



RDF Model

▫ RDF stands for “Resource Description Framework”

- **Resource**
 - สามารถเป็นทุกสิ่งที่เราสนใจหรือไม่ใช่พอดี
 - จะต้องระบุหรืออ้างอิงได้โดยไม่ซ้ำกันโดยผ่านสิ่งที่เรียกว่า URI
- **Description**
 - คือการอธิบาย Resource ผ่านทาง Properties และ relationships ระหว่าง Resources
- **Framework**
 - = เป็นการใช้งานผสมผสานกันของ Web-based Protocols ต่างๆ (URI, HTTP, XML, Turtle, JSON,...)
 - ถูกออกแบบอยู่บนพื้นฐานของโมเดลมาตรฐาน (formal model) ที่สื่อความหมาย (semantics) ได้แก่โครงสร้างที่เป็น triple

9



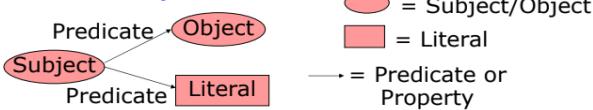
Subject or Resource

- Subject หรือ Resource ก็คือ “thing” หรือสิ่งที่เราสนใจที่กำลังพูดถึง เช่น
 - E.g. authors, books, publishers, places, people, hotels
- Resources ทุกด้านจะต้องอยู่ในรูปแบบของ URI ซึ่งเป็น Universal Resource Identifier
- A URI can be
 - a URL (Web address) or
 - some other kinds of unique identifier



RDF as a piece of Graph

▫ The RDF Graph



- **Subject** is the resource being described by predicate and object
- **Predicate** is a relation between the subject and **object** or **literal** (values of predicate)
- **Object** is a **resource**, **Literal** is a **string value**



What is URI/IRI?

▪ Where does URI come from?

- URI → <https://thewaltdisneycompany.com/Pluto>
- ดูเรียกว่า **Prefix** หรือ **Namespace** คือ **Resource** ซึ่งจะมีรูปแบบคล้าย URL แต่จะมีอยู่จริง ที่เรากำลังพูดถึง หรือไม่อยู่จริงก็ได้



What is URI/IRI?

▪ Where does IRI come from?

- IRI (International Resource Identifier) is an internet protocol standard จะถูกสร้างอยู่บน Uniform Resource Identifier (URI) protocol อีกที
- IRIs สามารถเพิ่มจำนวนตัวอักษรที่มากกว่า URIs โดยจะรองรับ Universal Character Set (Unicode/ISO10646), ซึ่งจะรวมภาษาได้ๆ ในโลกนี้ เช่น Chinese, Japanese, Korean, and Cyrillic characters.



RDF Statements

▫ โครงสร้างพื้นฐานของ RDF จะอยู่ในรูปแบบของ triple :

(subject, predicate, value) } statement
or **(resource, property, value)** }

- A **value** of predicate/property can be either a **resource** (object) or **literal**
 - **Literal** is an **atomic value**, such as **string**, **integer**, **date**, etc.



Properties

▫ Properties จะถูกใช้ในการอธิบายความสัมพันธ์ระหว่าง Resources

- E.g. “written by”, “age”, “title”, etc.

▫ Properties ก็จะต้องถูกเขียนให้อยู่ในรูปแบบของ URLs ด้วยเช่นเดียวกัน

▫ Advantages of using URIs:

- ทำให้ Resource นั้นมีความเป็นหนึ่งเดียว ไม่ซ้ำกัน (Unique)
- แก้ไขปัญหา Homonym words ที่จะเกิดขึ้น



What is URI/IRI?

▪ Where does URI come from?

- URI (Uniform Resource Identifier) จะมีความคล้ายกับ URL
- URI จะเป็น logical resources คือ ไม่มีอยู่จริง หรือ physical resource ที่มีอยู่จริงเหมือนกับ URL ก็ได้
- วัตถุประสงค์ของ URI คือการแก้ไขปัญหา homonym conflict โดยการทำให้การอ้างถึง resource แต่ละตัวต้องไม่ซ้ำกัน
- Hence, <https://dbpedia.org/resource/Pluto> is different from <https://thewaltdisneycompany.com/Pluto>



What is URI/IRI?

▪ Where does URI come from? (continue..)

- URIs จะจัดอยู่ที่ัวอักขระภาษาอังกฤษที่เป็น US-ASCII character set (ไม่รองรับ Unicode หรือภาษาที่หลากหลาย)

- กรณีใช้ URI กับ Web of data จะทำให้เราสามารถ click ที่ resource นั้นเพื่อจะเชื่อมโยงไปยังข้อมูลรายละเอียดของ resource นั้นได้

- URI จึงช่วยให้เกิดการเชื่อมโยงันระหว่าง Resources.



23



What is Linked Open Data?

- Linked Data คือการเชื่อมโยงข้อมูลต่างๆ ที่มีโครงสร้างเข้าด้วยกัน ซึ่งไม่เพียงแต่มุ่งยกระดับให้เข้าใจได้ แต่ยังรวมถึง computer ด้วย
- ถ้าข้อมูลที่มีการเชื่อมโยงนั้นเป็นที่บิดเบิกกับสาธารณะ เราจะเรียกว่า **Linked Open Data** หรือ **LOD**
- LOD จะเป็นพื้นฐานของ Semantic Web



This Example represents “Pluto” by using DBpedia Vocabulary in Graph Format



55



RDF as the Serialization Formats

- RDF** ไม่ใช้แค่เป็นการจัดรูปแบบของข้อมูล (data format) แต่จะเป็นการสร้างโมเดลข้อมูล (data model) เพื่อให้สามารถอธิบาย resources ต่างๆ ให้อยู่ในรูปแบบของ Triple ซึ่งได้แก่ **subject, predicate, object**
- จะมีรูปแบบของภาษาที่ถูกเรียกว่า **RDF serialization languages** ที่ถูก拿来เสนอขึ้นมาเพื่อให้สามารถนำไปแปลงเป็น **RDF model** คืออยู่ในรูปแบบที่เป็น triple ได้ ได้แก่ ภาษาต่อไปนี้
 - RDF/XML**
 - Turtle** (Terse RDF Triple Language)
 - N-Triples**
 - N-Quads**
 - RDFa** (Resource Description Framework in Attributes)
 - RDF/JSON** or **JSON-LD** (JSON-Linked Data)

6



DB → XML



9



RDF as the Serialization Formats

- Graph** จะเป็นรูปแบบการแทนโครงสร้างข้อมูลที่มนุษย์เข้าใจได้ง่าย
- แต่จุดประสังค์ของ Semantic Web คือต้องการที่จะทำให้ computers สามารถเข้าใจความหมายของข้อมูล นำข้อมูลไปประมวลผลได้อย่างถูกต้อง และสืบค้นข้อมูลได้ตรงกับความต้องการของมนุษย์ให้มากที่สุด
- ดังนั้น จึงต้องมีการแทนโครงสร้าง RDF ให้อยู่ในรูปแบบที่ computers สามารถเข้าใจได้ดีอยู่ในรูปแบบของ **RDF Serialization Languages** ได้แก่ภาษา **XML**, **N3**, **N-Triples**, **Turtle**, **RDFa** และ **JSON-LD** เป็นต้น

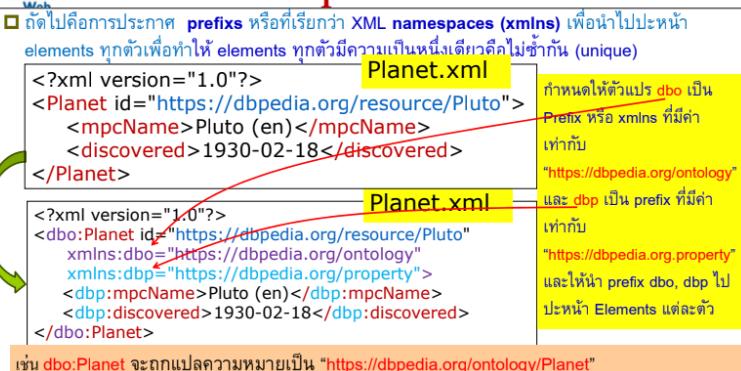


RDF/XML

- RDF/XML** เป็นภาษาที่เน้นโครงสร้างไวยากรณ์ (syntax) ซึ่งมีองค์กร **W3.org** เป็นผู้กำหนดรูปแบบโครงสร้าง โดยเป็นภาษาที่ถูกใช้เป็นภาษาแรกสุดในกลุ่มของ **Serialization Languages** ที่ถูกใช้ในการแทน **RDF Model**



Insert Namespace to XML Element



10



XML → RDF

<https://www.w3.org/RDF/Validator/>


11



Turtle- Terse RDF Triple Language

- Turtle** จะเป็นภาษาอีกภาษาหนึ่งที่ทำให้เราสามารถแทน **RDF graphs** ให้อยู่ในรูปแบบของภาษาที่ใกล้เคียงกับภาษาธรรมชาติของมนุษย์ คือประโยชน์ภาษาอังกฤษ
- Turtle** จะต้องมีการประกาศตัวแปรในรูปแบบของ **Prefixes** และใช้ตัวแปรที่เป็น **Prefixes** นั้นปะหน้า **Resources** เพื่อทำให้อยู่ในรูปแบบของ **URLs** ทำให้ไม่ต้องเขียน URIs ในรูปแบบที่ยาวเกินไป



Turtle- Terse RDF Triple Language



12



Turtle- Terse RDF Triple Language

- Example1 of Turtle:**
- ```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix dbp: <https://dbpedia.org/property/>.
@prefix dbo: <https://dbpedia.org/ontology/>.
@prefix dbr: <https://dbpedia.org/resource/>.

dbr:Pluto
 a dbo:Planet;
 dbp:mpcName "Pluto (en)" ;
 dbp:discovered "1930-02-18" .
```
- Annotations:**
- Resources:** `a` คือ keyword ที่มีความหมายแบบเดียวกับการเขียนแบบ triples ดังนี้: `(dbr:Pluto, rdf:type, dbo:Planet)`, `(dbr:Pluto, dbp:mpcName, "Pluto (en)")`, `(dbr:Pluto, dbp:discovered, "1930-02-18")`.

