โมเดลข้อมูลเชิงวัตถุ

โมเดลของข้อมูลเป็นการแสดงให้เห็นองค์ประกอบสิ่งต่าง ๆ (objects)ที่เราสนใจ ซึ่งจะมีข้อกำหนดที่กำหนดให้กับองค์ประกอบเหล่านั้น และแสดงความสัมพันธ์ระหว่างองค์ประกอบเหล่านั้น

หลักการของโมเดลข้อมูลเชิงวัตถุจะประกอบไปด้วยองค์ประกอบเหล่านี้คือ

1 object and object identifier: หมายถึงอ๊อปเจ็กใด ๆ ที่อยู่ในรูปแบบที่กำหนด และมีการกำหนดค่า id ที่ไม่ซ้ำกันเพื่อใช้ในการอ้างถึงอ๊อปเจ็กนั้น

2 attributes and methods: อ๊อปเจ็กทุก ๆ อ๊อปเจ็กจะมีคุณลักษณะ(state) คือเซตของค่าข้อมูลของ แอตทริบิวต์ของอ๊อปเจ็ก และการกระทำ(behavior) เป็นเซตของวิธีการ ส่วนของโปรแกรม ซึ่งมีการดำเนินการบนลักษณะของอ๊อปเจ็ก ซึ่งลักษณะและการกระทำจะถูกห่อหุ้มไว้ในอ๊อปเจ็ก และจะถูกอ้างถึงหรือถูกเรียกให้ทำงานจากภายนอกผ่านทางข่าวสาร(message)

3 class: หมายถึงการรวมกลุ่มของอ๊อปเจ็กทั้งหมดที่มีลักษณะและวิธีการที่เหมือนๆ กัน อ๊อปเจ็กหนึ่งๆ จะถูกสร้างขึ้นจากคลาสเพียงหนึ่งคลาส ซึ่งคลาสมีลักษณะเหมือนกับ abstract data type คลาสอาจจะเป็นข้อมูลพื้นฐานก็ได้เช่น integer string หรือ boolean เป็นต้น

4 Class hierarchy and inheritance: เป็นการสร้างคลาสใหม่(subclass)จากคลาสเดิม(superclass)ที่มีอยู่ก่อน คลาสที่สร้างขึ้นใหม่จะสืบทอดคุณลักษณะและวิธีการทั้งหมดจากคลาสเดิม โดยคลาสใหม่สามารถที่จะเพิ่มคุณลักษณะ และวิธีการใหม่ เพิ่มเข้าไปได้ จะแบ่งเป็นสองประเภทคือ single inheritance และ multiple inheritance

1. Object Structure

แนวทางเชิงวัตถุมีพื้นฐานมาจากการห้อหุ้มข้อมูล และโปรแกรม(encapsulating) ดังนั้นการติดต่อสื่อสารกันระหว่างอ๊อปเจ็กจะทำผ่านทางข่าวสาร(messages)เสมอ

โดยทั่วไปแล้ว อ๊อปเจ็กจะเกี่ยวข้องกับ

เซตของตัวแปรที่บรรจุข้อมูลของอ๊อปเจ็ก
เซตของข่าวสารซึ่งอ๊อปเจ็กจะตอบสนองต่อข่าวสารที่ได้รับจากภายนอก
เซตของวิธีการ ซึ่งแต่ละวิธีการจะเป็นโปรแกรมที่ดำเนินการกับข่าวสาร และส่งข้อมูลกลับ
สิ่งที่ทำให้ต้องมีการใช้ข่าวสารและวิธีการระหว่างอ๊อปเจ็ก พิจารณาเอนติ้นี้ employee ในระบบธนาคาร ทุก ๆ คนจะมีการคำนวณเงินรายได้ทั้งปี แต่มีความแตกต่างกันในเรื่องของการคำนวณ ตามประเภทของพนักงาน เช่น ผู้จัดการ ได้รับโบนัสตามกำไรที่ธนาคารทำได้ หรือพนักงานเคาเตอร์ จะได้รับโบนัสตามชั่วโมงการทำงาน เป็นต้น เราสามารถซ่อนโปรแกรมในส่วนของการคำนวณเงินเดือน โดยอ๊อปเจ็กจะรู้ว่าต้องคำนวณเงินรายได้ทั้งปีอย่างไร โดยที่อ๊อปเจ็กของพนักงานทุกคนมี interface เดียวกัน ในระบบเชิงวัตถุ เราสามารถที่จะแก้ไขนิยามของวิธีการ ตัวแปร โดยไม่มีผลกระทบต่อระบบ ซึ่งคุณสมบัตินี้เป็นข้อดีของการเขียนโปรแกรมเชิงวัตถุ

