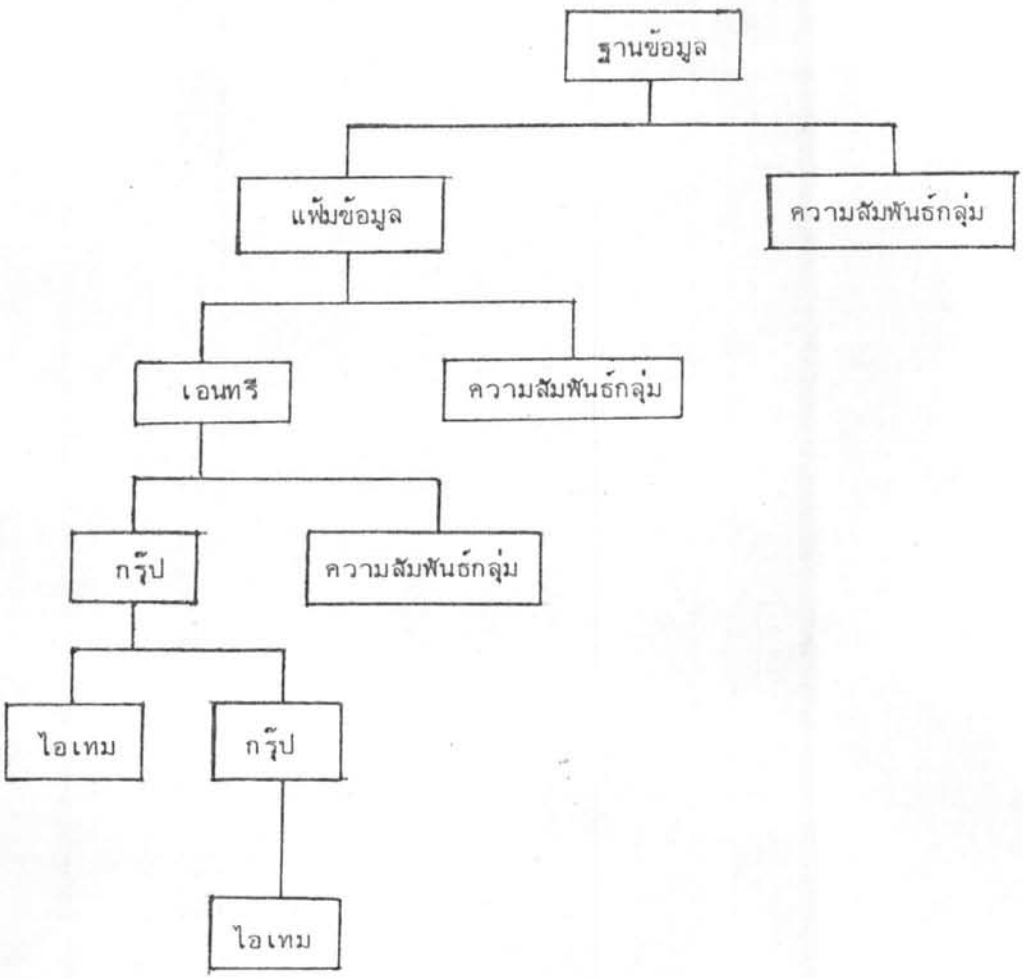
๓.๑.๓ <u>โททอล</u> ระบบจัดการฐานข้อมูลโททอลผลิตโดยบริษัทซินคอมดั้งอยู่เลขที่ 2300 Montana Avenue Cincinati, Ohio 45211 ถูกนำออกสู่วงการคอมพิวเตอร์ของบริษัทต่าง ๆ ดังเช่น ไอบีเอ็มทุกระบบ ฮันนีเวล(Honeywell) ยูนิแวค(Univac) เอ็นซีอาร์(NCR) คอนโทลดาต้า(Control Data) เป็นต้น และด้วยเหตุนี้โททอลจึงมีผู้ใช้ทั่วโลกประมาณ ๑,๓๕๐ ราย ซึ่งเป็นระบบจัดการฐานข้อมูลที่มีผู้ใช้มากที่สุด

โททอลสามารถทำงานได้ทั้งการประมวลงานต่อเนื่องและการประมวลงานในสายและยังสามารถติดต่อกับภาษาโฮสท์ คือโคบอล ฟอร์แทรน พีแอลวัน และแอสแซมเบอร์

ในประเทศไทยมีผู้ใช้โททอล ๒ บริษัทคือ บริษัทเอสโซสแตนดาร์ด ใช้ในงานบัญชี งานบุคคลากร การจัดการสินทรัพย์ และสินค้าคงคลัง บริษัทคาสเท็กซ์(ไทย)จำกัด ใช้ในงานบัญชีและสินค้าคงคลัง

๓.๒ โครงสร้างข้อมูล

คือโครงสร้างของระบบฐานข้อมูลส่วนที่เป็นตรรกภาพในความคิดของผู้ใช้ซึ่งเป็นอิสระจากโครงสร้างทางกายภาพ ดังนั้นโครงสร้างข้อมูลจึงบอกลักษณะของส่วนประกอบและความสัมพันธ์ต่าง ๆ ที่รวมกันทำให้เกิดเป็นระบบฐานข้อมูล โครงสร้างข้อมูลโดยทั่วไปที่นิยมกันมีโครงสร้างที่ประกอบด้วยไอเทม(Item) กรุ๊ป(Group) ความสัมพันธ์กลุ่ม(Group Relation) เอนทรี(Entry) แฟ้มข้อมูล(File) และฐานข้อมูล(Data Base) แสดงเป็นโครงสร้างข้อมูลได้ดังรูปที่ ๓.๒.๑



รูปที่ ๓.๒.๑ โครงสร้างข้อมูล

ทั้งดีบีทีจี ไอเอ็มเอส และโททอลเรียกส่วนประกอบต่าง ๆ ในโครงสร้างข้อมูลต่างกัน ดังในตารางที่ ๓.๒.๑

ตารางที่ ๓.๒.๑ แสดงความแตกต่างของระบบต่าง ๆ ที่เรียกส่วนประกอบในโครงสร้างระบบฐานข้อมูล

โครงสร้างทั่วไป	ดีบีทีจี	ไอเอมเอส (ดีเอ)	ไอเอ็มเอส (เอฟ)	โททอล
ไอเทม	data item	field	field	element
กรุ๊ป	record, data agregate	physical segment	logical segment	record
ความสัมพันธ์กลุ่ม	set	relationship	ไม่มี	linkage path
เอนทรี	record	record	record	data set
แฟ้มข้อมูล	data base	physical data base	logical data base	data base

รายละเอียดของส่วนประกอบในโครงสร้างข้อมูล คือ

๓.๒.๑ ไอเทม เป็นส่วนประกอบที่เล็กที่สุดในโครงสร้างข้อมูลซึ่งประกอบด้วย ชื่อของไอเทม ค่าที่บรรจุในไอเทม ความยาวและลักษณะ(picture) ของค่าที่บรรจุในไอเทม และระบบควบคุมในการอ้างถึง หรือการดึงข้อมูลไปใช้(access lock) ตัวอย่าง

ไอเทมชื่อ "อายุ" ค่าที่บรรจุเช่น "๒๓" "๒๕" "๕๐" มีค่าเป็นตัวเลขความยาวไม่เกิน ๒ หลัก

ไอเทมชื่อ "เพศ" ค่าที่บรรจุเป็น "ญ" หรือ "ช" นั่นคือ ค่าที่บรรจุเป็นพยัญชนะ ความยาวเพียง ๑ ตัวอักษร

ไอเทมสามารถแบ่งออกตามลักษณะของค่าที่บรรจุในไอเทม กล่าวคือ

๓.๒.๑.๑ ตัวเลข(numeric) คือค่าที่บรรจุในไอเทมเป็นตัวเลขที่สามารถใช้ในการคำนวณต่าง ๆ ได้ ดังเช่น ไอเทมชื่อ "เงินเดือน" ค่าที่บรรจุเป็นตัวเลขแสดงจำนวนเงินเดือน สามารถคำนวณหักภาษี หักเงินกู้ หาจำนวนเงินเดือนที่จ่ายจริงได้ เป็นต้น

๓.๒.๑.๒ สตริงไอเทม(String item) คือค่าที่บรรจุในไอเทมเป็นตัวอักษรหรือตัวเลขที่ไม่สามารถใช้ในการคำนวณได้ ตัวอย่าง ไอเทมชื่อ "เพศ" หรือ "ช" และไอเทมชื่อ "เลขประจำตัวนิสิต" ค่าที่บรรจุเป็น "๔๑๖๒๓๕" เป็นต้น สตริงไอเทมนี้ใช้เพื่อการอ้างถึงเท่านั้น

การเปรียบเทียบไอเทมชนิดตัวเลขแสดงในตารางที่ ๓.๒.๒และสตริงไอเทมแสดงในตารางที่ ๓.๒.๓

ตารางที่ ๓.๒.๒ การเปรียบเทียบไอเทมชนิดตัวเลข

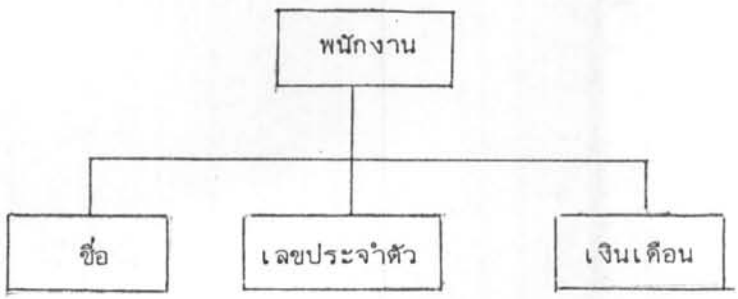
ระบบ	ชื่อเฉพาะระบบ (System term)	ความยาวที่บรรจุในไอเทม
โคบอล	numeric	๑-๘ หลักจะมีเครื่องหมายหรือไม่มีก็ได้
ดีบีทีจี	arithmetic	ความยาวขึ้นอยู่กับการกำหนดของผู้ผลิต
ไอเอ็มเอส	hexadecimal	ความยาวไม่เกิน ๒๕๕ หลัก
โททอล	arithematic	ความยาวตั้งแต่ ๑ หลักขึ้นไป

ตารางที่ ๓.๒.๓ การเปรียบเทียบไอเทมชนิดสตริงไอเทม

ระบบ	ชื่อเฉพาะระบบ	ความยาวที่บรรจุในไอเทม
โคบอล	alphabetic alphanumeric	๑ ตัวอักษรขึ้นไปและยอมรับค่าแบลงค์ (blank) ๑ ตัวอักษรขึ้นไป
ดีบีทีจี	Character bit	๑ คาร์แลคเตอร์ขึ้นไป ๑ บิทขึ้นไป
ไอเอ็มเอส	alphanumeric	๑ ถึง ๒๕๕ ตัวอักษรเท่านั้น
โททอล	alphanumeric	ตั้งแต่ ๑ ตัวอักษรขึ้นไป

๓.๒.๒ กรุ๊ป เกิดจากหลาย ๆ ไอเทมมารวมกลุ่มกัน หรือเกิดจากการร่วมกลุ่มของกรุ๊ปเองด้วย กรุ๊ปแบ่งออกเป็น ๒ ชนิด คือ

๓.๒.๒.๑ กรุ๊ปเดี่ยว(Single Group) คือ กรุ๊ปที่เกิดจากการรวมกลุ่มของไอเทม เช่น กรุ๊ปชื่อ "พนักงาน" อันประกอบด้วยไอเทมที่มีชื่อว่า ชื่อ เลขประจำตัว และเงินเดือน ดังตัวอย่างของโครงสร้างในรูปที่ ๓.๒.๒



รูปที่ ๓.๒.๒ แสดงโครงสร้างของกรุ๊ปเดี่ยว

หรือกรุ๊ปเดี่ยวอาจเกิดจากการนำข้อมูลหลายไอเทมมาต่อกันมีลักษณะคล้ายกับสตริงไอเทม ตัวอย่างเช่น การรวมไอเทมชื่อว่า วัน เดือน และปี ซึ่งมีค่าเป็น "๒๕" "๑๒" และ "๒๔" รวมเป็นกรุ๊ปเดี่ยวชื่อ "วันที่" เทียบเท่ากับ วัน เดือน ปี ที่มีค่าเป็น "๒๕๑๒๒๔"

๓.๒.๒.๒ <u>กรุ๊ปรวม(Compound Group)</u> คือกรุ๊ปที่รวมทั้งกลุ่มของไอเทมและกลุ่มของกรุ๊ปเข้าด้วยกัน ดังตัวอย่าง

กรุ๊ปชื่อ "ความสามารถ" = { รหัส, ชื่อความสามารถ }

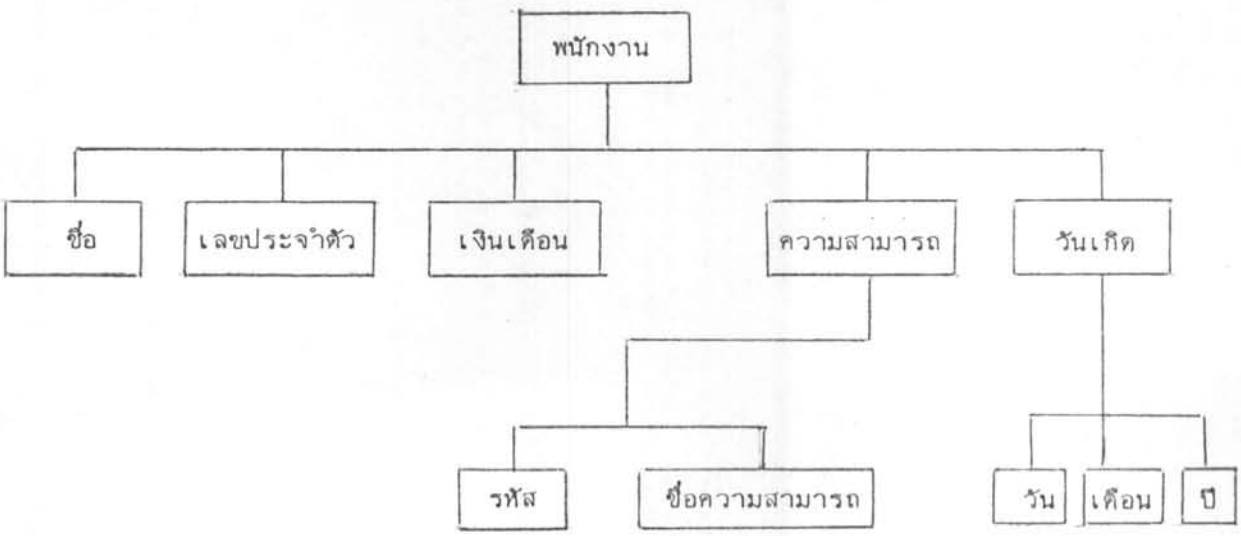
"วันเกิด" = { วัน, เดือน, ปี }

"พนักงาน" = { ชื่อ, เลขประจำตัว, เงินเดือน }

เมื่อรวมกันเป็นกรุ๊ปรวมจะเป็นดังนี้

"พนักงาน" = { ชื่อ, เลขประจำตัว, เงินเดือน, ความสามารถ = { รหัส, ชื่อความสามารถ }, วันเกิด = { วัน, เดือน, ปี } }

โครงสร้างของกรุ๊ปรวมชื่อ "พนักงาน" จะเป็นดังรูปที่ ๓.๒.๓



<u>รูปที่ ๓.๒.๓</u> แสดงโครงสร้างของกรุ๊ปรวมชื่อ "พนักงาน"

ในกรณีที่ไอเทมไม่มีกรุ๊ปในระดับที่สูงกว่าจะเรียกว่าพรินซิพอลไอเทม(principal item) จากตัวอย่าง กรุ๊ปรวมชื่อ "พนักงาน" ไอเทมชื่อว่า ชื่อ เลขประจำตัว และเงินเดือน เป็นพรินซิพอลไอเทม

สำหรับกรุ๊ปที่อยู่ในกรุ๊ปรวมเรียกว่า พรินซิพอลกรุ๊ป(principal group) จากตัวอย่าง ความสามารถ และวันเกิด เป็นพรินซิพอลกรุ๊ป

พรินซิพอลกรุ๊ปแบ่งเป็น ๒ ชนิด คือ กรุ๊ปซ้ำ(repeating group) และกรุ๊ปไม่ซ้ำ (non-repeating group)

กรุ๊ปซ้ำ คือ กรุ๊ปที่สามารถเกิดขึ้นได้หลาย ๆ ครั้งซ้ำกันภายใต้กรุ๊ปเดียวกัน จากตัวอย่าง กรุ๊ปชื่อว่า "ความสามารถ" จัดเป็นกรุ๊ปซ้ำเพราะในกรณีที่พนักงานมีความสามารถหลายอย่างก็จะมีข้อมูลความสามารถหลายรายการ

กรุ๊ปไม่ซ้ำ คือ กรุ๊ปที่เกิดขึ้นครั้งเดียวภายใต้กรุ๊ปเดียวกัน จากตัวอย่างกรุ๊ปชื่อว่า "วันเกิด" จัดเป็นกรุ๊ปไม่ซ้ำเพราะถือว่าทุกคนจะต้องมีวันเกิดเพียงวันเดียว

๓.๒.๓ <u>ความสัมพันธ์กลุ่ม</u> คือการสร้างความสัมพันธ์ระหว่างกลุ่มของกรุ๊ป ๒ กลุ่ม กลุ่มแรกเรียกว่า พาเรนกรุ๊ป(parent group) และกลุ่มที่สองเรียก ดีเพนเดนกรุ๊ป(dependent group) โดยที่ดีเพนเดนกรุ๊ปจะเป็นชนิดกรุ๊ปซ้ำหรือไม่ซ้ำก็ได้ แต่ไม่สามารถแสดงความหมายที่ชัดเจนได้ด้วยตนเองจะต้องประกอบกับพาเรนกรุ๊ปจึงจะได้ความหมายที่ชัดเจน ดังตัวอย่าง

กรุ๊ป "พนักงาน" { พนักงาน(นาย ก.), พนักงาน(นาย ข.) }

กรุ๊ป "ความสามารถ" { ความสามารถ(๑๑๑๐), ความสามารถ(๑๑๒๐), ความสามารถ(๑๑๓๐), ความสามารถ(๑๑๓๕) }

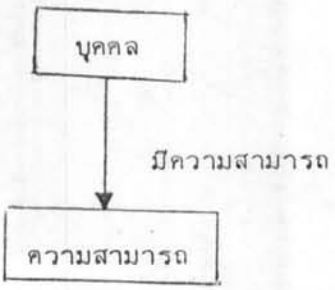
นำกรุ๊ปทั้งสองมาสร้างความสัมพันธ์กลุ่มชื่อ "มีความสามารถ" จะเป็น

$$\left\{\begin{array}{l} \langle \text{พนักงาน(นาย ก.)}, \text{ความสามารถ(๑๑๑๐)} \rangle, \\ \langle \text{พนักงาน(นาย ก.)}, \text{ความสามารถ(๑๑๓๕)} \rangle, \\ \langle \text{พนักงาน(นาย ข.)}, \text{ความสามารถ(๑๑๑๐)} \rangle, \\ \langle \text{พนักงาน(นาย ข.)}, \text{ความสามารถ(๑๑๒๐)} \rangle, \end{array}\right\}$$

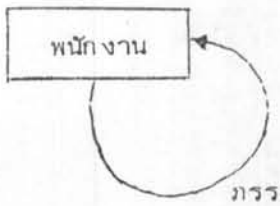
จากตัวอย่างนี้กรุ๊ป "พนักงาน" เป็นพาเรนกรุ๊ปและกรุ๊ป"ความสามารถ" เป็นดีเพนเดนกรุ๊ป โดยมีความสัมพันธ์กลุ่มชื่อว่า "มีความสามารถ"

<u>การแสดงโครงสร้างความสัมพันธ์กลุ่ม</u>

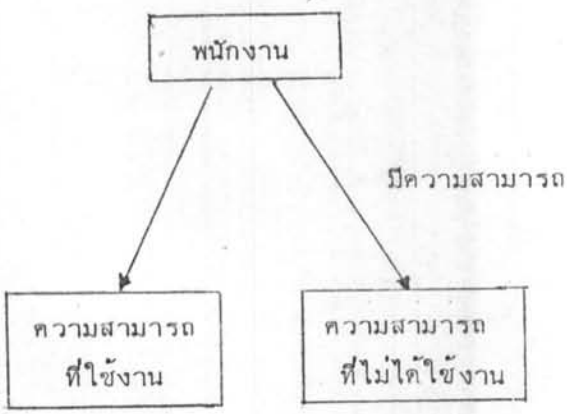
ความสัมพันธ์กลุ่ม ประกอบด้วย พาเรนกรุ๊ปและดีเพนเดนกรุ๊ปอย่างละหนึ่งกรุ๊ปขึ้นไป โดยมีหางลูกศรออกจากพาเรนกรุ๊ปเพียงอันเดียว หัวลูกศรชี้ไปยังดีเพนเดนกรุ๊ปแต่ละตัว ตัวอย่างการแสดงความสัมพันธ์กลุ่มดังเช่น รูปที่๓.๒.๔ แสดงความสัมพันธ์กลุ่มที่มีพาเรนกรุ๊ปและดีเพนเดนกรุ๊ป เพียงอย่างละ ๑ กรุ๊ปเท่านั้น รูปที่๓.๒.๕ แสดงความสัมพันธ์กลุ่มที่มีเพียงกรุ๊ปเดียวทำหน้าที่เป็นทั้งพาเรนกรุ๊ปและดีเพนเดนกรุ๊ป และรูปที่๓.๒.๖ แสดงความสัมพันธ์กลุ่มที่มีพาเรนกรุ๊ปเพียง ๑ กรุ๊ป แต่มีดีเพนเดนกรุ๊ปหลายกรุ๊ป



<u>รูปที่ ๓.๒.๔</u> แสดงโครงสร้างความสัมพันธ์กลุ่มชนิดที่มีดีเพนเดนกรุ๊ปเพียงกลุ่มเดียว



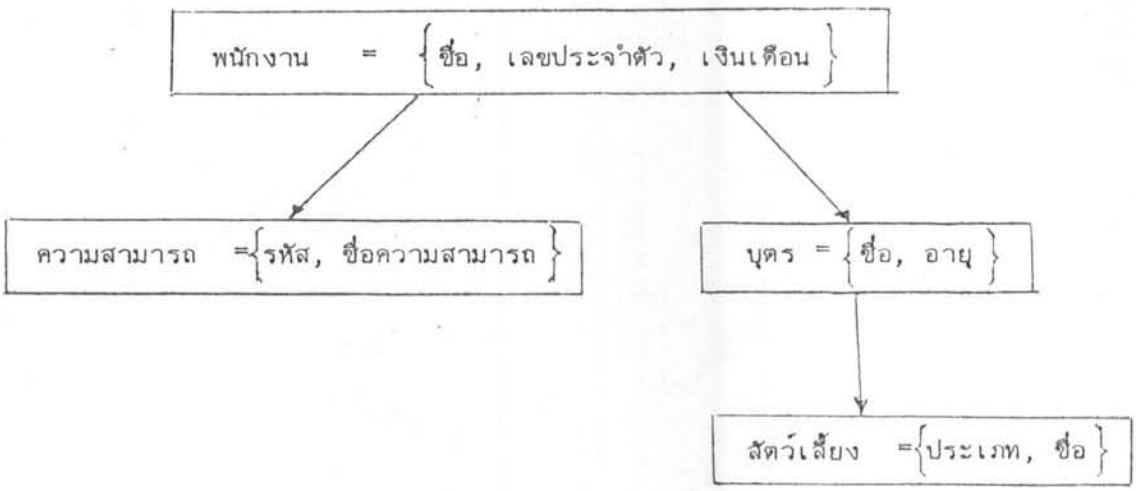
รูปที่ ๓.๒.๕ แสดงโครงสร้างความสัมพันธ์กลุ่มชนิดที่มี ๑ กรุ๊ปเท่านั้น



รูปที่ ๓.๒.๖ แสดงโครงสร้างความสัมพันธ์กลุ่มชนิดที่มีอินดีเพนเดนกรุ๊ปมากกว่า ๑ กรุ๊ป

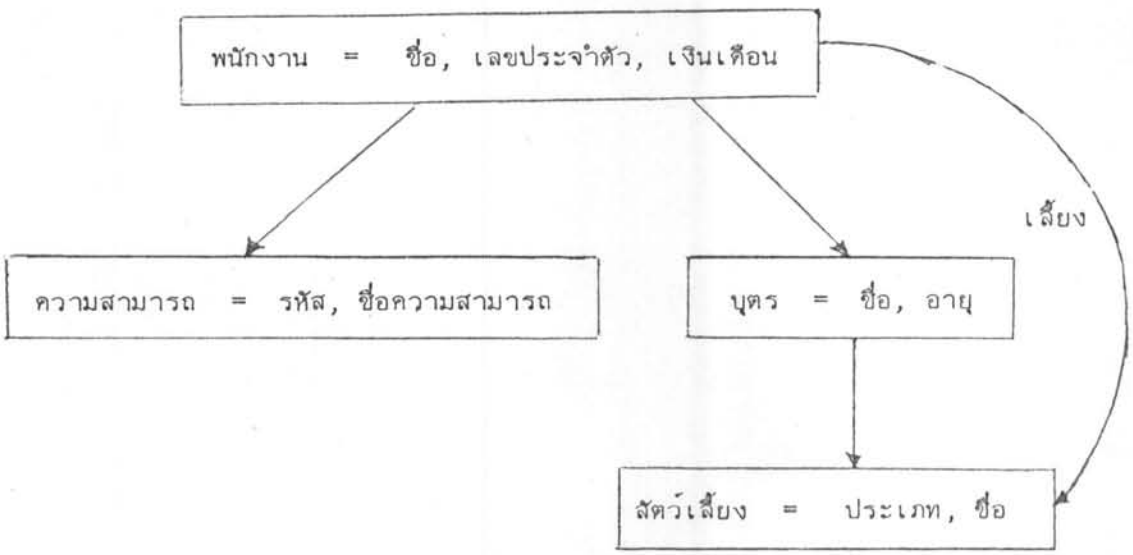
๓.๒.๔ <u>เอนทรี</u> คือกลุ่มของกรุ๊ปและความสัมพันธ์กลุ่มที่ประกอบเข้าด้วยกัน เอนทรีแบ่งตามลักษณะของโครงสร้างออกเป็น ๒ ชนิด คือ เอนทรีต้นไม้(tree entry) และเพลกเอนทรี (plex entry) โดยแต่ละชนิดมีรายละเอียดดังนี้

๓.๒.๔.๑ <u>เอนทรีต้นไม้</u> คือกลุ่มของกรุ๊ปที่มีความสัมพันธ์ที่จัดรูปร่างคล้ายต้นไม้ดังเช่น กต่ละกรุ๊ปมีพาเรนกรุ๊ปอย่างมากเพียงกรุ๊ปเดียว และมีเพียงกรุ๊ปเดียวเท่านั้นที่ไม่มีพาเรนกรุ๊ป ซึ่งเรียกกรุ๊ปนี้ว่ารู๊ทกรุ๊ป(rootgroup) ตัวอย่างโครงสร้างเอนทรีต้นไม้ ดังรูปที่ ๓.๒.๗



รูปที่ ๓.๒.๗แสดงโครงสร้างแบบเอนทรีต้นไม้

๓.๒.๔.๒ <u>เพลกเอนทรี</u> คือกลุ่มของความสัมพันธ์กลุ่มที่มีพาเรนกรุ๊ปเพียงกรุ๊ปเดียวและดีเพนเดนกรุ๊ปมีความสัมพันธ์กลุ่มแบบต้นไม้ แต่ความสัมพันธ์กลุ่มระหว่างกรุ๊ปอาจมีความสัมพันธ์กลุ่มไม่เป็นแบบต้นไม้ ดังตัวอย่างในรูปที่ ๓.๒.๘ เป็นการแสดงโครงสร้างเพลกเอนทรี จะพบว่าในตัวอย่างนี้ยังคงใช้โครงสร้างของเอนทรีต้นไม้ ดังรูปที่ ๓.๒.๗ โดยเพิ่มความสัมพันธ์กลุ่มชื่อว่า "เลี้ยง" ระหว่างกรุ๊ป "พนักงาน" และ "สัตว์เลี้ยง" ซึ่งไม่ใช่ความสัมพันธ์แบบต้นไม้



รูปที่ ๓.๒.๘ แสดงโครงสร้างแบบเพลกเอนทรี

การเปรียบเทียบลักษณะแฟ้มข้อมูลของระบบต่าง ๆ ขึ้นกับจำนวนเอนทรีที่ประกอบเป็นแฟ้มข้อมูล ดังรายละเอียดในตารางที่ ๓.๒.๔

ตารางที่ ๓๒.๔ แสดงการเปรียบเทียบลักษณะของแฟ้มข้อมูลของแต่ละระบบ

ระบบ	จำนวนเอนทรี
โคบอล	๑ เอนทรีขึ้นไป
ดีบีทีจี	๑ เอนทรีขึ้นไป
ไอเอ็มเอส(ดีเอ)	๑ เอนทรีเท่านั้น
ไอเอ็มเอส(เอฟี)	๑ เอนทรีขึ้นไป
โททอล	๑ หรือ มากกว่า ๑ เอ็นทรีขึ้นไป

๓.๒.๖ ระบบฐานข้อมูล คือกลุ่มของแฟ้มข้อมูลที่ได้รับการดูแลรักษาจากระบบจัดการฐานข้อมูลอาจจะประกอบด้วยแฟ้มข้อมูลเพียง ๑ แฟ้ม หรือมากกว่านั้น และความสัมพันธ์กลุ่มระหว่างแฟ้มข้อมูลก็อาจจะมีเพียง ๑ ความสัมพันธ์หรือมากกว่านั้นก็ได้เช่นกัน

ตาราง ๓.๒.๕ แสดงการเปรียบเทียบลักษณะโครงสร้างของระบบฐานข้อมูลของระบบต่าง ๆ

ระบบ	จำนวนแฟ้มข้อมูล
ดีบีทีจี	๑ แฟ้มเท่านั้น
ไอเอ็มเอส(ดีเอ)	๑ แฟ้มขึ้นไป
โททอล	๑ หรือ มากกว่า ๑ แฟ้มขึ้นไป

๓.๓ การเปรียบเทียบ ดีดีแอล

๓.๓.๑ ดีดีแอลของดีบีทีจี

๓.๓.๑.๑ การใช้ ดีดีแอลเพื่อเขียนสกีม่า

การใช้ดีดีแอลเพื่อเขียนสกีม่าให้สมบูรณ์ จะต้องประกอบไปด้วย เอ็นทรี่ (entry) ๔ ชนิด

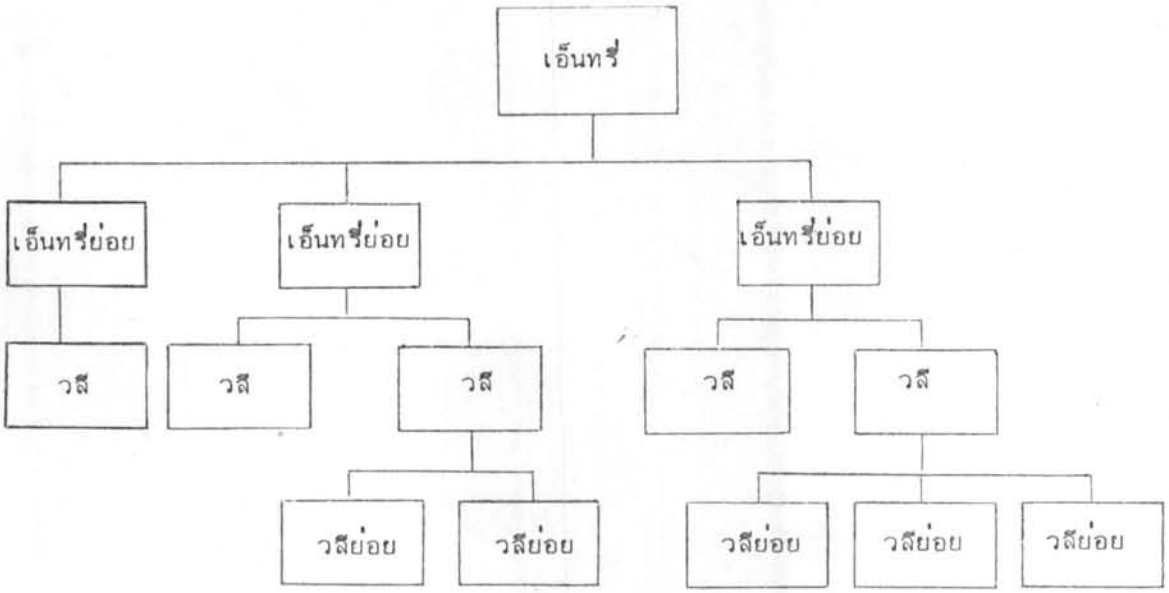
- สกีม่าเอ็นทรี่ (Schema entry)
- แอเรีย (area) เอ็นทรี่
- เอ็นทรี่ของระเบียนข้อมูล
- เซท (Set) เอ็นทรี่