13



## Turtle- Terse RDF Triple Language

### Example2 of Turtle

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix dbp: <https://dbpedia.org/property/> .
@prefix dbo: <https://dbpedia.org/ontology/> .
@prefix dbr: <https://dbpedia.org/resource/> .
```

```
dbr:Pluto
 a dbo:Planet;
 dbp:mpcName "Pluto (en)";
 dbp:discovered "1930-02-18";
 dbp:discoverer dbr:Clyde_Tombaugh.

dbr:Clyde_Tombaugh
 a dbo:Person;
 dbo:deathDate "1997-01-17".
```

ตัวชี้ไฟล์ชื่อ  
Planet2.ttl

20

## Turtle- Terse RDF Triple Language

### Example3 of Turtle

```
@base <http://example.org/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix rel: <http://www.perceive.net/schemas/relationship/> .
```

```
<green-goblin>
 rel:enemyOf <spiderman>;
 a foaf:Person ; # in the context of the Marvel universe
 foaf:name "Green Goblin".
```

```
<spiderman>
 rel:enemyOf <green-goblin>;
 a foaf:Person ;
 foaf:name "Spiderman" "Uomo Ragno"@it .
```

เราต้องมารู้ว่าภาษาที่เราใช้ @base เพื่อกำหนด namespace ที่จะใช้ในภาษาที่เราเขียน resources หากต้องการต่อตัวในไฟล์เพื่อจะมีประสิทธิภาพมากขึ้น

ถ้าใช้เครื่องหมาย , หมายความว่าแต่ละประโยค จะใช้ปริมาณ <Spiderman> และ Property foaf:name ร่วมกัน (เหมือนกัน) และเมื่อมีคำอธูม ก็จะ Property แยกต่างกัน

20



## Turtle- Terse RDF Triple Language

### Example3 of Turtle

```
@prefix : <http://example.org/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix rel: <http://www.perceive.net/schemas/relationship/> .
```

:green-goblin
 rel:enemyOf :spiderman ;
 a foaf:Person ; # in the context of the Marvel universe
 foaf:name "Green Goblin".

:spiderman
 rel:enemyOf :green-goblin ;
 a foaf:Person ;
 foaf:name "Spiderman", "Uomo Ragno"@it .

เรียกสามารถใช้ @prefix : เพื่อแทน @base ได้
 อีกด้วย และจะใช้ : ปะอยู่หน้า resource ที่ต้องการได้เลย

26



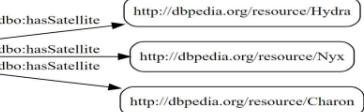
## Turtle- Terse RDF Triple Language

### Example4 of Turtle

```
@prefix dbo: <http://dbpedia.org/ontology/> .
@base <http://dbpedia.org/resource/> .
```

```
<Pluto> dbo:hasSatellite <Hydra>
 <Nyx>,
 <Charon> .
```

ถ้าใช้เครื่องหมาย , หมายความว่าแต่ละประโยค จะใช้ปริมาณ <Pluto> และ Property dbo:hasSatellite ร่วมกัน (เหมือนกัน) และจะมีคำอธูมของ Property แตกต่างกัน



27



## N-Triples

- N-Triples** จะเป็น **subset** ของ **Turtle** โดยจะตัดเรื่องการใช้ **namespace prefixes** ออกไป
- ในแต่ละ **triple** จะต้องมีการระบุ **URIs** ทุกด้านอย่างเต็มรูปแบบโดยไม่มีการใช้ตัว **prefix** ที่ทำให้การเขียนสั้นลง
- ลักษณะไฟล์ **N-Triples** อาจมีรูปแบบเดียวกับ **Turtle** และแม้แต่ **RDF/XML**
- อย่างไรก็ตาม **N-Triples** ถูกออกแบบให้มีรูปแบบที่ง่ายกว่า **Turtle** และง่ายต่อ **software** ที่จะทำการอ่านและแปลงข้อมูลให้อยู่ในรูปแบบของ **triple**
- N-Triples** จึงเหมาะสมที่จะใช้ในการแลกเปลี่ยนข้อมูลที่มีขนาดใหญ่ เพื่อใช้ประมวลผล **RDF graph** ที่มีขนาดใหญ่ด้วย **tool** หรือ **software** ที่มีลักษณะการประมวลผลความที่ต้องอ่านเป็นบรรทัดๆ

31



## N-Triples

### Example of N-Triples

```

<https://dbpedia.org/resource/Pluto> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <https://dbpedia.org/ontology/Planet> .
<https://dbpedia.org/resource/Pluto> <https://dbpedia.org/property/discovered> "1930-02-18" .
<https://dbpedia.org/resource/Pluto> <https://dbpedia.org/property/discoverer> <https://dbpedia.org/resource/Clyde_Tombaugh> .
<https://dbpedia.org/resource/Clyde_Tombaugh> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <https://dbpedia.org/ontology/Person> .
<https://dbpedia.org/resource/Clyde_Tombaugh> <https://dbpedia.org/ontology/deathDate> "1997-01-17" .

```



## RDFa –RDF in HTML attributes

- RDFa** เป็นภาษาที่อยู่ในกลุ่ม **Serialization languages** ภาษาหนึ่งที่ทำให้เราสามารถแทรกส่วนที่เป็น **RDF model** เข้าไปในเอกสาร **(X)HTML** ได้โดยการยังคงมีการใช้งาน **(X)HTML attributes** เดิมอยู่ แต่จะมีการแทรก **new attributes** เข้าไปได้ด้วย
- RDFa 1.0** ถูกออกแบบมาเพื่อรักษาของเอกสาร **XHTML** (W3C Recommendation 2008)
- RDFa 1.1** ถูกปรับปรุงจาก 1.0 และถูกออกแบบมาเพื่อรักษาของ **HTML5** (W3C Recommendation June 2012) ประกอบด้วย
  - RDFa Lite1.1** (จะแนะนำให้วิจัยตัวนี้)
  - RDFa 1.1**

4



## RDFa –RDF in HTML attributes

- RDFa** จะยังคงมีการใช้งาน **(X)HTML attributes** เดิมอยู่ เช่น **href**, **src**, **span** เป็นต้น และจะมีการเพิ่มเติม **HTML attributes** ใหม่ดังต่อไปนี้เข้าไปใน **(X)HTML** ด้วย
  - vocab**,
  - typeof**,
  - property**,
  - resource**,
  - prefix**



## RDFa Lite 1.1

You can visit this Web Site:  
<https://vocab.org/open/>

- ❑ ถ้าคำศัพท์ใน FOAF ไม่เพียงพอที่จะใช้อธิบาย Properties ทั้งหมดได้ เราอย่างสามารถ ก้าวหนทางกลุ่มคำศัพท์เพิ่มเติมได้โดยใช้ Prefix

```
<p vocab="http://xmlns.com/foaf/0.1/"
prefix="pa: http://open.vocab.org/terms/"
resource="Robert" typeof="Person">
My name is
Robert Lee
and you can give me a phone via
287399388.

cat and dog
</p>
```

50



## Example of using RDFa in HTML

Example 1 ⓘ  
No Markup Microdata RDFa JSON-LD Structure  
Example encoded as RDFa embedded in HTML.

```
div vocab="https://schema.org/" typeof="Person"
Jane Doe

Product Manager
<div property="address" typeof="PostalAddress">
203 Main Whitworth Institute
405 N. Whitworth
Seattle
WA
98052
</div>
(425) 123-4567
jane.doe@xyz.edu
Jane's home page:
janedoe.com
Graduate students:

```



## HTML5 Microdata ( $\mu$ Data)

- ❑ **HTML5 Microdata specification** จะประกอบด้วย
  - Microdata vocabularies** จะเป็นกลุ่มคำศัพท์ที่นำมาจาก <http://schema.org>
  - Microdata Global Attributes** ที่จะต้องแทรกอยู่ในเอกสาร html ได้แก่ attributes ต่อไปนี้
    - itemscope** (ถูกใช้เพื่อบอกจุดเริ่มต้นของ Microdata)
    - itemtype** (เป็นการประกาศประเภท (type) ของสิ่งที่เราจะ อธิบาย โดยนำคำมาจาก schema.org)
    - itemid** (ใช้เพื่อระบุถึงชื่อ resource ที่จะอธิบาย)
    - itemprop** (ใช้เพื่ออธิบายถึง property ของ resource)
    - itemref** (ใช้เพื่อการอ้างอิงไปยัง resource อื่น)



## Example of using Microdata in HTML

ตัวอย่างนี้เป็นการใช้ Microdata ฝังอยู่ในเอกสาร HTML ข้างต้น เพื่ออธิบายสินค้า (Product) ที่เป็น Microwave



## HTML5 Microdata ( $\mu$ Data)

❑ ตัวนี้เป็นตัวอย่างการเขียนเอกสาร HTML ที่มีการฝัง Microdata อยู่ภายใน

```
<p itemscope itemid="http://mydomain.com/Robert" itemtype="http://schema.org/Person">
My name is
Robert Lee
and you can give me a phone via
287399388.