วิธีการของอ๊อปเจ็กสามารถจำแนกได้ 2 กลุ่มคือ ประเภทอ่านอย่างเดียว หรือ ประเภท Update ถ้าเป็นวิธีการแบบอ่านอย่างเดียวจะไม่มีผลกระทบต่อข้อมูลในอ๊อปเจ็ก ในขณะที่ Update จะทำการเปลี่ยนแปลงข้อมูลภายในอ๊อปเจ็กได้ ซึ่งข่าวสารที่อ๊อปเจ็กจะตอบสนองว่าเป็นแบบอ่านอย่างเดียวหรือแบบ update นั้นขึ้นอยู่กับวิธีการ implement ข่าวสาร

สำหรับแอตทริบิวต์ที่ได้จากการคำนวณนั้น ในระบบเชิงวัตถุจะใช้ข่าวสารแบบอ่านอย่างเดียว กล่าวคือทุก ๆ แอตทริบิวต์ของเอนติตี้ต้องประกอบไปด้วยข่าวสาร 2 อันคือ ข่าวสารสำหรับการอ่านข้อมูลของแอตทริบิวต์ และอีกข่าวสารสำหรับการปรับปรุงข้อมูล

2. Object Classes

โดยปกติ ในฐานข้อมูลจะมีอ๊อปเจ็กที่คล้าย ๆ กัน หมายความว่าอ๊อปเจ็กมีการตอบสนองต่อข่าวสารเดียวกัน ใช้วิธีการเดียวกัน และมีตัวแปรชื่อเดียวกัน ชนิดข้อมูลเดียวกัน ดังนั้นเราจึงจัดกลุ่มของอ๊อปเจ็กที่ลักษณะคล้าย ๆ กัน แล้วกำหนดเป็นคลาส ๆ หนึ่ง และแต่ละอ๊อปเจ็กที่สร้างมาจากคลาสก็เรียกว่า เป็น instance ของคลาส อ๊อปเจ็กทุกตัวจะมีลักษณะเหมือนกัน จะแตกต่างกันตรงข้อมูลที่ถูกกำหนดให้กับตัวแปรในอ๊อปเจ็ก

เราจะนิยามคลาส employee ในรูปของคำสั่งลำลอง ซึ่งแสดงตัวแปรและข่าวสารซึ่งอ๊อปเจ็กจะตอบสนอง ในตัวอย่างนี้ยังไม่มีวิธีการที่จะมาจักการกับข่าวสาร

```
class employee {

/* Variables */

string name;

string address;

date start-date;

int salary;

/* Messages */

int annual-salary();

string get-name();

string get-address();

int set-address(string new-address);

int employment-length();

};
```

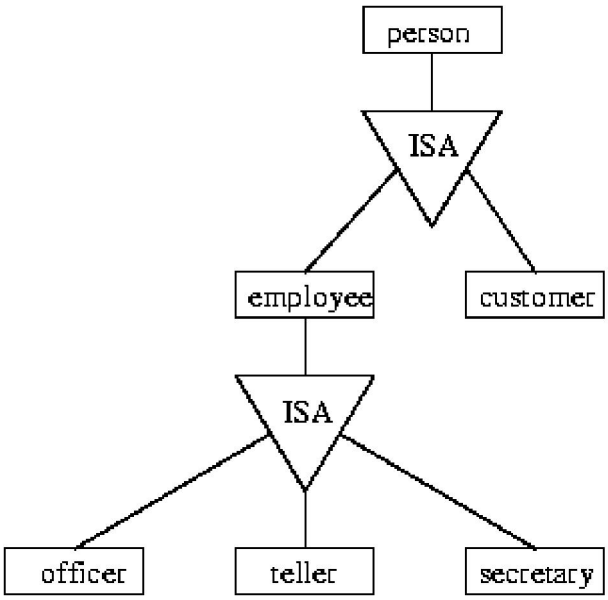
จากนิยามของคลาส employee จะประกอบไปด้วยตัวแปร name และ address ซึ่งมีชนิดข้อมูลเป็น string start-date มีชนิดข้อมูลเป็น date และ salary มีชนิดข้อมูลเป็น integer แต่ละอ๊อปเจ็กจะตอบสนองข่าวสาร 5 ข่าวสาร คือ

annual-salary get-name get-address set-address และ employment-length ข่าวสารแต่ละอันมีการกำหนดชนิดข้อมูลไว้ด้านหน้า หมายถึงชนิดของการตอบสนองข้อมูลกลับไปยังข่าวสารนั้น

