Faculty of
Information Technology
King Mongkut's Institute of Technology Ladkrabang

Chapter 11: Introduction to JSON Document and Schema

By Dr. Taravichet Titijaroonroj

Practical NoSQL Database

Outline

- JSON Document
- JSON Schema

Outline

- JSON Document
- JSON Schema

JavaScript Object Notation (JSON)

คือ Message ประเภทตัวอักษรที่ง่ายต่อการและทำความเข้าใจของมนุษย์ ได้แก่ JSON และ XML เป็นต้น

JavaScript Object Notation: JSON คือ การเก็บข้อมูลแบบกลุ่มของคุณลักษณะ

```
{
   [
        { "firstName":"John", "lastName":"Doe" },
        { "firstName":"Peter", "lastName":"Jones" }
   ]
}
```

Extensible Markup Language: XML คือ การเก็บข้อมูลแบบกลุ่มของ key-value แบบ Markup

```
<employees>
      <employee>
            <firstName>John</firstName> <lastName>Doe</lastName>
      </employee>
      <employee>
            <firstName>Peter</firstName> <lastName>Jones</lastName>
      </employee>
</employees>
```

JavaScript Object Notation (JSON)

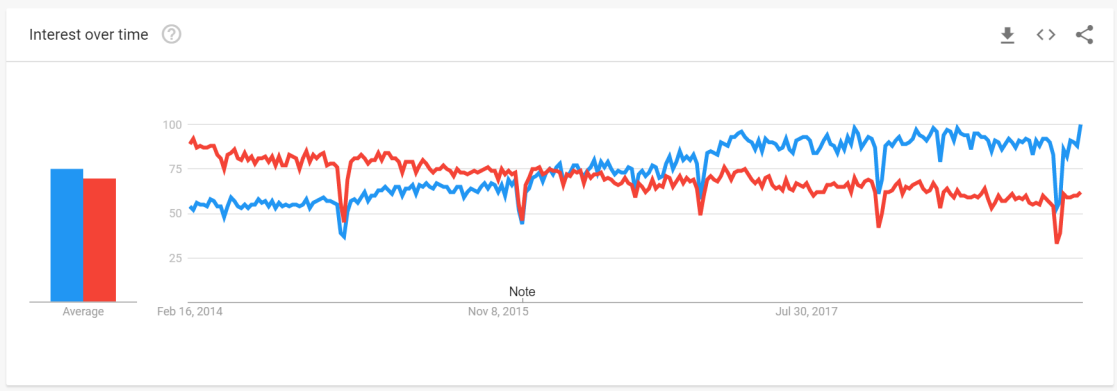
JavaScript Object Notation: JSON

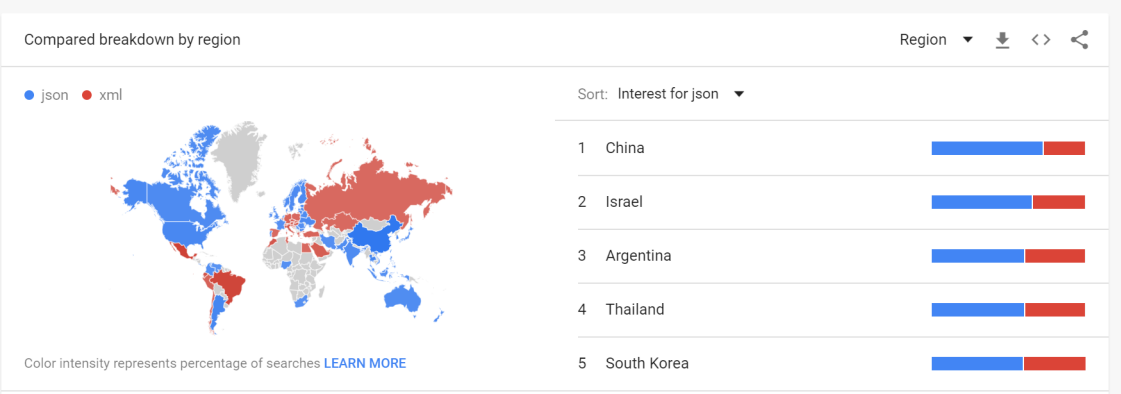
```
{
  [
     { "firstName":"John", "lastName":"Doe" },
     { "firstName":"Peter", "lastName":"Jones" }
  ]
}
```

Extensible Markup Language: XML

```
<employees>
      <employee>
            <firstName>John</firstName> <lastName>Doe</lastName>
      </employee>
      <employee>
            <firstName>Peter</firstName> <lastName>Jones</lastName>
      </employee>
</employees>
```

JSON และ XML คือ รูปแบบ (Format) สำหรับการรับส่งข้อมูลระหว่างเครื่องผู้ให้บริการ (Server) กับเครื่องผู้รับบริการ (Client) ซึ่งเป็นการคุยกันระหว่างแอพพลิเคชั่น





ความแตกต่างระหว่าง JSON และ XML ?