Computer Support
</p>
```

**itemscope, itemid, itemtype, itemprop** เป็นกลุ่มคำของ Microdata แต่ Person, name, telephone, image, และ jobTitle เป็นกลุ่มคำของ Schema.org



## JSON-LD

### Example of JSON document

```
{ "name": "Manu Sporny",
 "homepage": "http://manu.sporny.org/",
 "image": "http://manu.sporny.org/images/manu.png" }
```

Key: value pairs

ตัวอย่างนี้เป็นตัวอย่างเอกสาร JSON ที่เราต้นคีย์นัน ที่ประกอบด้วย key value pair ที่มีโครงสร้างง่ายๆ การเพิ่มโครงสร้าง JSON แบบนี้จะทำให้เราใช้งานเพิ่มความยืดหยุ่น ทั้งคีย์และ machine readable code ที่ไม่มีความซ้ำซ้อน ชื่อคีย์ name จะไม่ได้เป็น name ของค่า ทั้ง homepage และ image ก็เช่นกัน และถ้ามีเอกสาร หลบเล็กหลบเล็กที่มาระบุเช่นที่ซึ่งหัวข้อด้วยเด็กภาษา ก็ได้ จึงมีความก้าวหน้าของชื่อเดิมที่ดังนั้น จึงต้องมีการกำหนด identifier ที่อยู่ในรูปแบบของ namespace มากขึ้นหากค่าที่เป็น key เหล่านี้พื้นที่ต้องให้ unique ไม่มีความก้าวหน้าเกิดขึ้น การ reuse และ sharing เอกสารก็จะทำได้ง่าย



## JSON-LD

### Another Example of JSON-LD document

```
{ "@context": "https://schema.org",
 "name": "Manu Sporny",
 "url": "http://manu.sporny.org/",
 "image": "http://manu.sporny.org/images/manu.png" }
```

- ❑ อย่างไรก็ตาม เราอย่างสามารถเขียนอธิบายแบบนี้เพื่อ มีการประกาศ "@context" เพื่อเก็บ standard vocabulary ที่เราจะใช้ไว้ ซึ่งได้แก่ เว็บของ schema.org หลังจากนั้นก็เรียกใช้ ค่าคีย์ที่อยู่ใน schema.org ได้เลย โดยไม่ต้องปะ prefix ข้างหน้า คือท่าให้อธิบายในรูปแบบของ key: value pairs เมื่อเสร็จ



## JSON-LD

ในตัวนี้จะมีการแทรก @id เพื่อเข้าไปอธิบายลักษณะของค่า context เพื่อประกาศว่า resource ของๆ นั้นเป็นอะไร หรือเป็นรูปแบบของ URI  
และการประกาศ @type เพื่อแสดงถึง type ของ resource ที่อยู่ใน @id นั้นเอง

### Another Example of JSON-LD document

```
{ "@context": "https://schema.org",
 "@id": "https://www.mydomain.com/P0001",
 "@type": "Product",
 "name": "T-Shirt",
 "image": "http://www.mydomain.com/images/P0001.png",
 "manufacturer": {
 "@id": "https://www.product.com/C0001",
 "@type": "https://www.product.com/Company",
 "name": "Xuzang Company"
 } }
```

31



## Example of linking JSON-LD in HTML

Schema.org/Person

Example 1

No Markup Microdata RDFa JSON-LD Structure

Example encoded as JSON-LD in an HTML script tag.