3. Inheritance

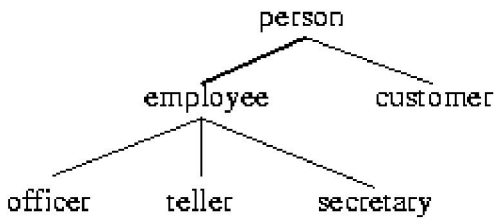
ในระบบฐานข้อมูลเชิงวัตถุจะประกอบไปด้วยคลาสจำนวนมาก อย่างไรก็ตามคลาสหลาย ๆ คลาสจะมีลักษณะคล้าย ๆ กัน ตัวอย่างเช่น พนักงานของธนาคาร จะมีลักษณะที่คล้ายกับลูกค้า เช่น ชื่อ ที่อยู่ โทรศัพท์ เป็นต้น

เพื่อแสดงให้เห็นถึงลักษณะที่คล้ายกันของคลาส เราจะแสดง E-R ไดอะแกรมในลักษณะของลำดับชั้นของ specialization (ความสัมพันธ์แบบ ISA) ดังรูปที่ 1



รูปที่ 1 แสดงลำดับชั้นของคลาสแบบ Specialization

หลักการของลำดับชั้นของคลาสจะเหมือนกับ specialization ใน E-R โมเดล รูปที่ 2 แสดงลำดับชั้นของคลาสที่เหมือน E-R ไดอะแกรมในรูปที่ 1



รูปที่ 2 แสดงลำดับชั้นของคลาสที่เหมือนกับ E-R ในรูปที่ 1

ลำดับชั้นของคลาสสามารถกำหนดเป็นคำสั่งลำลองได้ดังรูปที่ 3 ในที่นี่ยังไม่ขอกล่าวถึงวิธีการของแต่ละคลาส

```
class person {

string name;
```

```
string address;
};
class customer isa person {
int credit-rating;
};
class employee isa person {
date start-date;
int salary;
};
class officer isa employee {
int office-number;
int expense-account-number;
};
class teller isa employee {
int hours-per-week;
int station-number;
};
class secretary isa employee {
int hours-per-week;
string manager;
};
```

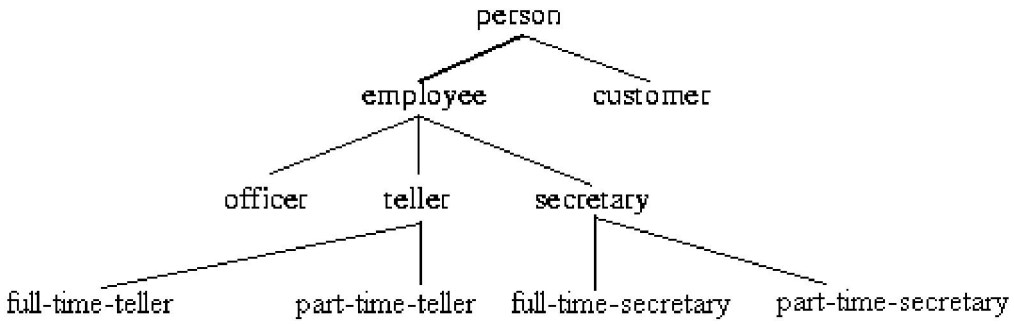
รูปที่ 3 แสดงคำสั่งลำลองในการนิยามลำดับชั้นของคลาส

คำว่า isa ใช้สำหรับบอกว่าคลาสนั้นมีลักษณะเช่นเดียวกันกับอีกคลาสหนึ่ง โดยเราเรียกคลาสที่ specialization ว่า subclasses ตัวอย่างเช่น employee เป็น subclass ของ person และ teller เป็น subclass ของemployee ในทางกลับกัน employee ก็เป็น superclass ของ teller

ข้อดีของการสืบทอดคุณสมบัติในระบบเชิงวัตถุคือ มีคุณลักษณะของ code-reuse นั่นคือวิธีการใด ๆ ของคลาส A สามารถที่จะถูกเรียกใช้โดยคลาส B ซึ่งเป็น subclass A ได้ทันที ทำให้ไม่ต้องเขียนคำสั่งในคลาส B อีก