กฎเกณฑ์การเขียนเอ็มทรี่ในสกีม่า

- ต้องเริ่มต้นด้วยสกีม่าเอ็นทรี่
- แอเรียเอ็นทรี่จะต้องสร้างขึ้นก่อนเอ็นทรี่ของระเบียนข้อมูล สำหรับทุก ๆ ระเบียนข้อมูลในแอเรียนั้น ๆ
- เอ็นทรี่ของระเบียนข้อมูล จะต้องสร้างขึ้นก่อนเซทเอ็นทรี่ สำหรับระเบียนข้อมูลที่รวมอยู่ในเซทนั้น ๆ

ลักษณะของเอ็นทรี่ อาจจะประกอบไปด้วยหลาย ๆ เอ็นทรี่ย่อย ซึ่งจะมีซ้ำ ๆ กันได้ในเอ็นทรี่ทั้งเอ็นทรี่และเอ็นทรี่ย่อย จะต้องประกอบไปด้วยอย่างน้อย ๑ วลี (Clause) ใช้อธิบายลักษณะ (Attribute) ของเอ็นทรี่หรือเอ็นทรี่ย่อย และแต่ละวลีจะประกอบไปด้วยวลีย่อย เอ็นทรี่และเอ็นทรี่ย่อยจะจบด้วยเครื่องหมายมหัพภาค (.) เสมอ

ตัวอย่างแสดงลักษณะของเอ็นทรี่



รูปที่ ๓.๓.๑ ตัวอย่างแสดงลักษณะของเอนทรีของดีบีทีจี

๓.๓.๑.๒ การใช้สัญญลักษณ์

๑. ทุก ๆ สมาชิก (elements) ในวลีประกอบไปด้วย อักษรตัวใหญ่ อักษรตัวเล็ก สัญญลักษณ์พิเศษและตัวอักษรพิเศษ

๒. อักษรตัวใหญ่และมีขีดเส้นใต้ หมายความว่า จำเป็นจะต้องใช้สมาชิกนี้เสนอ เมื่อใช้แบบฟอร์มนี้

๓. อักษรตัวใหญ่ไม่มีเส้นใต้ หมายความว่า เป็นการเลือกว่าจะใช้สมาชิกนี้หรือไม่ก็ได้

๔. อักษรตัวเล็กจะต้องถูกแทนด้วยเนม (name) หรือค่า (value)

๕. ความหมายของวงเล็บในแบบฟอร์มทั่ว ๆ ไปในรูปของสัญญลักษณ์พิเศษเป็นดังนี้

[a, b, c] อย่างน้อยไม่เกิดขึ้นเลย
อย่างมากเกิดขึ้นเพียง ๑

{ a, b, c } อย่างน้อยต้องเกิดขึ้น ๑
อย่างมากเกิดขึ้นเพียง ๑

‖ a, b, c ‖ อย่างน้อยต้องเกิดขึ้น ๑
อย่างมากเกิดขึ้นเพียง ๑ ของแต่ละประเภท

๖. กลุ่มของตัวอักษร ที่ใช้ใน DDL มี ๕๑ ตัว รวมถึงตัวอักษร, ตัวเลข และสัญญลักษณ์ (= , > , <)

Character

0, 1, ..., 9	ตัวเลข
A, B, ..., Z	ตัวอักษร
	space
+	plus sign
-	minus
*	asterisk
,	comma
;	semi-colon
.	period
"	quotation mark
(	left parenthesis
)	right parenthesis
/	stroke
$	national currency graphic

Symbol

=	equals sign
>	greatn than symbol
<	less than symbol

๗. เวอด (Word) เป็นการเรียงอักษรไม่เกิน ๓๐ ตัว ซึ่งตัวอักษรจะต้องเป็น 'A', ...'Z', 'O', ...'9', '-' และ '-' ต้องไม่อยู่หน้าหรือหลังสุด

๘. เนม (Names)

เนมมี ๒ ลักษณะ

- แบบปกติ

  เนมเป็นเวอดที่เริ่มด้วยตัวอักษร

- แบบพิเศษ (Escape form)

  เนมเป็นชุดของตัวอักษร โดยถูกจำกัดด้วย ๓

๙. ชนิดของเนม

- Data-base-data-name
- Record-name
- Area-name
- Set-name
- Lock-name
- Index-name
- Data-base-identifier
- Data-base-procedure
- Support-function
- Implementor-name
- Schema-name

๑๐. คำสงวน (Reserve words)

ACTUAL
ALIAS
ALL
ALLOWED
ALTER
ALWAYS
ARE
AREA
AREA-CODE
AREA-ID
ASCENDING (ASC)
AUTOMATIC (AUTI)
BINARY (BIN)
BIT
BY
CALC
CALL
CHAIN
CHARACTER (CHAR)
CHECK
CLOSE
COMPLEX
CURRENT
DATABASE-KEY (DBKEY)
DECIMAL (DEC)
DECODING
DELETE
DESCENDING (DESC)
DIRECT
DISPLAY
DUPLICATES (DUP)
DYNAMIC
ENCODING
EXCLUSIVE (EXCL)
FIND
FIRST
FIXED
FLOAT
FOR
GET
IN//
INDEX
INDEXED
INSERT
IS
KEY
LAST
LINKED
LOCATION (LOC)
LOCK
LOCKS
MANDATORY (MAND)
MANUAL
MEMBER
MEMBERS
MODE
MODIFY
NAME
NON-EXCLUSIVE (NEXCL)
NOT
OCCURRENCE
OCCURS
OF
ON
ONLY
OPEN
OPTIONAL (OPT)
OR
ORDER
OWNER
PICTURE (PIC)
POINTER-ARRAY (PTR)
PRIOR
PRIVACY
PROCEDURE (PROC)
PROTECTED (PROT)
RANGE
REAL
RECORD
RECORD-NAME
REMOVE
RESULT
RETRIEVAL (RETR)
SCHEMA
SEARCH
SELECTION
SELECTIVE
SET
SORTED
SOURCE
STORE
SYSTEM
TEMPORARY (TEMP)
THRU
TIMES
TO
UPDATE
USAGE
USING
VALUE
VIA
VIRTUAL
WITHIN

หมายเหตุ : ตัวย่อของคำสงวนอยู่ในวงเล็บ

สภาพแวดล้อมของระบบยอมให้ผู้บริหารข้อมูล (Data Administrator) สามารถใช้แก้ไขและปรับปรุงแต่ละสกีม่าภายใต้การควบคุมของตนเองได้ง่าย เช่น สามารถจัดระเบียบ, ควบคุมปฏิบัติงานของข้อคำสั่งต่าง ๆ, จัดระเบียบใหม่ และจัดโครงสร้างใหม่ การระบุเหล่านี้จะต้องมีการทำงานตามคุณสมบัติต่อไปนี้

ALTER	เป็นการยอมให้เปลี่ยนแปลงทุกสกีม่าด้วย ข้อยกเว้นของวลีไพรเวซี่ล็อค (privacy locks)
COPY	ระบุในดีดีแอลเพื่อการเขียนซับสกีม่า การที่ระบุในสกีม่านั้น เพื่อให้รวมอยู่ในซับสกีม่าด้วย
DISPLAY	อนุญาตให้ดูสกีม่าด้วย ข้อยกเว้นของวลีไพรเวซี่ล็อค
LOCKS	อนุญาตให้ดู, สร้าง, เปลี่ยนแปลงวลี ในสกีม่าด้วยไพรเวซี่ล็อค

๓.๓.๑.๓ ลักษณะโครงสร้างของการใช้ดีดีแอลกับสกีม่า

SCHEMA TYPES	SCHEMA entry AREA entry RECORD entry SET entry
SCHEMA ENTRY	SCHEMA clause ;Privacy schema clause ....
Schema clause	SCHEMA NAME IS schema-name
Privacy schema clause	PRIVACY LOCK [FOR { LOCKS | DISPLAY | COPY | ALTER }] IS { literal-1 | lock-name-1 | PROCEDURE data-Base-procedene-1 } [OR { literal-2 | lock-name-2 | PROCEDURE data-Base-procedene-2 }]...
AREA ENTRY	Area clause [;TEMPORARY area clause] [;PRIVACY area clause]... [;ON area clause]...
Area clause	AREA NAME IS area-name
Temporary area clause	AREA IS TEMPORARY

Privacy area clause

```
PRIVACY LOCK [ FOR [ [EXCLUSIVE] RETRIEVE ]
                   [ [PROTECTED]                ]
               [ [EXCLUSIVE] UPDATE ]
               [ [PROTECTED]        ]
               [ Support-function-1, support-function-2.... ] ... ] IS

  { literal-1                                  } [ [ OR { literal-2                                  } ] ]
  { lock-name-1                                }        { lock-name-2                                  }
  { PROCEDURE data-base-procedure-1            }        { PROCEDURE data-base-procedure-2              }
```

On area clause

```
ON [ [ OPEN [ FOR { [ EXCLUSIVE ]   { UPDATE }  } ] ]
                  { [ PROTECTED ]   { RETRIEVE } }
                  { [ NON-EXCLUSIVE ]          }
                  { EXCLUSIVE                  }
                  { PROTECTED                  }
                  { NON-EXCLUSIVE              }
   [ CLOSE ] ]

CALL data-base-procedure-3
```

```
RECORD ENTRY          | Record Sub-entry
                      |     [Data Sub-netry]...

Record Sub-entry      | RECORD clause
                      |     [;LOCATION clause]
                      |      ;WITHIN clause
                      |     [;ON record clause]....
                      |     [;PRIVACY record clause]...

Record clause         | RECORD NAME IS record-name-1

Location clause       | LOCATION MODE IS { DIRECT { data-base-data-name-1      }
                      |                  {        { data-base-indentifier-1   }
                      |                  { CALC [data-base-procedure-1]
                      |                  {      USING data-base-identifier-2
                      |                  {       ,data-base-identifier-3 ...
                      |                  {      DUPLICATES ARE [NOT ALLOWED]
                      |                  { VIA set-name-1 SET

Within clause         | WITHIN area-name-1 [{,area-name-2}...AREA-ID IS
                      |                     data-base-data-name-2]

On record clause      | ON [ INSERT                          ]  CALL data-base-procedure-2
                      |    [ REMOVE                          ]
                      |    [ STORE                           ]
                      |    [ DELETE                          ]
                      |    [ DELETE ONLY                     ]
                      |    [ DELETE SELECTIVE                ]
                      |    [ DELETE ALL                      ]
                      |    [ MODIFY                          ]
                      |    [ FIND                            ]
                      |    [ GET                             ]
```

Privacy record clause

```
PRIVACY LOCK FOR [ INSERT
                   REMOVE
                   STORE
                   DELETE
                   DELETE ONLY
                   DELETE SELECTIVE
                   DELETE ALL
                   MODIFY
                   FIND
                   GET ] IS { PROCEDURE data-base-procedure-3
                            { literal-1
                            { lock-name-1

[ OR { PROCEDURE data-base-procedure-4
     { leteral-2
     { lock-name-2 } ] ...
```

Data Sub-entry

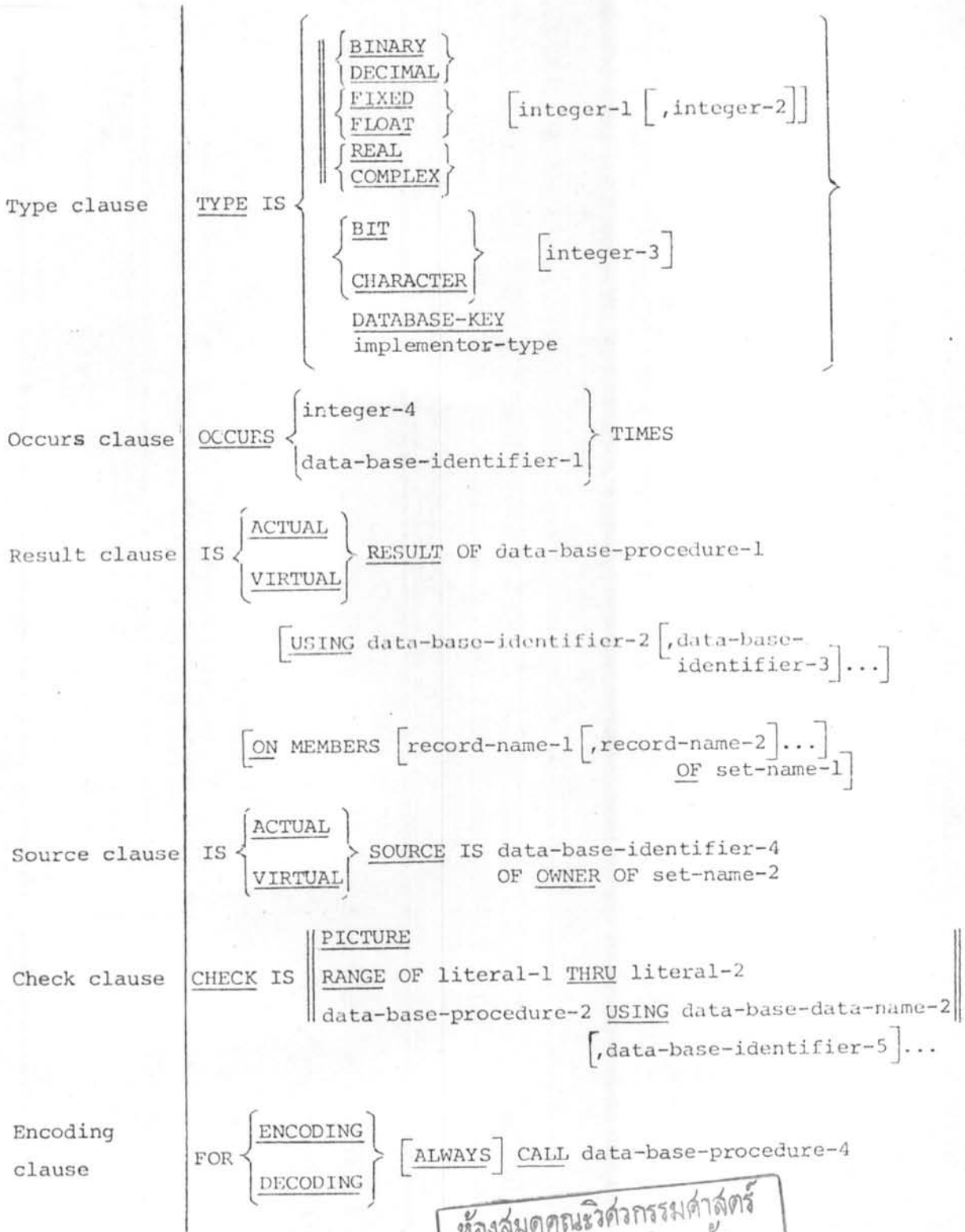
```
Data clause
   [;PICTURE clause]
   [;TYPE clause]
   [;OCCURS clause]
   [;RESULT clause]
   [;SOURCE clause]
   [;CHECK clause]
   [;ENCODING clause]
   [;ON data clause]...
   [;PRIVACY data clause]...
```

Data clause

```
[Level-number] data-base-data-name-1
```

Picture clause

```
PICTURE IS " { charcter-string-picture-specification } "
           { numeric-picture-specification           }
```



On data clause

```
ON [ STORE  ]
   [ GET    ]  CALL data-base-procedure-3
   [ MODIFY ]
```

```
[USING data-base-identifier-6[,data-base-identifier-7]..]
```

Privacy data clause

```
PRIVACY LOCK [ FOR [ STORE  ] ] IS { PROCEDURE data-base-procedure-5 }
             [     [ GET    ] ]   { literal-3                       }
             [     [ MODIFY ] ]   { lock-name-1                     }

[ OR { PROCEDURE data-base-procedure-6 } ]...
     { literal-4                       }
     { lock-name-2                     }
```

SET ENTRY

Set Sub-entry.

[Member sub-entry]...

Set sub-entry

SET clause
;MODE clause
;ORDER clause
[;ON set-clause]...
[;PRIVACY set-clause]...
;OWNER clause

Set clause

SET NAME IS Set-name-1

Mode clause

MODE IS { CHAIN LINKED TO PRIOR | POINTER-ARRAY DYNAMIC | Implementor-name }

Order clause

ORDER IS { [ALWAYS] { FIRST | LAST | NEXT | PRIOR } | SORTED [INDEXED [NAME IS index-name-1]] [WITHIN RECORD-NAME BY DATABASE-KEY [DUPLICATE ARE [FIRST | LAST | NOT] ALLOWED]] }

On set-clause

ON [[ORDER | INSERT | REMOVE]] CALL data-base-procedure-1

Privacy set-clause

PRIVACY LOCK [FOR [ORDER | INSERT | REMOVE | FIND]] IS { PROCEDURE data-base-procedure-2 | literal-1 | lock-name-1 }
[OR { PROCEDURE data-base-procedure-3 | literal-2 | lock-name-2 }]...

Owner clause

```
OWNER IS { record-name-1 }
         { SYSTEM        }
```

Member Sub-entry

```
MEMBER clause
   [;ASC/DES clause]
   [;SEARCH clause]...
     SET OCCURRENCE SELECTION clause
```

Member clause

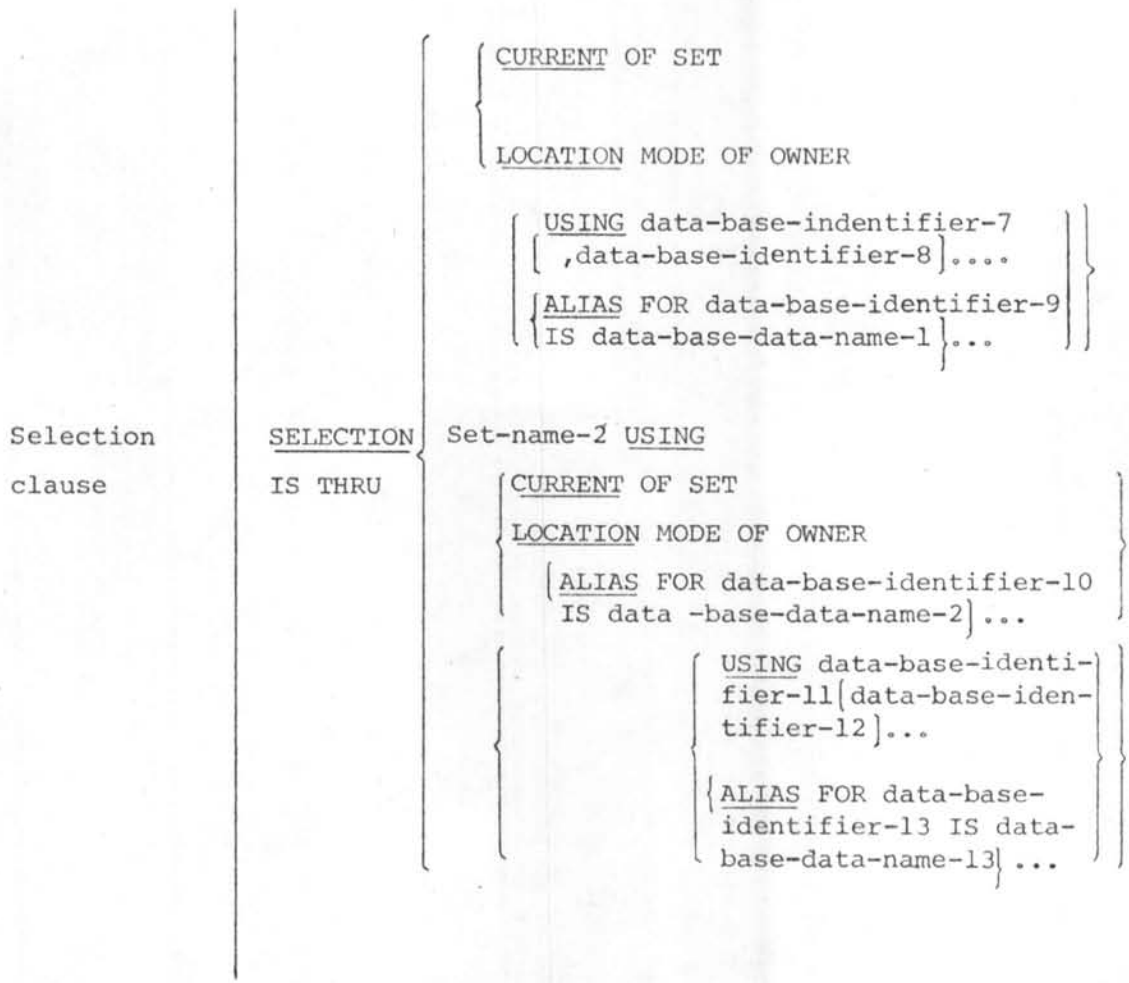
```
MEMBER IS record-name-1 { MANDATORY } { AUTOMATIC } [LINKEY TO
                        { OPTIONAL  } { MANUAL    } [OWNER  ]
   [DUPLICATES ARE NOT ALLOWED FOR data-base-identifier-1
                [,data-base-identifier-2]...]...
```

ASC/DES clause

```
{ ASCENDING  } [RANGE] KEY IS data-base-identifier-3
{ DECENDING  }         [,data-base-identifier-4]...
   [DUPLICATES ARE [FIRST] ALLOWED]
                   [LAST ]
                   [NOT  ]
```

SEARCH clause

```
SEARCH KEY IS data-base-identifier-5
              [,data-base-identifier-6]...
[USING { CALC                                  } ] DUPLICATES ARE
       { INDEX [NAME IS index-name-1]          }   [NOT] ALLOWED]
       { data-base-proceduer-1                 }
```


Selection clause
SELECTION IS THRU
Set-name-2 USING
CURRENT OF SET
LOCATION MODE OF OWNER
USING data-base-indentifier-7 ,data-base-identifier-8 ....
ALIAS FOR data-base-identifier-9 IS data-base-data-name-1 ...
CURRENT OF SET
LOCATION MODE OF OWNER
ALIAS FOR data-base-identifier-10 IS data -base-data-name-2 ...
USING data-base-identifier-11 data-base-identifier-12 ...
ALIAS FOR data-base-identifier-13 IS data-base-data-name-13 ...

SUB-SCHEMA

```
SUB-SCHEMA IDENTIFICATION DIVISION.
```

IDENTIFICATION DIVISION (กำหนดและตั้งชื่อ Sub-schema)

```
SUB-SCHEMA clause
[PRIVACY sub-schema clause] ...
[PRIVACY KEY sub-schema clause] .
```

SUB-SCHEMA clause

```
SUB-SCHEMA NAME IS sub-schema-name OF SCHEMA
     NAME schema-name
```

PRIVACY sub-schema clause

```
PRIVACY LOCK [ FOR [ LOCKS   ] ] IS { literal-1                    }
             [     [ DISPLAY ] ]    { lock-name-1                  }
             [     [ COMPILE ] ]    { PROCEDURE data-base-         }
             [     [ ALTER   ] ]    {           -procedure-1       }

   [ OR { literal-3                    } ] ...
   [    { lock-name-2                  } ]
   [    { PROCEDURE data-base-procedure-2 } ]
```

PRIVACY KEY sub-schema clause

```
PRIVACY KEY FOR COPY IS { literal-5          }
                        { implementor-name-1 }
```

SUB-SCHEMA DATA DIVISION (เป็นการตั้งชื่อและให้คุณลักษณะพิเศษของ areas, record และsets ของ Schema Sub-schema)

```
SUB-SCHEMA DATA DIVISION.
[RENAMING SECTION]
AREA SECTION
RECORD SECTION
[SET SECTION]
```

RENAMING SECTION (เพื่อสัมพันธ์กับชื่อใน Sub-schema)

```
[Renaming area clause] ...
[Renaming record clause] ...
[Renaming data clause] ...
[Renaming set clause] ...
[Renaming implementor clause] ... .
```

Renaming area clause

```
AREA NAME area-name-1 IN SCHEMA IS CHANGED TO area-name-2
     [,area-name-3 TO area-name-4] ...
```

Renaming record clause

```
RECORD NAME record-name-1 IN SCHEMA IS CAANGED TO
     record-name-2 [,record-name-3 TO record-name-4] ...
```

Renaming data clause

```
DATA NAME data-base-identifier-1 IN SCHEMA IS CHANGED TO
     data-base-data-name-1
     [,data-base-identifier-2 TO data-base-identifier-2]..
```

Renaming set clause

```
SET NAME set-name-1 IN SCHEMA IS CHANGED TO
     set-name-2 [,set-name-3 TO set-name-4] ...
```

Renaming implementor clause

```
IMPLEMENTOR NAME implementor-name-1 IN SCHEMA IS CHANGED
        TO implementor-name-2
        [ ,implementor-name-3 TO implementor-name-4 ] ...
```

AREA SECTION

```
COPY { ALL AREAS
     { area-name-1 [ ,area-name-2 ] ...
     {   [ Privacy lock area section clause ] ... } .
```

Privacy lock area section clause

```
PRIVACY LOCK [ FOR || [ EXCLUSIVE ]  RETRIEVAL                    ||
                  || [ PROTECTED ]                                ||
                  || [ EXCLUSIVE ]  UPDATE                        ||
                  || [ PROTECTED ]                                ||
                  ||   Support-function-1 [ ,support-             ||
                  ||                        -function-2 ] ...     ||
```

```
IS { literal-1                          }
   { lock-name-1                        }
   { PROCEDURE data-base-procedure-1    }

[ OR { literal-3                        } ] ...
     { lock-name-2                      }
     { PROCEDURE data-base-procedure-2  }
```

RECORD SECTION

```
Record Control Entry.
    [Data Description Entry. [Condition-name Entry.] ...]..
```

Record Control Entry

```
01  record-name-1 [ ; within record clause]
        [;Privacy lock record clause] ...
```

Within record clause

```
WITHIN area-name-1 [,area-name-2 ] ...
```

Privacy lock record clause

```
PRIVACY LOCK [ FOR [ INSERT                ] ]
             [     | REMOVE                | ]
             [     | STORE                 | ]
             [     | DELETE                | ]
             [     | DELETE ONLY           | ]
             [     | DELETE SELECTIVE      | ] IS
             [     | DELETE ALL            | ]
             [     | GET                   | ]
             [     | MODIFY                | ]
             [     | FINE                  | ]

    { PROCEDURE data-base-procedure-1 }
    { literal-1                       }
    { lock-name-1                     }

    [ OR { PROCEDURE data-base-procedure-2 } ] ...
         { literal-3                       }
         { lock-name-2                     }
```

Data description entry

```
level-number data-base-data-name-1
    [; Picture data clause]
    [; Usage data clause]
    [; Sign data clause]
    [; Occurs data clause]
    [; Privacy lock data clause] ...
```

Picture data clause

```
{PICTURE} IS character-string
{PIC     }
```

Usage data clause

```
[USAGE IS] {COMPUTATIONAL   }
           {COMP            }
           {COMPUTATIONAL-n }
           {COMP-n          }
           {DISPLAY         }
           {DATABASE-KEY    }
```

Sign data clause

```
[SIGN IS] {LEADING } [SEPARATE CHARACTER]
          {TRAILING}
```

Occur data clause

```
[;OCCURS integer-2 TIMES
   [INDEXED BY index-name-1 [,index-name-2] ...]

 ;OCCURS integer-1 TO integer-2 TIMES
    DEPENDING ON data-base-data-name-3
    [INDEXED BY index-name-1 [,index -name-2] ...]]
```

SET SECTION

```
SET SECTION

    COPY SET clause

    SET OCCURRENCE SELECTION clause

    [ PRIVACY SET SECTION clause ] ... .
```

COPY SET clause

```
COPY { set-name-1 , Set-name-2 ... }
     { ALL SETS                    } .
     { Set-name-3                  }
```

SET OCCURRENCE SELECTION clause

```
SET OCCURRENCE SELECTION FOR MEMBER record-name-1 THRU
                                    set-name-4

USING { CURRENT SET                                                    }
      { LOCATION MODE OF OWNER [ ALIAS FOR data-base-                  }
      {    identifier-1 IS data-base-data-name-1 ] ...                 }

  { Set-name-5 { USING data-base-identifier-2                          }
  {            {   [ , data-base-identifier-3 ] ...                    }
  {            { ALIAS FOR data-base-identifier-4                      }
  {            {   IS data-base-identifier-2 ...                       }
```

PRIVACY SET SECTION clause

```
[ PRIVACY LOCK [ FOR { ORDER  } ] IS { literal-1                    } ]
               { FINE   }           { lock-name-1                   }
               { REMOVE }           { PROCEDURE data-base-          }
               { INSERT }           {  -procedure-1                 }

    [ OR { literal-3                                  } ] ...
         { lock-name-2                                }
         { PROCEDURE data-base-procedure-2            }
```

Privacy lock data clause

```
PRIVACY LOCK [ FOR [ STORE ] ] IS { PROCEDURE data-base-procedure-3 }
             [ GET   ]            { literal-1                       }
             [ MODIFY ]           { lock-name-1                     }

    [ OR { PROCEDURE data-base-procedure-4 } ] ...
         { literal-3                       }
         { lock-name-2                     }
```

Condition-name entry

```
88  condition-name
        ; Value condition clause
```

Value condition clause

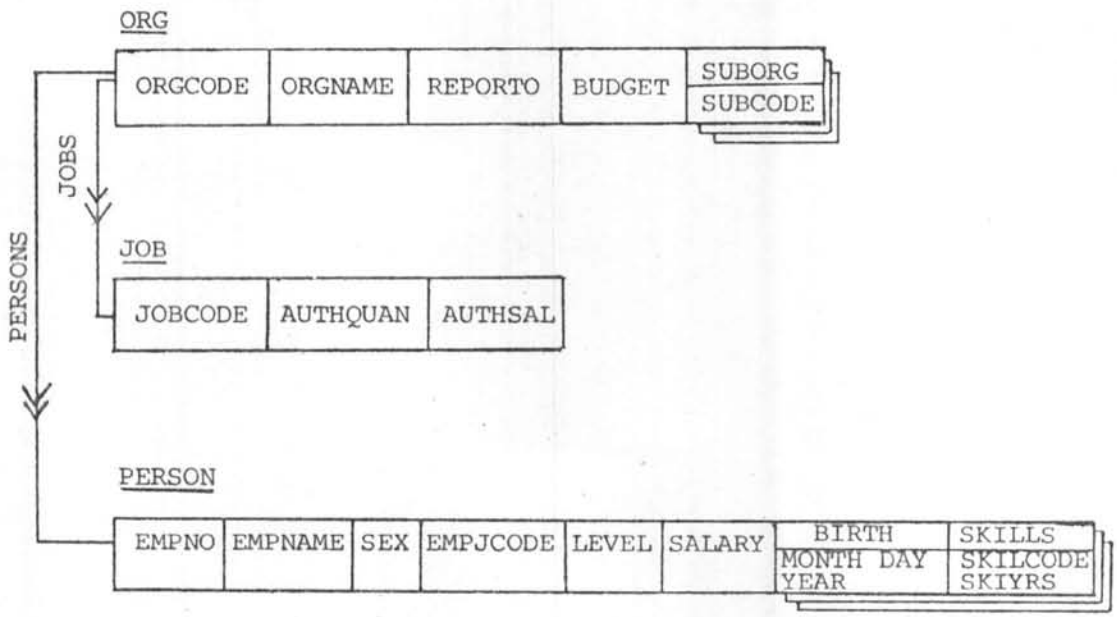
```
{ VALUE IS  } literal-1 [ { THRU    } literal-2 ]
{ VALUES ARE }           { THROUGH }

    [ ,literal-3 [ { THRU    } literal-4 ] ] ... .
                 { THROUGH }
```

๓.๓.๑.๔ ตัวอย่างการเขียนดีดีแอล

สมมุติบริษัทหนึ่งออกแบบฐานข้อมูลของงานบริหารบุคคลมีชื่อสกีม่าว่า "ORGDATA" ประกอบด้วยระเบียนข้อมูล ORG, JOB และ PERSON ซึ่งระเบียนข้อมูล ORG สามารถหยิบข้อมูลจาก PERSON และ JOB ได้ตาม PERSONS และ JOBS ตามลำดับ โครงสร้างฐานข้อมูล แสดงในรูป ๓.๓.๒