```
<script type="application/ld+json">
{
 "@context": "https://schema.org",
 "@type": "Person",
 "address": {
 "@type": "PostalAddress",
 "addressLocality": "Seattle",
 "addressRegion": "WA",
 "postalCode": "98052",
 "streetAddress": "20341 Whitworth Institute"
 },
 "colleague": [
 "http://www.xyz.edu/students/alicejones.htm",
 "http://www.xyz.edu/students/bobsmith.html"
],
 "email": "mailto:jane-doe@xyz.edu",
 "image": "janedoe.jpg",
 "jobTitle": "Professor",
 "name": "Jane Doe",
 "telephone": "(425) 123-4567",
 "url": "http://www.janedoe.com"
}</script>
```

ด้วยนี่ถ้าเราไม่มีการกำหนด @id หลักให้เอกสาร แต่ไประบุ @type เป็น "Person" เลย การนี้ให้คอมพิวเตอร์จะทำการสร้าง Blank node ขึ้นมาให้ท่านที่เป็น resource หลัก (Subject) ที่จะถูกเก็บอยู่ใน @id และกำหนดให้ Blank node นั้นมี type เป็น "Person" อีกที

ด้วยนี่ถ้าเรียกัน ไม่มีการกำหนด @id ให้เป็น value ของ property "address" แต่มีการระบุ @type อย่างเดียว คอมพิวเตอร์จะทำการสร้าง Blank node ขึ้นมาให้ท่านที่เป็น resource ที่เก็บอยู่ใน @id และกำหนดให้ Blank node นั้นมี type เป็น "PostalAddress"

เอกสาร JSON-LD นี้จะสามารถนำไปแทรกอยู่ในเอกสาร HTML ได้โดยตรง เพื่อช่วยให้ google สามารถสืบค้นข้อมูลในเอกสาร HTML ได้ง่ายและเร็วขึ้น



## What is Ontology?

- **Ontology** เป็นองค์ประกอบสำคัญของ Semantic Web
- คำว่า "ontology" มีจุดกำเนิดมาจากวิชาด้านปรัชญา (philosophy)
  - ซึ่งเป็นการศึกษาถึงธรรมชาติของสิ่งต่าง ๆ ที่เกิดขึ้น เพื่ออธิบาย ประเภทหรือชนิดของสิ่งต่าง ๆ ที่เราสนใจ (kinds of things) และ อธิบายถึงคุณลักษณะของสิ่งต่าง ๆ เหล่านั้น
- **Ontology** ได้ถูกหยิบมามุ่งใช้ในสาขา Computer Science โดยเฉพาะทางด้าน AI เพื่อช่วยในการ Knowledge sharing and reuse



## Ontologies in Computer Science

- ความหมายของ Ontology ที่ถูกใช้ในสาขา computer science จะถูกนิยามโดย T.R. Gruber จะให้ความหมายว่า
 

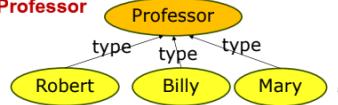
**"An ontology is an explicit and formal specification of a conceptualization"**

→ "ontology" เป็นการอธิบายอย่างชัดแจ้ง (explicit) ถึงสิ่งต่าง ๆ ที่ถูกเรียกว่าเป็น concept (conceptualization) โดยคำอธิบายนั้นจะเป็นคำอธิบายที่มีรูปแบบที่เป็นมาตรฐาน (formal specification)"
- ลักษณะที่โดดเด่นของ ontology คือ การมุ่งเน้นที่การ share สิ่งที่เรียกว่า concept ที่อยู่ใน domain ใด domain หนึ่งที่เราสนใจ เพื่อให้ค้นและ application systems สามารถเข้าใจความหมายนั้นได้ร่วมกัน



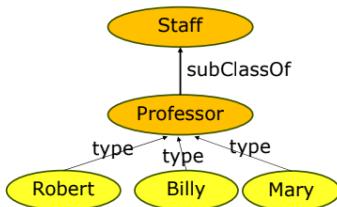
## Typical Components of Ontologies

- **Class** จะเป็นที่รวมของสิ่งที่เรียกว่า **Instances** ที่มีคุณสมบัติ (Properties) ที่เหลือเชื่อม
  - ด้วยของ classes ต่าง ๆ เช่น Professors, Staff, Students, Courses, Departments เป็นต้น
  - **Class Professors** ก็จะเป็นที่รวมของคนทุกคนที่เป็น ศาสตราจารย์ คนแต่ละคนนี้จะเรียกว่า Instance ที่อยู่ภายใต้ Class Professors เช่น ถ้า Robert, Billy, และ Mary เป็นศาสตราจารย์ แต่ละคนก็ถือว่าเป็น Instance ที่อยู่ใน class Professor



## Typical Components of Ontologies

- ➡ ถ้า Robert, Billy, และ Mary เป็น instances ที่อยู่ใน class Professor ห้องสมุดนี้ก็จะเป็น instances ของ class Staff ด้วย กล่าวคือทุกคนจะเป็นทั้ง Professor และเป็น Staff ด้วย



10



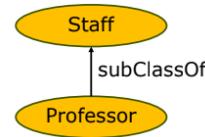
## Typical Components of Ontologies

- **Ontology** จะประกอบด้วยสิ่งต่อไปนี้
  - Concepts หรือ Classes (จะเลือกใช้คำว่า **Classes**)
  - ข้อมูลที่เรียกว่า Objects หรือ Instances ที่อยู่ภายใต้ Concept (จะเลือกใช้คำว่า **instances**)
  - ความสัมพันธ์ (relationships) ระหว่าง concepts เหล่านั้น
  - คุณสมบัติ (Properties) ของแต่ละ concept
  - การกำหนดเงื่อนไข (Restriction)



## Typical Components of Ontologies

- **Relationships** ระหว่าง classes สามารถอยู่ในรูปแบบของลำดับชั้น ของความสัมพันธ์ (class hierarchies)
  - เช่น ถ้า class C1 เป็น subclass ของ class C2 ดังนั้น instances ทุกดัวที่อยู่ใน class C1 ก็จะเป็น instances ที่อยู่ใน class C2 ด้วย
  - เช่น ถ้า class Professor เป็น subclass ของ class Staff ดังนั้น คนที่คนที่อยู่ใน class Professor ก็จะเป็นสมาชิกของ class Staff ด้วย ดังนั้น



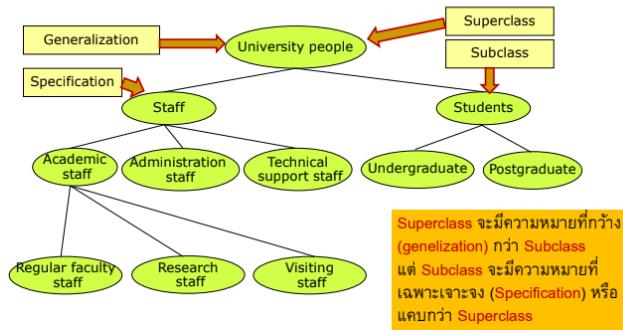
9



## Typical Components of Ontologies

- **คุณสมบัติ (Properties) ของ Class**
  - เช่น class Professor ก็จะมี Properties คือ name, address, email, teaches เป็นต้น
- **การกำหนดเงื่อนไข (restrictions)**
  - เช่น Professor จะสอน (teaches) ได้ไม่เกิน 3 วิชา เป็นต้น
  - หรือ คนที่เป็น Professor เท่านั้นที่จะให้คำปรึกษา (supervises) นักศึกษาระดับปริญญาโทหรือเอก (GraduateStudent) ได้

## Example of a Class Hierarchy



## The Role of Ontologies on the Web

- **Ontology** จะช่วยให้เราสามารถแบ่งปัน (share) ข้อมูลระหว่างกันได้ และทำให้เกิดการทำงานร่วมกันได้อย่างสื่อความหมาย (**semantic interoperability**)

- ยังช่วยแก้ปัญหา **synonym conflicts** ของคำที่เขียนต่างแต่ความหมายเหมือนกัน เช่น 'Staff' vs 'Employee' โดยจะใช้วิธีการเชื่อมความสัมพันธ์ระหว่างคำ
- และยังช่วยแก้ปัญหา **homonym conflicts** ของคำที่เขียนเหมือนกันแต่ความหมายต่างกันด้วย เช่น 'course' ที่แปลว่าวิชา และ 'course' ที่แปลว่าหลักสูตร โดยการใช้สิ่งที่เรียกว่า **URI/IRI**

## Ontology Engineering

- **Ontology Engineering** เป็นกระบวนการในการพัฒนา ontology เพื่อสร้างองค์ประกอบต่างๆของ ontology ขึ้นมาดังที่ได้กล่าวไปแล้ว
- ชั้นการพัฒนา ontology จะไม่ใช่ว่าจะสร้างเพียงแค่ครั้งเดียวแล้วก็ใช้ได้เลย
- แต่อาจจะต้องมีการแก้ไขปรับปรุงหลายครั้ง (iterative process) จนกว่าจะได้เป็น ontology ที่ครอบคลุมทุกสิ่งที่เราต้องการ

### Step 1. Determine the domain and scope of the ontology (2)

#### Competency Questions

- เมื่อเราทราบ Domain ของ ontology ที่แนัดแล้ว เราจะต้องหา scope หรือขอบเขตของ ontology ที่เราจะสร้างให้ได้ เราจำเป็นต้องพยายามสร้าง กลุ่มคำถามต่างๆขึ้นมา และกลุ่มคำตอบที่เราคิดว่าเราจะได้จาก ontology ยกตัวอย่างเช่น ในการพัฒนา ontology ของมหาวิทยาลัย