4. Multiple Inheritance

โดยทั่วไปแล้วโครงสร้างของคลาสแบบลำดับชั้นก็เพียงพอต่อการแสดงโมเดลของข้อมูลแล้ว แต่ในบางกรณี เช่นเราต้องการจำแนกระหว่าง teller และ secretaries แบบ full-time และ part-time ในรูปที่ 2 ซึ่งเราก็สามารถสร้าง subclass ของ part-time-teller full-time-teller part-time-secretary และ full-time-secretary ดังรูปที่ 4

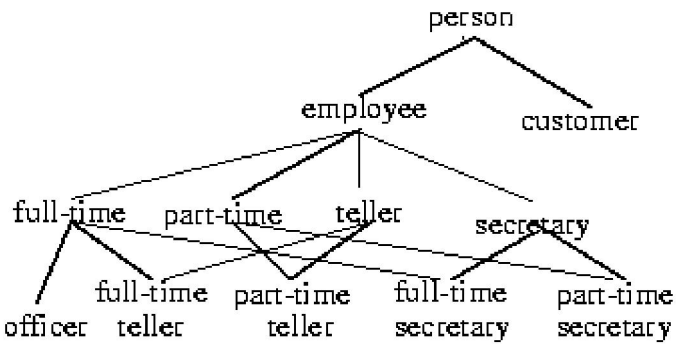


รูปที่ 4 แสดงลำดับชั้นของคลาส ของพนักงาน full-time และ part-time

แต่จะเกิดปัญหาสองอย่างคือ (1) ปัญหาความไม่สอดคล้องในการปรับปรุงข้อมูล เนื่องมาจาก ตัวแปรและวิธีการของพนักงาน full-time ต้องถูกกำหนดสองครั้ง คือ full-time-teller และ full-time-secretary ในทำนองเดียวกันกับพนักงาน part-time เมื่อไรก็ตามที่ต้องมีการเปลี่ยนแปลงข้อมูลของพนักงาน ก็ต้องมีการเปลี่ยนแปลงสองที่ (2) ลำดับชั้นของคลาสไม่สามารถแสดงพนังงานกลุ่มอื่นที่ไม่ใช่ full-time หรือ part-time ได้

วิธีการ Multiple inheritance เป็นวิธีการที่ยอมให้คลาสหนึ่งสืบทอดคุณสมบัติและวิธีการจากหลาย ๆ คลาสได้

ความสัมพันธ์ระหว่าง class และ subclass แสดงได้โดย directed acyclic graph (DAG) ซึ่งใช้ในกรณีที่คลาส มี superclass มากกว่าหนึ่งคลาส



รูปที่ 5 แสดงคลาส DAG

จากรูปที่ 5 เรากำหนดคลาส part-time และคลาส full-time ขึ้นมาเพื่อเก็บข้อมูลและวิธีการของพนักงาน part-time และ full-time จากนั้น สร้างคลาส part-time-teller ซึ่งเป็น subclass ของ teller และ part-time คลาส part-time-teller จะสืบทอดคุณสมบัติ และวิธีการของทั้ง teller และ part-time มา ซึ่งจะเห็นว่าไม่เกิดความซ้ำซ้อนอย่างที่เกิดขึ้นในรูปที่4 แล้ว

เมื่อมีการสืบทอดแบบหลายคลาส อาจจะมีความคลุมเคลือเกิดขึ้นในกรณีที่ตัวแปรหรือวิธีการที่สืบทอดมาจากหลายคลาสมีชื่อเดียวกัน

ตัวอย่างเช่น เรากำหนดการจ่ายเงิน pay ของพนักงานแต่ละประเภทดังนี้

full-time: จ่ายเงินอยู่ระหว่าง 0 ถึง 100,000 ของเงินรายได้ทั้งปี
part-time: จ่ายเงินอยู่ระหว่าง 0 to 20 เป็นค่าแรงต่อชั่วโมง.
teller: จ่ายเงินอยู่ระหว่าง 0 to 20,000 ของเงินรายได้ทั้งปี
secretary: จ่ายเงินอยู่ระหว่าง 0 to 25,000 ของเงินรายได้ทั้งปี
พิจารณาคลาส part-time-secretary ซึ่งสืบทอดตัวแปร pay มาจากคลาส part-time หรือ secretary ผลลัพธ์ที่ได้มีหลายกรณี ขึ้นอยู่กับวิธีการว่าจะจัดการอย่างไร