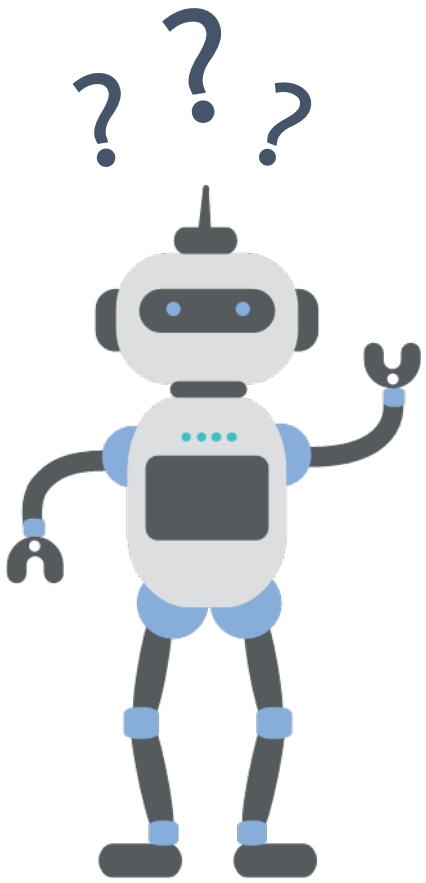
JSON	XML
JSON stands for JavaScript Object Notation and is based on JavaScript language.	XML is short for Extensive Markup Language and is derived from SGML.
It supports text and number data types including integer and strings, and arrays and objects.	It has no direct support for array.
It's a language-independent data-interchange format which supports only UTF-8 encoding.	It's an independent data format which supports different encodings.
It does not contain start and end tags.	It contains start and end tags.
It does not support native objects.	It gets support of objects via attributes and elements.
No support for Namespaces.	Namespaces are supported in XML.

DifferenceBetween.net

ความแตกต่างระหว่าง JSON และ XML ?

XML

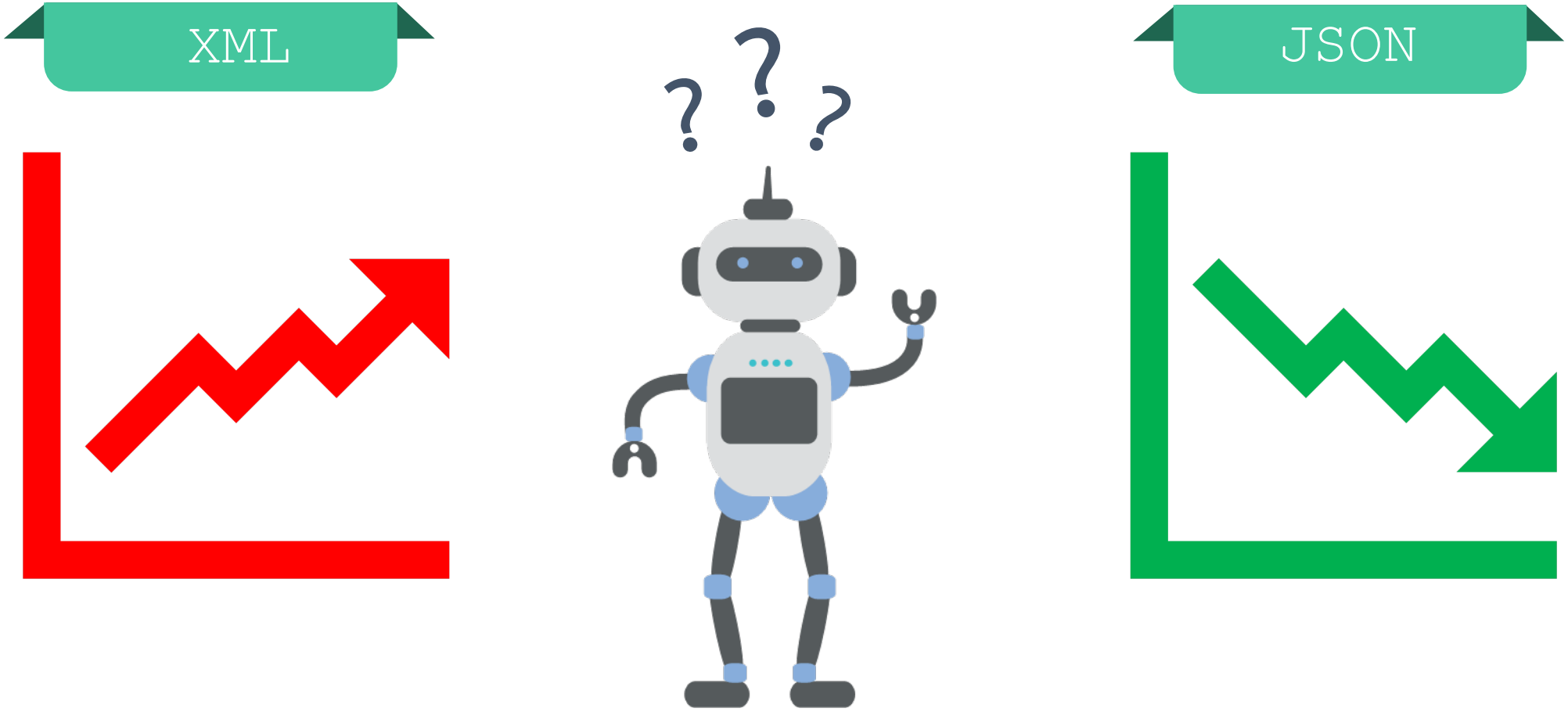
- เป็นภาษาที่พัฒนามาจาก SGML
- **ไม่รองรับ**ชนิดข้อมูลอาร์เรย์
- ตัวโครงสร้างภาษา**มี**คำขึ้นหน้าและคำลงท้าย
- รองรับข้อมูลประเภทวัตถุ (Object)