  - พนักงานสายผู้สอน (AcademicStaff) จะมีความแตกต่างจาก พนักงาน (staff) ทั่วไปอย่างไร
  - พนักงานสายผู้สอนสามารถแบ่งออกได้เป็นกี่กลุ่ม (Professor, AssociateProfessor, AssistantProfessor, Lecturer)
  - พนักงานแต่ละคนสังกัด (workIn) ภาควิชาอะไร คณะอะไร
  - คนที่เป็น Professor สามารถให้คำปรึกษาแก่นักศึกษาลุ่มใหม่ได้บ้าง (UndergraduateStudent หรือ GraduateStudent) เป็นต้น

20

### Step 1. Determine the domain and scope of the ontology

- ในขั้นตอนแรกสุดของการพัฒนา ontology คือจะต้องทำการหา Domain และ scope ของ ontology โดยจะต้องตอบคำถามต่อไปนี้ให้ได้
  - โดเมนที่ ontology จะครอบคลุมนั้นคืออะไร?
    - เช่น เรากำลังพัฒนา ontology ที่อยู่ในโดเมนของระบบงานในมหาวิทยาลัย หรือ โดเมนของการท่องเที่ยว หรืออาหารขยะของ เป็นต้น
  - เราจะใช้ ontology เพื่อให้สามารถตอบปัญหาอะไรได้บ้าง?
  - จะต้องตั้งค่าตามอย่างไร และต้องการค่าตอบอย่างไรจาก ontology
  - ควรจะใช้และค่อยปรับปรุงแก้ไข ontology?

### Step 2. Enumerate important terms in the ontology

- ขั้นตอนถัดไป จะนำคำถามที่สร้างไว้ในขั้นตอนที่ 1 หันหมวดมาทำการสกัด หาคำ (terms) ต่างๆที่เป็นคำสำคัญ โดยมีเป้าหมายเพื่อหา
  - มีคำใดบ้างที่เป็นคำนาม (noun) ที่จะสามารถนำมาใช้เป็น Concepts หรือ Classes
  - มีคำใดบ้างที่ทำหน้าที่เป็นเมื่อยอดคำ (verb) ที่จะสามารถนำมาทำเป็น properties ให้กับ classes ต่างๆได้
  - ตัวอย่างของคำที่สามารถสกัดได้จากคำถามต่างๆใน domain มหาวิทยาลัย เช่น Staff, Academic Staff Member, Lecturer, Professor, Faculty, Department, Subject, teaches, workIn, taughtBy, supervises, supervisedBy, name, address, phone, email เป็นต้น

20

### Step 3. Define the classes and the class hierarchy

- ต่อไปก็จะทำการหาความสัมพันธ์ระหว่าง Classes ในลักษณะของลำดับชั้น (class hierarchy) โดยการพิจารณาว่า ถ้า class นี้มี instances เป็นแบบนี้ instances เหล่านี้จะสามารถเป็นสมาชิกที่อยู่ใน class อื่นได้อีกหรือไม่ โดยยึดตามกฎเกณฑ์ดังนี้
  - ถ้า class A เป็น superclass ของ class B ดังนั้น instances ทุกด้านของ class B ก็จะต้องเป็น instances ของ class A ด้วย
  - subclass แต่ละตัว จะสามารถมี properties เป็นของตนเองได้ และยังสามารถรับการสืบทอด properties ที่สืบมาจาก superclass ได้อีกด้วย

### Step 3. Define the classes and the class hierarchy

- ขั้นตอนถัดไปคือการกำหนด Class (หรือ Concept) ขึ้นมา ซึ่งจะเป็นที่รวมของสิ่งที่เรียกว่า instances (หรือ objects) ซึ่ง classes จะได้แก่กลุ่มคำที่เป็นคำนาม (noun) ทั้งหลาย เช่น Staff, Academic Staff Member, Lecturer, Professor, Faculty, Department, Subject เป็นต้น
- เช่น class "Staff" ก็จะได้แก่กลุ่มคนที่เป็นพนักงานทุกคน เช่น Robert, Billy, Mary เป็นต้น ซึ่งพนักงานแต่ละคนนี้ก็จะถูกเรียกว่าเป็น instances ที่อยู่ใน class "Staff"

### Step 3. Define the classes and the class hierarchy

- จะสามารถเขียนความสัมพันธ์ระหว่าง Class A และ B โดยถ้า Class B เป็น subclass ของ Class A ในรูปแบบของรูปภาพได้ดังนี้

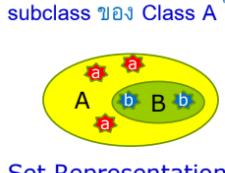
### Step 3. Define the classes and the class hierarchy(2)

#### The Class hierarchy process

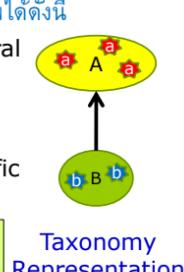
- ในการสร้างความสัมพันธ์แบบลำดับชั้นของ classes นี้เราอาจทำการสร้างโดยพิจารณาจาก class บนชั้น (top-level class) ซึ่งจะเป็น superclass ซึ่งจะเป็น class ที่มีความหมายที่กว้างกว่า class ที่อยู่ด้านล่าง เช่น ระหว่าง Staff, Lecturer และ Professor

- ➔ Staff จะถูกพิจารณาให้เป็น superclass ของ Lecturer กับ Professor ซึ่งสอง classes นี้จะถูกพิจารณาให้เป็น subclass ของ Staff เพราะมีความหมายแคบกว่า

- ถ้า subclasses ใดๆ สามารถ share properties ร่วมกันได้ เราสามารถสร้าง class ใหม่ขึ้นมาที่เป็น class ตรงกลาง (middle-level class) คือระหว่าง superclass และ subclasses เหล่านั้น เช่น AcademicStaff จะถูกสร้างขึ้นมาอย่างบัน class Lecturer และ Professor เพราะทั้งสอง class นี้มี Property ร่วมกันคือสามารถสอน (teaches) นักศึกษาได้เหมือนกัน



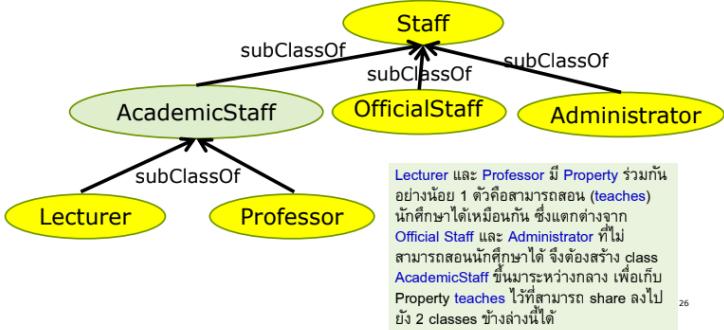
จากภาพทั้งสองนี้ จะพบว่า instances ทั้งหมดของ B ก็จะเป็น instances ของ A ด้วย



24



### Step 3. Define the classes and the class hierarchy (3)



### Step 4. Define the properties of classes(2)

#### ตัวอย่างเช่น

class Staff จะมี Properties ได้แก่

→ workIn, name, address, phone, email

class AcademicStaff จะมี Properties ได้แก่

→ teaches

class Subject จะมี Properties ได้แก่

→ taughtBy



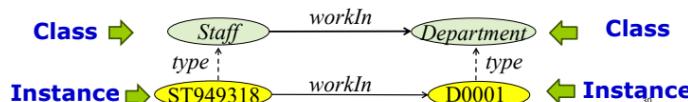
### Step 4. Define the properties of classes(4)

ObjectProperty จะเป็น Property ที่ใช้เชื่อมความสัมพันธ์ระหว่าง Resource หนึ่งกับอีก Resource หนึ่ง (ซึ่ง Resource อาจจะเป็น class หรือ instance ก็ได้)



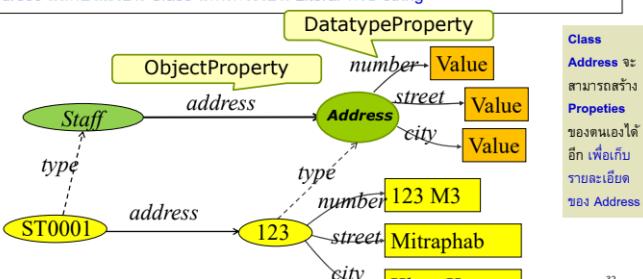
#### ยกตัวอย่างเช่น

จะต้องเขียนข้อความจากทางลูกศูนย์ไปที่หัวหน้า (Staff, workIn, Department) ซึ่งจะถูกเขียนว่า (Staff, workIn, Department)



### Step 4. Define the properties of classes(6)

นำกรงเรื่องของแบบให้ address มีลักษณะเป็น ObjectProperty ก็ได้ โดยกำหนดให้ value ของ address มีลักษณะเป็น Class แทนที่จะเป็น Literal หรือ string

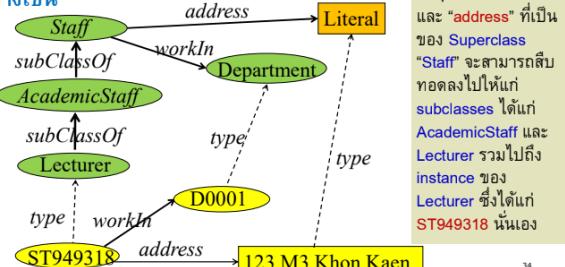


32



### Step 4. Define the properties of classes(8)

#### ตัวอย่างเช่น



34



### Step 4. Define the properties of classes

- หลังจากที่ได้ทำการกำหนด classes และความสัมพันธ์ระหว่าง classes ได้แล้ว ขั้นตอนถัดไปคือการกำหนด Properties (หรือบางที่จะเรียกว่า Predicate) ให้กับ classes ต่างๆเหล่านั้น
- ในการกำหนด Properties จะเป็นการนำคำต่างๆที่ทำหน้าที่เหมือนเป็นคำกริยา (verb) จาก step 2 มาทำให้เป็น Properties สำหรับ class
- ตัวอย่างเช่น คำว่า workIn, teaches, taughtBy, name, address, phone, email เป็นต้น
- Property แต่ละตัวนี้ จะต้องถูกนิยามพิจารณาว่าควรจะต้องเป็น Property ของ class ไหน เพื่อจะได้นำไปบูรณาไว้กับ class นั้น (ให้ class นั้นทำหน้าที่เป็นประธาน (Subject) ของ Property นั้น)

27