สืบทอดมาทั้งสองคลาส แล้วเปลี่ยนชื่อของ เป็น part-time-pay และ secretary-pay
เลือกตัวใดตัวหนึ่ง โดยดูจากลำดับของการสร้างคลาส
บังคับให้เลือกเลยว่าจะใช้ pay ตัวไหน
แสดงผลว่าเกิดความผิดพลาด
5.Object Identity

Object identity: อ๊อปเจ็กจะยังคงรักษาความเป็น identity แม้ว่าข้อมูลของตัวแปรหรือวิธีการของ อ๊อปเจ็กมีการเปลี่ยนแปลง แนวคิดของ identity นี้ไม่ได้นำไปใช้กับทูเปิลของฐานข้อมูลเชิงสัมพันธ์ ซึ่งในระบบเชิงสัมพันธ์นั้น ทูเปิลของรีเลชันจะแตกต่างกันด้วยข้อมูลที่ถูกเก็บอยู่

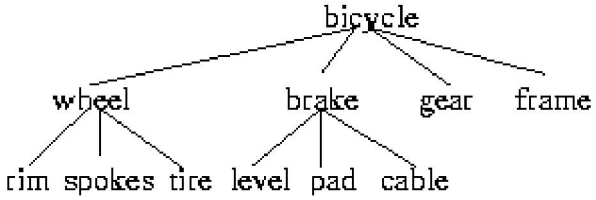
รูปแบบต่าง ๆ ของ identity

Value: หมายถึงค่าของข้อมูลที่ใช้ในการ identity เช่น primary key
Name: เป็นชื่อที่ผู้ใช้กำหนดเพื่อใช้ในการ identity เช่น ชื่อแฟ้มในระบบแฟ้มข้อมูล
Built-in: เป็นการกำหนด identity โดยอัตโนมัติให้กับข้อมูล หรือภาษาโปรแกรม โดยผู้ใช้ไม่จำเป็นต้องกำหนด identifier รูปแบบนี้ถูกใช้ในระบบเชิงวัตถุ แต่ละอ๊อปเจ็กจะถูกกำหนด identifier โดยอัตโนมัติ เมื่ออ๊อปเจ็กถูกสร้างขึ้น ในทางปฏิบัติ Object identity จะถูกกำหนดให้มีค่าไม่ซ้ำกัน เรียกว่า OID ซึ่งค่าของ OID นี้จะไม่สามารถมองเห็นได้โดยผู้ใช้ทั่วไป แต่จะถูกใช้โดยระบบเพื่อระบุอ๊อปเจ็กแต่ละอ๊อปเจ็ก

6. Object Containment

อ๊อปเจ็กที่สามารถบรรจุอ๊อปเจ็กอื่น ๆ ได้จะถูกเรียกว่า อ๊อปเจ็กเชิงซ้อน(complex object)หรืออ๊อปเจ็กผสม (composite object) ซึ่งสามารถบรรจุซ้อนกันได้หลายชั้น นั่นคือเป็นลักษณะความสัมพันธ์ที่อ๊อปเจ็กหนึ่งจะประกอบด้วยอ๊อปเจ็กหลาย ๆ อ๊อปเจ็กประกอบเข้าด้วยกัน

ตัวอย่าง A bicycle design database



รูปที่ 6 แสดงลำดับชั้นความสัมพันธ์ระหว่างอ๊อปเจ็ก

รูปที่ 6 แสดงลำดับชั้นความสัมพันธ์ระหว่างอ๊อปเจ็ก ซึ่งความสัมพันธ์ในลักษณะนี้ เป็นลักษณะของ is-part-of ซึ่งเป็นคนละแบบกับ is-a ที่เกิดจากการสืบทอดคลาส

ลักษณะการประกอบกันของอ๊อปเจ็กนี้เป็นหลักการที่สำคัญอย่างหนึ่งในระบบเชิงวัตถุ เนื่องจากจะทำให้การมองข้อมูลเริ่มมองจากจุดเล็ก ๆ ก่อน เช่น คนออกแบบล้อ ก็จะมุ่งไปกับการออกแบบล้อ โดยไม่สนใจกับส่วนอื่นเช่น เกียร์หรือ เบรก จากนั้นพนักงานฝ่ายการตลาดก็จะทำการกำหนดราคาจักรยาน โดยนำเอาข้อมูลส่วนต่าง ๆ ที่เป็นส่วนประกอบของจักรยาน มาคำนวณราคาต่อไป

ที่มา http://www.srisangworn.go.th/home/databaselearnx/ms1t1-13.htm