JSON

- เป็นภาษาที่พัฒนามาจาก JavaScript
- **รองรับ**ชนิดและโครงสร้างข้อมูลหลากหลาย (arrays)
- ตัวโครงสร้างภาษา**ไม่มี**คำขึ้นหน้าและคำลงท้าย (Start and end tags)
- รองรับข้อมูลประเภทวัตถุ (Object)

ความแตกต่างระหว่าง JSON และ XML ?



- คำฟุ่มเฟือย (Verbosity) และความซับซ้อน (Complexity) เชิงโครงสร้างไวยากรณ์ระหว่าง XML และ JSON

ความแตกต่าง Relational DB กับ JSON

	Relational DB	JSON
Structure	ตาราง	โครงสร้างการซ้อน (Object / array)
Schema	แน่นอน	ยืดหยุ่น
Query	ในรูปแบบ SQL	ใช้ได้อย่างหลากหลาย

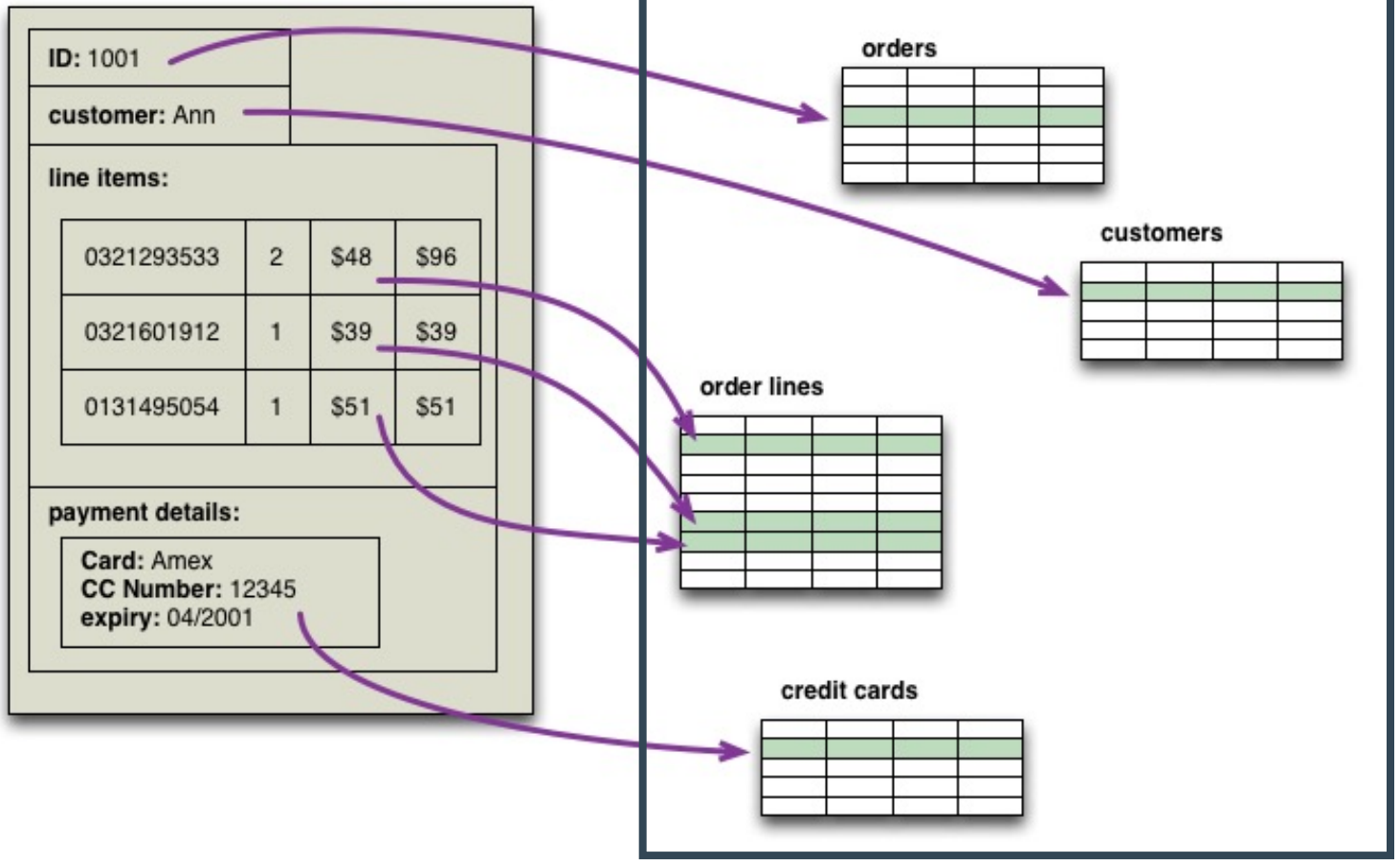
ความแตกต่าง Relational DB กับ JSON

JSON ย่อมาจากคำว่า “JavaScript Object Notation” โดยมี Douglas Crockford เป็นผู้พัฒนา JSON เป็นรูปแบบ**การแลกเปลี่ยนข้อมูลและการจัดเก็บข้อมูล**ที่ง่ายและรวดเร็วโดยไม่ยึดติดกับภาษาที่ใช้ในการพัฒนาโปรแกรม หรือสามารถกล่าวได้ว่าJSON เป็น “Wire Protocol” ที่ใช้ในการรับส่งข้อมูลระหว่างกัน ซึ่งสะดวกต่อการอ่านและเขียนโดยมนุษย์ นอกจากนั้น ยังสะดวกต่อการนำไปวิเคราะห์และสังเคราะห์โดยคอมพิวเตอร์เช่นกัน