### Step 4. Define the properties of classes(3)

#### ประเภทของ Properties

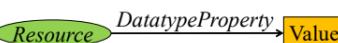
► Properties จะสามารถแบ่งออกได้เป็น 2 ประเภทได้แก่

- ObjectProperty
- DatatypeProperty

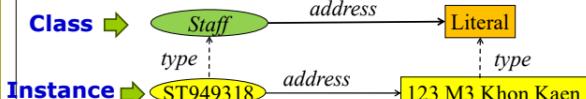


### Step 4. Define the properties of classes(5)

DatatypeProperty จะเป็น Property ที่ใช้เชื่อมความสัมพันธ์ระหว่าง Resource หนึ่งกับข้อมูลที่เป็น Literal หรือ XML Schema datatype. (ซึ่ง Resource อาจจะเป็น class หรือ instance ก็ได้)



จะต้องเขียนข้อความจากทางลูกศูนย์ไปที่หัวหน้า (Staff, address, Literal)



31



### Step 4. Define the properties of classes(7)

#### Inheritance Mechanism

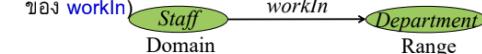
- Subclasses ทุกด้านจะสามารถรับ properties ของตนเองแล้ว ยังสามารถสืบทอด (inherit) properties มาจาก superclass ได้ด้วย
- Properties ที่ถูกกำหนดให้กับ superclass จะสามารถสืบทอด properties เหล่านั้นลงไป subclasses ต่างๆของ superclass นั้นได้อีกด้วย โดยสามารถสืบทอดลงไปจนถึง instances ต่างๆที่อยู่ใน subclasses เลยทีเดียว



### Step 5. Define the restrictions of the properties-facets

#### Domain and range of a property

- Class ซึ่งทำหน้าที่เป็นประธาน (Subject) ของ Property จะเรียก Class นั้นว่าเป็น Domain ของ property นั้น
  - ➔ เช่น Class "Staff" จะเป็น domain ของ Property "workIn"
- Class ซึ่งทำหน้าที่เป็นกรรม (Object) ของ Property จะเรียก Class นั้นว่าเป็น Range ของ property นั้น
  - ➔ เช่น Class "Department" จะเป็น range ของ Property "workIn" (แต่ instance แต่ละตัวที่อยู่ใน Staff หรือ Department เช่น ST949318 หรือ D0001 จะไม่ถูกเรียกว่าเป็น domain, หรือ range ของ workIn)



35

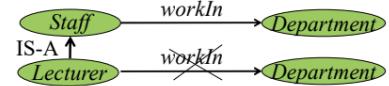


## Step 5. Define the restrictions of the properties-facets (2)

### □ Domain and range of a property (2)

- Class ซึ่งเป็น subclass ของ Superclass จะสามารถสืบทอด properties ของ superclass ลงมาที่ subclass นั้นได้ โดยที่ไม่ต้องสร้าง properties เหล่านั้นที่ซ้ำกันให้กับ subclass อีก

→ เช่น ถ้า class "Lecturer" เป็น subclass ของ class "Staff" และ class "Staff" เป็น domain ของ property "workIn" และมี class "Department" เป็น range ของ "workIn" ดังนั้น ไม่ต้องกำหนดให้ "workIn" มี domain เป็น class "Lecturer" ด้วย เพราะคอมพิวเตอร์จะสืบทอด Property "workIn" ให้กับ class "Lecturer" อัตโนมัติ



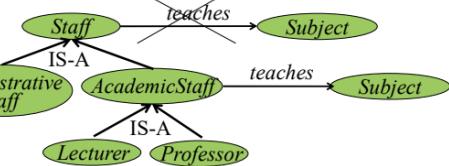
36



## Step 5. Define the restrictions of the properties-facets (3)

### □ Domain and range of a property (3)

- การกำหนด Property ให้กับ class จะต้องแน่ใจว่า Property นั้นจะถูกสืบทอด (inherit) ไปยัง subclasses ข้างล่างได้ถูกต้องหรือไม่ดังด้วย. ต่อไปนี้



ตัวนี้ จึงควรร่าง property "teaches" ให้ domain เป็น AcademicStaff จะเหมาะสมมาก ซึ่งก็จะสามารถถูกต่อไปยัง Lecturer และ Professor ได้ หากทั้งคู่ ก็สองอย่างล้วนเกิดขึ้นที่สามารถใช้ property "teaches" ได้

Property "teaches" ถูกกำหนดให้มี domain เป็น Staff และว่า Property "teaches" จะสามารถสืบทอดไปยัง subclasses ของ Staff ได้ทั้งหมด สำหรับ AdministativeStaff ซึ่ง เป็นผู้บริหารศึกษาการ teaches ให้กับ ชั้นเรียนอาจไม่ค่อยการได้ AdministativeStaff สอน ได้แต่การให้ AcademicStaff เก็บนั้นที่สอนได้



## Step 6. Create instances

### □ ขั้นตอนสุดท้ายคือการสร้าง instances ให้กับ class แต่ละ class ใน ontology

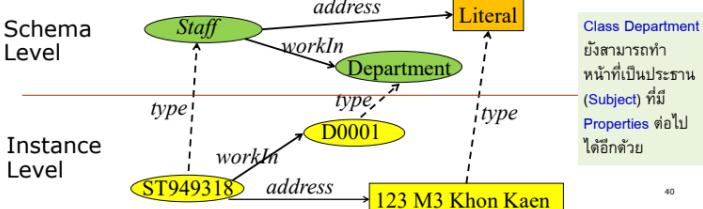
#### □ ในการสร้าง instance ให้กับ class มีขั้นตอนคือ

- เลือก class ที่จะกำหนด instances ลงไว้
- ทำการสร้าง instance ให้กับ class นั้น
- ต้องเข้าใจว่า instances ทุกด้านของ class นั้นจะสามารถสืบทอด (inherits) properties ทุกด้านของ class นั้นได้ รวมไปถึงจาก superclass ของ class นั้นด้วย
- ทำการกำหนด property values ให้กับ instance แต่ละตัว



## Step 6. Create instances(2)

❖ ข้อ workIn เป็น objectProperty ที่มี value เป็น class Department ดังนั้นใน instance level ก็ต้องกำหนด value ของ workIn ให้เป็น instance เช่น "D0001" ที่อยู่ใน class Department ส่วน address เป็น datatype property ที่มี value เป็น Literal ดังนั้นใน instance level ก็ต้องกำหนดค่า value ของ address ให้มีค่าเป็น string ที่สอดคล้องกับ Literal ด้วย



40



## An Instance or a Class?

- ตัวอย่างเช่น ถ้าเรามีการกำหนดว่ากลุ่มวิชาทาง Computer Science (Cscourse) จะประกอบด้วยกลุ่มวิชาด้าน Web Technology และกลุ่มวิชาด้าน Network
- กลุ่มวิชาด้าน Web Technology ก็จะประกอบด้วยวิชา "322436" และ "322736" เป็นต้น
- กลุ่มวิชาด้าน Network ก็จะประกอบด้วยวิชา "322588" และ "322725" เป็นต้น
- จากข้อมูลดังกล่าว จะสามารถออกแบบว่าข้อมูลใดควรออกแบบให้เป็น classes ข้อมูลใดควรออกแบบให้เป็น instances บ้าง



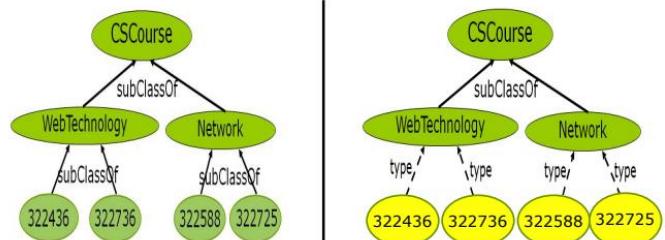
## An Instance or a Class?

- จะรู้ได้อย่างไรสิ่งที่กำลังพูดถึงนี้จะออกแบบให้เป็น Instance หรือเป็น Class ดี? เราจะต้องพิจารณาจากคุณสมบัติของ class และ instance ดังนี้
- Class จะเป็นที่รวมของ group of instances ที่มีการ share properties ที่เหมือนกัน
- Class จะสามารถถูกแปลง成 subclasses ได้
- สำหรับ Instance นั้น จะเป็นชิ้นของข้อมูลหรือของโครงสร้างเรียกว่า object ที่สามารถถูกเก็บอยู่ใน Class อีกที
- Instance แต่ละ instance จะไม่สามารถถูกแปลง成อีกให้เป็น subInstances ได้

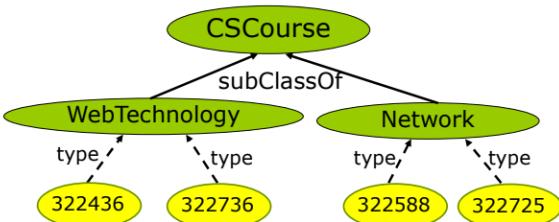


## An Instance or a Class? (2)

- เราควรออกแบบให้ "322436", "322736", "322588", "322725" มีลักษณะเป็น Classes หรือเป็น Instances ดี?



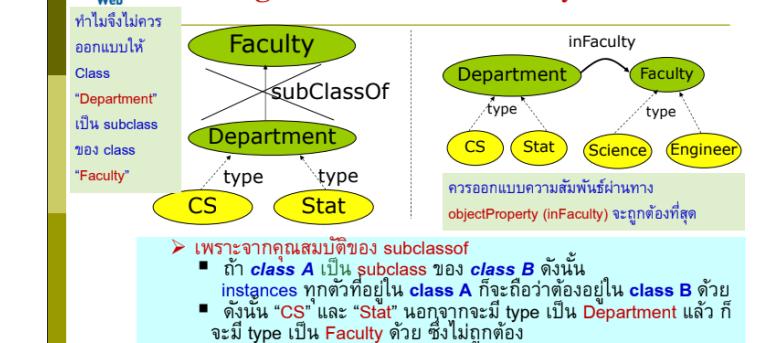
## An Instance or a Class? (3)



▪ เราไม่ควรออกแบบให้วิชาแต่ละวิชาเป็น 322436 หรือ 322736 มีลักษณะเป็น subclasses ของ WebTechnology class เพราะวิชาแต่ละวิชาที่จะไม่สามารถมี instances อุปกรณ์เช่นเหล็กนี้ได้อีก และไม่สามารถแบ่งย่อยออกเป็นกลุ่มของวิชาอย่างๆ (subclass) ได้อีก จึงควรออกแบบให้มีลักษณะเป็น instance จะเหมาะสมที่สุด



## Ensuring that the class hierarchy is correct



- เพราะจากคุณสมบัติของ subclassof