ชื่อสกีม่า "ORGDATA"



รูปที่ ๓.๓.๒ โครงสร้างฐานข้อมูลในลักษณะของสกีม่า

ตัวอย่างการเขียนสกีม่าใน ดีดีแอล

```
SCHEMA NAME IS ORGDATA
      ;PRIVACY LOCK FOR COPY IS ORGSUB
AREA NAME IS ORGPART
RECORD NAME IS ORG
       ;PRIVACY LOCK FOR DELETE IS ORGDEL
       ;PRIVACY LOCK IS SESAME
       01  ORGCODE  PICTURE IS "9(4)"
       01  ORGNAME  TYPE IS CHARACTER 25
       01  REPORTO  PICTURE IS "9999"
       01  BUDGET   TYPE DECIMAL FLOAT ; IS ACTUAL RESULT OF SALSUM
             ON MEMBERS OF PERSONS
       01  NOSUBORG TYPE BINARY
       01  SUBORG   OCCURS NOSUBORG TIMES
           02 SUBCODE PICTURE "9999"

RECORD NAME IS JOB
       ;PRIVACY LOCK FOR INSERT IS JOBINS
       01  JOBCODE  PICTURE "9999"
       01  AUTHQUAN PICTURE "99"
       01  AUTHSAL  TYPE FLOAT

RECORD NAME IS PERSON
       ;PRIVACY LOCK FOR INSERT IS PERINS
       01  EMPNO    PICTURE "9(5)"
       01  EMPNAME  TYPE CHARACTER 20
       01  SEX      PICTURE "A"
       01  EMPJCODE PICTURE "9999"
       01  LEVEL    PICTURE "X(4)"
       01  SALARY   PICTURE "9(5)V99"
           ;PRIVACY LOCK FOR GET IS PROCEDURE AUTHENT
```

```
01  BIRTH
    02  MONTH PICTURE "99"
    02  DAY   PICTURE "99"
    02  YEAR  PICTURE "99"
01  NOSKILLS TYPE BINARY
01  SKILLS   OCCURS NOSKILLS TIMES
    02  SKILCODE PICTURE "9999"
    02  SKLYRS   PICTURE "99"
SET NAME IS JOBS ; ORDER IS SORTED
    OWNER IS ORG
    MEMBER IS JOB OPTIONAL AUTOMATIC ; ASCENDING KEY IS
        JOBCODE DUPLICATES NOT ALLOWED

SET NAME IS PERSONS ; ORDER IS SORTED
    OWNER IS ORG
    MEMBER IS PERSON OPTIONAL AUTOMATIC ; ASCENDING KEY IS
        EMPJCODE , EMPNO DUPLICATES NOT ALLOWED
```

โคบอลดีวีแอล ประกอบด้วย ๓ ดิวิชั่น

- ไอเด็นทิฟิเคชั่น ดิวิชั่น (Identification Division)
- ดาต้า ดิวิชั่น (Data Division)
- โพรซีเซอ ดิวิชั่น (Procedure Division)

ไอเด็นติฟิเคชั่น ดิวิชั่น

บรรทัดที่เพิ่มขึ้นจากภาษาโคบอลเป็นไพรเวซี่ ซึ่งมีหรือไม่มีก็ได้ เมื่อใช้บรรทัดเหล่านี้จะต้องอยู่หลังโปรแกรม-ไอดี (PROGRAM-ID) ดังตัวอย่าง

```
IDENTIFICATION DIVISION.
PROGRAM-ID.
.....   ┐
.....   │
.....   ├   เอ็นทรี่ของไพรเวซี่ดี (PRIVACY KEY)
.....   │
.....   ┘

.....   ┐
.....   │
.....   ├   เอ็นทรี่ของโคบอลมาตรฐาน
.....   │
.....   ┘
```

๓.๓.๑.๕ ลักษณะโครงสร้างของการใช้ดีดีแอลกับซับสกีม่า

เอ็นทรี่ของไพรเวซี่ ศัย์ในดีดีแอลมีประโยชน์เพื่อให้เข้าถึง สกีม่า หรือซับสกีม่า ที่ถูกบีด ประกอบด้วย

- วลีของไพรเวซี่ คอมไพล (Privacy compile) ใช้ในการ invoke ในดาต้า ดิวิชั่น เพื่อระบุซับสกีม่า
- วลีของไพรเวซี่ คี แอเรีย (Privacy key area)
- วลีของไพรเวซี่ คีของระเบียนข้อมูล (Privacy key record)
- วลีของไพรเวซี่ คีของข้อมูล (Privacy key data)
- วลีของไพรเวซี่ คีเซท (Privacy key set)

ดาต้า ดิวิชั่น

ส่วนเพิ่มเติมของโคบอล ดีดีแอลในดาต้า ดิวิชั่น เป็น เซ็คชั่น (Section) ใหม่ เรียกว่า สกีม่าเซ็คชั่น (Schema Section) และจะต้องเป็นเซ็คชั่นแรกในดาต้า ดิวิชั่น โดยมีคุณสมบัติ ดังนี้

- เป็นการอ้างอิงถึง ซับสกีม่า ที่กำหนด
- ทำให้ระบบได้ทำการเก็บและตั้งชื่อทุก ๆ เนื้อที่ที่ทำงานของโปรแกรมที่จำเป็น

ตัวอย่าง

```
DATA DIVISION.

SCHEMA SECTION

INVOKE clause

.....
.....        } เซ็คชั่นของมาตรฐานโคบอล
.....
.....
```

การใช้ดีดีแอล เพื่อเขียนโคบอล ซับสกีม่านั้น จะต้องแบ่งออกเป็น ๒ ส่วนตามลำดับ ดังนี้

- ซับสกีม่า ไอเด็นทิฟิเคชั่น ดิวิชั่น (Sub-schema Identification Division)
- ซับสกีม่า ดาต้า ดิวิชั่น (Sub-Schema Data Division) ประกอบด้วย ๔ เซ็คชั่น ตามลำดับ

[RENAMING SECTION]
AREA SECTION
RECORD SECTION
[SET SECTION]

Sub-schema Identification Division เป็นการกำหนดและตั้งชื่อซับสกีม่า โดยสามารถกำหนด ไพรเวซี่ ล็อค และไพรเวซี่ คีย์ของ ซับสกีม่าได้

Sub-schema Data Division เป็นการตั้งชื่อและให้คุณลักษณะพิเศษของแอเรีย ระเบียนข้อมูลของสกีม่า ที่มีอยู่ในซับสกีม่า สามารถเปลี่ยนแปลงชื่อให้สัมพันธ์กับชื่อในซับสกีม่า โดยใช้ RENAMING SECTION

๓.๓.๒ ดีดีแอลของไอเอ็มเอส

ต่อไปนี้จะกล่าวถึง โปรแกรมย่อยสองระดับ ซึ่งเทียบได้กับ ดี.ดี.แอล ของ ดีบีทีจี เพื่ออธิบายลักษณะทางกายภาพและทางตรรกของฐานข้อมูลและโครงสร้างงาน (Application Structure) สำหรับแต่ละโปรแกรมคำสั่งงาน

๓.๓.๒.๑ โปรแกรมย่อยระดับแรกเรียกว่า ดี.บี.ดี (DBD หรือ Data Base Description)

ประกอบด้วยกลุ่มประโยค (Set) ของ IMS ประเภท "Macro Instruction" ที่ผู้ใช้เขียนขึ้นเพื่อกำหนดลักษณะของฐานข้อมูลตามที่ต้องการ แล้วประโยคเหล่านี้จะถูกส่งให้แก่ โปรแกรมการสร้าง ดี.บี.ดี (DBD Generation Procedure) ต่อไป

กลุ่มประโยคในการสร้าง ดีบีดี (DBD Generation Macro Instruction) จะถูกนำไปเก็บไว้ในที่เก็บโปรแกรมก่อนการแปลเป็นภาษาเครื่อง เมื่อถูกเรียกมาทำงาน (execute) แล้ว จะสร้างเป็น "IMS DBD CSECT" แล้วถูกเก็บและนำไปรวมไว้ในที่เก็บโปรแกรมคำสั่งเครื่อง เพื่อการทำงานต่อไป

ประโยคที่ใช้ในการสร้างดีบีดี (DBDGEN Control Statement)

ในที่นี้จะแสดงเฉพาะรูปแบบและลำดับของประโยค ความหมายของสัญญลักษณ์ในรูปแบบของแต่ละประโยค

๑. อักษรภาษาอังกฤษตัวพิมพ์ใหญ่ หมายถึง คำที่สำคัญต่าง ๆ ภายในประโยค และถ้าเขียนประโยคนี้ต้องเขียนให้ถูกต้องตามที่ได้แสดงเอาไว้ในรูปแบบ

๒. อักษรภาษาอังกฤษตัวพิมพ์เล็ก หมายถึง ชื่อที่ผู้ใช้เป็นผู้ตั้งขึ้น ซึ่งมีลักษณะเป็นอักษรผสมตัวเลข

๓. ประโยคที่ใช้ในการสร้าง ดีบีดี มีลักษณะเป็น "Free Format" กล่าวคือค่อนข้างจะอิสระ หมายถึงว่า เวลาเขียนไม่จำเป็นต้องคำนึงถึงว่าจะต้องเขียนที่สดมภ์ใด ซึ่งถ้าเป็นโปรแกรมคำสั่งงานที่เขียนด้วยภาษาตัวแปลต่าง ๆ เช่น ภาษาอาพีจี ต้องคำนึงถึงหลักไวยากรณ์ และการลงรหัสตามสดมภ์ที่กำหนดไว้

๔. สัญญลักษณ์ที่เป็น "Operation code" ต้องเริ่มหลังสดมภ์ที่ ๑ (เช่น เครื่องหมาย =)

๕. อักษรที่เป็น "Operand" ต้องแยกออกจาก "Operation Code" อย่างน้อยหนึ่งช่องว่าง (Blank)

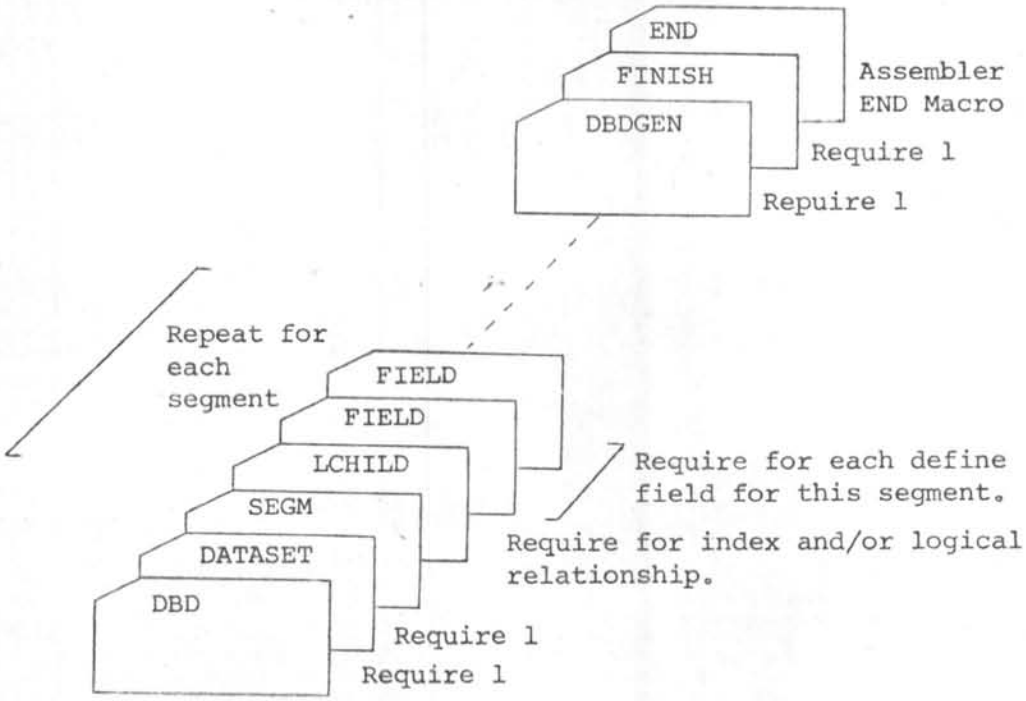
๖. แต่ละ "Operand" แยกกันด้วยเครื่องหมายจุลภาค (Comma) และต้องเรียงตามลำดับ

๗. กรณีที่ไม่สามารถเขียนให้จบประโยคได้ (คือ ต้องเขียนเกินสดมภ์ที่ ๗๒) ต้องมีอักษรอะไรก็ได้ที่สดมภ์ที่ ๗๒ และประโยคที่ต่อจากประโยคที่แล้วต้องเริ่มที่สดมภ์ที่ ๑๖

๘. เครื่องหมาย [ ] หมายถึง สามารถละข้อความ หรือตัวอักษรที่อยู่ภายในเครื่องหมายนั้นได้ (Optionnal)

๙. เครื่องหมาย { } หมายถึง ให้เลือกเอาข้อความหรือตัวอักษรที่อยู่ภายในเครื่องหมายนั้น ๑ อย่าง (Choice)

๑๐. เครื่องหมาย ------ หมายถึง มีได้มากกว่า ๑ พารามิเตอร์ที่ใช้ใน "Operand" เดียวกัน



รูปที่ ๓.๓.๓ โครงสร้างและลำดับของกลุ่มประโยคที่ใช้ในการสร้าง ดีบีดี

	Operation	General Use	Number Used in Each DBD Generation							
			SHSAM	SHISAM	HSAM	HISAM	HDAM	HIDAM	INDEX	LOGICAL
	DBD	Defines data base name	1	1	1	1	1	1	1	1
	DATASET	Defines data file within data base	1	1	1	1	1	1	1	1
	SEGM	Defines segment within data base	1	1	1–255	1–255	1–255	1–255	1	1–255
	[LCHILD]	Defines index relationship	0	0	0	0	0–255**	1–255**	1	0
		or								
	[LCHILD]	Defines logical relationship between segments	0	0	0	0	0–255**	0–255**	0	0
	[FIELD]	Defines a field within a segment	0–255	1–255	0–1000	1–1000	1–1000*	1–1000*	1–1000	0
	[XDFLD]	Specifies the field to be indexed by the secondary index	0	0	0	0	0–1000*	0–1000*	0	0
	DBDGEN	Indicates the end of DBD generation statements	1	1	1	1	1	1	1	1
	FINISH	Checks for successful DBD generation	1	1	1	1	1	1	1	1
	END	Indicates end of DBD generation input to the DOS/VS assembler	1	1	1	1	1	1	1	1

Notes: *The number of XDFLD and FIELD statements together must not exceed 255 per SEGM statement and must not exceed 1000 per DBD generation. A maximum of 32 XDFLD statements may be used per SEGM statement.

**The total number of LCHILD statements must not exceed 255.

รูปที่ ๓.๓.๔ ประโยคชนิดต่าง ๆ ที่ใช้ในการสร้าง ดีบีดี

๑. ประโยค ดี.บี.ดี. มีเพียงประโยคเดียว ใช้บ่งบอกถึงชื่อของการจัดแฟ้มข้อมูล (File Organization) ที่ใช้ และบอกชื่อของฐานข้อมูล

```
         DBD     NAME=dbdname
                        { HSAM                                  }
                        { HISAM                                 }
                        { HIDAM                                 }
                 ,ACCESS={ INDEX                                }
                        { SHSAM                                 }
                        { SHISAM                                }
                        { HDAM,RMNAME=(mod-name, { 1    }       }
                                                 { anch }       }
                                          [,rbn ,bytes  )]      }
```

๒. ประโยค DATASET ประโยคนี้ จะใช้บ่งบอกถึงสื่อที่ใช้เก็บฐานข้อมูล ขนาดของระเบียนข้อมูลทางกายภาพ, ขนาดของระเบียนข้อมูล, และถ้าสื่อเก็บฐานข้อมูลที่ใช้เป็นเทปแม่เหล็ก ประโยคนี้จะใช้บ่งบอกถึงชื่อของแฟ้มข้อมูลที่เป็นเอ้าพุทด้วย ประโยคนี้มีได้หนึ่งประโยค

โครงร่าง

```
         DATASET DD1=fname1
                        {3330}
                        {3340}
                        {3350}
                 ,DEVICE={2314}
                        {TAPE}

                 [,DD2=fname2 ]
                 [,DEVADDR=(SYSnnn-1,SYSnnn-2) ]
                 [,OVFLW=fname3 ]
                 [,BLOCK=(blk-fact-1 [,blk-fact-2 ]) ]
                 [,RECORD-(rec-len-1 [,rec-len-2]) ]
                 [,SCAN={cyls}]
                        {3   }
                 [,FRSPC=({fbff,fspf})]
                          {0    0   }
```

๓. ประโยค SEGM ประโยคนี้ใช้สำหรับแต่ละเซ็กเม้นท์ ในการทำ ดีบีดีเจน ครั้งหนึ่ง ๆ จะมีจำนวนเซ็กเม้นท์ได้อย่างมากไม่เกิน ๒๕๕ เซ็กเม้นท์ ประโยคนี้ใช้อธิบายรายละเอียดของเซ็กเม้นท์ เช่น พาเร้นท์เซ็กเม้นท์, ตัวบ่งชี้, จำนวนไบท์ของเซ็กเม้นท์ และถ้าเป็นเซ็กเม้นท์ที่มีความยาวแปรเปลี่ยนได้ จะต้องบอกถึงจำนวนไบท์ต่ำสุดและสูงสุดของเซ็กเม้นท์นั้น ๆ นอกจากนี้ยังสามารถใช้บอกถึงกฎเกณฑ์ในการแก้ไขปรับปรุงเซ็กเม้นท์อีกด้วย

โครงร่าง

	SEGM	NAME = seg-name1 [,PARENT={0 | (seg-name2,[{SNGL | DBLE}]) [,(lpseg-name[,V,db-name1])])}] [,BYTES={bytes | (max-bytes,min-bytes)}] [,COMPRTN=(mod-name[,D,INIT])] [,POINTER=one or more keyword options] [,RULES=([{P | L | V}][{P | L | V}][{P | L | V}][{,FIRST | ,LAST | ,HERE}])] [,SOURCE=((seg-name3[,D,db-name2]))]

๔. ประโยค FIELD

ประโยคนี้ใช้กับทุก ๆ ฟิลด์ (Sequence field) เพื่ออธิบายฟิลด์ต่าง ๆ ของเซ็กเม้นท์ โดยเฉพาะเป็นเซ็กเม้นท์ที่โปรแกรมคำสั่งงานใช้ "IMS CALL Statement Segment Search Argument (SSA)" ในการสร้าง ดีบีดี ครั้งหนึ่ง ๆ มีการเขียนประโยคนี้ได้ไม่เกิน ๑๐๐๐ และไม่เกิน ๒๕๕ ประโยคต่อ ๑ เซ็กเม้นท์

โครงร่าง

	FIELD	NAME=(fld-name1[,SEQ[{,U}]]) {,M} ,BYTES=bytes ,START=pos [,TYPE={X} {P} {C}]

๔. ประโยค LCHILD

ประโยคนี้ใช้ ๑ ครั้ง สำหรับแต่ละเซ็กเม้นท์ที่มีความสัมพันธ์เชิงดัชนี หรือความสัมพันธ์เชิงตรรก ซึ่งแยกได้ ดังนี้

๔.๑ สำหรับความสัมพันธ์เชิงดัชนีเฉพาะการจัดแฟ้มข้อมูลแบบดัชนี, แบบเข้าถึงโดยตรงตามลำดับขั้น (เอสแดม) และแบบดัชนีเข้าถึงโดยตรงตามลำดับขั้น (ไฮแดม) จะเป็นประโยคที่ใช้อธิบายชื่อของเซ็กเม้นท์ที่เป็น "Index target" หรือเซ็กเม้นท์ของดัชนีฐานข้อมูล (Index data base)

โครงร่าง ของประโยค LCHILD สำหรับดัชนีปฐมภูมิ เมื่อใช้ฐานข้อมูลแบบดัชนีเข้าถึงโดยตรงตามลำดับขั้น หรือสำหรับดัชนีทุติยภูมิ เมื่อใช้ฐานข้อมูลแบบเข้าถึงโดยตรงหรือแบบดัชนีเข้าถึงโดยตรง

	LCHILD	NAME=(seg-name1,db-name) ,POINTER=INDX

โครงร่าง ของประโยค LCHILD เมื่อใช้ฐานข้อมูลแบบดัชนี

	LCHILD	NAME=(seg-name1,db-name) [,POINTER=SNGL] ,INDEX={fld-name } {xdfld-name}

๕.๒ สำหรับความสัมพันธ์เชิงตรรกจะใช้ประโยคนี้อธิบายความสัมพันธ์ระหว่างพาเร้นท์เชิงตรรก (Logical parent Segment) กับ ชายด์เซ็กเม้นท์-เชิงตรรก (Logical child segment) เฉพาะการจัดแฟ้มข้อมูลแบบเข้าถึงโดยตรงตามลำดับขั้น และแบบดัชนีเข้าถึงโดยตรงตามลำดับขั้น

โครงร่าง

	LCHILD	NAME=(seg-name1[,db-name]) {NONE} [,POINTER={SNGL}] {DBLE} [,PAIR=seg-name2] {FIRST} [,RULES={LAST }] {HERE }

๖. ประโยค XDFLD

ประโยคนี้ใช้เฉพาะกรณีความสัมพันธ์แบบดัชนีทุติยภูมิ สำหรับ ดีบีดี ที่มีวิธีการเข้าถึงข้อมูลเป็นแบบเข้าถึงโดยตรงตามลำดับขั้น หรือแบบดัชนีเข้าถึงโดยตรงตามลำดับขั้นจะใช้ประโยชน์เพียงครั้งเดียว เพื่ออธิบายถึงความสัมพันธ์แบบดัชนีทุติยภูมิ และประโยคนี้จะตามหลังประโยค LCHILD ที่อธิบายเซ็กเม้นท์นั้นว่าตัวบ่งชี้ (Pointer) เป็น "INDEX" และทั้งประโยค LCHILD และ XDFLD ต้องตามหลังประโยค FIELD ที่อธิบายถึงฟิลด์ (Sequence field) ของเซ็กเม้นท์นั้น ประโยค XDFLD มีได้มากที่สุด ๓๒ ครั้ง ใน ๑ เซ็กเม้นท์ และประโยค XDFLD รวมกับประโยค FIELD ต้องไม่เกิน ๒๕๕ ประโยคต่อ ๑ เซ็กเม้นท์ และไม่เกิน ๑๐๐๐ ประโยคต่อการสร้าง ดีบีดี ครั้งหนึ่ง ๆ

โครงร่าง

	XDFLD	NAME=xdfld-name [,SEGMENT=iss-name] ,SRCH=list1 [,SUBSEQ=list2] [,DDATA=list3] [,NULLVAL=value1] [,EXTRIN=mod-name]

๗. ประโยค DBDGEN

เป็นประโยคสุดท้ายของการสร้าง ดีบีดี และมีเพียงประโยคเดียว

โครงร่าง

	DBDGEN	

๔. ประโยค FINISH

เป็นประโยคที่ใช้บอกสำหรับ "Source-level compatability กับ IMS/VS" ประโยคนี้มีเพียงประโยคเดียว

โครงร่าง

	FINISH	

๙. ประโยค END

เป็นประโยคที่บอกว่าจบประโยคที่เป็นอินพุทของโปรแกรมระบบคำสั่งเครื่อง (OS/VS ASSEMBLER) และมีเพียงประโยคเดียว

โครงร่าง

	END	

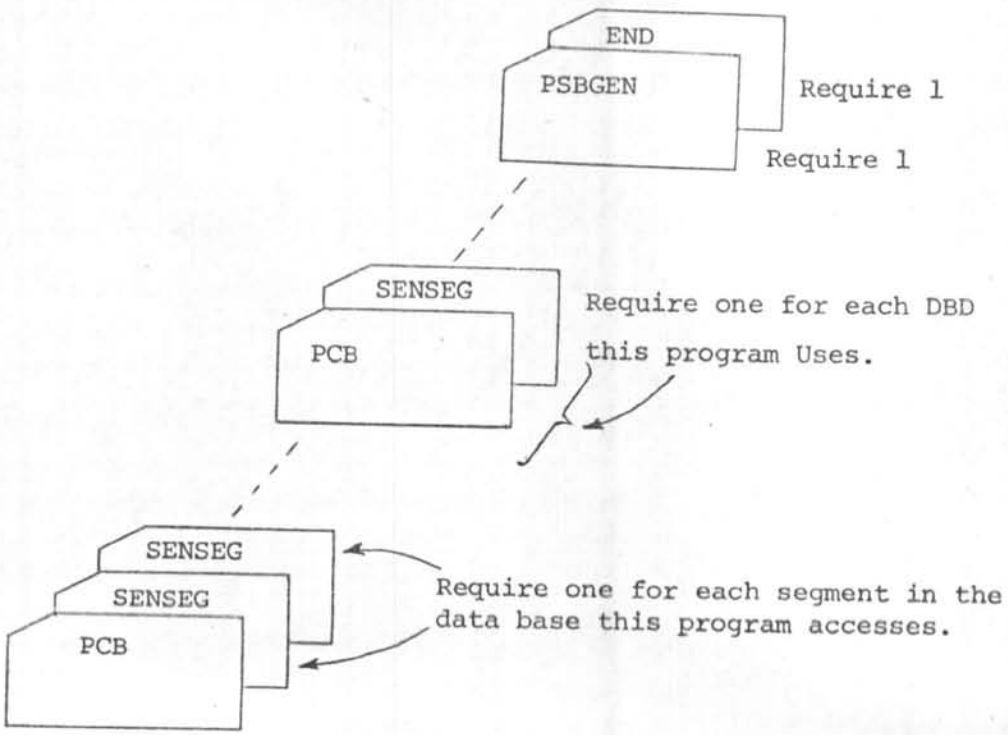
ชุดประโยคคำสั่งควบคุมเมื่อทำการสร้าง ดีบีดี (JCL) เรียงตามลำดับ ดังนี้

```
// JOB     DBDGEN
// OPTION CATAL
// EXEC    ASSEMBLY
          :   }
          :   }  DBD GENERATION CONTROL STATEMENT
          :   }
/*
//    EXEC LNKEDT
/&
```

๓.๓.๒.๒ โปรแกรมย่อยระดับที่สองเรียกว่า "PSB" หรือ "Program Specification Block"

ประกอบด้วย กลุ่มประโยคของ ไอเอ็มเอส ประเภท "Macro Instruction" ที่ผู้ใช้เขียนขึ้นเพื่อกำหนดลักษณะของฐานข้อมูล (จากโครงสร้างฐานข้อมูลทางกายภาพที่มีอยู่แล้ว สำหรับใช้ในแต่ละโปรแกรมคำสั่งงาน)

กลุ่มประโยคในการสร้าง พีเอสบี จะถูกนำไปเก็บไว้ในที่เก็บโปรแกรมก่อนการแปลงเป็นภาษาเครื่อง เมื่อถูกเรียกมาทำงานแล้วจะสร้างเป็น "PSB CSECT" แล้วถูกเก็บและนำไปรวมไว้ในที่เก็บโปรแกรมคำสั่งเครื่อง เพื่อการทำงานต่อไป



รูปที่ ๓.๓.๕ โครงสร้างและลำดับของกลุ่มประโยคที่ใช้ในการสร้าง พีเอสบี

๑. ประโยค PCB

ประโยคนี้ใช้สำหรับแต่ละฐานข้อมูลที่โปรแกรมคำสั่งงานต้องการ

โครงร่าง

	PCB	TYPE=DB ,DBNAME=name ,PROCOPT={{[G][I][R][D] }[P][E] } {A } {L[S] } {GO[P] } ,KEYLEN=value {MULTIPLE} [,POS={SINGLE }] [,PROCSEQ=index-db-name]

๒. ประโยค SENSEG

ประโยคนี้ใช้สำหรับแต่ละ เซ็กเม้นท์และประโยคนี้จะต้องปรากฏในลักษณะ "Hierarchical Sequence" เหมือนในการสร้าง ดีบีดี

โครงร่าง

	SENSEG	NAME=name, [PARENT={name}] {0 } [,PROCOPT={[G][I][R][D]}[P][E] {A }

๓. ประโยค PSBGEN

เป็นประโยคสุดท้ายของการสร้าง พีเอสบี และมีเพียงประโยคเดียว

โครงร่าง

	PSBGEN	LANG={COBOL / IMS / ASSEM / RPG} ,PSBNAME=name

๔. ประโยค END

เป็นประโยคที่บอกว่าจบประโยคที่เป็นอินพุทของโปรแกรมระบบคำสั่งเครื่อง OS/VS ASSEMBLER และมีเพียงประโยคเดียว

โครงร่าง

	END	

ชุดประโยคคำสั่งควบคุมเมื่อทำการสร้าง พีเอสบี (JCL) เรียงตามลำดับ ดังนี้

```
//  JOB     PSBGEN
///  OPTION CATAL
 //  EXEC    ASSEMBLY
                    :
                    :   } PSBGEN Control Statement
                    :
/*
//  EXEC    LNKEDT
/&
```

๓.๓.๒ ตัวอย่างการสร้างดีบีดีของไอเอ็มเอส โดยใช้โครงสร้างในรูปที่ ๓.๓.๒ หน้า 62 ๕๕

```
IMS

DBD       NAME = ORGDATA

SEGM      NAME = ORG, BYTES = 38, START = 1
FIELD     NAME = (ORGCODE, SEQ, U), BYTES = 2, TYPE = P
FIELD     NAME = ORGNAME, BYTES = 25, START = 3, TYPE = C
FIELD     NAME = REPORTO, BYTES = 2, START = 28, TYPE = P
FIELD     NAME = BUDGET, BYTES = 4, START = 30, TYPE = P

SEGM      NAME = JOB, BYTES = 8, FREQ = 31, PARENT = ORG
FIELD     NAME = (JOBCODE, SEQ, U), BYTES = 2, TYPE = P

SEGM      NAME = SUBORG, BYTES = 2, PARENT = ORG
FIELD     NAME = (SUBCODE, SEQ, U), BYTES = 2, TYPE = P

SEGM      NAME = PERSON, BYTES = 40, PARENT = ORG
FIELD     NAME = (EMPJCODE, SEQ, M), BYTES = 2, TYPE = P
FIELD     NAME = (EMPNO, SEQ, U), BYTES = 3, TYPE = P

SEGM      NAME = SKILLS, BYTES = 3, PARENT = PERSON
FIELD     NAME = (SKILCODE, SEQ, U), BYTES = 2, TYPE = P

DBDGEN

FINISH

END
```

๓.๓.๓ ดีดีแอลของโททอล

๓.๓.๓.๑ DATA BASE DEFINITION (DBD)

หน้าที่โดยสรุปคือ ระบุพารามิเตอร์ต่าง ๆ ของฐานข้อมูลสำหรับการจัดฐานข้อมูลแบบโททอล ภาษาที่ใช้ในการทำงานของขั้นตอนนี้คือ ดีบีดีแอล (Data Base Definition Language) ซึ่งมีกฎเกณฑ์และโครงร่างโดยย่อ ๆ ดังนี้

กฎเกณฑ์และสัญญลักษณ์ที่ใช้ในโครงร่างของภาษา ดีบีดีแอล

๑. ทุกคำสั่งต้องเริ่มที่สดมภ์ที่หนึ่งของบัตรเจาะรู

๒. การเว้นช่องว่างไว้เป็นการจบคำสั่งแต่ละคำสั่ง

๓. ตัวอักษรภาษาอังกฤษที่เป็นตัวพิมพ์ใหญ่ และพวกเครื่องหมายวรรคตอนต่าง ๆ ต้องเขียนตามที่ได้แสดงเอาไว้

๔. เครื่องหมายวงเล็บใหญ่[ ] ใช้สำหรับแสดงว่าคำสั่งใดที่อยู่ภายในเครื่องหมายวงเล็บใหญ่นี้จะเขียนลงไปก็ได้หรือไม่เขียนก็ได้

๕. เครื่องหมาย "mmmm" หมายความถึง ให้แทนที่ได้ด้วยชื่อของ "Single entry data set"

๖. เครื่องหมาย "vvvv" หมายความถึง ให้แทนที่ได้ด้วยชื่อของ "Variable entry data set"

๗. เครื่องหมาย "xxxx", "kkkk", "ssss" แทนที่ได้ด้วยตัวอักษร A ถึง Z เลข 0 ถึง9 และสัญญลักษณ์พิเศษ # , $, @ นี้เท่านั้น