ความแตกต่าง Relational DB กับ JSON

JSON

Relational Database

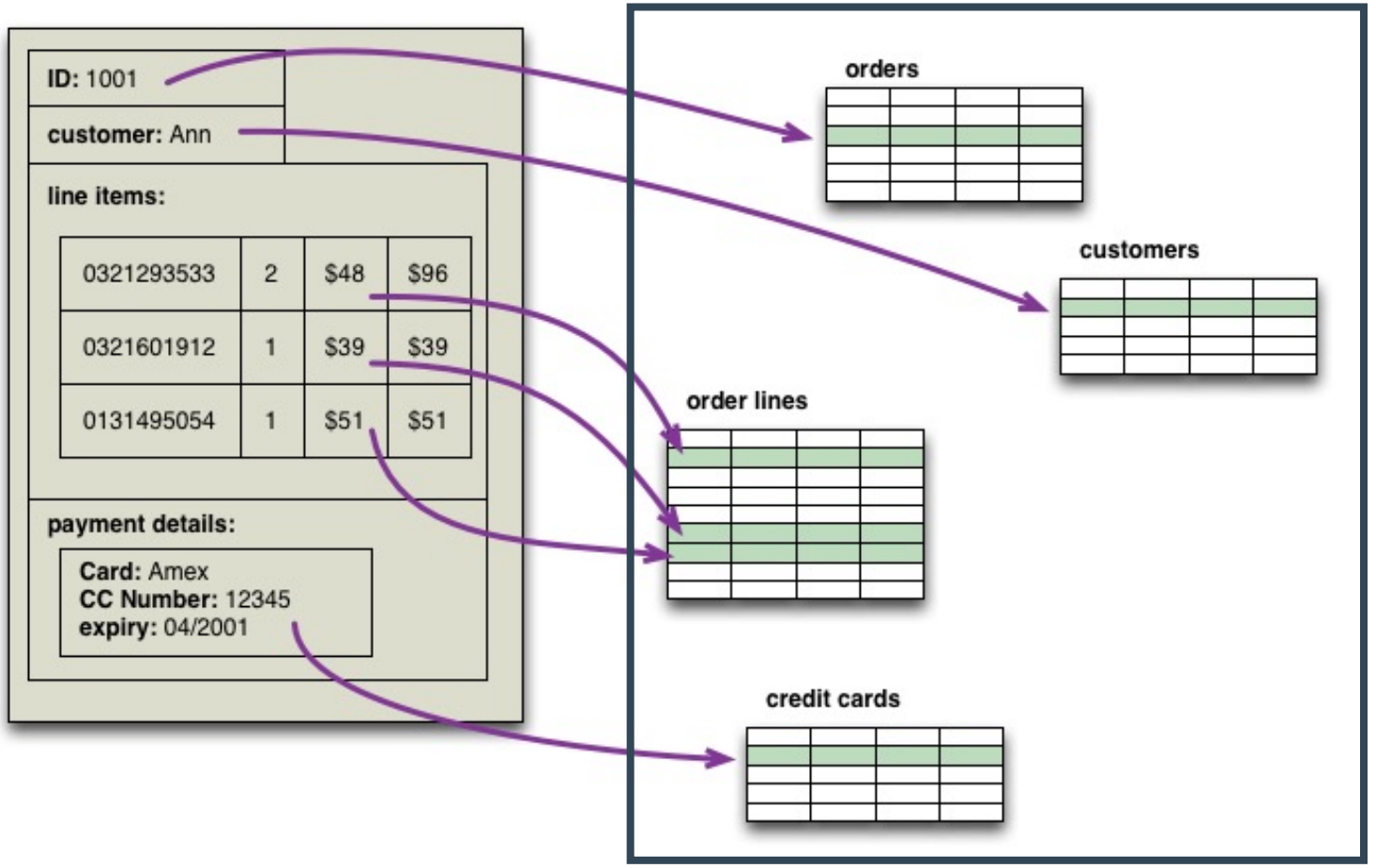


ความแตกต่าง Relational DB กับ JSON

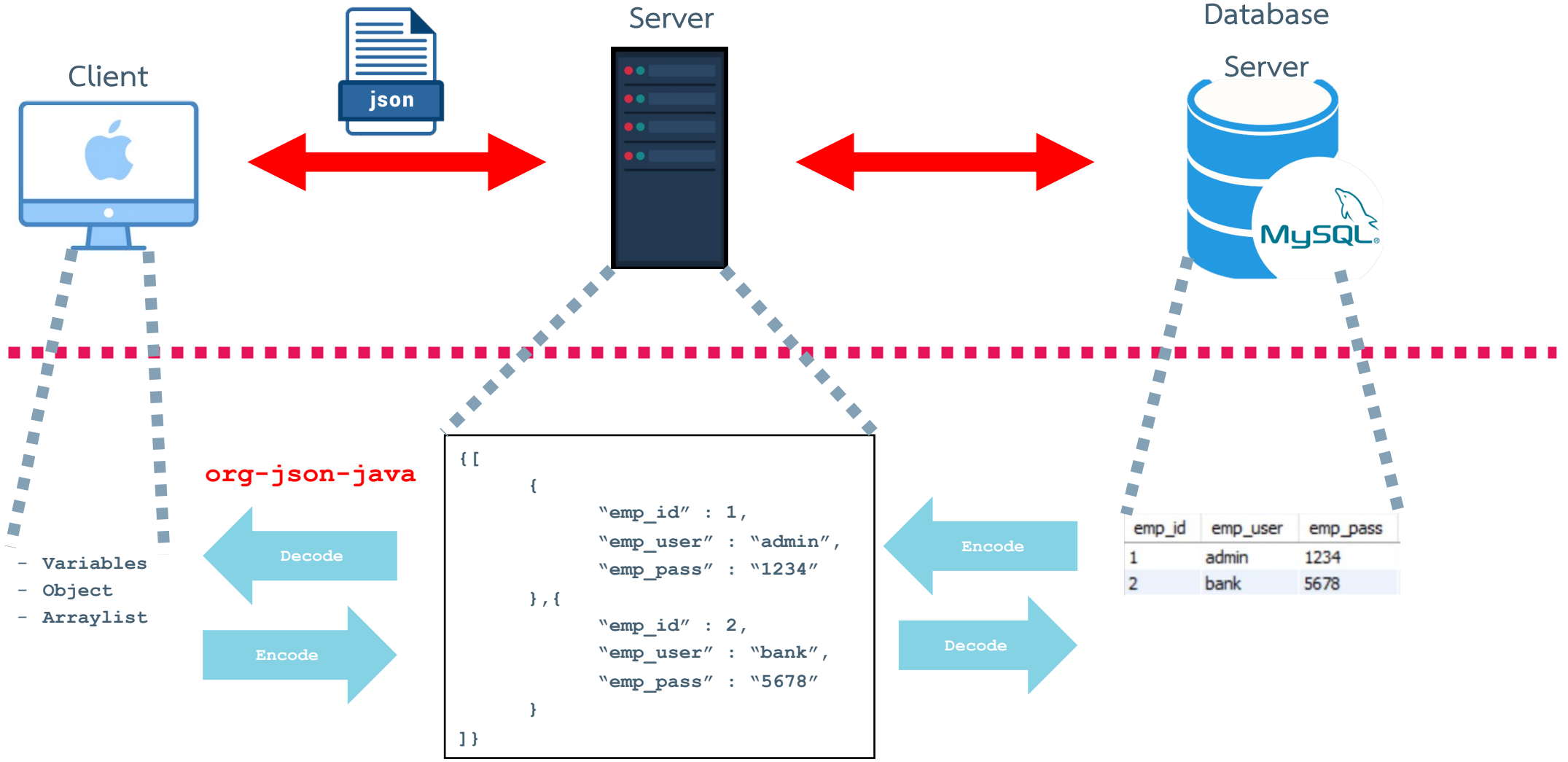
JSON

```
{
  "ID" :  1001,
  "customer" : "Ann",
  "Line_Items" : [
        [0321293533,2,48,96],
        [0321601912,1,39,39],
        [0131495054,1,51,51]
   ],
  "Payment_Details" : {
        "Card" : "Amex",
        "CC_Number" : 12345,
        "expiry" : "04/2001"
   }
}
```