  - ถ้า class A เป็น subclass ของ class B ดังนั้น instances ทุกด้านที่อยู่ใน class A ก็จะต้องอยู่ใน class B ด้วย
  - ดังนั้น "CS" และ "Stat" น่าจะมาจากมี type เป็น Department และมี type เป็น Faculty ด้วย ซึ่งไม่ถูกต้อง



## Domain of Discourse

- จากการบ้านที่แล้ว ที่ให้ถูกระบุลงข้อความต่อไปนี้ให้เป็นโครงสร้างของอนโนทेशันระดับ Schema level จะมีข้อดังนี้อย่างไรบ้าง

นักศึกษา (Student) แต่ละคนจะสามารถลงทะเบียน (enroll) วิชา (Subject) แต่ละวิชาได้ โดยวิชา (Subject) แต่ละวิชาจะสามารถสมัครเข้าเรียน (relatedTo) กับวิชา (Subject) อื่นๆได้ด้วย นอกจากนี้ วิชา (Subject) แต่ละวิชาจะต้องอยู่ใน (inProgram) หลักสูตร (Program) ซึ่งหลักสูตร (Program) ก็จะสังกัด (inDepartment) ภาควิชา (Department) อีกที่ นักศึกษา (Student) แต่ละคน จะต้องถูกดูแล (supervisedBy) โดยอาจารย์สายผู้สอน (AcademicStaff) ซึ่งอาจารย์สายผู้สอน (AcademicStaff) นี้จะทำงาน (workIn) ในภาควิชา (Department) ซึ่งภาควิชา (Department) นี้จะถูกควบคุม (controlledBy) คณะ (Faculty) อีกที่ และอาจารย์สายผู้สอน (AcademicStaff) ก็จะประกอบด้วยกลุ่มอาจารย์ที่เป็นอาจารย์ (Lecturer) ผู้ช่วยศาสตราจารย์ (AssistantProfessor) รองศาสตราจารย์ (AssociatedProfessor) หรือ ศาสตราจารย์ (Professor) ก็ได้



## Step 1: Define the Classes

นักศึกษา (Student) แต่ละคนจะสามารถลงทะเบียน (enroll) วิชา (Subject) แต่ละวิชาได้ โดยวิชา (Subject) แต่ละวิชาจะสามารถสมัครเข้าเรียน (relatedTo) กับวิชา (Subject) อื่นๆได้ด้วย นอกจากนี้ วิชา (Subject) แต่ละวิชาจะต้องอยู่ใน (inProgram) หลักสูตร (Program) ซึ่งหลักสูตร (Program) ก็จะสังกัด (inDepartment) ภาควิชา (Department) อีกที่ นักศึกษา (Student) แต่ละคน จะต้องถูกดูแล (supervisedBy) โดยอาจารย์สายผู้สอน (AcademicStaff) ซึ่งอาจารย์สายผู้สอน (AcademicStaff) นี้จะทำงาน (workIn) ในภาควิชา (Department) ซึ่งภาควิชา (Department) นี้จะถูกควบคุม (controlledBy) คณะ (Faculty) อีกที่ และอาจารย์สายผู้สอน (AcademicStaff) ก็จะประกอบด้วยกลุ่มอาจารย์ที่เป็นอาจารย์ (Lecturer) ผู้ช่วยศาสตราจารย์ (AssistantProfessor) รองศาสตราจารย์ (AssociatedProfessor) หรือ ศาสตราจารย์ (Professor) ก็ได้



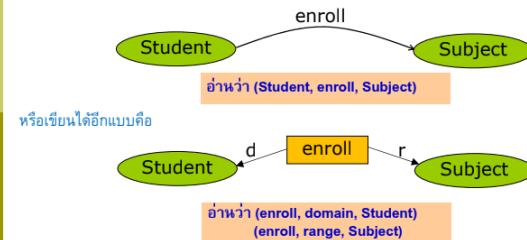
## Step 2: Define the Properties

นักศึกษา (Student) แต่ละคนจะสามารถลงทะเบียน (enroll) วิชา (Subject) แต่ละวิชาได้ โดยวิชา (Subject) แต่ละวิชาจะสามารถสมัครเข้าเรียน (relatedTo) กับวิชา (Subject) อื่นๆได้ด้วย นอกจากนี้ วิชา (Subject) แต่ละวิชาจะต้องอยู่ใน (inProgram) หลักสูตร (Program) ซึ่งหลักสูตร (Program) ก็จะสังกัด (inDepartment) ภาควิชา (Department) อีกที่ นักศึกษา (Student) แต่ละคน จะต้องถูกดูแล (supervisedBy) โดยอาจารย์สายผู้สอน (AcademicStaff) ซึ่งอาจารย์สายผู้สอน (AcademicStaff) นี้จะทำงาน (workIn) ในภาควิชา (Department) ซึ่งภาควิชา (Department) นี้จะถูกควบคุม (controlledBy) คณะ (Faculty) อีกที่ และอาจารย์สายผู้สอน (AcademicStaff) ก็จะประกอบด้วยกลุ่มอาจารย์ที่เป็นอาจารย์ (Lecturer) ผู้ช่วยศาสตราจารย์ (AssistantProfessor) รองศาสตราจารย์ (AssociatedProfessor) หรือ ศาสตราจารย์ (Professor) ก็ได้



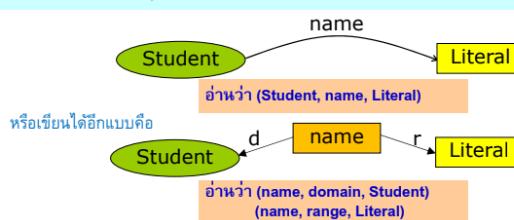
## Step 2: Define the Properties (2)

- Property** ที่ใช้เชื่อมความสัมพันธ์ระหว่าง Classes หรือระหว่าง Instances ด้วยกัน จะเรียก Property นั้นว่า **ObjectProperty**

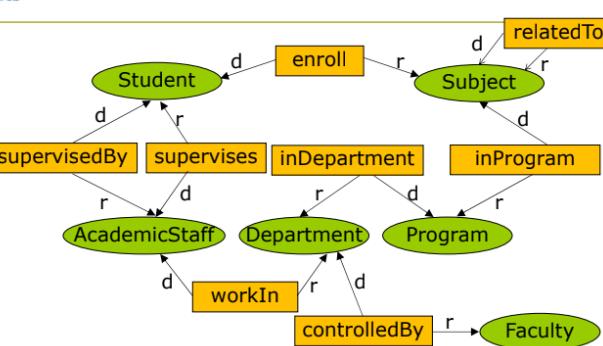


## Step 2: Define the Properties (3)

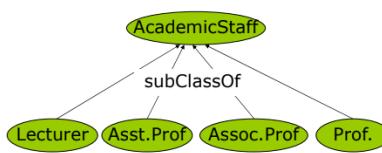
- Property** ที่ใช้เชื่อมความสัมพันธ์ระหว่าง Class กับ Literal หรือระหว่าง Instance กับ String จะเรียก Property นั้นว่า **DatatypeProperty**
- Note:** Literal จะถูกสร้างให้เป็น Class ประเภทหนึ่ง ซึ่งเก็บข้อมูลเป็น String แต่เราสามารถกำหนดค่า value of DatatypeProperty ให้เป็นประเภท date, integer, decimal, หรืออื่นๆที่ได้โดยอิมเมจจาก XML Schema Datatype.



## Step 2: Define the Properties (4)



## Step 3: Define a Class Hierarchy



## A Class Hierarchy: Transitivity

- จะต้องแนใจว่าความสัมพันธ์ระหว่าง classes ที่เป็นแบบลำดับชั้นนั้น มีความถูกต้องตามคุณสมบัติ Transitivity (Transitive Relationship)

- ความสัมพันธ์ระหว่าง classes จะมีลักษณะของคุณสมบัติ Transitive Relationship กล่าวคือ ถ้า C เป็น subclass ของ B และ B เป็น subclass ของ A ดังนั้น C ก็จะเป็น subclass ของ A ด้วย



## A Class Hierarchy: Multiple Inheritance

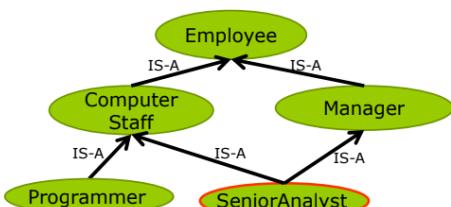
### Multiple Inheritance

- สามารถกำหนดให้ class หนึ่งสามารถเป็น subclass ของ class อื่น ได้มากกว่าหนึ่ง class
- ถ้า class A เป็น subclass ของ class B และ class C ดังนั้น instances ทุกด้วยของ A ก็จะเป็น instances ที่รวมอยู่ใน class B และ C ด้วย
- นอกจากนี้ Class A ก็จะสามารถสืบทอด Properties และ restrictions ต่างๆมาจากการพ่อแม่ (Parent classes) ซึ่งได้แก่ class B และ class C ด้วย

## A Class Hierarchy: Multiple Inheritance

### Multiple Inheritance

- For example,



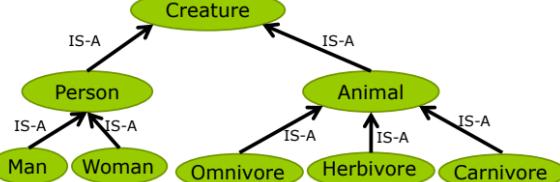
ด้วยว่าถ้า **SeniorAnalyst** จะเป็น subclass ของ **ComputerStaff** และ **Manager** classes ดังนั้น คน ก็จะเป็น **SeniorAnalyst** ก็จะ ก็ถือว่าเป็น **ComputerStaff**, **Manager** และ **Employee** นอกจากนี้ **SeniorAnalyst** นอกจากจะมี Properties ที่เป็น ของตนเองแล้ว ก็ยังสามารถสืบทอด Properties ต่างๆที่อยู่ใน **ComputerStaff**, **Manager** และ **Employee** มาใช้ได้หมดด้วย

15

## A Class Hierarchy: Properties Inheritance

### Properties Inheritance

- ถ้าเรามีโครงสร้าง ontology ดังต่อไปนี้



## A Class Hierarchy: Properties Inheritance

### Properties Inheritance

#### Class Man

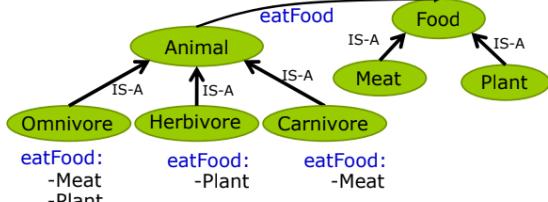
- สามารถสืบทอด properties ทุกด้วยจาก class Person
- สามารถสร้าง Properties ของตนเองเพิ่มขึ้นมาได้ด้วย เช่น ObjectProperty: hasWife(Woman)

#### Class Woman

- สามารถสืบทอด properties ทุกด้วยจาก class Person
- สามารถสร้าง Properties ของตนเองเพิ่มขึ้นมาได้ด้วย เช่น ObjectProperty: hasHusband(Man)