๘. "n" หมายถึง แทนที่ด้วยตัวเลข

๑. ประโยคซึ่งใช้เป็นประโยคนำในการทำ ดีบีเจน (Prologue Statements)

มีอยู่ ๕ ชนิด เรียงตามลำดับ ดังนี้

๑.๑ ประโยคแรกที่ใช้สำหรับการทำ ดีบีเจน มีโครงร่าง ดังนี้

```
BEGIN-DATA-BASE-GENERATION:
```

๑.๒ ประโยคที่สองใช้สำหรับบอกชื่อของฐานข้อมูลซึ่งจะเป็นอักษรปนตัวเลขไม่เกิน ๖ ตัว มีโครงร่าง ดังนี้

```
DATA-BASE-NAME = XXXXXX
```

๑.๓ ประโยคที่สามบอกลักษณะงาน มีโครงร่าง ดังนี้

```
VERSION = BATCH
```

๑.๔ ประโยคที่สี่ใช้ร่วมกับประโยคที่ห้าซึ่งใช้บ่งชื่อของพื้นที่ซึ่งเป็น อินพุท/เอ้าพุท มีโครงร่าง คือ

```
SHARE - IO:
```

๑.๕ ประโยคที่ห้าใช้บอกชื่อของพื้นที่ซึ่งเป็น อินพุท/เอ้าพุท ซึ่งประโยคนี้มีได้หลายครั้ง (ถ้าไม่มีการกำหนดชื่อจะมีได้ไม่เกิน ๑๖ พื้นที่ซึ่งเป็นชื่อของอินพุท/เอ้าพุท) โครงร่าง คือ

```
IOAREA = XXXX
```

๒. ประโยคที่ใช้อธิบายเกี่ยวกับแฟ้มข้อมูลหลัก (Master Data Set Statement)

๒.๑ ประโยคแรกที่ใช้บอกว่าต่อไปนี้จะเป็นการอธิบายถึงแฟ้มข้อมูลหลักมีโครงร่าง คือ

```
BEGIN-MASTER-DATA-SET:
```

๒.๒ ประโยคสองจะใช้สำหรับบ่งบอกชื่อของแฟ้มข้อมูล มีโครงร่างดังนี้

```
DATA-SET-NAME = mmmm
```

๒.๓ ประโยคสามใช้บอกชื่อของพื้นที่ อินพุท/เอ้าพุท ที่แฟ้มข้อมูลหลักใช้อยู่ และชื่อของพื้นที่ อินพุท/เอ้าพุท ต้องตรงกับที่ได้เขียนไว้ในประโยคตอนต้น ๆ (SHARE-IO:) โครงร่างของประโยค คือ

```
IOAREA = XXXX
```

๒.๔ ประโยคสี่เป็นประโยคแรกก่อนที่จะเริ่มอธิบายถึงฟิลด์ต่าง ๆ ในระเบียนข้อมูลหลัก โครงร่าง คือ

```
MASTER-DATA:
```

๒.๕ ประโยค ROOT ใช้สำหรับบ่งชื่อของฟิลด์ที่เป็นของระบบเครื่องและกำหนดความยาว ไว้เท่ากับ ๘ มีโครงร่าง ดังนี้ (kkkk ควรเป็นชื่อของแฟ้มข้อมูลหลัก)

```
kkkkROOT = 8
```

๒.๖ ประโยค CONTROL KEY ใช้บอกชื่อของฟิลด์ที่เป็นคีย์ของระเบียนข้อมูลหลัก

kkkk เป็นชื่อของฟิลด์ที่เป็นคีย์ของระเบียนข้อมูลหลัก

n เป็นความยาวของคีย์

p เป็นเลขลำดับ

มีโครงร่าง คือ

```
[.p.] kkkk CTRL = n
```

๒.๗ ประโยค LINK ใช้สำหรับบ่งบอกเส้นทางเชื่อมต่อ โดยที่

mmmm เป็นชื่อของแฟ้มระเบียนข้อมูลหลัก

xx เป็นรหัสที่ใช้เชื่อมต่อ

8 เป็นความยาวคงที่

มีโครงร่าง ดังนี้

```
mmmm LKXX = 8
```

๒.๘ ประโยคนี้มีได้ตามจำนวนชื่อฟิลด์ที่เป็นข้อมูลของระเบียนข้อมูล มีโครงร่างดังนี้

```
[.p.] ssssxxxx [= n]
```

๒.๙ ประโยค END-DATA : มีไว้เพื่อบอกให้ทราบว่าหมดฟิลด์ที่เป็นข้อมูลแล้ว มีโครงร่าง ดังนี้

```
END-DATA :
```

๒.๑๐ ประโยคชนิดที่สิบ ใช้บอกว่าสื่อกลางในการเก็บข้อมูลเป็นอย่างไร ถ้าไม่มีประโยคนี้เครื่องจะใช้ตัว ๒๓๑๔ เป็นสื่อกลาง ซึ่งมีโครงร่างดังนี้

```
DEVICE = { TELEX
         { 2311
         { 2314
         { 3330
```

๒.๑๑ ประโยคที่ใช้บอกว่าแฟ้มข้อมูลนี้ เก็บข้อมูลได้เท่าใด (n ต้องมากกว่า ๓) มีโครงร่าง คือ

```
TOTAL-LOGICAL-RECORDS = n
```

๒.๑๒ ประโยคที่สิบสองใช้บอกจำนวนแทรกซ์ทั้งหมดที่ใช้สำหรับแฟ้มข้อมูลนี้มีโครงร่าง คือ

```
TOTAL-TRACKS = n
```

๒.๑๓ ประโยคที่ใช้บอกความยาวทั้งหมดของระเบียนข้อมูล (ซึ่งจะรวมทั้งความยาวของคีย์, ตัวเชื่อมต่อ, ข้อมูล, ROOT) มีโครงร่างดังนี้

```
LOGICAL-RECORD-LENGTH = n
```

๒.๑๔ ประโยคที่สิบสี่ ใช้บอกจำนวนบล๊อกซ์ต่อหนึ่งแทรกซ์ มีโครงร่างดังนี้

```
LOGICAL-BLOCKS-PER-TRACK = n
```

๒.๑๕ ประโยคที่สิบ ใช้บอกจำนวนสื่อกลางการเก็บข้อมูลที่สามารถขยายไปได้ (ปกติ ๑, สูงสุด ๑๖) สำหรับเก็บแฟ้มข้อมูลนี้ มีโครงร่างคือ

```
DISK-EXTENTS = n
```

๒.๑๖ ประโยคที่สิบหกจะใช้ต่อเมื่อมีการเปลี่ยนแปลงการใช้ไททอล-๔ มาเป็น ไททอล-๗ คือใช้แฟ้มข้อมูลเก่าได้มีโครงร่าง ดังนี้

```
OLD-FILE = YES
```

๒.๑๗ ประโยคสุดท้ายสำหรับบอกว่าจบการอธิบายแฟ้มข้อมูลหลักชุดนี้แล้ว มีโครงร่างคือ

```
END-MASTER-DATA-SET:
```

๓. ประโยคที่ใช้อธิบายแฟ้มข้อมูลแปรเปลี่ยน (Variable Data Set Statement)

๓.๑ ประโยคแรกใช้บ่งบอกว่าต่อไปนี้จะเป็นการอธิบายถึงแฟ้มข้อมูลแปรเปลี่ยน ซึ่งมีโครงร่าง ดังนี้

```
BEGIN-VARIABLE-ENTRY-DATA-SET:
```


๓.๒ ประโยคที่สองใช้บอกชื่อของแฟ้มข้อมูล มีโครงร่างดังนี้

```
DATA-SET-NAME = VVVV
```

๓.๓ ประโยคที่สามใช้บอกชื่อพื้นที่ของ อินพุท/เอ้าพุท สำหรับแฟ้มข้อมูลอันนี้ มีโครงร่างคือ

```
IOAREA = XXXX
```

๓.๔ ประโยคที่สี่ใช้แสดงว่าต่อไปนี้จะเป็นส่วนของระเบียนข้อมูลที่เป็นชนิด "BASE DATA" ซึ่งมีโครงร่าง คือ

```
BASE-DATA:
```

๓.๕ ประโยคที่ห้าบอกถึงชื่อของส่วนที่เป็นรหัสของระเบียนข้อมูล ซึ่งมีความยาว ๒ ตัวอักษร คงที่มีโครงร่าง คือ

```
VVVVCODE = 2
```

๓.๖ ประโยคที่หกใช้อธิบายแต่ละฟิลด์ที่เป็นข้อมูลภายในระเบียนข้อมูลมีโครงร่าง คือ

```
[.P.] SSSSXXXX [ = n]
```

๓.๗ ประโยคชนิดที่เจ็ดเป็นประโยคสำหรับบอกถึงชื่อเส้นทางเชื่อมต่อที่ใช้เชื่อมกับแฟ้มข้อมูลหลักมีโครงร่าง คือ

```
MMMMLKxx = 8
```

๓.๘ ประโยคชนิดที่แปดใช้สำหรับบอกชื่อรหัสของระเบียนข้อมูลมีความยาว ๒ ตัวอักษร เป็นตัวเลขก็ได้มีโครงร่าง ดังนี้

```
RECORD-CODE = XX
```

๓.๙ ประโยคที่บอกว่าจบการอธิบายฟิลด์ข้อมูลคือ ประโยคที่มีโครงร่าง ดังนี้

```
END-DATA:
```

๓.๑๐ ประโยคต่าง ๆ ต่อไปก็เหมือนกับที่ใช้ตอนท้ายของแฟ้มข้อมูลหลัก เพื่ออธิบายถึงสื่อเก็บข้อมูลทางกายภาพต่าง ๆ มีโครงร่าง ดังนี้

```
DEVICE = { TELEXDD
         { 2311
         { 2314
         { 3330
TOTAL-LOGICAL-RECORD = n
TOTAL-TRACKS = n
LOGICAL-RECORD-LENGTH = n
LOGICAL-BLOCKS-PER-TRACK = n
DISK-EXTENTS = n
```

๓.๑๑ ประโยคพิเศษอีกชนิดก็คือ ประโยคที่ใช้สำหรับระบุจำนวนเปอร์เซ็นต์ซึ่งใช้ในการจัดการเกี่ยวกับพื้นที่ในสื่อเก็บข้อมูลมีโครงร่าง คือ (n ตามปกติควรเป็น ๘๐ แต่เราสามารถกำหนดได้จาก ๐ ถึง ๑๐๐)

```
CYLINDER-LOAD-LIMIT = n
```

๓.๑๒ ประโยคที่บอกว่าจบการอธิบายเกี่ยวกับแฟ้มข้อมูลนี้แล้วคือ ประโยคที่มีโครงร่าง

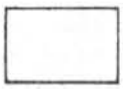
```
END-VARIABLE-ENTRY-DATA-SET:
```

๔. ประโยคสุดท้ายเพื่ออธิบายว่าจบโปรแกรมคำสั่ง ดีบีเจน มีโครงร่าง คือ

```
END-DATA-BASE-GENERATION:
```

๓.๓.๓.๒ ตัวอย่างการใช้ดีดีแอลเพื่อทำดีบีเจ็น

สมมติว่าได้ออกแบบโครงสร้าง และความสัมพันธ์ของแฟ้มข้อมูลในระบบหนึ่ง คือ บริษัทซื้อสินค้ามาจากโรงงานทีละมาก ๆ แล้วขายส่งให้พ่อค้ารายย่อย โดยมีลักษณะโครงสร้างฐานข้อมูลตามรูปที่ ๓.๓.๖, ๓.๓.๗ จากนั้นเป็นดีดีแอลของโครงสร้างนี้ ซึ่งเรียกว่า การทำดีบีเจน



-----Master File

--------Variable File

CUST ----------Customers

VEND ----------Vendors

INVT ----------Inventory Item

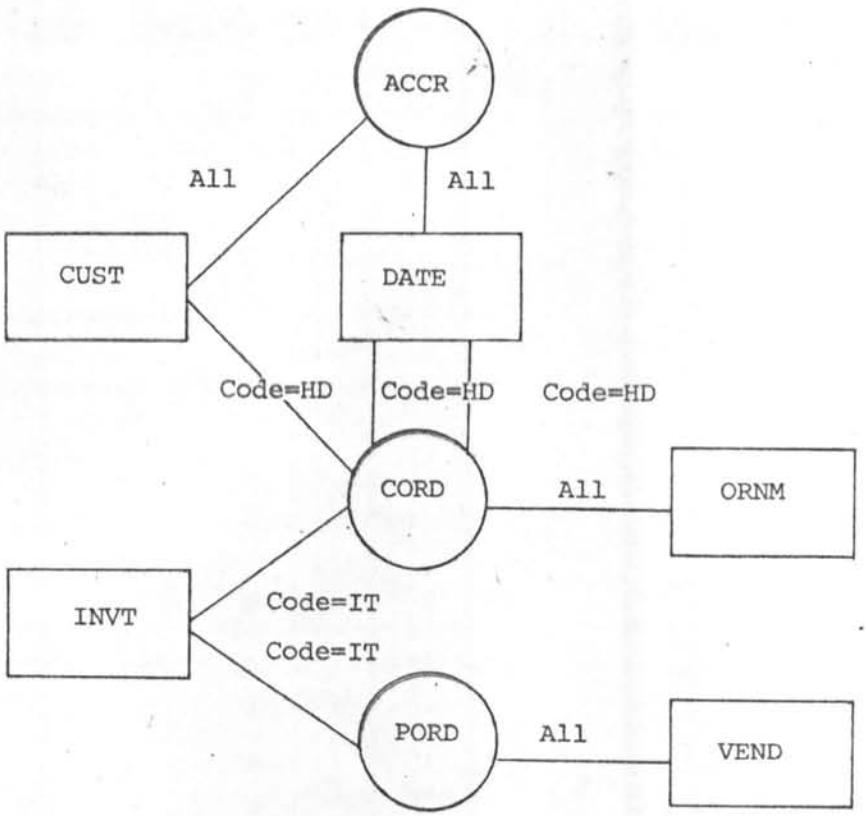
ORNM ----------Order Number

DATE ----------Date

CORD --------Customer Order

PORD --------Purchase Order

ACCR --------Account recievable



รูปที่ ๓.๓.๖. แสดงโครงสร้างฐานข้อมูลในลักษณะสกีม่า

CUSTOMER

data field name	Customer number	Name	Address	City state
byte	6	30	30	20
	key			

ORDER NUMBER

Order number
12
key

VENDOR

Vender number	Name	Address	City state	Load time
7	25	30	30	5
key				

DATE

Date
4
key

INVENTORY ITEM

Item number	Description	Cost	Price	Quantity on hand	Quantity on order
8	30	5	5	4	4
key					

PURCHASE ORDER

Record code	Vender number	Order number	Line number	Date released	Date due	Carrier
HD	7	5	2	6	6	15
	key					

Record code	Vender number	Order number	Line number	Item number	Quantity	Unused bytes
IT	7	5	2	8	5	14
	key			key		

รูปที่ ๓.๓.๗ แสดงฟอร์มระเบียบต่าง ๆ ของแฟ้มข้อมูล

CUSTOMER ORDER

Record code	Order number	Line number	Customer number	Date recieved	Date to ship	Total value	Total item	Terms	Unused
HD	12	2	6	4	4	10	2	10	19
	-- key-		- key- -	-key- -	- key-				

Record code	Order number	Line number	Item number	Quantity	Price	Weight	Unused
IT	12	2	8	4	5	4	34
	- - key-		- -key -				

Record code	Order number	Line number	Comment
CM	12	2	55
	- - key-		

ACCOUNT RECEIVABLE

Record code	Customer number	Sequence number	Invoice number	Due date net	Due date gross	Net amount	Gross amount	Amount paid
BL	6	2	6	6	6	5	5	5
	- -key - -							

Record code	Customer number	Sequence number	Invoice number	Check number	Date received	Check amount	Unused
CK	6	2	6	6	6	5	10
	- -key - -						

รูปที่ ๓.๓.๗.(ต่อ) แสดงฟอร์มระเบียบต่าง ๆ ของแฟ้มข้อมูล

```
BEGIN-DATA-BASE-GENERATION:
DATA-BASE-NAME=BASE01
SHARE-IO1
IOAREA=MASI
IOAREA=VARI

BEGIN-MASTER-DATA-SET1
DATA-SET-NAME=CUST
IOAREA=MAS1
MASTER-DATA:
CUSTROOT=8
CUSTCTRL=6
CUSTLRCO=8                    (LINK TO CUSTOMER ORDER HEADER RECORDS)
CUSTLKAR=8                    (LINK TO INVOICES AND PAYMENTS)
CUSTDAT1=80
.1.CUSTNAME=30
.1.CUSTADDR=30
.1.CUSTCTYS=20
END-DATA:
DEVICE=2314
TOTAL-TRACKS=20
LOGICAL-BLOCKS-PER-TRACK=4
END-MASTER-DATA-SET:

BEGIN-MASTER-DATA-SET:
DATA-SET-NAME=VEND
IOAREA=MAS1
MASTER-DATA:
VENDROOT=8
VENDCTRL=7
VENDLKPO=8                    (LINK TO OPEN PURCHASE ORDER)
VENDDAT1=90
.1.VENDNAME=25
.1.VENDADDR=30
.1.VENDCTYS=30
.1.VENDLTME=5
END-DATA:
DEVICE=2314
TOTAL-TRACKS=20
LOGICAL-BLOCKS-PER-TRACK=4
END-MASTER-DATA-SET:

BEGIN-MASTER-DATA-SET:
DATA-SET-NAME=INVT
MASTER DATA:

                              *NOTE*  PRIVATE IOAREA GENERATED.
```

```
INVTROOT=8
INVTCTRL=8
INVTLKCO=8                          (LINK TO OPEN CUSTOMER ITERMS)
INVTLKPI=8                          (LINK TO OPEN VENDOR ITEMS)
INVTDAT1=40
.1. INVTDESC=30
.1. INVTCOST=5
.1. INVTPRIC=5
INVTDAT2=8
.1. INVTONHD=4
.1. INVTORDR=4
END-DATA:
DEVICE=2314
TOTAL-TRACKS=20
LOGICAL-BLOCKS-PER-TRACK=4
END-MASTER-DATA-SET:

BEGIN-MASTER-DATA-SET:
DATA-SET-NAME=ORNM
MASTER-DATA:
                                    *NOTE*  PRIVATE IOAREA GENERATED.
ORNMROOT=8
ORNMCTRL=12
ORNMLKCO=8                          (LINK TO ORDER RECORDS)
END-DATA:
DEVICE=2314
TOTAL-TRACKS=20
LOGICAL-BLOCKS-PER-TRACK=4
END-MASTER-DATA-SET:

BEGIN-MASTER-DATA-SET:
DATA-SET-NAME=DATE
MASTER-DATA:
                                    *NOTE*  PRIVATE IOAREA GENERATED.
DATEROOT=8
DATECTRL=4    YYMMDD PACKED
DATELKAR=8
DATELKDR=8                          (LINK TO DATE RECEIVED)
DATELKDS=8                          (LINK TO DATE TO SHIP)
END-DATA:
DEVICE=2314
TOTAL-TRACKS=20
LOGICAL-BLOCKS-PER-TRACK=4
END-MASTER-DATA-SET:
```

```
BEGIN-VARIABLE-ENTRY-DATA-SET:
DATA-SET-NAME=PORD
IOAREA=VARI
BASE-DATA:
PORDCODE=2
PORDVEND=7                          (BASE VENDOR CONTROL)
VENDLKPO=8
PORDPNUM=5
PORDLINE=2
PORDDAT1=27
RECORD-CODE=HD
.1.PORDIDAT=6
.1.PORDDUDT=6
.1.PORDCARR=15
RECORD-CODE=IT
.1.PORUITEM=8
INVTLKPI=8
.1.PORDIQTY=5
END-DATA:
                                    *NOTE  SPECIFIED LENGTH EXCEEDS
                                           CALCULATED.
DEVICE=2314
TOTAL-LOGICAL-RECORDS=37122
LOGICAL-RECORD-LENGTH=51
LOGICAL-BLOCKS-PER-TRACK=2
END-VARIABLE-ENTRY-DATA-SET:

BEGIN-VARIABLE-ENTRY-DATA-SET:
DATA-SET-NAME=CORD
BASE-DATA:
                                    *NOTE*  PRIVATE IOAREA GENERATED.
CORDCODE=2
CORDORNM=12                         (LINK TO ORDER NUMBER)
ORNMLKCO=8
CORDLINE=2
CORDDATA=55
RECORD-CODE=HD
.1.CORDCUST=6
CUSTLKCO=8                          (LINK TO CUSTOMER NUMBER)
.1.CORDDATR=4
DATELKDR=8                          (LINK TO DATE ORDER RECEIVED)
.1.CORDDATS=4
DATELKDS=8                          (LINK TO DATE TO SHIP)
.1.CORDVALE=5   TOTAL VALUE OF THE ORDER S9(07)V99 PACKED
.1.CORDTOTI=2   TOTAL NUMBER OF ITEMS    S999      PACKED
.1.CORDTERM=10  TERMS OF THE ORDER
RECORD-CODE=IT
.1.CORDITEM=8
INVTLKCO=8
```

```
.1.CORDQTYX=4
.1.CORDIRCE=5
.1.CORDSLBS=4
RECORD-CODE=CM
                                        *NOTE*  SPECIFIED LENGTH EXCEEDS
.1.CORDCOMT=55                                  CALCULATED.
END-DATA:
DEVICE=2314
TOTAL-LOGICAL-RECORDS=73508
LOGICAL-RECORD-LENGTH=79
LOGICAL-BLOCKS-PER-TRACK=1
END-VARIABLE-ENTRY-DATA-SET:

BEGIN-VARIABLE-ENTRY-DATA-SET:
DATA-SET-NAME=ACCR
IOAREA=VARI
BASE-DATA:
ACCRCODE=2
ACCRCUST=6                              (BASE CUSTOMER CONTROL)
CUSTLKAR=8
.1.ACCRNUMB=4
DATELKAR=8
.1.ACCRSEXC=2
ACCRSEQS=2
ACCRDAT1=33
RECORD-CODE=BL
.1.ACCRINUM=6
.1.ACCRNDAT=6
.1.ACCRGDAT=6
.1.ACCUNANT=5
.1.ACCRGAMT=5
.1.ACCRPAID=5
RECORD-CODE=CK
.1.ACCRRINO=6
.1.ACCRCHKN=6
.1.ACCRDREC=6
.1.ACCRCAMT=5
END-DATA:
                                        *NOTE*  SPECIFIED LENGTH EXCEEDS
DEVICE=2314                                     -CALCULATED.
TOTAL-LOGICAL-RECORDS=71626
LOGICAL-RECORD-LENGTH=59
LOGICAL BLOCKS PER TRACK=2
END-VARIABLE-ENTRY-DATA-SET:

END-DATA-BASE-GENERATION:
```

๓.๔ การเปรียบเทียบดีเอ็มแอล

๓.๔.๑ ดีเอ็มแอลของดีบีทีจี

๓.๔.๑.๑ คำสั่งในดีเอ็มแอล

a. OPEN : CLOSE

b. INSERT : REMOVE

c. STORE : DELETE

d. KEEP : FREE

e. FIND

f. GET

g. MODIFY

h. MOVE

i. ORDER

ในระบบความผิดพลาดและเงื่อนไขข้อยกเว้น (exception conditions) ซึ่งติดต่อกับผู้ใช้โปรแกรม จะส่งผลออกมายังที่เก็บค่าพิเศษ ลักษณะเป็น ๖ ชนิด เพื่อแสดงเงื่อนไขความผิดพลาด

```
ERROR-STATUS        PICTURE IS 9999.
ERROR-AREA          PICTURE IS X(30).
ERROR-SET           PICTURE IS X(30).
ERROR-RECORD        PICTURE IS X(30).
ERROR-TYPE          PICTURE IS 99.
ERROR-COUNT         PICTURE IS 99.
```

ERROR-STATUS : เป็นรหัสแสดงชนิดของความผิดพลาดที่พบเริ่มแรก (ตัวเลข ๒ ตัวแรก) และคำสั่งในดีเอ็มแอลที่เกิดขึ้นขณะนั้น (ตัวเลข ๒ ตัวหลัง)

ERROR-SET : ชื่อของเซทที่เกิดความผิดพลาด

ERROR-RECORD : ชื่อของระเบียนข้อมูลที่เกิดความผิดพลาด

ERROR-AREA ชื่อแอเรียที่เกิดความผิดพลาด

ERROR-TYPE แสดงชนิดของการกระทำสลับกันโดยรัน ยูนิท (run unit) พร้อม ๆ กัน ในกรณีที่มีหลาย ๆ รัน ยูนิต

ERROR-COUNT จำนวนความผิดพลาดทั้งหมดที่ระบบพยายามกระทำต่อคำสั่งในดีเอ็มแอล และเมื่อเกิดความผิดพลาด สามารถใช้คำสั่งยูส (USE statement) ระบุขั้นตอนย่อยในโปรแกรมคำสั่งที่จะทำงาน (กำหนดไว้หลังคำสั่งยูส เมื่อรหัสเงื่อนไขความผิดพลาดที่ระบุไว้นั้นเกิดขึ้น

๓.๔.๑.๒ ลักษณะการใช้ ดีเอ็มแอล

IDENTIFICATION DIVISION

Privacy key entry (เขียนหลัง Program-id)

[Privacy compile clause]

[Privacy key area clause] ....

[Privacy key record clause] ....

[Privacy key data clause] ....

[Privacy key set clause] ....

Privacy key compile clause

PRIVACY KEY FOR COMPILE IS { literal-1 | implementor-name-1 }

Privacy key area clause

PRIVACY KEY [FOR [[EXCLUSIVE | PROTECTED] RETRIEVAL | [EXCLUSIVE | PROTECTED] UPDATE]] OF

{ area-name-1 [, area-name-2] ... AREA | ALL AREAS } IS { PROCEDURE { Procedure-name-2 | literal-6 | identifier-6 } | litnal-2 | identifier-2 }

Privacy key record clause

```
PRIVACY KEY [ FOR [ INSERT | REMOVE | STORE | DELETE | DELETE ONLY | DELETE SELECTIVE | DELETE ALL | GET | MODIFY | FIND ] ] OF { record-name-1 [,record-name-2]... | RECORD | ALL RECORDS }
```

Privacy key data clause

```
IS { PROCEDURE { Procedure-name-3 | literal-7 | identifier-7 } | literal-3 | identifier-3 }
```

```
PRIVACY KEY [ FOR [ GET | MODIFY | STORE ] ]
OF data-base-identifier-1 [,data-base-identifier-2] ...
DATA-ITEM
IS { PROCEDURE { Procedure-name-4 | literal-8 | identifier-8 } | literal-4 | identifier-4 }
```

Privacy key set clause

```
PRIVACY KEY [ FOR [ ORDER | FIND | REMOVE | INSERT ] ] OF { Set-name-1 ,Set-name-2 ... | SET | ALL SETS }

IS { [ PROCEDURE { Procedure-name-5 | literal-9 | identifier-9 } ] | literal-5 | identifier-5 }
```

DATA DIVISION

Invoke clause

```
INVOKE SUB-SCHEMA sub-schema-name OF SCHEMA schema-name
```

PROCEDURE DIVISION

CLOSE

```
CLOSE { ALL [FOR SET set-name-1 ],set-name-2 ...] | AREA area-name-1 [,area-name-2]... }
```

DELETE

```
DELETE [record-name] [ ONLY | SELECTIVE | ALL ]
```

FIND

```
Record-
FIND Selection- [ SUPPRESS { ALL | RECORD | AREA | { SET | set-name-1 [,set-name-2]... } } CURRENT UPDATES ]
expression
```

Record-selection-expression

```
[record-name-1] USING identifier-1 [record-name-2 RECORD]
[OWNER IN set-name-3 OF] CURRENT OF [set-name-4 SET]
                                    [area-name-1 AREA]
                                    [RUN-UNIT]

[NEXT]
[PRIOR]
[FIRST]
[LAST]      [record-name-3] RECORD OF [set-name-5 SET]
[integer-1]                           [area-name-2 AREA]
[idenitifier-2]

OWNER RECORD OF set-name-6 SET
[NEXT DUPLICATE WITHIN] record-name-4 RECORD
record-name-5 VIA [CURRENT OF] Set-name-7
    [USING data-base-identifier-3 [,data-base-identifier-4]...]
NEXT DUPLICATE WITHIN set-name-8
    USING data-base-idenfifier-5 [,data-base-identifier-6]...
```

FREE

```
FREE [ALL]
```

GET

```
GET [record-name]
    record-name; data-base-identifier-1 [,data-base-identifier-2] ...
```

IF

```
IF set-name-1 SET [NOT] EMPTY; [statement-1      ] [;ELSE [statement 2]]
                               [NEXT SENTENCE    ]       [NEXT SENTENCE]

   RECORD [NOT] [MEMBER] OF [set-name-2] SET; [statement-3  ]
                [OWNER ]    [ANY     ]       [NEXT SENTENCE]

        [;ELSE [STATEMENT-4  ]]
               [NEXT SENTENCE]
```

INSERT

```
INSERT [record-name] INTO { set-name-1 [,set-name-2]...
                          { ALL SETS
```

KEEP

```
KEEP
```

MODIFY

```
MODIFY { [record-name] [USING data-base-identifier-1
       {   [,data-base-identifier-2] ....]
       { record-name; data-base-identifier-3
       {   [,data-base-identifier-4] ....
       {  [USING data-base-identifier-5
       {    [,data-base-identifier-6] ...]
```

MOVE

```
MOVE { CURRENCY STATUS FOR { RUN-UNIT
     {                     { record-name RECORD
     {                     { area-name AREA
     {                     { set-name SET
     {                                          } TO identifier-1
     {
     {            RUN-UNIT
     { AREA-NAME FOR record-name RECORD  TO identifier-3
     {             area-name AREA
     {             set-name SET
     {             identifier-2
```

OPEN

```
OPEN { ALL [FOR SET set-name-1 [,set-name-2] ....
     {     [USING-MODE IS [EXCLUSIVE] { RETRIEVAL } ]
     {                    [PROTECTED] { UPDATE    } ]
     {
     { AREA area-name-1 [,area-name-2] ... [USAGE-MODE IS
     {     [EXCLUSIVE] { RETRIEVAL } ]
     {     [PROTECTED] { UPDATE    } ]
```

```
REMOVE      REMOVE [record-name] { FROM set-name-1 [,set-name-2] ....
                                 { FROM ALL SETS

STORE       STORE record-name [ SUPPRESS { ALL                                          }
                              [          { || RECORD                                 || }
                              [          { || AREA                                   || }
                              [          { || { SETS                                 || }
                              [          { || { set-name-1 [,set-name-2] ... }       || }

                      CURRENCY UPDATES ]

USE         USE IF ERROR-STATUS [IS integer-1 [,integer-2] ...]

ORDER       ORDER set-name-1 SET [FOR RUN-UNIT]
               { { ASCEDING  } KEY IS || RECORD-NAME                                  }
               { { DECENDING }        || DATA BASE-KEY                                }
               {                      || data-base-identifier-1 [,data-base-          }
               {                      || identifier-2]                                }
               {                                                                      }
               { FOR record-name-1 { { ASCENDING  }     || DATABASE-KEY        ||     }
               {                   { { DECENDING  }     || data-base-          ||     }
               {                                        || identifier-3        ||     }
```

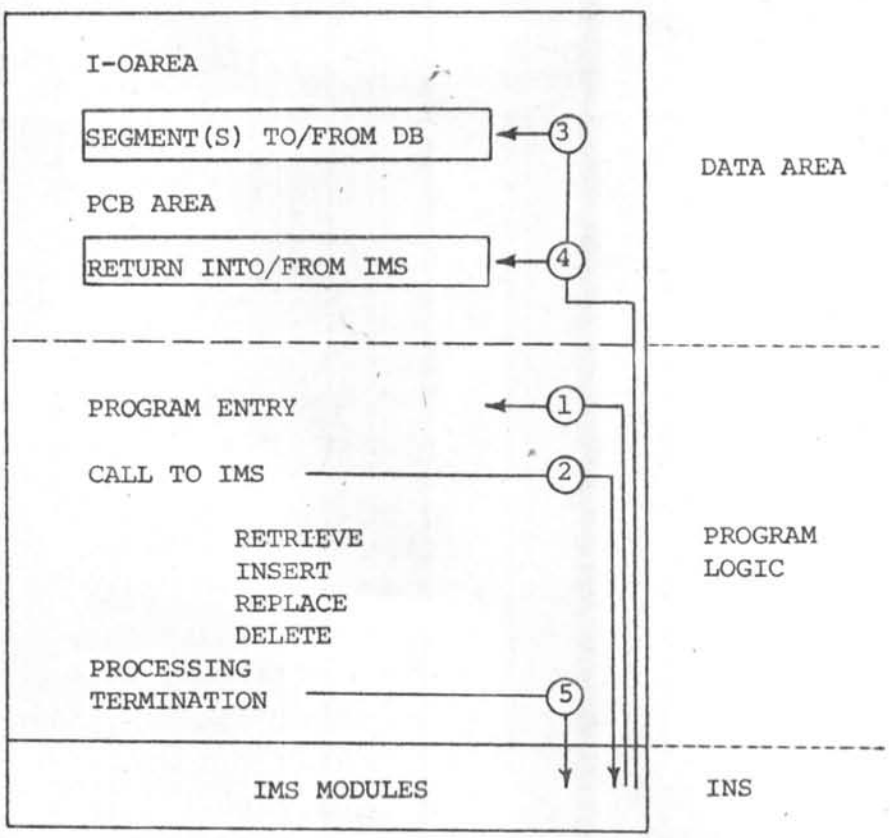
๓.๔.๑ ตารางสรุปรหัสแสดงสถานะภาพการผิดพลาด