Relational Database



ความแตกต่าง Relational DB กับ JSON



JavaScript Object Notation (JSON)

Introducing JSON

العربية Български 中文 Český Dansk Nederlands English Esperanto Français Deutsch Ελληνικά עברית Magyar Indonesia
Italiano 日本 한국어 فارسی Polski Português Română Русский Српско-хрватски Slovenščina Español Svenska Türkçe Tiếng Việt

ECMA-404 The JSON Data Interchange Standard.

JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. It is based on a subset of the JavaScript Programming Language, Standard ECMA-262 3rd Edition - December 1999. JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language.

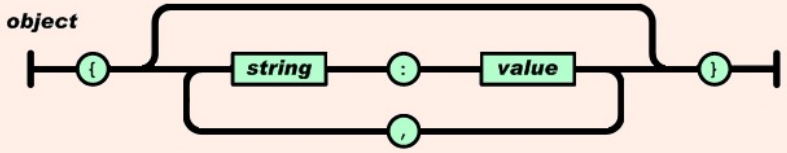
JSON is built on two structures:

- A collection of name/value pairs. In various languages, this is realized as an *object*, record, struct, dictionary, hash table, keyed list, or associative array.
- An ordered list of values. In most languages, this is realized as an *array*, vector, list, or sequence.

These are universal data structures. Virtually all modern programming languages support them in one form or another. It makes sense that a data format that is interchangeable with programming languages also be based on these structures.

In JSON, they take on these forms:

An *object* is an unordered set of name/value pairs. An object begins with { (left brace) and ends with } (right brace). Each name is followed by : (colon) and the name/value pairs are separated by , (comma).



object
{}
{ *members* }
members
pair
pair , *members*
pair
string : *value*
array
[]
[*elements*]
elements
value
value , *elements*
value
string
number
object
array
true
false
null

string
""

- พัฒนามาจากภาษา JavaScript
- Douglas Crockford เป็นผู้คิดค้น JSON

JavaScript Object Notation (JSON)

โครงสร้างหลักของ JSON ประกอบด้วย 2 โครงสร้างย่อย ได้แก่

- **Object** เป็นการเก็บข้อมูลภายใต้เครื่องหมายปีกกา {...} แบบไม่มีลำดับในรูปแบบ Key – Value โดยอาศัยเครื่องหมาย Colon (:) สำหรับการแบ่งแยก Key และ Value นอกจากนี้ ยังอาศัยเครื่องหมาย Comma (,) มาช่วยแบ่งกรณีมี Key – Value มากกว่า 1 คู่ ตัวอย่างเช่น

```
{ "firstName":"John", "lastName":"Doe" }
```

- **Array** เป็นการเก็บข้อมูลภายใต้เครื่องหมายก้ามปู [...] แบบมีลำดับในรูปแบบ Value โดยอาศัยเครื่องหมาย Comma (,) มาแบ่งแยกระหว่าง Value ตัวอย่างเช่น

1. obj เก็ฐทั้ง key&value ในขณะที่ array เก็บเหพาะ value)
2.

```
[ "html", "xml", "css" ]
```

JavaScript Object Notation (JSON)

สำหรับ Value ของโครงสร้าง Object หรือ Array ใน JSON สามารถมีชนิดข้อมูลได้หลากหลาย ได้แก่

- **ข้อความ (string)** อาศัยเครื่องหมาย “....” เป็นตัวกำหนดขอบเขตเปิด-ปิด
- **ตัวเลข (number)** เป็นเลขจำนวนเต็ม, ทศนิยม หรือ Exponential (E) notation ในระบบเลขฐาน 10
- **ค่าความจริง (boolean)** เป็นค่า true หรือ false
- **ค่าว่าง (null)**
- Object เป็นโครงสร้างที่รองรับการซ้อนกันได้
- Array เป็นโครงสร้างที่รองรับการซ้อนกันได้

```
Human = {
    "name": `Bank',          // String
    "age": 28,               // int
    "college" : true,        // boolean
    "offices" : [ '3350DMC', '3437NQ' ],    // List/Array
    "skills" : { "java": 10,                // Object
       "C": 10,
       "php": 5,
       "python" : '7'
    }
};
```