## A Class Hierarchy: Properties Inheritance

ในการแปลงโครงสร้าง ontology นี้ให้เป็นภาษา Ontology ที่คอมพิวเตอร์เข้าใจ เช่น ภาษา OWL (Ontology Web Language) เราจะสามารถกำหนดให้ subclass แต่ละชั้นเมื่อไหร่ก็ได้ในการ สืบทอด Properties มาจาก parent class และ สามารถกำหนดให้ Properties เหล่านั้นมี range ที่แตกต่างจากที่ได้รับมาจาก parent class ได้ด้วย (จะได้เรียนอีกรอบในบท OWL)



21

## A Class Hierarchy: Subclass

### คุณลักษณะของ Subclass

- Subclass** จะสามารถมี properties ที่เป็นของตนเอง ซึ่งจะไม่มี properties เหล่านี้อยู่ใน superclass
- Subclass** สามารถสืบทอด properties ทุกด้วยจาก Superclasses ทั้งหลายมาได้หมด เพื่อนำมาใช้กับตัวเอง
- Subclass** สามารถมีการกำหนดเงื่อนไข หรือ restrictions ที่แตกต่างจาก superclass ได้
- Subclass** สามารถทำการเปลี่ยนแปลงเงื่อนไขของ Properties ที่ได้รับจาก superclass ได้ เช่น เปลี่ยนแปลงค่า Range ของ Property ให้เป็นแบบอื่น เป็นต้น

## A Class Hierarchy: Properties Inheritance

### Properties Inheritance

เราสามารถกำหนด Properties ต่าง ๆ ให้กับแต่ละ Class ได้ดังนี้

#### Class Creature

- DataProperty: habitat,

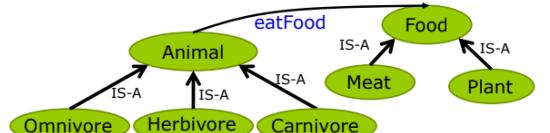
- ObjectProperty: eatFood (kind of food)

#### Class Person

- สามารถสืบทอด properties ทุกด้วยจาก class Creature
- สามารถสร้าง Properties ของตนเองเพิ่มขึ้นมาได้ด้วย เช่น DataProperty: Name, dateOfBirth, Nationality

## A Class Hierarchy: Properties Inheritance

### ให้พิจารณา property "eatFood" ที่ถูกสืบทอดลงไปที่ class Animal



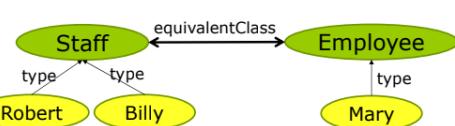
จากโครงสร้าง ontology นี้ subclasses ทุกด้วยของ Animal จะสามารถสืบทอด property "eatFood" มาจาก Animal ได้ เช่น "eatFood" นี้จะมี range เป็น Food ซึ่งได้แก่ Meat หรือ Plant ดังนั้น ก็จะหมายความว่า Animal ทุกด้วยจะเป็น Omnivore (สัตว์กินทุกอย่าง เช่นเนื้อ, Herbivore (สัตว์กินพืช) และ Carnivore (สัตว์กินเนื้อ) ก็จะสามารถกินได้ทั้ง พืช (Plant) และเนื้อ (Meat) เมื่อมันกินหมู ดูเป็นสิ่งที่เราไม่ต้องการให้เป็นแบบนี้

20

## Equivalent Classes

ถ้ามีการกำหนดให้ class หนึ่งมีความเท่าเทียม (equivalence) กับอีก class หนึ่ง แสดงว่า classes ทั้งสองนี้จะมีการบรวม instances ให้เป็นกลุ่มเดียวกัน

- Classes ทั้งสองนี้จะถูกเรียกว่าเป็น synonymous classes กัน



ถ้ามีการสืบทอดกับคอมพิวเตอร์ให้ทำการจัด instances ทุกคนที่มี type เป็น Staff (หรือ Employee) คอมพิวเตอร์จะจดจำชื่อ 3 คน (Robert, Billy, และ Mary)

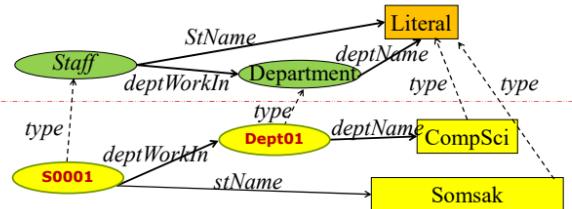
## Ontology Design VS Database Design

### Staff Table

StfID	StName	DeptWorkIn
S0001	Somsak	Dept01

### Department Table

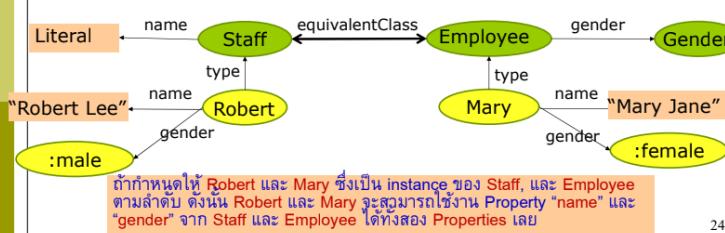
DeptID	DeptName
Dept01	CompSci



24

## Equivalent Classes

นอกจากนี้แล้วมีการกำหนดให้ class หนึ่งมีความเท่าเทียม (equivalence) กับอีก class หนึ่ง Properties ของแต่ละ class จะสามารถใช้งานร่วมกันได้ด้วย เพราะคอมพิวเตอร์จะมองว่าทั้งสอง class นี้จะเป็น class เดียวกัน



## Ontology Design VS Database Design

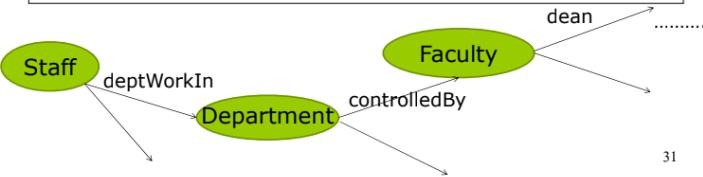
- ❑ จากตัวอย่างข้างต้น จะพบว่า
- Class จะเปรียบเทียบได้กับ Entity หรือ Table (ตาราง) ใน Database
  - ➔ ในดบ.นี้จะได้แก่ class "Staff" และ "Department"
- ObjectProperty จะทำหน้าที่เปรียบเหมือนกับเป็นคีย์นอก (foreign key) ใน Database ที่ใช้เชื่อมความสัมพันธ์ระหว่างตาราง (table)
  - ➔ ในดบ.นี้จะได้แก่ property "deptWorkIn" ที่ใช้เชื่อมระหว่าง class "Staff" และ "Department"

## Ontology Design VS Database Design

- ❑ ObjectProperty และ DataProperty จะถูกใช้เชื่อมความสัมพันธ์ระหว่าง ประชาน (Subject) หรือที่เรียกว่า domain of property ไปยัง value of property หรือที่เรียกว่า range of property ที่สามารถเขียนให้อยู่ในรูปแบบของ direct graph ได้ดังในหน้าต่อไปนี้

## Ontology Design VS Database Design

- ❑ Note: Object จะสามารถทำหน้าที่เป็น Subject ของประโยคดัดแปลงไปได้อีก ซึ่งก็จะมี Properties ที่เป็น ObjectProperty หรือ DatatypeProperty ต่อไปได้อีกเรื่อยๆ
- ❑ ทราบได้ที่ยังมี ObjectProperty ก็จะสามารถสร้างความสัมพันธ์ต่อไปได้เรื่อยๆ เป็นลักษณะของ Semantic Network



## Ontology Design VS Database Design

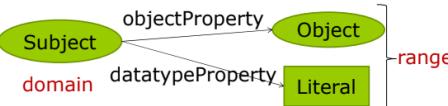
- ❑ Schema Level (in triple format)
  - (Staff, type, Class)
  - (Department, type, Class)
  - (deptWorkIn, type, ObjectProperty)
  - (deptWorkIn, domain, Staff)
  - (deptWorkIn, range, Department)
  - (stName, type, DatatypeProperty)
  - (stName, domain, Staff)
  - (stName, range, Literal)
  - (deptName, type, DatatypeProperty)
  - (deptName, domain, Department)
  - (deptName, range, Literal)

## Ontology Design VS Database Design

- ❑ นอกจากนี้จะพบว่า
- DatatypeProperty จะเปรียบเหมือนกับเป็น Attributes ธรรมดaicain Table ของ Database ที่เก็บข้อมูลเป็น string, integer, date, decimal หรืออื่นๆ
  - ➔ ในดบ.นี้จะได้แก่ property "stName", "deptName" ซึ่งทั้งสองตัวนี้จะเก็บข้อมูลเป็น string ทั้งคู่ ที่สอดคล้องกับ Literal
- Instance จะเปรียบเหมือนกับ Primary key ที่อยู่ใน Table ของ Database
  - ➔ ในดบ.นี้จะได้แก่ "S0001" และ "Dept01"
  - ➔ ชื่อ "S0001" จะเป็น instance ที่อยู่ใน class "Staff"
  - ➔ ส่วน "Dept01" จะเป็น instance ที่อยู่ใน class "Department"

## Ontology Design VS Database Design

- ❑ A simple direct graph

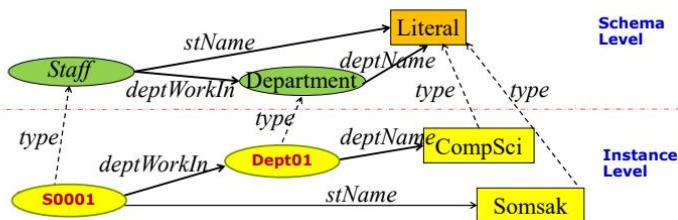


- ❑ A triple form of statement

(Subject, objectProperty, Object)  
(Subject, datatypeProperty, Literal)

## Ontology Design VS Database Design

- ❑ Schema Level vs Instance Level



## Ontology Design VS Database Design

- ❑ Instance Level or Document Level

- There are seven statements or seven triples
  - (S0001, type, Staff)
  - (S0001, deptWorkIn, Dept01)
  - (S0001, stName, "Somsak")
  - (Dept01, type, Department)
  - (Dept01, deptName, "CompSci")
  - ("CompSci", type, Literal)
  - ("Somsak", type, Literal)