CONDITION	MAJOR CODE	01	02	03	04	05	06	07	08	09	10	11	12
	COMMAND	CLOSE	DELETE	FIND	FREE	GET	KEEP	INSERT	MODIFY	OPEN	ORDER	REMOVE	STORE
Area not open		01	01	01	-	-	-	-	-	-	01	-	01
Database-key inconsistent with area-name		-	-	02	-	-	-	-	02	-	-	-	02
Object record altered by concurrent run-unit		-	03	-	-	-	-	03	03	-	-	03	-
Data-items invalid or inconsistent		-	-	04	-	-	-	-	04	-	-	-	-
Violation of DUPLICATES NOT ALLOWED clause		-	-	-	-	-	-	05	05	-	-	-	05
Current of set, area, or record-name not known		-	-	06	-	-	-	06	-	-	06	-	-
End of set or area		-	-	07	-	-	-	-	-	-	-	-	-
Referenced record or set-name not in sub-schema		-	08	-	-	-	-	-	-	-	-	-	-
Incorrect usage mode for area		-	09	-	-	-	-	09	09	-	09	09	09
Privacy breach attempted		-	10	10	-	10	-	10	10	10	10	10	10
Media space not available		-	-	-	-	-	-	-	-	-	-	-	11
Database-key not available		-	-	-	-	-	-	-	-	-	-	-	12
No current record of run-unit		-	13	-	13	13	13	13	13	-	-	13	-
Object record is mandatory automatic in named set		-	-	-	-	-	-	14	-	-	-	-	-
Object record is mandatory in named set		-	-	-	-	-	-	-	-	-	-	15	-
Record already a member of named set		-	-	-	-	-	-	16	-	-	-	-	-
Deleted record involved		-	-	17	-	-	-	-	-	-	-	-	-
Implicitly referenced area not available		-	18	18	-	-	-	18	18	-	-	18	18
Conversion of value of data-item not possible		-	-	-	-	19	-	-	19	-	-	-	19
Current record of run-unit not of record-name		-	20	-	-	20	-	20	20	-	-	20	-
Affected area not open		-	21	-	-	-	-	21	21	-	-	21	21
Record not current member of named or implied set		-	-	22	-	-	-	-	22	-	-	22	-
Illegal area-name		-	-	23	-	-	-	-	23	-	-	-	23
Set occurrence would span temp. and perm. areas		-	-	-	-	-	-	24	24	-	-	-	24
No set occurrence satisfies argument values		-	-	-	-	-	-	-	25	-	-	-	25
No record satisfies r-s-e specified		-	-	26	-	-	-	-	-	-	-	-	-
CHECK clause violation		-	27	-	-	-	-	27	27	-	-	27	27
Area already open		-	-	-	-	-	-	-	-	28	-	-	-
Violation of optional deadlock protection rule		-	-	-	-	-	-	-	-	29	-	-	-
Unqualified DELETE attempted on non-empty set		-	30	-	-	-	-	-	-	-	-	-	-
Removed record involved		-	-	50	-	-	-	50	-	-	-	-	-
Deleted record involved		-	-	51	-	-	-	51	-	-	-	-	-
Virtual data-item not available		-	-	-	-	52	-	-	-	-	-	-	-
Invalid value for virtual result data-item		-	-	-	-	53	-	-	-	-	-	-	-
Value of string data-item truncated in UWA		-	-	-	-	54	-	-	-	-	-	-	-

๓.๔.๒ ดีเอ็มแอลของไอเอ็มเอส

๓.๔.๒.๑ องค์ประกอบต่าง ๆ ของโปรแกรมคำสั่งงานที่ใช้ไอเอ็มเอส

โปรแกรมย่อยที่ทำหน้าที่เชื่อมโยงระหว่างระบบบริหารฐานข้อมูลกับโปรแกรมคำสั่งงาน ซึ่งเปรียบได้กับ "ดีเอ็มแอล" ของ ดีบีทีจี นั้น ใน ไอเอ็มเอส ต้องใช้ผ่านประโยค CALL ซึ่งโปรแกรมคำสั่งงานจะต้องมีโครงสร้าง ดังรูปที่ ๓.๔.๑



รูปที่ ๓.๔.๑ สิ่งที่จำเป็นต้องมีในโครงสร้างของโปรแกรมคำสั่งงาน

๓.๔.๒.๑.๑ ส่วนนำฐานข้อมูลเข้ามาเกี่ยวข้องในโปรแกรมคำสั่งงาน (Program Entry)

ประโยคนี้ต้อง เป็นประโยคแรกหลังจากประโยค PROCEDURE DIVISION ในภาษาโคบอล ประโยคนี้ทำหน้าที่บอกโปรแกรมระบบคำสั่งเครื่องว่า โปรแกรมคำสั่งงานนี้ต้องการใช้ระบบบริหารฐานข้อมูล โดยจะใช้ พีเอสบี อะไรบ้าง

โครงร่าง

```
ENTRY  'DLITCBL'  USING  DBPSB-1,DBPSB-2,----
```

๓.๔.๒.๑.๒ ส่วนที่ใช้ติดต่อกับระบบบริหารฐานข้อมูล

ส่วนนี้เป็นประโยคที่ใช้ติดต่อกับระบบบริหารฐานข้อมูล โดยแจ้งให้ทางระบบบริหารฐานข้อมูลทราบว่า ขณะนี้โปรแกรมคำสั่งงานกำลังต้องการที่จะหยิบข้อมูล เพิ่มเติมข้อมูล ลบข้อมูล หรือแก้ไขข้อมูลบางประการ ภายในฐานข้อมูล ซึ่งจะทำงานในลักษณะใดก็ขึ้นอยู่กับส่วนของ "Call-function" ที่ใช้ในประโยค CALL

โครงร่าง

```
CALL  'CBLTDLI'  USING  call-function,
                        pcb-name,
                        user-i-o-area,
                        ssa1,ssa2,---,ssan.
```

๓.๔.๒.๑.๓ ส่วนของพื้นที่ซึ่งใช้เป็นอินพุทและเอ้าพุทภายในโปรแกรมคำสั่งงาน (I/O AREA)

ส่วนนี้เป็นพื้นที่ซึ่งผู้เขียนโปรแกรมคำสั่งงานต้องเขียนไว้ในส่วนของ "WORKING STORAGE SECTION" ของภาษาโคบอล เพื่อเอาไว้ใช้ประโยชน์ในกรณีต่าง ๆ เช่น เพื่อต้องการให้เอา ข้อมูลในพื้นที่นี้ไปใส่ในฐานข้อมูลในกรณีของการใช้คำสั่ง ISRT (ย่อมาจาก INSERT หมายถึงให้ทำการเพิ่มเติมข้อมูลลงไปในฐานข้อมูล) หรือให้เอาข้อมูลที่ได้โดยการหยิบจากฐานข้อมูลมาใส่ไว้ในพื้นที่นี้ในกรณีของการใช้คำสั่ง GET ซึ่งเป็น "call-function" ชุดหนึ่ง เป็นต้น

ลักษณะของพื้นที่ขึ้นอยู่กับรูปร่างของแต่ละเซ็กเม้นท์

๓.๔.๒.๑.๔ ส่วนของพื้นที่จำลองแบบซึ่งใช้รับข่าวสารจากระบบบริหารฐานข้อมูล (PCB Mask Area)

ส่วนนี้เป็นพื้นที่ใน LINKAGE SECTION ในภาษาโคบอล ซึ่งจะทำหน้าที่คอยรับข้อมูลต่าง ๆ จากระบบบริหารฐานข้อมูล เช่น รับข้อมูลว่าการทำงานของประโยค CALL สำเร็จหรือไม่ เพราะเหตุใด ซึ่งข้อมูลที่รับมานั้นจะอยู่ในลักษณะของรหัสสถานะ (STATUS CODE) โครงร่างของพื้นที่นี้จะเป็นดังรูปที่ ๓.๔.๒

FORMAT PROGRAM COMMUNICATION BLOCK (PCB)

GENERATED BY UTILITY		MASK WRITTEN IN COBOL	
BYTES	FUNCTION	01 SKILLDB PCB.	
8	NAME OF DATABASE	02 DBD-NAME	PIC X(8).
2	SEGMENT HIERARCHICAL LEVEL	02 SEG-LEVEL	PIC XX.
2	IMS STATUS CODE	02 STATUS-CODE	PIC XX.
4	IMS PROCESSING OPTIONS	02 PROC-OPTIONS	PIC XXXX.
4	RESERVED FOR IMS	02 RESERVE-DLI	PIC S9(5) COMP.
8	SEGMENT NAME FEEDBACK	02 SEGNAME-FB	PIC X(8).
4	LENGTH OF FEEDBACK KEY	02 LENGTH-FB-KEY	PIC S9(5) COMP.
4	NUMBER OF SENSITIVE SEGMENT	02 NUMB-SENS-SEGS	PIC S9(5) COMP.
N	KEY FEEDBACK AREA	02 KEY-FB-AREA.	
		03 SKILL-KEY	PIC X(8).
		03 NAME-KEY	PIC X(42).
		03 EDUC-KEY	PIC X(18).

รูปที่ ๓.๔.๒. โครงร่างของพื้นที่จำลองแบบซึ่งใช้รับข้อมูลจากระบบบริหารฐานข้อมูล (PCB Mask Area Format)

ส่วนประกอบต่าง ๆ ของพื้นที่จำลองแบบซึ่งใช้รับข้อมูลจากระบบบริหารฐานข้อมูล ตามรูปที่ ๓.๔.๒ อธิบายได้ ดังนี้

๑. ชื่อของ พีซีบี จะต้องเป็นชื่อเดียวกับ พีเอสบี ที่ได้สร้างไว้ใช้กับแต่ละโปรแกรมคำสั่งงานของภาษาโคบอล ชื่อนี้ต้องอยู่ที่ระดับ 01 (level 01)

๒. ชื่อของฐานข้อมูลต้องเป็นฟิลด์แรกของ พีซีบี เพื่อใช้เก็บชื่อของฐานข้อมูล โดยจะถูกจัดการ โดยส่วนของ ดีบีดี พื้นที่ส่วนนี้มีความยาว ๘ ไบท์

๓. ส่วนของ "Segment Hierarchical Level Indicator" ใช้พื้นที่ส่วนนี้เพื่อบอกระดับ (level number) ของเซ็กเม้นท์สุดท้ายที่ถูกเรียก (Call) ไปแล้ว

๔. ส่วนของรหัสสถานะของ ไอเอ็มเอส (IMS Status Code) ใช้บอกผลของการใช้คำสั่ง CALL ถ้าประโยคคำสั่ง CALL ทำงานสำเร็จส่วนนี้จะเป็นที่ว่างหรือรหัสสถานะอื่น ๆ ก็ได้ โดยขึ้นอยู่กับผลของการเรียก พื้นที่ส่วนนี้ยาว ๒ ไบท์

๕. ส่วนของ "IMS Processing Options" เป็นพื้นที่ซึ่งใช้เก็บรหัสที่บอก ไอเอ็มเอส เกี่ยวกับรายละเอียดในการประมวลผลของฐานข้อมูลด้วยโปรแกรมคำสั่งงาน มีความยาว ๔ ไบท์ และจะไม่เกิดการเปลี่ยนแปลงค่าเมื่อมีการเรียกครั้งหนึ่ง ๆ ค่าภายในพื้นที่นี้ จะถูกกำหนดในช่วงของการสร้าง พีเอสบี ภายใต้พารามิเตอร์ของ พีซีบี "PROCOPT"

๖. พื้นที่ที่สำรองเอาไว้สำหรับ ไอเอ็มเอส (Reserved Area For IMS) เป็นส่วนที่ ไอเอ็มเอส ใช้เชื่อมติดต่อกันภายในกับโปรแกรมคำสั่งงาน พื้นที่นี้ยาว ๔ ไบท์

๗. พื้นที่สำหรับรับชื่อของเซ็กเม้นท์ (Segment Name Feedback Area) เป็นส่วนที่ ไอเอ็มเอส ใช้เก็บชื่อของเซ็กเม้นท์สุดท้าย ที่ได้ถูกทำการเรียกสำเร็จไปแล้ว มีความยาว ๘ ไบท์

๘. พื้นที่สำหรับบอกค่าความยาวคีย์ (Length of Key Feedback Area) เป็นส่วนที่บอกความยาวของ "key feedback area" พื้นที่นี้มีความยาว ๔ ไบท์

STATUS CODE	DATA BASE CALLS: GU GHU	GN GHN	GNP GHNP	DLET	REPL	ISRT (LOAD)	ISRT (ADD)	CHKP	CALL COMPLETED	ERROR IN CALL	I/O OR SYSTEM ERROR	DESCRIPTION
AB	X	X	X	X	X	X	X			X		SEGMENT I/O AREA REQUIRED, NONE SPECIFIED IN CALL
AC	X	X	X			X	X			X		HIERARCHICAL ERROR IN SSAs
AD		✓								X		INVALID FUNCTION PARAMETER
AH						X	X			X		CALL REQUIRES SSAs, NONE PROVIDED
AI	X	X	X	X	X	X	X				X	DATA MANAGEMENT OPEN ERROR
AJ	X	X	X	X	X	X	X			X		INVALID SSA QUALIFICATION FORMAT OR COMMAND CODE
AK	X	X	X			X	X			X		INVALID FIELD NAME IN CALL
AM	X	X	X	X	X	X	X			X		CALL FUNCTION NOT COMPATIBLE WITH PROCESSING OPTION OR SEGMENT OR PATH SENSITIVITY
AO	X	X	X	X	X	X	X				X	I/O ERROR
DA					X					X		SEGMENT KEY FIELD HAS BEEN CHANGED
DJ				X	X					X		NO PRECEDING SUCCESSFUL GET HOLD CALL
DX				X						X		VIOLATED DELETE RULE
GA		ⓧ	ⓧ						X			CROSSED HIERARCHICAL BOUNDARY INTO HIGHER LEVEL (RETURNED ONLY ON CALLS WITH NO SSA SPECIFIED)
GB		ⓧ										END OF DATA SET, LAST SEGMENT REACHED
GE	ⓧ	ⓧ	ⓧ				ⓧ					SEGMENT OR PARENT SEGMENT NOT FOUND
GK		ⓧ	ⓧ						X			DIFFERENT SEGMENT TYPE AT SAME LEVEL RETURNED (RETURNED ON UNQUALIFIED CALLS ONLY)
GP			X							X		A GNP CALL AND NO PARENT ESTABLISHED, OR REQUESTED SEGMENT LEVEL NOT LOWER THAN PARENT LEVEL
II							ⓧ					SEGMENT TO INSERT ALREADY EXISTS IN DATA BASE
IX							X			X		VIOLATED INSERT RULE
LB						ⓧ						SEGMENT TO INSERT ALREADY EXISTS IN DATA BASE
LC						ⓧ						KEY FIELD OF SEGMENTS OUT OF SEQUENCE
LD						ⓧ						NO PARENT FOR THIS SEGMENT HAS BEEN LOADED
LE						ⓧ						SEQUENCE OF SIBLING SEGMENT NOT THE SAME AS DBD SEQUENCE
NA					X					X		DATA IN SEARCH OR SUBSEQUENCE FIELD HAS BEEN CHANGED
NE				X	X				X		X	INDEX MAINTENANCE CANNOT FIND SEGMENT
NI				X	X	X	X				X	INDEX MAINTENANCE UNABLE TO OPEN INDEX DATA BASE
					X	X	X			X		DUPLICATE KEY FOUND FOR INDEX DATA BASE
NO				X	X	X	X				X	I/O ERROR
RX					X					X		VIOLATED REPLACE RULE
V1					X	X	X			X		INVALID LENGTH FOR VARIABLE LENGTH SEGMENT
XD								X				ERROR DURING DATA BASE BUFFER WRITE OUT
XH								X				DATA BASE LOGGING NOT ACTIVE
bb	ⓧ	ⓧ	ⓧ	ⓧ	ⓧ	ⓧ	ⓧ	ⓧ	X			GOOD, NO STATUS CODE RETURNED, PROCEED!

รูปที่ ๓.๔.๓ ตารางสรุปรหัสสถานะต่าง ๆ ที่ได้จากการใช้ คำสั่ง CALL

๙. พื้นที่ซึ่งใช้เก็บค่าจำนวนของเซ็กเม้นท์ (Number of Sensitive Segment) เป็นส่วนที่เก็บค่าบอกจำนวนของชนิดของเซ็กเม้นท์ในฐานข้อมูล หรืออาจกล่าวได้ว่าใช้เก็บค่าจำนวนของเซ็กเม้นท์ในลักษณะของโครงสร้างข้อมูลเชิงตรรก ซึ่งถูกกำหนดไว้แล้วในส่วนของ พีซีบี ยาว ๔ ไบท์

๑๐. ส่วนที่ใช้เก็บค่าคีย์ที่ส่งกลับมา (Key Feedback Area) ไอเอ็มเอส จะใช้ส่วนนี้สำหรับเก็บค่าคีย์ที่ถูกส่งกลับมา พื้นที่นี้จะยาวเท่าใดต้องดูจากผลรวมของความยาวของคีย์ฟิลด์ของทุกเซ็กเม้นท์ในเส้นทางตามลำดับชั้น (Hierarchical path) ที่เรียกว่า "Concatenated key" ที่ยาวที่สุดเป็นเกณฑ์

๓.๔.๒.๑.๕ <u>ส่วนที่บอกการจบของการติดต่อกับฐานข้อมูล (Termination)</u>

ส่วนนี้จะเป็นประโยคที่บอกให้ระบบบริหารฐานข้อมูลเบิกติดต่อกับฐานข้อมูล และให้ระบบเครื่องกลับไปอยู่ในสภาพปกติเหมือนเดิมก่อนการทำงานของโปรแกรมคำสั่งงานนี้

โครงร่าง

Language	ANS COBOL	IMS	ASSEMBLY
Command	GOBACK	RETURN	RETURN (14,12)

๓.๔.๒.๒ พารามิเตอร์ภายในประโยค CALL

ไอเอ็มเอส มีกลุ่มของฟังค์ชั่น (Function) ซึ่งทำให้ผู้เขียนโปรแกรมคำสั่งงานสามารถใช้ข้อมูลในฐานข้อมูลได้ โดยใช้ประโยค CALL ซึ่งมีลักษณะ ดังนี้

โครงร่าง

```
CALL 'CBLTDLI' USING call-function,
                     pcb-name,
                     user-i-o-area,
                     ssa1,ssa2,---,ssan.
```

๓.๔.๒.๒.๑ call-function แบ่งตามวัตถุประสงค์ของการใช้งานได้ ดังนี้

๑. ใช้ในการนำ (load) ข้อมูลลงในฐานข้อมูลในช่วงการทำการสร้างฐานข้อมูลครั้งแรก ได้แก่ ISRT (INSERT)

๒. ใช้ในการดึงข้อมูลจากฐานข้อมูลมาใช้ ได้แก่

GU␣␣	ย่อมาจาก	GET UNIQUE
GN␣␣	" "	GET NEXT
GNP␣	" "	GET NEXT WITHIN PARENT
GHU␣	" "	GET HOLD UNIQUE
GHN␣	" "	GET HOLD NEXT
GHNP	" "	GET HOLD NEXT WITHIN PARENT

(␣ หมายถึง ช่องว่าง)

๓. ใช้ในการปรับปรุงแก้ไขข้อมูลในฐานข้อมูล ได้แก่

ISRT	ย่อมาจาก	INSERT	(ใช้เพิ่มเติมเซ็กเม้นท์ในฐานข้อมูล)
DLET	"	DELETE	(ใช้ลบเซ็กเม้นท์ในฐานข้อมูลทิ้ง)
REPL	"	REPLACE	(ใช้แก้ไขเนื้อหาบางประการในเซ็กเม้นท์ของฐานข้อมูล)

การเรียกใช้ call-function จะทำโดยตรงเลยไม่ได้ ต้องเรียกผ่านพื้นที่ซึ่งตั้งเอาไว้เก็บค่าของฟังค์ชั่นเหล่านี้ ในภาษาโคบอล พื้นที่ส่วนนี้จะต้องกำหนดไว้ในส่วนของ "WORKING-STORAGE SECTION" ดังตัวอย่างเช่น

```
01  CALL-FUNC.
    03  FUNC-GU    PIC  X(4) VA 'GUbbb'.
    03  FUNC-GHU   PIC  X(4) VA 'GHUb'.
    03  FUNC-ISRT  PIC  X(4) VA 'ISRT'.
```

และเวลาจะเรียกใช้ในโปรแกรมคำสั่งงานก็ต้องเป็นในลักษณะนี้

```
CALL 'CBLTDLI' USING FUNC-GU,
                     ........,
                     ........,
                     ........,
```

หมายเหตุ การเรียกใช้ call-function ประเภท DLET หรือ REPL จะใช้ได้ก็ต่อเมื่อการเรียกครั้งที่แล้ว เป็นการเรียกที่ใช้ call-function ประเภท GET HOLD เท่านั้น

๓.๔.๒.๒.๒ pcb-name เป็นชื่อของ พีซีบี ที่เราต้องการจะเรียกใช้ในการติดต่อกับฐานข้อมูลชื่อของ พีซีบี นี้ จะถูกสร้างไว้แล้วใน พีเอสบี และเก็บไว้ในที่เก็บโปรแกรมในรูปภาษาเครื่องที่พร้อมจะทำงานได้ ซึ่งชื่อนี้จะเป็นตัวบอกให้ ไอเอ็มเอส ทราบว่าโปรแกรมคำสั่งงาน ต้องการที่จะทำงานกับข้อมูลในลักษณะเฉพาะแบบใดแบบหนึ่งของโครงสร้างข้อมูลแบบมีลำดับขั้น (Hierarchical data structure)

๓.๔.๒.๒.๓ user-i-o-area เป็นพื้นที่ซึ่งใช้สำหรับทำงาน (work area) สำหรับโปรแกรมคำสั่งงาน ไม่ว่าจะเป็นการดึงข้อมูลออกมาจากเซ็กเม้นท์หรือการนำเอาข้อมูลใส่ลงในเซ็กเม้นท์ของฐานข้อมูล ซึ่งลักษณะการใช้พื้นที่นี้จะมี ดังนี้

๑. ถ้าใช้ร่วมกับ call-function ประเภท ISRT หมายถึง ให้เอาข้อมูลที่อยู่ในพื้นที่นี้เติมลงไปในฐานข้อมูล

๒. ถ้าใช้ร่วมกับ call-function ประเภท GETหมายถึง ให้ถึงเอาข้อมูลจากฐานข้อมูลมาใส่ไว้ในพื้นที่นี้

๓. ถ้าใช้ร่วมกับ call-function ประเภท REPL หมายถึง ให้เอาข้อมูลที่อยู่ในพื้นที่นี้ทับลงไปในข้อมูลเดิมในเซ็กเม้นท์ที่เพิ่งทำการหยิบมาได้ด้วยการ GET HOLD ครั้งที่แล้ว

๔. ถ้าใช้ call-function ประเภท DLET อย่างเดียวในโปรแกรมคำสั่งงาน พื้นที่ส่วนนี้ไม่ต้องกำหนดไว้ในโปรแกรมคำสั่งงานนั้นก็ได้

๓.๔.๒.๒.๔ ssa1, ssa2,....,ssan. ย่อมาจาก "Segment Search Argument" ในภาษาโคบอล จะเป็นพื้นที่ซึ่งผู้เขียนโปรแกรมคำสั่งงานกำหนดเอาไว้ในส่วนของ WORKING-STORAGE SECTION เพื่อวัตถุประสงค์ ดังนี้

๑. บ่งถึงเซ็กเม้นท์ที่ต้องการจะอ้างอิง อันได้แก่ เซ็กเม้นท์ที่มีชื่ออยู่ในเอสเอสเอ อันสุดท้ายของการเรียก

๒. บอกเส้นทางตามลำดับขั้น (Hierarchical Path) ของเซ็กเม้นท์ที่กำลังอ้างถึง

๓. บอกเงื่อนไขเพื่อให้ได้มาซึ่งข้อมูลที่สอดคล้องตามเงื่อนไขนั้น

เอส เอส เอ แบ่งออกเป็น ๒ ชนิด ดังนี้

๑. เอสเอสเอ ที่ไม่ระบุเงื่อนไข (Unqualified SSA) ใช้อธิบายเพียงว่าต้องการเซ็กเม้นท์ที่มีชื่อใน เอสเอสเอ เท่านั้น โดยไม่ได้เจาะจงไปว่าต้องเป็นข้อมูลใดที่สอดคล้องกับเงื่อนไขใด

โครงร่าง

SEGMENT NAME	⊬
๘ ไบท์	๑ ไบท์

๒. เอสเอสเอ ที่ระบุเงื่อนไข (Qualified SSA) ใช้อธิบายถึงเงื่อนไขซึ่งเจาะจงลงไปเลยว่าต้องการข้อมูลที่คล้องตามเงื่อนไขว่าอย่างไร

โครงร่าง

SEGMENT NAME	(	key field name	Rd. Oper.	COMparative value ขึ้นอยู่กับค่าของคีย์ฟิลด์ยาวได้	)
๘ ไบท์	๑ ไบท์	๘ ไบท์	๒ ไบท์	๑ ถึง ๒๕๕ ไบท์	๑ ไบท์

- ชื่อของเซ็กเม้นท์ (SEGMENT NAME) เป็นชื่อของเซ็กเม้นท์ที่ได้กำหนดเอาไว้ในส่วนของ พีซีบี นี้ สำหรับโปรแกรมคำสั่งงานใด ๆ ส่วนของพื้นที่นี้ยาว ๘ ไบท์

- สัญญลักษณ์เริ่มต้น (Begin Qualification Character) เป็นส่วนที่มีค่า เป็นวงเล็บเปิด "(" ซึ่งมีความยาว ๑ ไบท์
- สัญญลักษณ์ที่แสดงเงื่อนไข (Relation Operator) ซึ่งตัว "Operator" เหล่านี้มีความหมายต่าง ๆ กัน ดังนี้

Operator	ความหมาย
¬ = หรือ = ¬ หรือ EQ	มีค่าเท่ากัน
= หรือ GE	มีค่ามากกว่าหรือเท่ากัน
= หรือ LE	มีค่าน้อยกว่าหรือเท่ากัน
¬ หรือ ¬ หรือ GT	มีค่ามากกว่า
¬ หรือ ¬ หรือ LT	มีค่าน้อยกว่า
= หรือ NE	มีค่าไม่เท่ากัน

พื้นที่ส่วนนี้จะยาว ๒ เบท์

- ส่วนค่าเปรียบเทียบ (Comparative value) เป็นพื้นที่ที่ใช้เก็บค่าที่จะนำมาเปรียบเทียบกันภายในเซ็กเม้นท์ของฐานข้อมูล โดยถือตามชื่อของคีย์ฟิลด์ ดังนั้น พื้นที่ส่วนนี้จะต้องมีความยาวเท่ากับชื่อของคีย์ฟิลด์
- สัญญลักษณ์จบ (End Qualification Character) เป็นส่วนที่มีค่าเป็นวงเล็บปิด ")" ซึ่งมีความยาว ๑ ไบท์

ตัวอย่างของ เอสเอสเอ ที่ระบุเงื่อนไข

```
01  QUAL-SSA.
    03  SEG-NAME    PIC X(8)   VA 'CUSTOMER'.
    03  BEG-PAR     PIC X      VA '('.
    03  KEY-NAME    PIC X(8)   VA 'IDEN    '.
    03  RELATION    PIC XX     VA '= '.
    03  KEY-VALUE   PIC X(7)   VA '1587642'.
    03  END-PAR     PIC X      VA ')'.
```

หมายเหตุ ในกรณีที่ประโยค CALL หนึ่ง ๆ มี เอส เอส เอ หลายตัว ลำดับที่บ่ง เอสเอสเอ จะต้องเรียงตามลำดับของ "hierarchical sequence" ของเซ็กเม้นท์ตามเส้นทาง (path) ที่กำลังอ้างถึง

๓.๔.๓ ดีเอ็มแอลของโททอล ในโททอลเรียกว่า DATA BASE MANAGEMENT PHASE

ในขั้นนี้เป็นการให้โปรแกรมคำสั่งงานทำการติดต่อเรียกใช้, ปรับปรุงแก้ไขข้อมูล ภายในฐานข้อมูลได้โดยผ่าน ดีเอ็มแอล ซึ่งจะมีคำสั่งต่าง ๆ ที่เป็นตัวพารามิเตอร์อยู่ในประโยคคำสั่ง CALL ในโปรแกรมคำสั่งงาน ซึ่งโครงสร้างของ ดีเอ็มแอล จะมีลักษณะตามรูปที่ ๓.๔.๔

ฟอร์มของคำสั่ง CALL เมื่อใช้ในโปรแกรมภาษาโคบอล

```
CALL 'DATBAS'  USING dmicommand,status(,file)(,reference)(,linkpath)
                     (,controlkey)(,qualifier)(,argement)(,length)
                     (,datalist)(,dataarea)(,access)(,dbmod)(,task)
                     (,options)(,end)
```

ฟอร์มของคำสั่ง CALL เมื่อใช้ในโปรแกรมภาษาแอสแซมเบอร์

```
CALL DATBAS,[dmlcommand,status(,file)(,reference)(,linkpath)
            (,controlkey)(,qualifier)(,argument)(,length)(,datalist)
            (,dataarea)(,access)(,dbmod)(,task)(,options)(,end)]
```

ฟอร์มของคำสั่ง CALL เมื่อใช้ในโปรแกรมภาษาพีแอลวัน

```
CALL DATBAS [dmlcommand,status(,file)(,reference)(,linkpath)(,controlkey)
            (,qualifier)(,argument)(,length)(,datalist)(,dataarea)
            (,access)(,dbmod)(,task)(,options)(,end)]
```

รูปที่ ๓.๔.๔ แสดงฟอร์มของคำสั่ง CALL ที่ใช้ในโปรแกรมทำงาน

ความหมายของพารามิเตอร์บางตัวที่สำคัญ

๑. dml command เป็นตัวกำหนดหน้าที่ที่จะจัดการกับข้อมูลในฐานข้อมูล เช่น ให้อ่าน, เพิ่มข้อมูล, ลบข้อมูล เป็นต้น ความยาวของทุกคำสั่งจะเท่ากับ ๔ ตัวอักษรคงที่ และแต่ละคำสั่งจะต้องกำหนดตัวพารามิเตอร์ต่าง ๆ ที่ตามมาให้ถูกต้องและครบถ้วน ตามรูปที่ ๓.๔.๕

๒. status มีความยาว ๔ ตัวอักษรและต้องกำหนดในทุกคำสั่ง ซึ่งระบบบริหารฐานข้อมูลโททอลจะส่งรหัสสถานะภาพ(status code) นี้ กลับมาทุกครั้งเพื่อแสดงผลที่ได้หลังจากทำตามคำสั่งใน ๑ แล้ว ถ้าสำเร็จตามต้องการจะส่งรหัส "***" ให้ ดังนั้นรหัสนี้ต้องถูกตรวจสอบหลังจากออกคำสั่งใน ๑ แล้วทุกครั้งในโปรแกรมคำสั่งงาน

๓. file มีความยาว ๔ ตัวอักษรซึ่งเป็นชื่อของแฟ้มข้อมูล (ในโปรแกรมย่อย ดีบีดี) ที่จะเข้าจัดการ ถ้าระบุว่า "ALL" แล้ว จะหมายถึงทุกแฟ้มข้อมูล

๔. reference มีความยาว ๔ ตัวอักษร

- ถ้าใช้เป็นจุดอ้างอิงภายใน (internal reference point) จะหมายถึงตำแหน่งเริ่มเข้าถึงและเมื่อสิ้นสุดการทำงาน "reference" จะบอกถึง record สุดท้ายที่ทำงานถึง
- ถ้าใช้ ๔ ตัวหลังของเส้นทางเชื่อมต่อ (Link path) คือ "LKXX" จะบอกถึงการจัดการว่าให้เริ่มต้นที่ระเบียนข้อมูลแรกหรือสุดท้ายของเส้นทางลูกโซ่ (chain) เมื่อเสร็จสิ้นการทำงาน "reference" จะมีค่าเป็น "END"

๕. link path มีความยาว ๘ ตัวอักษรคือ "mmmmLKxx" โดยที่ mmmm คือชื่อของแฟ้มข้อมูลหลัก (ชื่อของ link path ต้องเป็นไปตามที่ระบุเอาไว้ในโปรแกรมย่อย ดีบีดี)

๖. control key ความยาวแปรเปลี่ยนได้ตามที่ผู้ใช้กำหนดชื่อและขนาดต้องตรงกับที่ระบุไว้ในโปรแกรมย่อย ดีบีดี

- โททอลใช้คีย์นี้ในการทำการคำนวณ (Hashing) หาตำแหน่งของระเบียนข้อมูล
- เชื่อมระเบียนข้อมูลหลักกับระเบียนข้อมูลแปรเปลี่ยน

๗. qualifier ความยาวแปรตามที่ผู้ใช้กำหนดใช้สำหรับพวกคำสั่งควบคุม (CNTRL)

- กำหนดหน้าที่ (function) ที่จะทำ, ชื่อของงานหรือแฟ้มข้อมูล หรือชื่อของโปรแกรมย่อย (modules) ที่จะทำงาน (process)
- สำหรับคำสั่ง FINDX , RDNXT ฟิลด์นี้จะใช้ดูแลตำแหน่ง " current" ในแฟ้มข้อมูลที่จะทำงาน

๘. data-list ความยาวแปรตามผู้ใช้กำหนดเป็นชุด (string) ของข้อมูล หรือคีย์ควบคุม หรือรหัสระเบียนข้อมูล ซึ่งชื่อและขนาดต้องเป็นไปตามที่ระบุเอาไว้ใน ดีบีดี ตัว data-list ไม่สามารถรวม ROOT ของแฟ้มข้อมูลหลัก หรือตัวเชื่อมต่อ (LINK) เพราะว่าเป็นฟิลด์ของระบบเครื่อง

๙. data-area เป็นพื้นที่ของ อินพุท/เอ้าพุท ในหน่วยความจำ (ผู้ใช้ต้องกำหนด) สำหรับ data-list โดยเป็น อินพุทหรือเอ้าพุท ก็ได้แล้วแต่ความต้องการ รูปแบบของ data-area ต้องสอดคล้องกับของ data-list

๑๐. end ความยาว ๔ ตัวอักษรและต้องกำหนดในทุกคำสั่ง (เหมือนกับ Status) เพื่อบอกถึงการสิ้นสุดของกลุ่มของพารามิเตอร์

รูปที่ ๓.๔.๕ พารามิเตอร์ที่ใช้ในแต่ละคำสั่ง

Command	Required Parameters														
	status	file	reference	linkpath	controlkey	qualifier	argument	length	datalist	dataarea	access	dbmod	task	options	end
ADD-M	X	X			X				X	X					X
ADDVA	X	X	X	X	X				X	X					X
ADDVB	X	X	X	X	X				X	X					X
ADDVR	X	X	X	X	X				X	X					X
CLOSX	X	X													X
CNTRL*	X					X		X		X					X
DEL-M	X	X			X				X	X					X
DELVD	X	X	X	X	X				X	X					X
FINDX	X	X				X	X		X	X					X
OPENX	X	X													X
RDNXT	X	X				X			X	X					X
READD	X	X	X	X	X				X	X					X
READM	X	X			X				X	X					X
READR	X	X	X	X	X				X	X					X
READV	X	X	X	X	X				X	X					X
RQLOC	X	X			X					X					X
SINOF	X												X		X
SINON	X										X	X	X	X	X
WRITM	X	X			X				X	X					X
WRITV	X	X	X	X	X				X	X					X

* Applicable to all control functions.