JavaScript Object Notation (JSON)

Example 1

```
{"skill": {"web":[{ "name": "html", "years": 5},{ "name": "css",  "years": 3}],"database":[{ "name": "sql", "years": 7}]}}
```

Example 2

```
{
   "skill": {
    "web":[
       { "name": "html", "years": 5},
       { "name": "css",  "years": 3}
    ],
    "database":[
       { "name": "sql", "years": 7}
    ]
   }
}
```

JavaScript Object Notation (JSON)

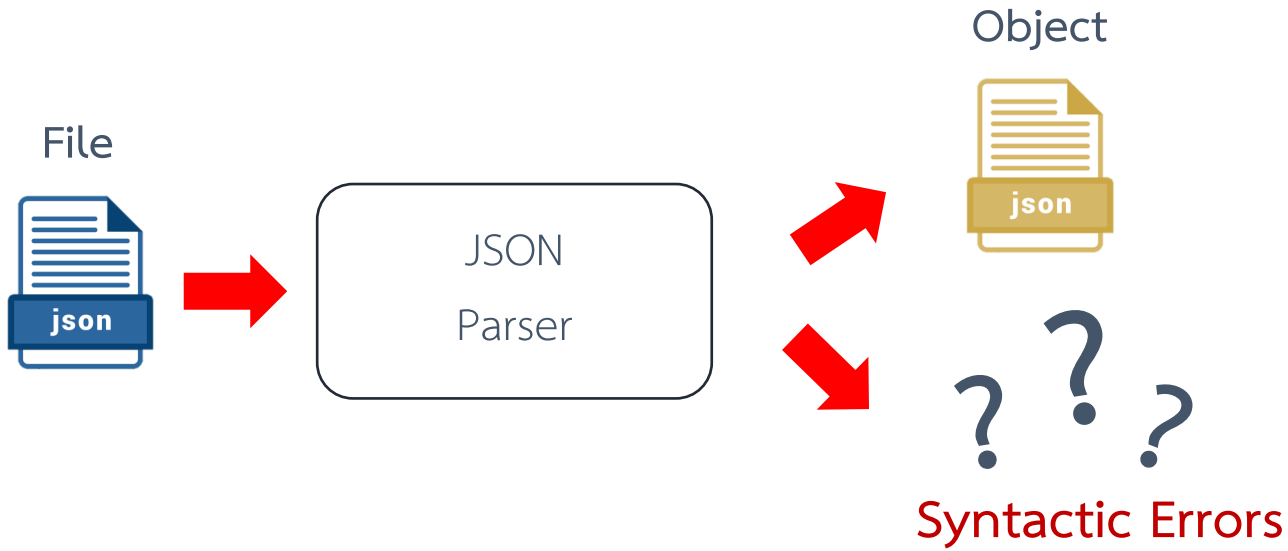
Example 3

```
{
    "detail": {
        "book":[
            { "name": "html",
              "price": 450
            },
            { "name": "css",
              "price": 350
            }]
        "e-book":[
            { "name": "sql",
              "price": 70
            }]
    }
}
```

Outline

- JSON Document
- JSON Schema

การตรวจสอบความถูกต้องไฟล์ JSON



JSON Parser คือ ตัวดำเนินการสำหรับการแปลงระหว่าง JSON String ไปเป็น JSON Object ซึ่งจะตรวจสอบได้แต่รูปแบบไวยากรณ์ (Syntactics) แต่ไม่สามารถตรวจสอบเชิงความหมาย (Semantics) ได้ หรือสามารถกล่าวได้ว่า JSON Parser ไม่สามารถตรวจสอบ ชนิดข้อมูล ขอบเขตของข้อมูล และ เงื่อนไข

JSON Schema

คือ ไฟล์ที่ใช้ระบุถึงชนิดข้อมูล (data type) เงื่อนไข (condition) ขอบเขตค่า (range) และคำอธิบาย (description) เพื่อใช้ในการตรวจสอบ (validation) และอธิบายรายละเอียด (documentation) นอกจากนี้ ยังช่วยในการตรวจความครบถ้วนและความถูกต้องสำหรับการส่งข้อมูลระหว่างโปรแกรม ซึ่งมีลักษณะคล้ายคลึงกับ XML Schema อย่างไรก็ตาม JSON Schema ยังไม่มีการกำหนดมาตราฐานของนามสกุลไฟล์ที่แน่นอน (แนะนำให้ใช้ .schema.json)