๑. ชนิดของคำสั่งใน ดีเอ็มแอล แบ่งเป็น ๔ กลุ่มใหญ่ ๆ ดังนี้

๑.๑) หยิบข้อมูล (Retrieve)

๑.๒) แก้ไขปรับปรุงเปลี่ยนแปลงข้อมูล (Update)

๑.๓) เริ่มต้น/จบปิด (Initiate/Terminate)

๑.๔) ควบคุม (Control)

๑.๑ ชนิดหยิบข้อมูลออกมาใช้ หน้าที่ของคำสั่งชนิดนี้แบ่งเป็น

๑. เข้าถึงระเบียนข้อมูลที่ต้องการโดยตรง ได้แก่ คำสั่ง READM, READV, READR, READD

๒. เข้าถึงระเบียนข้อมูลตามลำดับโดยกำหนดจุดเริ่มต้นได้แก่คำสั่ง RDNXT

๓. เข้าถึงระเบียนข้อมูลด้วยวิธีเฉพาะได้แก่ คำสั่ง FINDX, RQLOC

๔. เข้าถึงข้อมูลโดยไม่ผ่านคำสั่งของโททอล เพราะว่าแฟ้มข้อมูลของโททอลนั้นเข้ากันได้กับ (compatible) วิธีการเข้าถึงข้อมูลทางกายภาพของเครื่องอยู่แล้ว (เช่นใช้โปรแกรมคำสั่งเครื่องเพื่อทำการคัดลอกข้อมูล (copy) หรือการจัดเรียงข้อมูล เป็นต้น

๑.๑.๑ คำสั่งประเภทเข้าถึงโดยตรง

๑. คำสั่ง READM

มีหน้าที่อ่านระเบียนข้อมูลหนึ่ง โดยตรงในแฟ้มข้อมูลหลักที่มีคีย์ควบคุมตามที่ระบุไป ด้วยการสุ่มค่าของคีย์ให้เป็นตำแหน่งของระเบียนข้อมูล (ทำการคำนวณ (Hashing)) ถ้าพบก็จะดึงค่าตามชื่อใน data-list มาไว้ที่ data-area

๒. คำสั่ง READV

มีหน้าที่อ่านระเบียนข้อมูลในแฟ้มข้อมูลแปรเปลี่ยนอย่างตรงไปข้างหน้า (forward) ตามเส้นทางเชื่อมต่อที่ระบุ ถ้าจะให้อ่านหมดทั้งเส้นทางลูกโซ่ (chain) จะต้องกำหนด LKxx ใน reference ซึ่งบอกจุดเริ่มต้น ส่วน control key จะถูกใช้ในการเข้าถึงระเบียนข้อมูลหลักก่อน เพื่อเอาหมายเลขระเบียนข้อมูลสัมพันธ์ (relative record no.)

ของระเบียนข้อมูลแรกในเส้นทางลูกโซ่มาเข้าถึงในแฟ้มข้อมูลแปรเปลี่ยน ถ้าสำเร็จผู้ใช้ก็จะได้ระเบียนข้อมูลแรกมาทำงาน จากนั้นถ้าต้องการระเบียนข้อมูลต่อมาในเส้นทางลูกโซ่ก็ให้ทำซ้ำอีก (loop) (ซึ่ง READV จะถูกทำงานใหม่ (execute)) ถ้ามีการกำหนดจุดอ้างอิงภายใน ใน reference ตำแหน่งนั้นก็จะถูกเข้าถึงจนไปถึงระเบียนข้อมูลสุดท้าย

๓. คำสั่ง READR

หน้าที่คือ อ่านย้อนกลับ (reverse) ซึ่งการเข้าถึงจะกลับกันกับคำสั่ง READV

๔. คำสั่ง READD

หน้าที่คือ อ่านระเบียนข้อมูลในแฟ้มข้อมูลแปรเปลี่ยน-โดยตรงด้วยการกำหนดจุดอ้างอิงภายใน ใน reference (อนึ่ง สำหรับการเข้าถึงระเบียนข้อมูลในแฟ้มข้อมูลแปรเปลี่ยนนั้น การระบุชื่อของแฟ้มข้อมูล กับคีย์ จะไม่เพียงพอ เนื่องจากคีย์หนึ่ง ๆ นั้น อาจมีหลาย Sub-chain ได้)

๑.๑.๒ คำสั่งประเภทเข้าถึงตามลำดับ

๑. คำสั่ง RDNXT

หน้าที่คือ อ่านระเบียนข้อมูลแรกจนถึงระเบียนข้อมูลสุดท้ายตามลำดับในแฟ้มข้อมูลหลักหรือแฟ้มข้อมูลแปรเปลี่ยนอย่างกายภาพ นอกจากนี้ยังสามารถกำหนดระเบียนข้อมูลแรกที่จะเข้าถึงได้ดังนี้

- สำหรับแฟ้มข้อมูลหลักกำหนดด้วยคีย์ควบคุม หรือจุดอ้างอิงภายใน
- สำหรับแฟ้มข้อมูลแปรเปลี่ยนกำหนดด้วยจุดอ้างอิงภานใน หรือใช้เส้นทางเชื่อมต่อ ซึ่งจะไปเริ่มที่หัวของเส้นทางลูกโซ่แล้วไหลตามเส้นทางลูกโซ่ไป, ทั้งหมดนี้จะต้องกำหนดใน qualifier เท่านั้น

๑.๑.๓ คำสั่งประเภทเข้าถึงโดยวิธีเฉพาะ

๑. คำสั่ง FINDX

หน้าที่คล้ายคำสั่ง RDNXT แต่ตรงที่สามารถทำการเปรียบเทียบตัวข้อมูลที่อ่านได้กับ

argument ที่ระบุไป โดยที่ใน argument จะต้องมีรายการ (list) ชื่อของข้อมูลเครื่องหมายการเปรียบเทียบ (มากกว่า, น้อยกว่า, เท่ากับ, ไม่เท่ากับ, มากกว่าหรือเท่ากับ, น้อยกว่าหรือเท่ากับ) และรายการค่าคงที่ (ข้อมูลที่ไปเปรียบเทียบ) การเปรียบเทียบจะกระทำกับข้อมูล ๑ ฟิลด์ต่อ ๑ ค่าคงที่ โดยสอดคล้องตามลำดับ ถ้าเป็นจริงหมด ระเบียนข้อมูลนั้นจึงจะกระทำกับข้อมูล ๑ ฟิลด์ต่อ ๑ ค่าคงที่ โดยสอดคล้องตามลำดับ ถ้าเป็นจริงหมดระเบียนข้อมูลนั้นจึงจะถูกหยิบเอาไปใช้ ซึ่งจะเห็นได้ว่าคำสั่งนี้ทำการ SCAN/COMPARE/RETRIEVE

๒. คำสั่ง RQLOC

คำสั่งนี้ทำหน้าที่คำนวณหา (request location) จุดอ้างอิงภายใน (relative 'home' location) ของแฟ้มข้อมูลหลัก แล้วเก็บที่ data-area โดยจะทำการสุ่มค่าของคีย์ควบคุมที่ระบุไป จะเห็นว่าไม่มีการทำ อินพุท/เอ้าพุท โดยปกติคำสั่งนี้ไม่ถูกใช้ในโปรแกรมคำสั่งงาน แต่ใช้ในการเขียนโปรแกรมคำสั่งเครื่องสนับสนุน (System Support Routine) เช่น การ Load, การ recovery เป็นต้น

๑.๒ ชนิดแก้ไข ปรับปรุง เปลี่ยนแปลงข้อมูล แบ่งออกเป็นกลุ่มตามหน้าที่ ดังนี้

- เพิ่มข้อมูลใหม่เข้าไป ได้แก่ คำสั่ง ADD-M, ADDVA, ADDVB (, ADDVC ซึ่งมีหน้าที่เหมือน ADDVA แต่ใช้ในโททอลรุ่นเก่า)
- ลบข้อมูลเก่าออก ได้แก่คำสั่ง DEL-M, DELVD
- แก้ไขข้อมูลเก่า ได้แก่คำสั่ง WRITM, WRITV, ADDVR
- ทำการเพื่อให้ได้ข้อมูลเก่ากลับคืนมา หลังจากทำลาย ได้แก่ WRITD
- แก้ไขข้อมูลโดยไม่ผ่านคำสั่งของโททอล

๑.๒.๑ คำสั่งประเภทเพิ่มข้อมูลใหม่เข้าไป

๑. คำสั่ง ADD-M

หน้าที่คือ เพิ่มระเบียนข้อมูลใหม่ (ใน data-area) เข้าไปในแฟ้มข้อมูลหลัก โดย control key (ต้องแน่ใจว่าไม่ซ้ำกับของเดิม) จะถูกทำการคำนวณหาหมายเลขความสัมพันธ์ ถ้าไปตกตำแหน่งที่มีระเบียนข้อมูลเก่าอยู่แล้ว จะมีการหาที่ว่างถัดไปให้ระเบียนข้อมูลใหม่อยู่

ROOT ของระเบียนข้อมูลเก่าจะชี้ไปที่ตำแหน่งของระเบียนข้อมูลใหม่ และ ROOT ของระเบียนข้อมูลใหม่จะชี้ไปที่ตำแหน่งของระเบียนข้อมูลเก่า ('home' location) เป็นลักษณะของเส้นทางลูกโซ่ (chain)

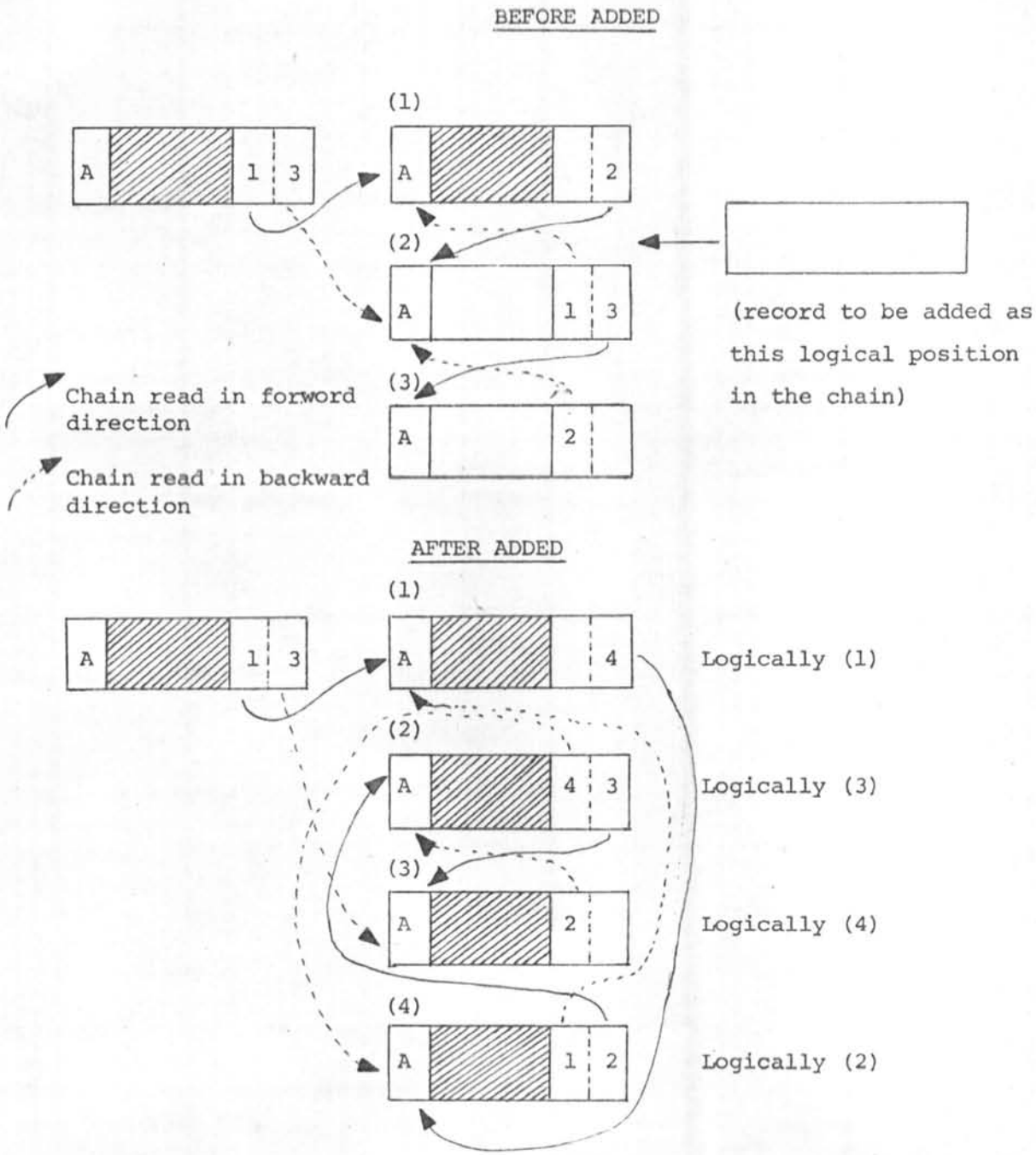
(คีย์ที่ไม่ซ้ำกันเลยนั้น มีโอกาสที่จะถูกทำการคำนวณ แล้วได้หมายเลขความสัมพันธ์เดียวกันได้ โอกาสที่ได้ซ้ำกันมากหรือน้อยขึ้นอยู่กับ เทคนิคหรือสูตรของการคำนวณที่ใช้ ซึ่งมีหลายสูตรแล้วแต่จะคิดขึ้น)

๒. คำสั่ง ADDVA (Add Variable After)

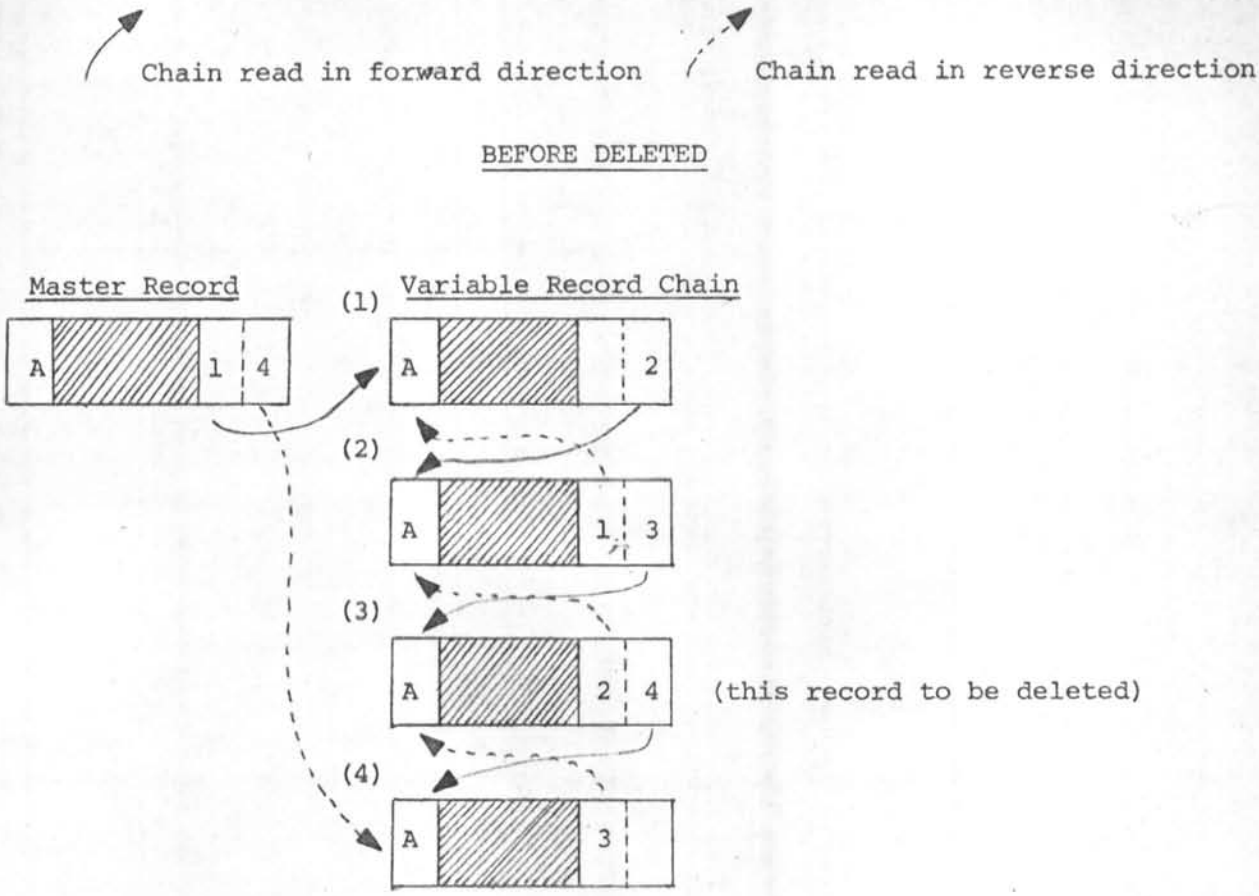
ทำหน้าที่เพิ่มเติมระเบียนข้อมูลใหม่ หลังจุดอ้างอิงภายใน ใน reference เข้าไปในแฟ้มข้อมูลแปรเปลี่ยน อย่างตรรก และจะไปเชื่อมกับเส้นทางลูกโซ่ตามชื่อที่กำหนดในเส้นทางเชื่อมต่อเท่านั้น ส่วนเส้นทางลูกโซ่อื่นที่ไม่ได้กำหนดในเส้นทางเชื่อมต่อ (ระเบียนข้อมูลหนึ่ง ๆ ส่วนใหญ่จะอยู่ในหลายเส้นทางลูกโซ่) ระเบียนข้อมูลนี้จะไม่ถูกเชื่อม (โททอลจะจัดการ การเชื่อมของเส้นทางลูกโซ่เอง) คีย์ต่าง ๆ ตามเส้นทางเชื่อมต่อจะต้องมีอยู่ในระเบียนข้อมูลหลักที่อยู่ในเส้นทางลูกโซ่เดียวกันแล้ว

๓. คำสั่ง ADDVB (Add Variable Before)

ทำหน้าที่คล้ายกับคำสั่ง ADDVA เพียงแต่ระเบียนข้อมูลใหม่จะเพิ่มเข้าไปหน้าตำแหน่งที่ระบุใน reference



รูปที่ ๓.๔.๖ Variable record chain ก่อนและหลังการเพิ่ม



รูปที่ ๓.๔.๗ Variable record chain ก่อนและหลังการถูกลบทิ้ง

๑.๒.๒ :คำสั่งประเภทลบข้อมูลเก่าออก

๑. คำสั่ง DEL-M

หน้าที่คือ ลบข้อมูลเก่า (ทางตรรก) ออกจากแฟ้มข้อมูลหลักด้วยการทำให้เป็นช่องว่าง ทั้งระเบียนข้อมูล เพื่อให้เป็นที่ว่างสำหรับระเบียนข้อมูลใหม่ Control key ที่ระบุจะถูกคำนวณหาตำแหน่งที่ระเบียนข้อมูลที่ต้องการลบอยู่ ก่อนที่จะใช้คำสั่งนี้ต้องแน่ใจว่า ตัวระเบียนข้อมูลแปรเปลี่ยนที่เชื่อมกับตัวระเบียนข้อมูลหลักนี้ ถูกลบทิ้งออกจากแฟ้มข้อมูลแปรเปลี่ยนหมดแล้ว (ต้องสั่งให้ลบหมดเอง) มิฉะนั้น ระเบียนข้อมูลหลักจะไม่ถูกลบ นอกจากนี้ต้องทำ

- ควรจะอ่านข้อมูลออกมาตรวจสอบดูก่อนว่าใช่ระเบียนข้อมูลที่ต้องการลบหรือไม่ (กันพลาด)
- ไม่ควรใช้คำสั่ง RDNXT ร่วมกับคำสั่งนี้ เพราะคำสั่งนี้จะทำให้ระเบียนข้อมูลต่าง ๆ (หลังระเบียนข้อมูลที่ถูกลบ) ถูกจัดที่ใหม่ (relocate) เพื่อประสิทธิภาพและประหยัดที่ ดังนั้นคำสั่ง RDNXT จะเลื่อน (SCAN) อ่านผิดจากที่ต้องการได้

๒. คำสั่ง DELVD (Delete Variable Direct)

ทำหน้าที่ลบข้อมูลเก่า (ในเชิงตรรก) ออกจากแฟ้มข้อมูลแปรเปลี่ยน ด้วยการทำให้เป็นช่องว่างทั้งระเบียนข้อมูล เพื่อให้เป็นที่ว่างสำหรับการเพิ่มระเบียนข้อมูลใหม่ ตัวจุดอ้างอิงภายใน ใน reference จะกำหนดตำแหน่งของระเบียนข้อมูล, เส้นทางเชื่อมต่อทั้งหมดในระเบียนข้อมูลนั้นจะถูกแก้ไขให้ไหลถูกต้องโดยอัตโนมัติ

๑.๒.๓ :คำสั่งประเภทแก้ไขข้อมูลเก่า

๑. คำสั่ง WRITM

ทำหน้าที่แก้ไขข้อมูลในระเบียนข้อมูลหลัก โดยทำตามลำดับดังนี้

- หาตำแหน่งของระเบียนข้อมูลโดยคำนวณจาก Control key ที่ระบุไป แล้วหยิบข้อมูล
- เขียนข้อมูลใหม่ที่ระบุลงไปใน Data-area
- คืนระเบียนข้อมูลไว้ที่ตำแหน่งเดิม

คำสั่งนี้ไม่สามารถแก้ไขคีย์, เส้นทางเชื่อมต่อได้ หากต้องการทำต้องลบแล้วเพิ่มทั้งระเบียนข้อมูลหรือใช้คำสั่ง

๒. คำสั่ง WRITV

ทำหน้าที่คล้ายคำสั่ง WRITM เพียงแต่ใช้สำหรับแฟ้มข้อมูลแปรเปลี่ยนและตำแหน่งของระเบียนข้อมูลที่จะลบ ถูกกำหนดด้วยจุดอ้างอิงภายใน ใน reference ที่ระบุไป

๓. คำสั่ง ADDVR (Add Variable Replace)

ทำหน้าที่เปลี่ยนเส้นทางลูกโซ่ของระเบียนข้อมูลในแฟ้มข้อมูลแปรเปลี่ยน (เชิงตรรก) จุดอ้างอิงภายใน ใน reference จะระบุถึงตำแหน่งของระเบียนนั้น จากนั้นคีย์ควบคุมใหม่ใน data-area จะถูกตรวจสอบเทียบกับของเดิมในระเบียนข้อมูล ถ้าค่าของคีย์ต่างกัน ระเบียนข้อมูลนั้นจะถูกทำให้หลุดออกจากเส้นทางลูกโซ่เก่าและจะถูกใส่ในเส้นทางลูกโซ่ใหม่ตามคีย์ควบคุมใหม่เป็นระเบียนสุดท้าย (ตามการไหลของเส้นทางลูกโซ่) ถ้าค่าของคีย์เหมือนกัน เส้นทางลูกโซ่เก่าก็ยังอยู่ นอกจากนี้ยัง

- สามารถเปลี่ยนได้หลายเส้นทาง ลูกโซ่นั่นคือ กำหนดไปหลายคีย์
- สามารถเปลี่ยนรหัสระเบียนข้อมูลได้ ซึ่งจะให้ผลในลักษณะเพิ่มหรือลดเส้นทางเชื่อมต่อ

๑.๒.๔ คำสั่งประเภทกระทำให้ได้ข้อมูลกลับมาหลังจากถูกทำลาย

๑. คำสั่ง WRITD

หน้าที่คือ กระทำให้ได้ข้อมูลกลับคืนมาหลังถูกทำลาย (recovery) (ใช้คำสั่งนี้ร่วมกับ Utilities ในการทำ) ขึ้นมาใหม่จากตัว "log image" (ที่เก็บข้อมูลซึ่งได้เปลี่ยนแปลง) โดยปกติคำสั่งนี้จะไม่ใช้กันในงานโปรแกรมธรรมดา ที่ไม่ได้เกี่ยวข้องกับงานลักษณะทางกายภาพและโครงสร้างของระเบียนข้อมูล

๑.๓ กลุ่มคำสั่งชนิดเริ่มต้น/จนปิด แบ่งออกเป็นกลุ่มตามหน้าที่ได้ดังนี้

- เริ่ม/เลิกการทำงานของงาน (Task) ได้แก่คำสั่ง SINON, SINOF
- เริ่ม/เลิกการทำงานของแฟ้มข้อมูล ได้แก่ คำสั่ง OPENX, CLOSX, CNTRL

อนึ่ง ถ้าอยู่ในลักษณะงานแบบทางสายประเภท Teleprocessing cuvironment แล้ว โปรแกรมคำสั่งงาน หรือโปรแกรมสำเร็จรูปที่เป็นมาตรฐานของระบบเครื่อง (System Standard Routines) จะไม่ต้องเกี่ยวข้องกับกลุ่มคำสั่งเหล่านี้เลย แต่ถ้าอยู่ในงานแบบท์แล้วต้องมีการเขียนคำสั่งเหล่านี้เอาเอง

๑.๓.๑ [illegible]คำสั่งเริ่ม/เลิกการทำงานของงาน

๑. คำสั่ง SINON (Sign-on)

ทำหน้าที่ต่าง ๆ ที่สามารถให้งานใด ๆ (กำหนดใน task) เริ่มทำงานได้ กระทำมากหรือน้อย ขึ้นอยู่กับภายใต้สภาวะใด (environment) ดังที่กล่าวตอนต้น สิ่งที่กระทำได้แก่

- ทำการกำหนดที่ (addressing หรือ load)หรือใส่ตัว TOTAL/7 Nucleus และโปรแกรมดีบีดี (ที่ถูกระบุใน dbdmod) อย่างพลศาสตร์ (Dynamic)
- กำหนดออปชั่นของการล็อกค์ (log) (กำหนดใน Option), ชนิดของการจัดการ (retrieve, update, recovery เป็นต้น) จะกำหนดใน Acess, และอื่น ๆ
- ทำการใส่ (load) ตัวติดต่อ (interface) ล็อกค์และโปรแกรมย่อยต่าง ๆ อย่างพลศาสตร์

อนึ่ง

- การล็อกค์เป็นการบันทึกข้อมูล ณ จุดที่กำหนดในระหว่างการทำงาน เช่น ก่อนการแก้ไขข้อมูล หลังการแก้ไขข้อมูล เป็นต้น
- แฟ้มข้อมูลล็อกค์จะถูกเปิดทุกครั้งที่ทำงานตามคำสั่ง SINON

๒. คำสั่ง SINOF (Sign-off)

ทำหน้าที่ตรงกันข้ามกับคำสั่ง SINON ดังนี้

- ปิดแฟ้มข้อมูลทั้งหมดที่ได้เปิดมา รวมทั้งแฟ้มข้อมูลล็อกด้วย
- ปลดปล่อย งานนั้นจากโททอล, "logging options", modules ต่าง ๆ ที่ได้ใส่เข้ามา และทำการติดต่อ (interface) ทั้งหมด

๑.๓.๒ คำสั่งเริ่ม/เลิกการทำงานของแฟ้มข้อมูล

๑. คำสั่ง OPENX

หน้าที่คือ เปิดแฟ้มข้อมูลหลักและ/หรือ แฟ้มข้อมูลแปรเปลี่ยน (กำหนดชื่อเป็น list ได้ในตัว file) ที่ได้ให้ไว้ในโปรแกรมย่อยดีบีดี ให้อ่านได้อย่างเดียวสำหรับงานนั้น อนึ่งหน้าที่ของคำสั่งนี้เป็นส่วนหนึ่งของคำสั่ง CNTRL (reserve) การเปิดจะทำอย่างตรรกก่อน ถ้าทำได้ (กรณีแฟ้มข้อมูลเคยเปิดมาแล้ว) แต่ถ้าไม่ได้ ก็จะทำอย่างกายภาพ เช่น มีการส่ง (issue) คำสั่งเปิดของโปรแกรมคำสั่งเครื่อง (O.S.) และทำการตรวจสอบหมายเลข (check lebel) เป็นต้น

๒. คำสั่ง CLOSX

ทำหน้าที่ตรงข้ามกับคำสั่ง OPENX คือทำการปิดแฟ้มข้อมูล การทำอย่างตรรกนั้นจะทำเครื่องหมายว่าได้ปิดแล้ว (โดยการตั้งค่า lock byte ในส่วนของระเบียนข้อมูลควบคุมแฟ้มข้อมูล (file control record) ของแต่ละแฟ้มข้อมูลที่กำหนด) ส่วนการทำอย่างกายภาพนั้น จะมีการสั่งคำสั่งการปิดจากโปรแกรมคำสั่งเครื่อง อนึ่ง หน้าที่ของคำสั่งนี้เป็นส่วนหนึ่งของคำสั่ง SINOF (ซึ่งปิดแฟ้มข้อมูลอย่างกายภาพด้วย)

๓. คำสั่ง CNTRL (release)

ทำหน้าที่ปลดเปลื้องแฟ้มข้อมูล (ที่ระบุใน qualifier) ซึ่งงานนี้ได้จับจองไว้ในการทำการแก้ไขข้อมูล ทำได้โดยเขียนจาก ไอ/โอ บัฟเฟอร์ (I/O Buffer) ของแฟ้มข้อมูล (ที่ได้ทำการแก้ไขข้อมูลแล้ว) ลงไปบนตัวฐานข้อมูลจริง ๆ จากนั้นงานนี้ก็จะทำการอ่านแฟ้มข้อมูลเหล่านี้ได้เท่านั้น โดยเปิดโอกาสให้งานอื่น ได้รับจองบ้าง มักใช้คำสั่งนี้ในงานประเภทแบชท์ ส่วนงานประเภท "multi-tasking" การปลดเปลื้องจริง ๆ นี้จะทำต่อเมื่อไม่มีงานอื่นใช้แฟ้มข้อมูลเหล่านี้

อนึ่ง ปกติคำสั่งนี้ไม่ใช้กับงานทั่วไป เนื่องจากสามารถเปลี่ยนแปลงข้อมูลในฐานข้อมูลตัวจริงได้ การใช้คำสั่งนี้จึงต้องอยู่ในความรับผิดชอบของผู้บริหารฐานข้อมูล เพราะต้องมีการแจ้งให้งานอื่นที่เข้าถึงแฟ้มข้อมูลเหล่านั้นทราบ หากมีการเปลี่ยนแปลงความสัมพันธ์ของข้อมูล

๔. คำสั่ง CNTRL (release)

ทำหน้าที่ตรงข้ามกับคำสั่ง CNTRL (release) และคล้ายกับคำสั่ง OPENX เพียงแต่เป็นการจับจองแฟ้มข้อมูลที่ระบุใน qualifier เพื่อทำการแก้ไขข้อมูล ดังนั้นงานอื่น ๆ จึงไม่สามารถเข้าไปทำการแก้ไขข้อมูลได้ แต่อ่านได้ การเปิดจะทำอย่างตรรกก่อน ถ้าทำได้ (โดยไปตั้งค่าใน lock byte ในระเบียนข้อมูลควบคุมแฟ้มข้อมูล เพื่อยึด) ถ้าไม่ได้ จึงจะทำการเปิดอย่างกายภาพ (จะต้องเคยเปิดอย่างกายภาพมาก่อนและได้ถูกปิดอย่างตรรกไปแล้ว จึงจะเปิดอย่างตรรกได้อีก)

๔. คำสั่ง CNTRL (Share)

ทำหน้าที่คล้ายคำสั่ง CNTRL (reserve) คำสั่งนี้ใช้ได้ต่อเมื่อแฟ้มข้อมูลต้องไม่เกิดความสับสน

- ใน Sigle Task Mode คำสั่งนี้เหมือนคำสั่ง CNTRL (reserve) คือให้เอกสิทธิ์แก่งานใดงานหนึ่งเท่านั้น
- ใน multi-tasking mode การแก้ไขข้อมูล จะยอมให้หลาย ๆ งานใช้แฟ้มข้อมูลร่วมกันได้ในขณะนั้น

๑.๔ ชนิดของคำสั่งประเภทควบคุม

ปกติคำสั่งกลุ่มนี้ ไม่ใช้งานโดยทั่ว ๆ ไป ผู้บริหารฐานข้อมูลและวิศวกรระบบมักเป็นผู้ใช้ เนื่องจากเป็นคำสั่งที่ทำการควบคุม ดูแลตัวฐานข้อมูล และเครื่องมืออุปกรณ์ของมัน คำสั่งกลุ่มนี้ ได้แก่ CNTRL (reserve record, release record, inform, listen, logmark, logquiet, purge)

๑. คำสั่ง CNTRL (reserve record)

ทำหน้าที่จับจองระเบียนข้อมูลในแฟ้มข้อมูลต่าง ๆ อย่าง explicit ใช้ได้ในงานประเภท multi-tasking mode เท่านั้น

อนึ่ง
- ระเบียนข้อมูลที่ต้องการนี้ต้องถูกจับจองอย่าง implicit มาก่อนโดยการอ่าน
- คำสั่งนี้กระทำกับ ๑ ระเบียนข้อมูลต่อ ๑ แฟ้มข้อมูลเท่านั้น เพราะการอ่านกระทำได้เพียงครั้งละ ๑ ระเบียนข้อมูล
- เมื่อใช้คำสั่งนี้แล้ว ระเบียนข้อมูลนั้นจะไม่สามารถถูกเข้าถึง โดยงานอื่นจนกว่าใช้คำสั่ง CNTRL (release record) หรือได้ปิด/ปลดปล่อยแฟ้มข้อมูลที่ระเบียนข้อมูลนั้นอยู่ หรือใช้คำสั่ง

๒. คำสั่ง CNTRL (release record)

ทำหน้าที่ส่งข่าวสาร (ระบุในฟิลด์ที่กำหนดใน data-area) จากงานหนึ่งไปงานอื่น หรืองานของผู้บริหารฐานข้อมูลในขอบเขต (region) หรือ partition เดียวกัน คำสั่งนี้ใช้ได้ใน multi-tasking เท่านั้น และไม่เกี่ยวข้องกับแฟ้มข้อมูล ปกติจะไม่ใช้ในงานทั่วไป (ถ้าจะใช้ต้องผ่านผู้บริหารฐานข้อมูล)

๓. คำสั่ง CNTRL (inform)

ทำหน้าที่ส่งข่าวสาร (ระบุในพื้นที่ที่กำหนดใน data-area) จากงานหนึ่งไปงานอื่น หรือ งานของผู้บริหารฐานข้อมูล ในขอบเขตหรือ partition เดียวกัน คำสั่งนี้ใช้ได้ใน multi-tasking เท่านั้น และไม่เกี่ยวข้องกับแฟ้มข้อมูล ปกติจะไม่ใช้ในงานทั่วไป ถ้าจะใช้ต้องขออนุญาตจากผู้บริหาร ฐานข้อมูล

๔. คำสั่ง CNTRL (listen)

ทำหน้าที่รับข่าวสารจากการสั่งด้วยคำสั่ง CNTRL (inform) จากงานของผู้บริหาร ฐานข้อมูล หรืองานอื่น ๆ ทั้งนี้ต้องมีการตรวจสอบสถานะภาพ (status) ก่อนว่าส่งมาหรือยัง

๕. คำสั่ง CNTRL (logmark)

ทำหน้าที่เขียนข้อมูล ๑ ระเบียนข้อมูลของผู้ใช้ลงไปในแฟ้มข้อมูลล็อกค์ เพื่อให้เป็นเครื่องหมาย check point ของตัวเองหรือเป็นเอกสารอ้างอิง ข้อมูลนั้นอยู่ในพื้นที่ของผู้ใช้ (ที่ระบุใน data-area) ชื่อของงานและเวลาจะอยู่หน้าข้อมูลนั้น (ความยาวของข้อมูลกำหนดใน length ในรูปของจำนวน ระเบียนข้อมูลล็อกค์ที่จะใช้)

๖. คำสั่ง CNTRL (logquiet)

ทำหน้าที่ - บันทึกระเบียนข้อมูลที่ได้ทำการแก้ไขไปแล้ว

- แก้ไขค่าในตัว cylinder control records ในฐานข้อมูล
- ทำเครื่องหมาย check point ไว้ด้วยการเขียน "quiet record" ลงในแฟ้มข้อมูลล็อกค์ ทั้งนี้ยังสามารถใส่ข้อมูลของผู้ใช้เองตามลงไปอีกได้ เช่นเดียวกับคำสั่ง CNTRL (logmark)

คำสั่งนี้จะกระทำต่อหลายแฟ้มข้อมูลได้ ส่วนการทำเพื่อให้ได้ข้อมูลคืนมาหลังจากถูกทำลาย ก็จะมาเริ่มที่จุด "quiet" นี้เอง

๗. คำสั่ง CNTROL (purge)

ทำหน้าที่โยกย้ายโปรแกรมย่อย (module) จากรายการของ moduler ที่ active และล้าง (purge) modules ที่สอดคล้องออกอย่างกายภาพ คำสั่งนี้อาจจะใช้เมื่ออยู่ใน partition ที่เล็กและพบว่า ตัว coreไม่พอ แต่ประสิทธิภาพที่ได้จะลดลง เพราะอาจมีการใส่เข้ามาอีก ถ้าต้องการ (reload) และต่อไปนี้เป็น modules ที่ล้างออกไม่ได้ : TOTAL modules, DBD module, I/O modules, logging module

๓.๕ การเปรียบเทียบโครงสร้างทางกายภาพ

เนื่องจากดีบีทีจีเป็นการเขียนขึ้นเพื่อเป็นแนวความคิด จึงแสดงโครงสร้างทางกายภาพไว้เพียงว่าเป็นแบบใช้ตัวบ่งชี้ (pointer array) ดังนั้นในหัวข้อนี้จะกล่าวถึงเฉพาะไอเอ็มเอสและโททอล

๓.๕.๑ โครงสร้างทางกายภาพของไอเอ็มเอส

การใช้ ไอเอ็มเอสกับฐานข้อมูลนั้น มีวิธีการหลายอย่างเพื่อให้ได้ข้อมูลนั้น ๆ มา และในที่นี้จะอธิบายถึงขั้นตอนและวิธีการที่ ไอเอ็มเอส ใช้ในการเข้าถึงฐานข้อมูลนั้น ด้วยเหตุที่ฐานข้อมูลไอเอ็มเอส นั้นใช้ลักษณะความสัมพันธ์ของข้อมูลในรูปของการจัดแบบแผนภูมิต้นไม้ ดังนั้น ในส่วนนี้จะอธิบายศัพย์ที่ประกอบเป็นการจัดแฟ้มข้อมูลของฐานข้อมูลแบบมีลำดับขั้น (Hierarchical Data Base Organization) ก่อน

๓.๕.๑.๑ ศัพท์พื้นฐาน

๑. เซ็กเม้นท์ (Segment) คือ กลุ่มของข้อมูลซึ่งมีความสัมพันธ์ในลักษณะคล้าย ๆ กัน เช่น กลุ่มชื่อของวิศวกรที่จบการศึกษาทางด้านคอมพิวเตอร์ เป็นต้น ในแต่ละเซ็กเม้นท์ที่เก็บในสื่อกลางสำหรับเก็บข้อมูลนั้นจะแบ่งออกเป็น ๒ ส่วน คือ

๑.๑ ส่วนควบคุมข้อมูลของระบบ คือ ส่วนที่ ไอเอ็มเอส ใช้ควบคุมข้อมูลเหล่านั้น

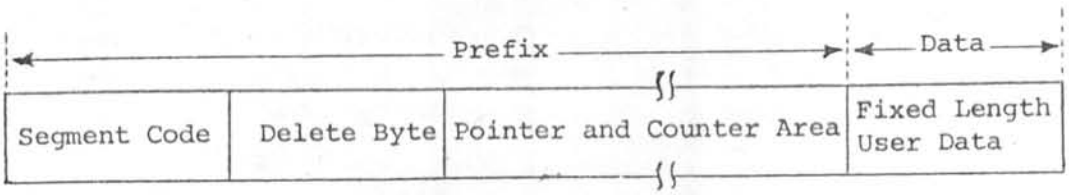
๑.๒ ส่วนของข้อมูล (Data Portion) คือ ข้อมูลที่โปรแกรมต่าง ๆ จะนำไปใช้งาน ซึ่งความยาวของส่วนข้อมูลนั้นต้องคงที่ ยกเว้น เราจะเข้าถึงข้อมูลโดยวิธี HIDAM & HDAM เท่านั้น ซึ่งอาจมีขนาดความยาวคงที่ หรือแปรเปลี่ยนได้ ในกรณีที่เป็นแบบความยาวแปรเปลี่ยน ขนาดของความยาวจะแปรเปลี่ยนตั้งแต่ค่าต่ำสุดจนถึงค่าสูงสุดที่ได้กำหนดไว้และขนาดของความยาวที่กำหนดไว้นั้น ต้องไม่ยาวเกินกว่าระเบียนข้อมูลเชิงกายภาพของสื่อเก็บข้อมูล (Device) ที่เก็บฐานข้อมูลนั้น

สำหรับตำแหน่งของเซ็กเม้นท์นั้นจะเป็นตัวบอกถึงความสัมพันธ์ของเซ็กเม้นท์นั้นกับฐานข้อมูลในลักษณะของการจัดโครงสร้างแบบมีลำดับขั้น

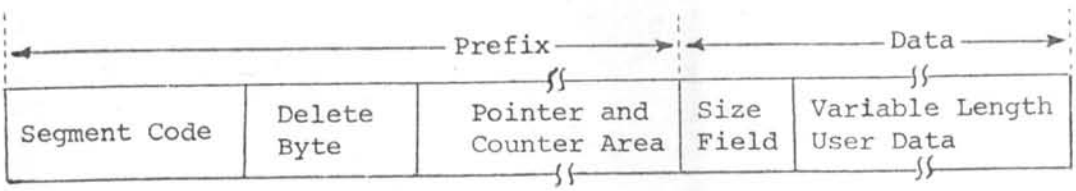
๒. โครงร่างของเซ็กเม้นท์ (Segment Format) มีอยู่ ๒ ชนิดคือ

๒.๑ เซ็กเม้นท์ที่มีความยาวคงที่ (Fixed Length Segment)

๒.๒ เซ็กเม้นท์ที่มีความยาวแปรเปลี่ยน (Variable Length Segment)



รูปที่ ๓.๕.๑ เซ็กเม้นท์ที่มีความยาวคงที่



รูปที่ ๓.๕.๒ เซ็กเม้นท์ที่มีความยาวแปรเปลี่ยน

๓. รหัสของเซ็กเม้นท์ (Segment Code) เพื่อเป็นการบอกชื่อของเซ็กเม้นท์ในฐานข้อมูลมีขนาด ๑ ไบท์ ในส่วนควบคุมข้อมูลของระบบซึ่งจะเก็บเลขตั้งแต่ ๐ ถึง ๒๕๕ ที่จะใช้แทนชื่อของเซ็กเม้นท์นั้น ๆ ซึ่งตัวเลขนี้จะถูกกำหนดให้เรียงจากน้อยไปมาก โดยเริ่มจากรูทเซ็กเม้นท์ (Root Segment) และเรื่อยไปตามลำดับในโครงสร้างแบบมีลำดับขั้น

๔. Delete Byte ไบท์นี้ใช้โดย ไอเอ็มเอส ในการบอกสถานะภาพ (Status) ของเซ็กเม้นท์ นั้น

๕. ฟิลด์ (fields) คือ ข้อมูลที่เล็กที่สุด ซึ่งสามารถเข้าใจได้ในส่วนของข้อมูล (Data Portion) ของแต่ละเซ็กเม้นท์ ฟิลด์แบ่งได้เป็น ๒ ชนิดคือ Sequence field หรือ Key field ซึ่งมันจะเป็นตัวบอกอันดับของข้อมูล (Occurence) ในเซ็กเม้นท์นั้น และในแต่ละเซ็กเม้นท์จะมี key field ได้เพียงหนึ่งหรือไม่มีเลยก็ได้ อีกชนิดหนึ่งคือ Data field

หรือฟิลด์ที่เป็นข้อมูล ซึ่งทั้ง ๒ ชนิดสามารถอ้างอิงได้ โดยโปรแกรมคำสั่งงาน

๓.๕.๑.๒ การจัดแฟ้มข้อมูลในฐานข้อมูลและวิธีการเข้าถึงข้อมูล (Data Base Organizaiton and Access Methods) ไอเอมเอส มีการจัดแฟ้มข้อมูลในฐานข้อมูลเป็น ๒ ชนิด คือ

- การจัดแฟ้มข้อมูลแบบต่อเนื่องอย่างมีลำดับขั้น (Hierarchical Sequential (HS))
- การจัดแฟ้มข้อมูลแบบเข้าถึงโดยตรงอย่างมีลำดับขั้น (Hierarchical Direct (HD))

ซึ่งการจัดแฟ้มข้อมูลทั้งสองนั้นมีวิธีการเข้าถึงข้อมูลดังนี้

การจัดแฟ้มข้อมูลแบบต่อเนื่องอย่างมีลำดับขั้น

1. Simple Hierarchical Sequential Access Method (Simple HSAM)
2. Hierarchical Sequential Access Method (HSAM)
3. Simple Hierarchical Indexed Sequential Access Method (Simple HISAM)
4. Hierarchical Indexed Sequential Access Method (HISAM)

การจัดแฟ้มข้อมูลแบบเข้าถึงโดยตรงอย่างมีลำดับขั้น

1. Hierarchical Direct Access Method (HDAM)
2. Hierarchical Indexed Direct Access Method (HIDAM)

๑. นิยามของการจัดแฟ้มข้อมูลแบบต่อเนื่องอย่างมีลำดับขั้น คือ การเก็บเซ็กเม้นท์ของการจัดแฟ้มข้อมูลแบบนี้ จะเก็บแบบต่อเนื่องกันไป โดยที่ในหนึ่งระเบียนข้อมูลของฐานข้อมูลเชิงกายภาพ (Physical Data Base Record) จะประกอบด้วยจำนวนหลายระเบียนข้อมูลเชิงตรรก (Logical record) จำนวนระเบียนข้อมูลเชิงตรรกที่ประกอบเป็นแต่ละหนึ่งระเบียน

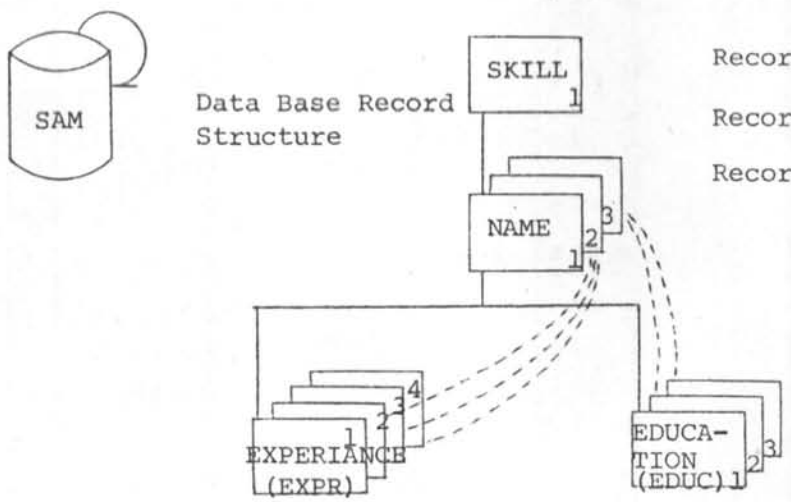
ข้อมูลของฐานข้อมูลเชิงกายภาพนั้นขึ้นอยู่ที่ว่าเราจัดแฟ้มข้อมูลแบบใด (HSAM หรือ HISAM)

๑.๑ การเข้าถึงข้อมูลแบบต่อเนื่องอย่างมีลำดับขั้นอย่างง่าย (Simple HSAM)

การจัดแฟ้มข้อมูลแบบนี้จะใช้กับฐานข้อมูลที่มีเฉพาะรูทเซ็กเมนท์เท่านั้น โดยการเก็บข้อมูลในแฟ้มข้อมูลนั้นจะนำข้อมูลเก็บแบบต่อเนื่อง (Sequential) ลงในระเบียนข้อมูลเชิงกายภาพ ดังนั้นสื่อกลางที่ใช้เก็บฐานข้อมูลชนิดนี้อาจเป็นเทปแม่เหล็กหรือจานแม่เหล็กก็ได้

๑.๒ การเข้าถึงข้อมูลแบบต่อเนื่องอย่างมีลำดับขั้น (HSAM)

การเก็บเซ็กเมนท์นั้นจะเก็บโดยมี ๒ ส่วนคือ ส่วนควบคุมข้อมูลของระบบและส่วนของข้อมูล และเก็บแบบต่อเนื่องลงในระเบียนข้อมูลเชิงกายภาพต่อเนื่องกันไป ส่วนที่ว่างข้างท้ายของระเบียนข้อมูลเชิงกายภาพจะถูกเติมด้วยเลขศูนย์ฐานสอง (Binary Zeros) สื่อกลางที่ใช้สำหรับ HSAM นี้อาจเป็นเทปแม่เหล็กหรือจานแม่เหล็กก็ได้



Record1	SKILL1	NAME1	EXPR1	EDUC1
Record2	NAME2	EXPR2	EXPR3	EXPR4
Record3	NAME3	EDUC2	EDUC3	

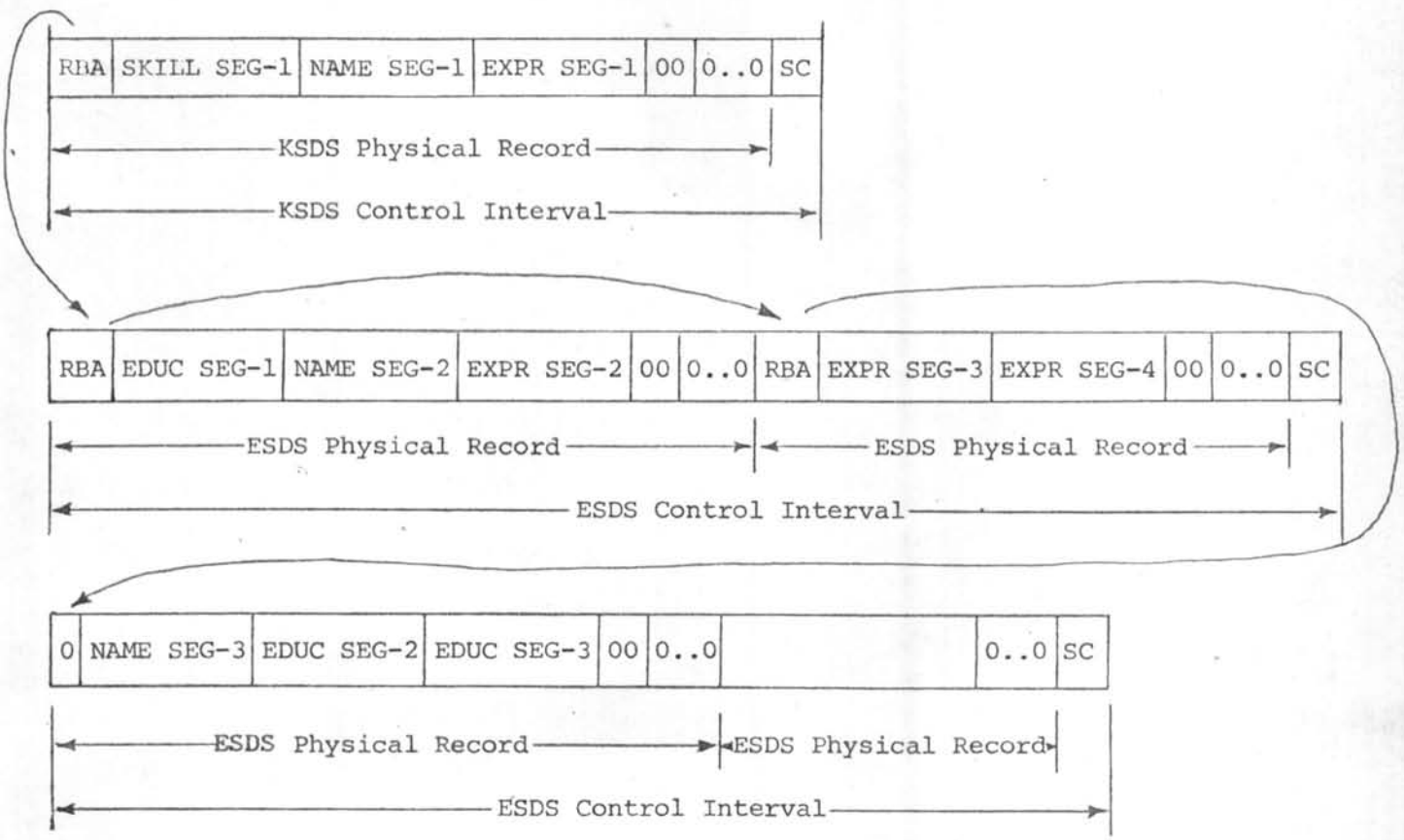
รูปที่ ๓.๔.๓ การจัดเก็บระเบียนข้อมูลของ HSAM

๑.๓ การเข้าถึงข้อมูลแบบดัชนีต่อเนื่องอย่างมีลำดับขั้นอย่างง่าย (Simple HISAM)

ฐานข้อมูลซึ่งใช้การเข้าถึงข้อมูลแบบนี้จะใช้หลักการของการเข้าถึงข้อมูลแบบดัชนีต่อเนื่อง (Indexed Sequential) โดยที่ฐานข้อมูลนี้ต้องมีเฉพาะรูทเซ็กเมนท์เท่านั้น แต่ละระเบียนข้อมูลจะถูกเก็บเฉพาะข้อมูลลงในหนึ่งระเบียนข้อมูลของ DOS/VS VSAM Key Sequenced file (KSDS)

๑.๔ การเข้าถึงข้อมูลแบบดัชนีต่อเนื่องอย่างมีลำดับขั้น (HISAM)

ฐานข้อมูลซึ่งใช้การเข้าถึงข้อมูลแบบนี้จะใช้หลักการของการเข้าถึงข้อมูลแบบดัชนีต่อเนื่อง ซึ่งการจัดฐานข้อมูลแบบนี้ต้องมี ๒ แฟ้มข้อมูลด้วยกันคือ Key Sequenced file (KSDS) และ Entry Sequenced file (ESDS) ซึ่งใน KSDS จะเก็บระเบียนข้อมูลของฐานข้อมูลทุกระเบียนข้อมูลโดยหนึ่งระเบียนข้อมูลของฐานข้อมูล ต่อหนึ่งระเบียนข้อมูลเชิงกายภาพ ในระเบียนข้อมูลเชิงกายภาพนั้น จะประกอบด้วย ๒ ส่วนคือ ส่วนควบคุมข้อมูลของระบบและส่วนของข้อมูล โดยส่วนของข้อมูลนั้นจะเก็บรูทเซ็กเม้นท์และระเบียนข้อมูลอื่นที่ต่อจากนั้นต่อเนื่องกันไปจนเต็มระเบียนข้อมูลเชิงกายภาพนั้น และในส่วนควบคุมข้อมูลของระบบจะเก็บตัวบ่งชี้ (Pointer) เพื่อชี้ไปยังแฟ้มข้อมูล ESDS ของฐานข้อมูลชุดนั้น สำหรับระเบียนข้อมูลที่เหลือที่ตามหลังส่วนที่เก็บในส่วนของข้อมูลของ KSDS แล้วจะถูกเก็บไว้ใน ESDS



รูปที่ ๓.๕.๔ การจัดเก็บระเบียนข้อมูลของ (Logical Record/Block Format)

การแก้ไขปรับปรุง HISAM

๑.๔.๑ การเพิ่มเติมรู้ทเซ็กเม้นท์ จะทำเหมือนกับการเพิ่มระเบียนข้อมูลใน Key Sequence file ของ VSAM คือจะแทรกเซ็กเม้นท์ลงไปเพื่อให้ได้ตามลำดับของโครงสร้างแบบมีลำดับขั้น

๑.๔.๒ การเพิ่มเติมเซ็กเม้นท์ที่อยู่ภายใต้รู้ทเซ็กเม้นท์ สามารถทำได้ ๔ กรณี ดังนี้

ก. กรณีที่มีที่มากพอให้เซ็กเม้นท์นั้นเข้าไปในระเบียนข้อมูลของ KSDS ก็ให้แทรกเซ็กเม้นท์ลงไปตามลำดับที่ควรจะเป็น

ข. เซ็กเม้นท์นั้นสามารถใส่ลงในระเบียนข้อมูลของ KSDS ได้ แต่ต้องย้ายเซ็กเม้นท์อื่นที่ต่อท้ายออกไปแล้วนำไปใส่ในระเบียนข้อมูลเชิงกายภาพอันอื่นต่อไป

ค. กรณีที่เซ็กเม้นท์นั้นไม่สามารถใส่ลงในระเบียนข้อมูลได้ จำต้องเคลื่อนย้ายเซ็กเม้นท์ต่อท้ายรวมทั้งเซ็กเม้นท์ที่จะเอามาเพิ่มใหม่ไปใส่ไว้ในระเบียนข้อมูลเชิงกายภาพอันอื่นต่อไป

ง. กรณีที่เซ็กเม้นท์ที่ทำการเพิ่มเติมใหม่นั้นยาวมากไม่สามารถใส่ในระเบียนข้อมูลเดิมได้ ต้องเคลื่อนย้ายเซ็กเม้นท์ต่อท้าย รวมทั้งเซ็กเม้นท์ที่ต้องการเพิ่มใหม่ไปไว้ในระเบียนข้อมูลเชิงกายภาพใหม่ จากนั้นเซ็กเม้นท์ที่ต่อท้ายเซ็กเม้นท์ที่ทำการเพิ่มเติมใหม่ก็ไม่สามารถใส่ลงในระเบียนข้อมูลเชิงกายภาพที่เพิ่มขึ้นใหม่ได้ จึงต้องเพิ่มระเบียนข้อมูลเชิงกายภาพขึ้นใหม่อีก

๑.๔.๓ การลบข้อมูลทั้งหรือการเลิกใช้ข้อมูลส่วนนั้น จะทำโดยการตั้งสัญญลักษณ์แสดงการเลิกใช้ (Set Delete flag) ภายในส่วนควบคุมข้อมูลของระบบของเซ็กเม้นท์เท่านั้น ซึ่งจะทำให้ชายต์เซ็กเม้นท์เชิงกายภาพ (Physical child segment) ของมันทั้งหมดถูกเลิกใช้ไปด้วย เซ็กเม้นท์ที่ถูกเลิกใช้นี้ไม่สามารถจะนำมาใช้ใหม่ได้ (แต่ที่ตรงนั้นยังมีข้อมูลอยู่ไม่ได้เป็นที่ว่าง)

๒. นิยามของการจัดแฟ้มข้อมูลแบบเข้าถึงโดยตรง คือ เซ็กเม้นท์ของการจัดแฟ้มข้อมูลแบบนี้จะเก็บไว้ใน ESDS โดยจะมีคีย์ชี้ไปยังเซ็กเม้นท์เหล่านั้นทุกเซ็กเม้นท์

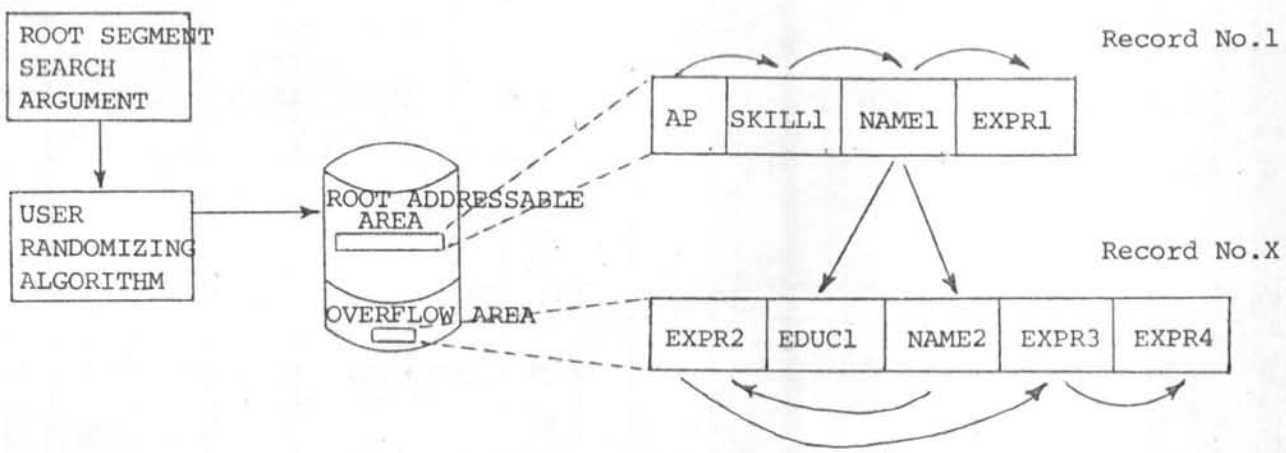
๒.๑ การเข้าถึงข้อมูลแบบเข้าถึงโดยตรงอย่างมีลำดับขั้น (HDAM)

ฐานข้อมูลที่ใช้การเข้าถึงแบบนี้จะมีระเบียนข้อมูลทั้งหมดเก็บอยู่ใน ESDS ฐานข้อมูลแบบ HDAM จะประกอบด้วยสองส่วน คือ

๑. พื้นที่สำหรับเก็บรู้ทเซ็กเม้นท์ (Root segment addressable area) ในส่วนนี้โปรแกรมย่อยสำหรับคำนวณที่อยู่ (Randomizing Module) จะจัดเก็บรู้ทเซ็กเม้นท์ทั้งหมดลงในพื้นที่นี้อีกทั้งจะเก็บค่าจากเซ็กเม้นท์อื่นอีกถ้ามีที่เหลือพอ

๒. พื้นที่สำหรับเก็บข้อมูลส่วนเกิน (Overflow area) เป็นพื้นที่สำหรับเก็บข้อมูลที่เหลือจากการเก็บในพื้นที่ส่วนแรก การใช้

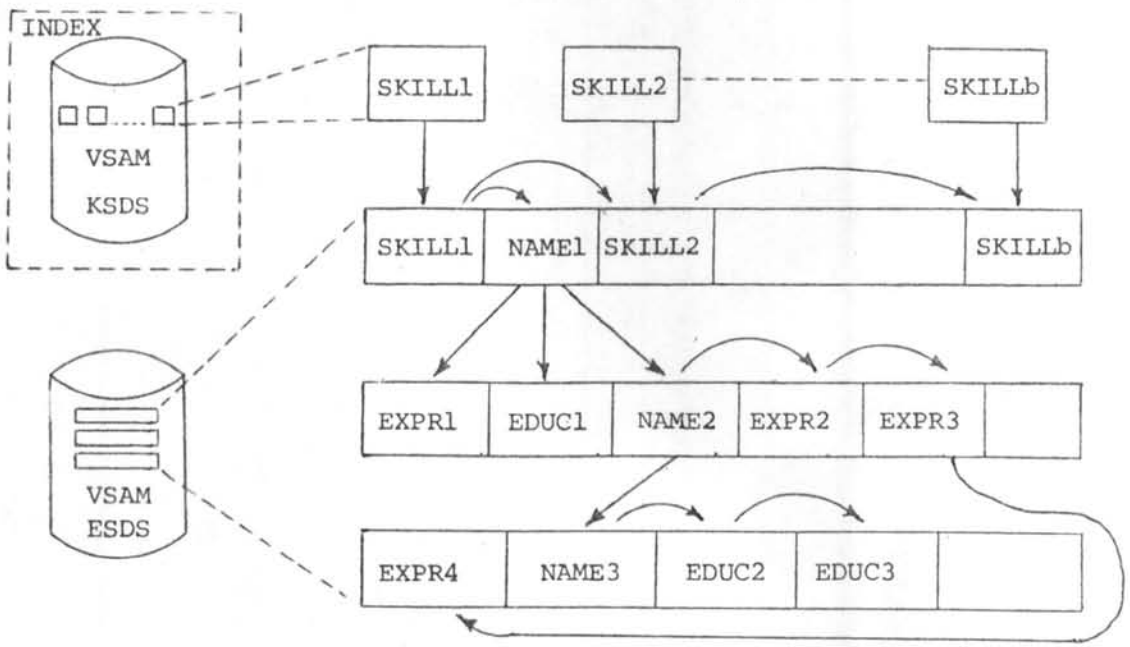
การใช้ฐานข้อมูลแบบ HDAM นี้ ผู้ใช้ต้องกำหนดโปรแกรมย่อยสำหรับคำนวณที่อยู่ให้แก่ไอเอ็มเอสเพื่อใช้ในการเปลี่ยนคีย์ฟิลด์ให้เป็นที่อยู่ (address) เพื่อใช้อ้างอิงถึงข้อมูลซึ่งค่าที่ได้จากการคำนวณมีอยู่สองส่วนคือ Relative Block Number (RBN) ซึ่งใช้บอกว่าเซ็กเม้นท์นั้นเก็บในบลอกคีใด และอีกส่วนคือ Anchor pointer Number (APN) ซึ่งค่านี้จะบอกตำแหน่งของรู้ทเซ็กเม้นท์นั้นในบลอกคีนั้น ๆ



รูปที่ ๓.๕.๕ การจัดเก็บระเบียนข้อมูลของ HDAM

๒.๒ การเข้าถึงข้อมูลแบบดัชนีเข้าถึงโดยตรงอย่างมีลำดับขั้น (HIDAM)

ฐานข้อมูลที่ใช้การเข้าถึงแบบนี้ ตัว HIDAM จะสร้างฐานข้อมูลเพิ่มขึ้นอีกชุดหนึ่งสำหรับ Root sequence โดยเฉพาะซึ่งเรียกว่าฐานข้อมูลดัชนี (Index Data Base) ซึ่งระเบียนข้อมูลเชิงตรรกในฐานข้อมูลดัชนีนั้นจะมีตัวดัชนีบ่งชี้ (Index Pointer) เพื่อชี้ไปยังที่เก็บรู้ทเซ็กเม้นท์ใน ESDS ตัวต่อตัว การสร้างฐานข้อมูลแบบ HIDAM นั้นผู้ใช้ต้องเรียงเซ็กเม้นท์ในลำดับที่เหมาะสม โดยที่ "key sequence" ของรู้ทเซ็กเม้นท์ต้องเรียงจากน้อยไปหามาก (Ascending) กรณีที่มีการเพิ่มรู้ทเซ็กเม้นท์ ฐานข้อมูลดัชนีจะใช้หลักการของ "Twin Pointer" เพื่อบอกถึงระเบียนข้อมูลที่เพิ่มขึ้น



รูปที่ ๓.๕.๖ การจัดเก็บระเบียนข้อมูลของ HIDAM

๒.๓ การเพิ่มเติมข้อมูลลงไปในฐานข้อมูลที่มีการเข้าถึงแบบเข้าถึงโดยตรง

การแก้ไขแฟ้มข้อมูลแบบเข้าถึงโดยตรงอย่างมีลำดับขั้น โดยการเพิ่มเติมข้อมูลมีข้อควรคำนึงถึงหลายประการเพื่อว่าหลังจากการเพิ่มเติมข้อมูลแล้ว แฟ้มข้อมูลยังคงมีประสิทธิภาพใกล้เคียงกับตอนใส่ข้อมูลใหม่ ๆ การเพิ่มเติมข้อมูลลงใน HDAM และ HIDAM ใช้หลักการอันเดียวกัน ปัจจัยที่

นำมาประกอบได้แก่การใช้ Bit Map, CI, space available chain, available length field. กฎทั่วไปในการเพิ่มเติมข้อมูลลงในฐานข้อมูลคือ พยายามเพิ่มเซ็กเม้นท์ใหม่ให้ใกล้กับเซ็กเม้นท์ที่เกี่ยวข้องมากที่สุด เช่น พยายามเพิ่มเติมข้อมูลให้อยู่ในบล๊อกค์เดียวกัน หรือถ้าต้องอยู่บล๊อกค์อื่นก็พยายามให้อยู่บน track หรือ Cylinder เดียวกัน ถ้าหาที่แทรกไม่ได้ก็ต้องไปอยู่ท้ายแฟ้มข้อมูล

๒.๔ การเลิกใช้ข้อมูลหรือลบข้อมูลบางส่วนทิ้งในฐานข้อมูลที่มีการเข้าถึงแบบเข้าถึงโดยตรง มีหลักการ ดังนี้

๑. ตั้งสัญญลักษณ์แสดงการเลิกใช้ที่เซ็กเม้นท์นั้น ๆ แล้วจึงไปลบค่าในส่วนควบคุมข้อมูลระบบของเซ็กเม้นท์ที่มีค่าชี้มายังเซ็กเม้นท์นี้

๒. แก้ไข space available chain ในบล๊อกค์

๓. แก้ไข Bit Map และถ้าเซ็กเม้นท์ที่ถูกเลิกใช้ไปนั้นอยู่ติดกับที่ว่างเดิมก็ให้รวมเป็นที่ว่างอันเดียวกัน

๓.๔.๒ โครงสร้างทางกายภาพของโททอล

โครงสร้างของระบบบริหารฐานข้อมูลโททอลเป็นแบบร่างแห ซึ่งมีความยืดหยุ่นที่จะเชื่อมโยงกันระหว่างแฟ้มข้อมูลต่าง ๆ การเชื่อมโยงกันภายในระบบบริหารฐานข้อมูลเป็นแบบระบบที่เรียกว่า "Embedded Link System" คือข้อมูลในแฟ้มข้อมูลเชื่อมกันด้วยตัวบ่งชี้ (list หรือ link หรือ pointer) ชนิดของความสัมพันธ์แบ่งได้ ๓ แบบ ตามลักษณะของระบบบริหารฐานข้อมูลโดยทั่ว ๆ ไป คือ Embedded Link, Directory Based และ Inverted File/Index ส่วนแฟ้มข้อมูลโททอลแบ่งเป็น ๒ ชนิด คือ

๑. แฟ้มข้อมูลหลัก (Master file) จะใช้กับระเบียบข้อมูลหลัก (Master record)

๒. แฟ้มข้อมูลแปรเปลี่ยน (Variable file) จะใช้กับระเบียนข้อมูลแปรเปลี่ยน (Variable record)

๓.๔.๒.๑ แฟ้มข้อมูลหลัก มีลักษณะโดยสรุป ดังนี้

๑. แต่ละระเบียนข้อมูลในแฟ้มข้อมูลหนึ่งมีแบบฟอร์มเดียวกัน ความยาวเท่ากัน

๒. ระเบียนข้อมูลถูกเก็บอย่างสุ่ม

๓. สามารถเข้าถึงระเบียนข้อมูลได้โดยตรง หรือตามลำดับก็ได้ถ้าระเบียนข้อมูลนั้นผ่านการจัดเรียงมาแล้ว

ดังนั้น ถ้าระเบียนข้อมูลใดมีความยาวเท่ากัน, มีโครงร่างเหมือนกัน, มีการเปลี่ยนแปลงน้อย, มีคีย์เดียวในการเข้าถึงข้อมูล ก็ให้เก็บไว้ในแฟ้มข้อมูลหลัก

โครงร่างของระเบียนข้อมูลหลักเป็น ดังนี้

ROOT	KEY	$DATA_1$	- - - -	$DATA_n$	$LINK_1$	- - - - - - - -	$LINK_m$

ความหมายของแต่ละส่วนในระเบียนข้อมูลหลัก

๑. ROOT เป็นฟิลด์ของระบบเครื่อง (System field) มีไว้ในการณีซึ่งที่อยู่ (Address) ของระเบียนข้อมูลใหม่ถูกคำนวณและกำหนด (Hashing) ให้ตกอยู่ในตำแหน่งทางกายภาพที่มีระเบียนข้อมูลเก่าอยู่แล้ว ตัว ROOT นี้จะเป็นตัวกำหนดให้ไปอยู่ในตำแหน่งทางกายภาพอื่น ๆ โดยจะมีตัวบ่งชี้ (pointer) ชี้ถึงกันด้วย (ตำแหน่งที่ระเบียนข้อมูลต่าง ๆ ตกร่วมกัน เรียกว่า "HOME"

๒. KEY ใช้ฟิลด์นี้ในการเข้าถึง (access) และมีได้เพียงคีย์เดียว

๓. DATA เป็นข้อมูลต่าง ๆ ซึ่งไม่เปลี่ยนแปลงง่าย ๆ

๔. LINK เป็นฟิลด์ของระบบเครื่อง โดยจะมี ๑ เส้นทางเชื่อมโยง (link path) ต่อ ๑ แฟ้มข้อมูลแปรเปลี่ยน โดย ๑ เส้นทางนี้จะเก็บแบ่งเป็น ๒ ส่วน ส่วนแรกจะเก็บหมายเลขระเบียนข้อมูลสัมพันธ์ (relative record number) ของระเบียนข้อมูลแปรเปลี่ยนอันแรก ส่วนที่สองจะเก็บของระเบียนข้อมูลสุดท้ายในเส้นทางลูกโซ่นั้น (chain)

๓.๔.๒.๒ แฟ้มข้อมูลแปรเปลี่ยน มีลักษณะโดยสรุป ดังนี้

๑. มีคีย์หลาย ๆ อันได้ (multiple keys) เพื่อสะดวกในการหยิบ (retrieve) ข้อมูลได้หลาย ๆ ทาง

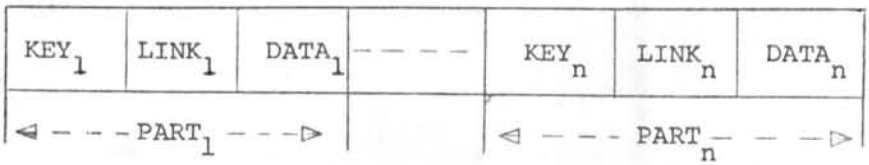
๒. สามารถหยิบข้อมูลที่ต่อรวมกัน (chained) โดยใช้คีย์เพียงอันเดียว

๓. สามารถเพิ่มเติมข้อมูล (add) หรือหยิบข้อมูลเป็นแบบลำดับได้ดี

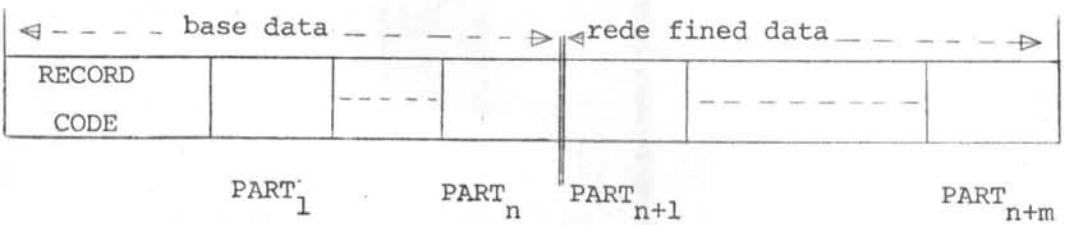
๔. ข้อมูลในเส้นทางลูกโซ่จะเข้าถึงได้โดยต้องผ่านระเบียนข้อมูลหลัก

แฟ้มข้อมูลชนิดนี้จะมีโครงร่างของระเบียนข้อมูลอยู่ ๒ ชนิด ดังนี้

โครงร่างของ STANDARD VARIABLE RECORD (SVR)



โครงร่างของ CODED VARIABLE RECORD (CVR)



STANDARD VARIABLE RECORD มีลักษณะคล้ายระเบียนข้อมูลหลักแต่ต่างกันที่มีได้หลายคีย์ เพื่อที่จะเชื่อมต่อได้กับหลาย ๆ แฟ้มข้อมูลหลัก

CODE VARIABLE RECORD แบ่งเป็น ๒ ส่วน

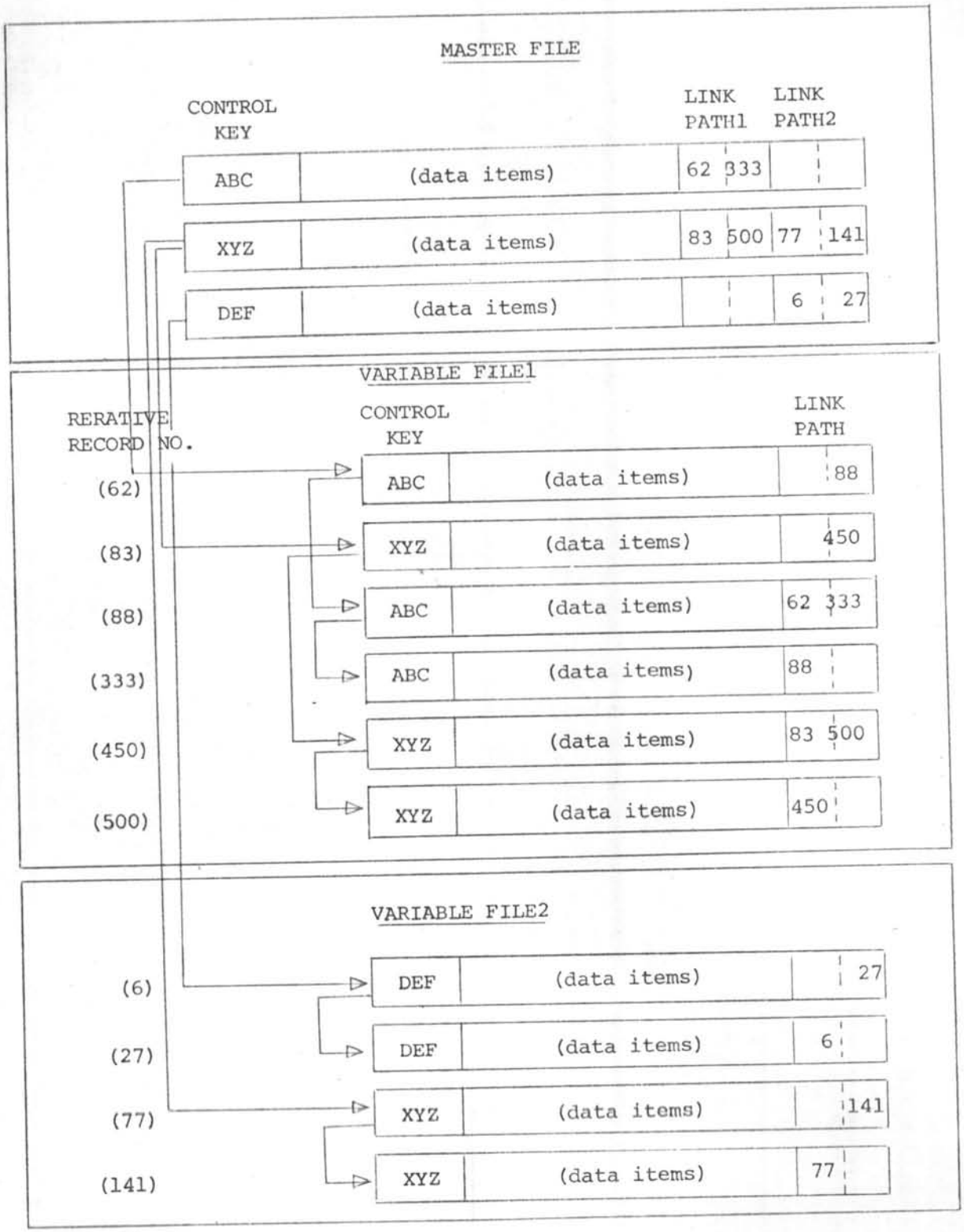
ส่วนแรกเป็นข้อมูลพื้นฐาน (base data) โดยมีความยาวคงที่ (fixed format) ซึ่งทุกระเบียนข้อมูลในแฟ้มข้อมูลนั้น ต้องมีรูปแบบและความยาวเหมือนกัน

ส่วนที่สองเป็น "redefined data" โดยมีลักษณะแปรเปลี่ยน (variable format) ซึ่งรูปแบบและความยาวไม่เหมือนกัน

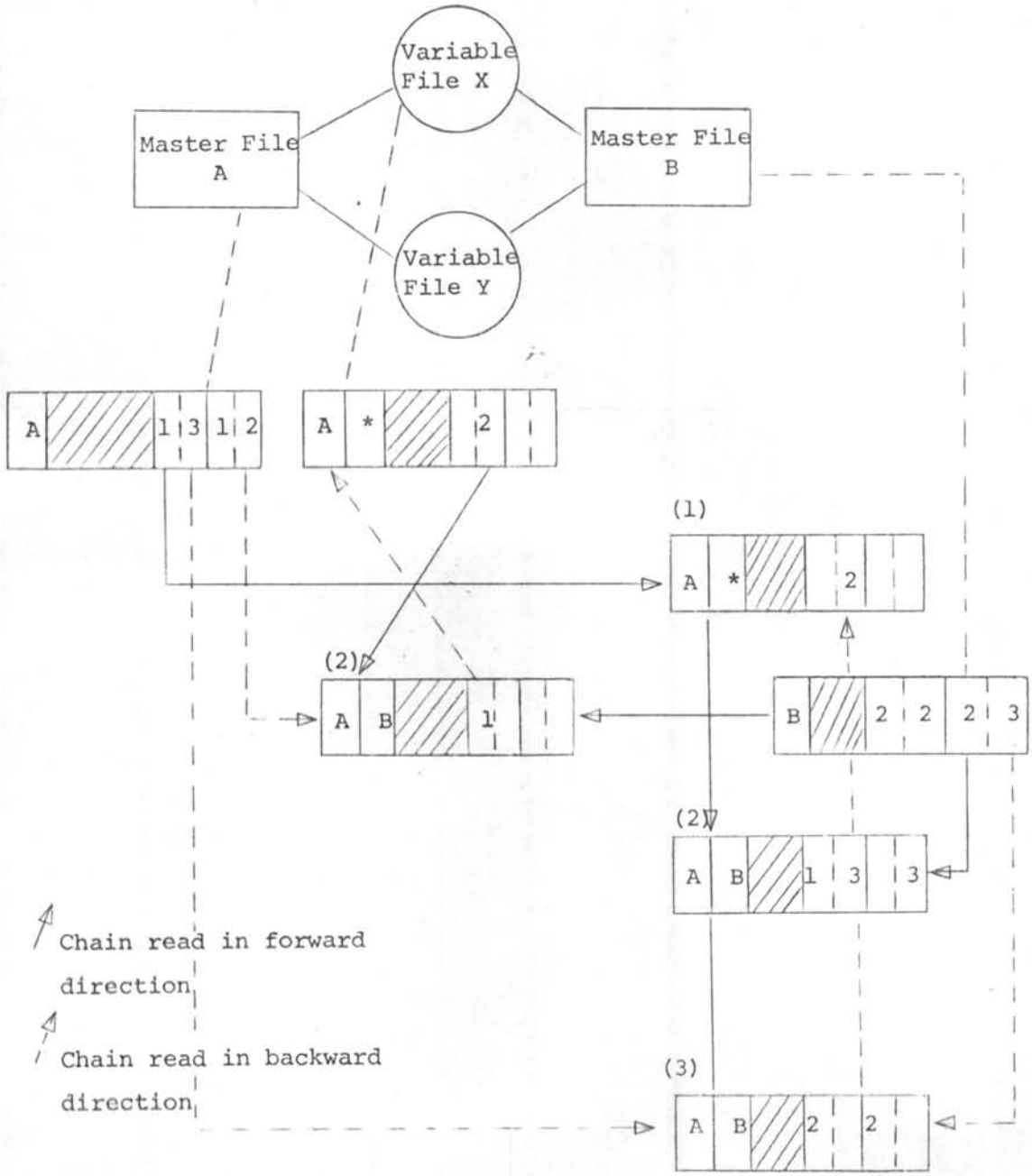
ในแต่ละ PART ประกอบด้วย ๑ คีย์, ๑ เส้นทางเชื่อมต่อ, และข้อมูลต่าง ๆ (ในบาง PART อาจไม่มีข้อมูลเลยก็ได้)

LINK เป็นตัวเชื่อมต่อระเบียนข้อมูลต่าง ๆ ที่มีคีย์ตรงกัน และตรงกับในระเบียนข้อมูลหลัก ซึ่ง LINK แบ่งเป็น ๒ ส่วน ส่วนแรกชี้ไปยังหมายเลขระเบียนข้อมูลซึ่งเป็นแหล่งที่มา (SOURCE) ส่วนหลังเป็นส่วนที่ชี้ไปยังหมายเลขระเบียนข้อมูลปลายทาง (destination) ทำให้สามารถเข้าถึงข้อมูลได้ทั้งตรงไปข้างหน้า (forward) และย้อนกลับ (backward)

RECORD CODE เป็นตัวแบ่งแยกความแตกต่างของส่วน "redefine data" ว่ามีรูปแบบใด



รูปที่ ๓.๕.๗ แสดงถึงการเชื่อมโยงของระเบียนข้อมูลหลักกับระเบียนข้อมูลแปรเปลี่ยน



รูปที่ ๓.๕.๘ แสดงความสัมพันธ์ของแฟ้มข้อมูลหลักและแฟ้มข้อมูลแปรเปลี่ยนชนิดหลาย ๆ แฟ้มข้อมูล

๓.๔.๒.๓ การติดต่อกันระหว่างระบบบริหารฐานข้อมูลกับโปรแกรมคำสั่งงาน

ในรูปที่ ๓.๔.๔ ส่วนล่างได้แสดงขั้นตอนการเรียกใช้ฐานข้อมูลโดยโปรแกรมคำสั่งงาน ดังนี้

๑. คำสั่ง CALL ในโปรแกรมคำสั่งงานจะถูกเริ่มใช้งาน (execute)

๒. คำสั่งควบคุม (control) จะถูกส่งจากโปรแกรมคำสั่งงานไป ดีเอ็มแอล (DML)

๓. พารามิเตอร์ของคำสั่ง CALL ถูกตรวจสอบเพื่อหาว่าจะต้องมีการกระทำสิ่งใดบ้าง

๔. ถ้าต้องการข้อมูลในฐานข้อมูลโปรแกรมย่อยของ ดีบีดี จะถูกส่งเข้ามาในส่วนความจำหลัก เพื่อใช้เป็นที่อ้างอิง (reference) (ข้อมูลในโปรแกรมย่อยของ ดีบีดี นี้จะบ่งลักษณะและความสัมพันธ์ของแฟ้มข้อมูลต่าง ๆ)

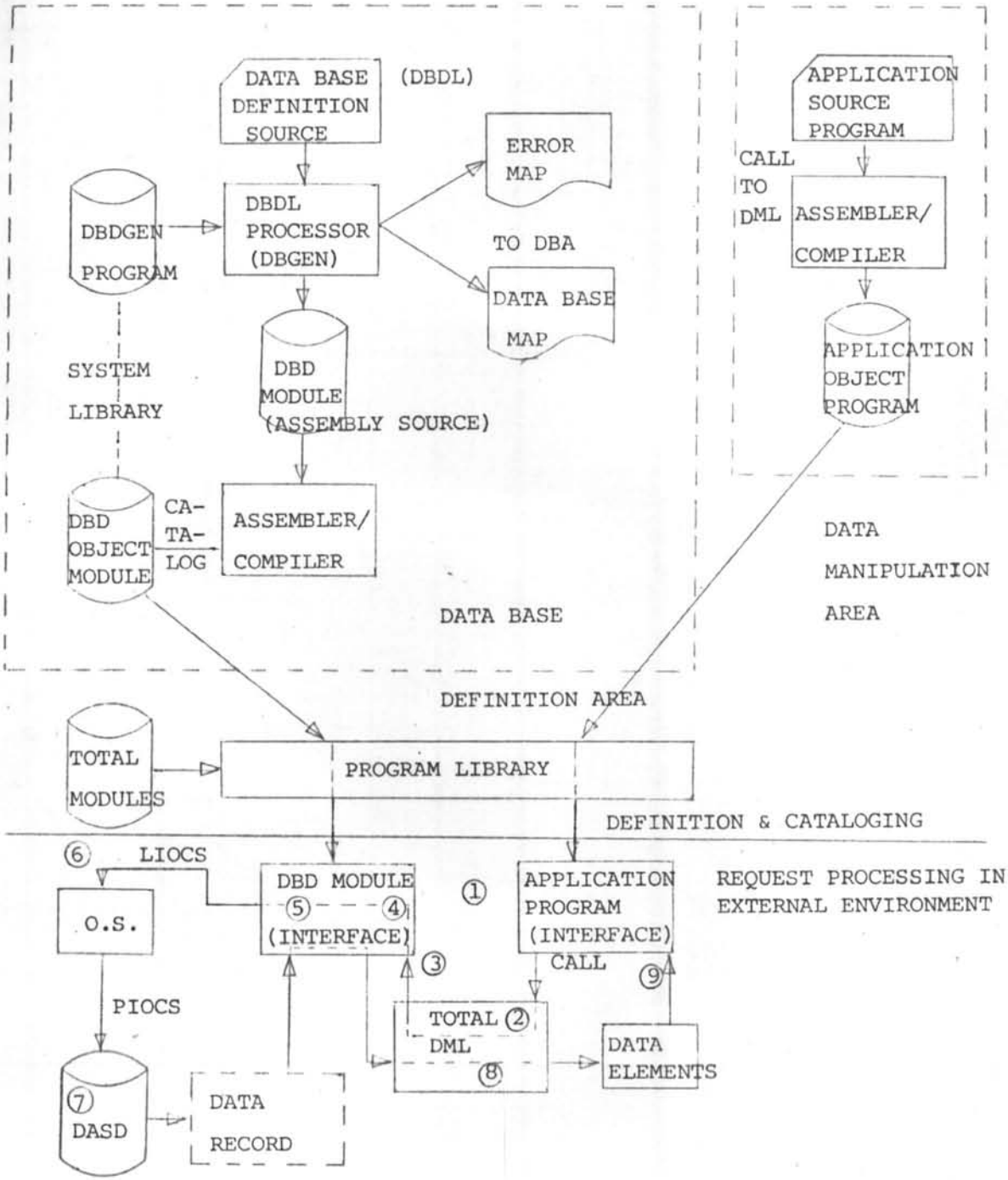
๕. แจ้งความต้องการ อินพุท/เอ้าพุท แก่ระบบโปรแกรมคำสั่งเครื่อง (Operating System) โดยอาศัยข้อมูลในโปรแกรมย่อย ดีบีดี ร่วมกับตัวพารามิเตอร์ของคำสั่ง CALL

๖. โปรแกรมย่อย "LIOCS" จะเป็นตัวกลางติดต่อระหว่างโททอลกับโปรแกรมคำสั่งเครื่อง

๗. โปรแกรมคำสั่งเครื่องจะจัดการในจานแม่เหล็ก โดยผ่านโปรแกรมย่อย "PIOCS" แล้วได้ระเบียนข้อมูลที่ต้องการ ซึ่งจะถูกส่งไปเก็บไว้ในพื้นที่เก็บข้อมูล (buffer) ของโปรแกรมย่อย ดีบีดี

๘. ดีเอ็มแอล จะเลือกข้อมูลตามที่ต้องการของระบบโปรแกรมคำสั่งงาน ส่งไปไว้ในพื้นที่เก็บข้อมูลของโปรแกรมคำสั่งงาน เพื่อที่จะได้ใช้งานต่อไป

๙. โปรแกรมคำสั่งงานรับข้อมูลที่ ดีเอ็มแอล ส่งมาให้



รูปที่ ๓.๕.๙ การติดต่อระหว่างระบบบริหารฐานข้อมูลกับโปรแกรมคำสั่งงาน

๓.๖ การเปรียบเทียบส่วนประกอบในการทำงานร่วมกับระบบจัดการฐานข้อมูล การเปรียบเทียบแสดงในตาราง ๓.๖.๑ มีเพียง ๒ ระบบคือ ไอเอ็มเอส และ โททอล เพราะดีบีทีจีเป็นระบบจัดการฐานข้อมูลที่สร้างขึ้นเพื่อเป็นแนวทางของงานจัดการฐานข้อมูล มิได้มีการสร้างโปรแกรมเพื่อใช้งานจริง ๆ

ตารางที่ ๓.๖.๑ แสดงการเปรียบเทียบส่วนประกอบในการทำงานร่วมกับระบบจัดการฐานข้อมูล

ระบบ	ส่วนประกอบในการทำงานร่วมกับดีบีเอ็มเอส	
	เครื่องคอมพิวเตอร์	ระบบคำสั่งดำเนินงาน
ไอเอ็มเอส	ไอบีเอ็ม ๓๖๐ ๓๗๐, ๓๐๓X, ๔๐ XX	โอเอส, โอเอสวีเอส ไอเอสเอ็มเอฟที (OS/MFT) ไอเอสเอ็มวีที (OS/MVT)
โททอล	ใช้ได้กับเครื่องคอม- พิวเตอร์ ส่วนมากทั้ง มินิคอมพิวเตอร์ คอม- พิวเตอร์ขนาดกลางและ ขนาดใหญ่และของหลาย ๆ บริษัท เช่น เครื่องคอม- พิวเตอร์ของไอบีเอ็มรุ่น๓๖๐ ๓๗๐ และ SYSTEM ๓	ใช้ได้กับระบบคำสั่งดำเนินงาน ที่เหมาะสมกับเครื่องนั้น ดอส, ดอสวีเอส, โอเอส, โอเอสวีเอส