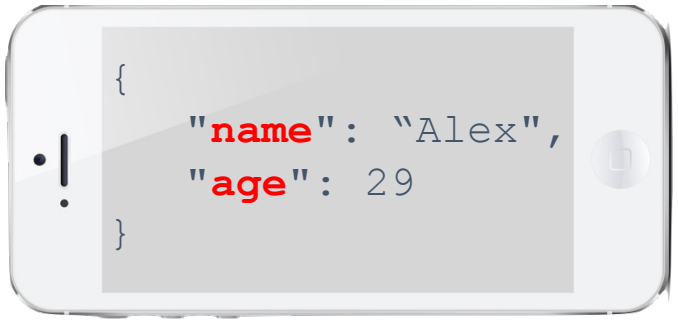
ตัวอย่างเช่น JSON Schema

JSON Schema

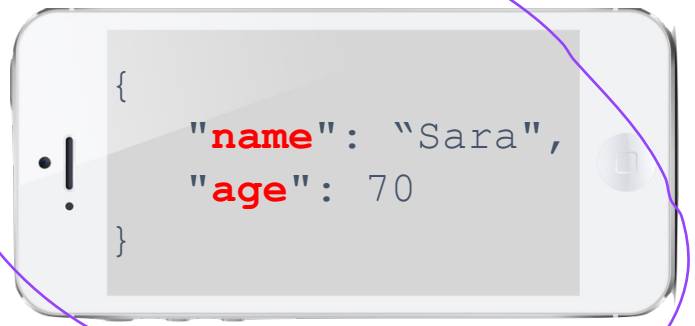
```
{   "$schema": "http://json-schema.org/draft-04/schema#",
    "title": "Person",
    "description": "A person",
    "type": "object",
    "properties":{
        "name":{
            "description": "A person's name",
            "type": "string"
        },
        "age":{
            "description": "A person's age",
            "type": "number",
            "minimum": 18,
            "maximum": 64
        }
    },
    "required": ["name", "age"]
}
```

ตัวอย่างเช่น JSON Schema

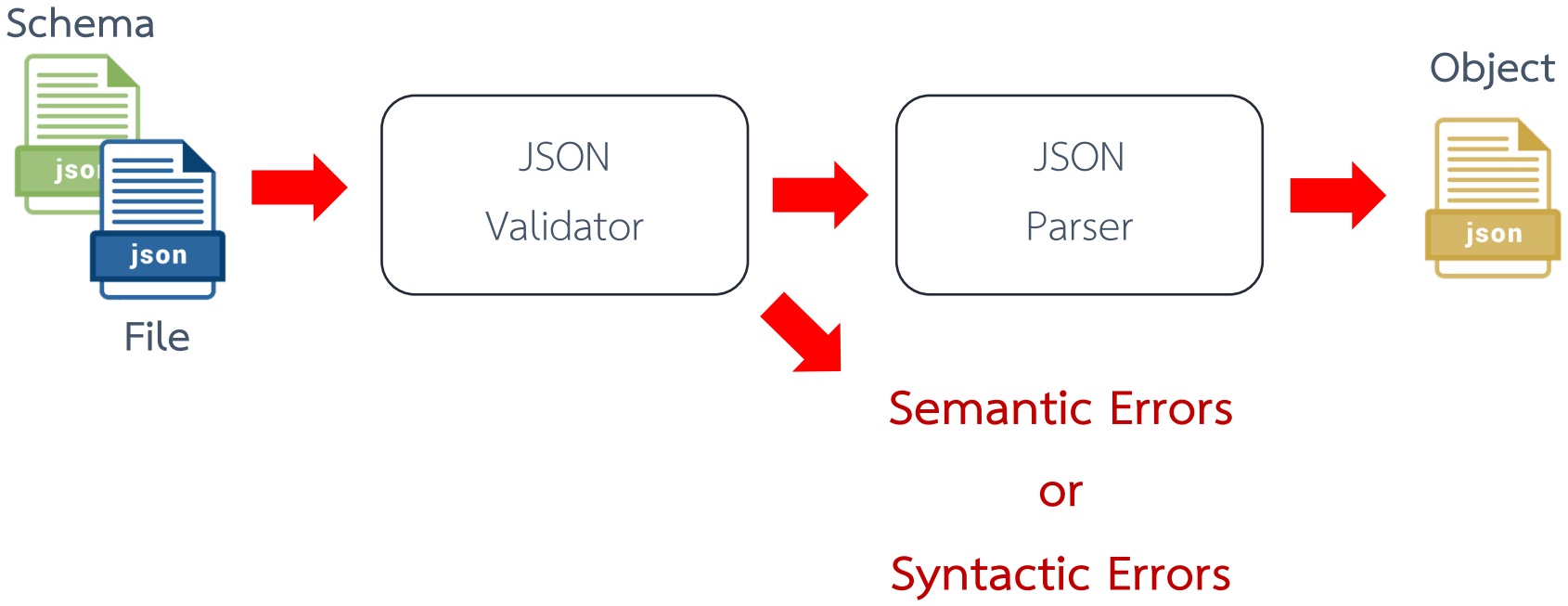
JSON File



JSON File

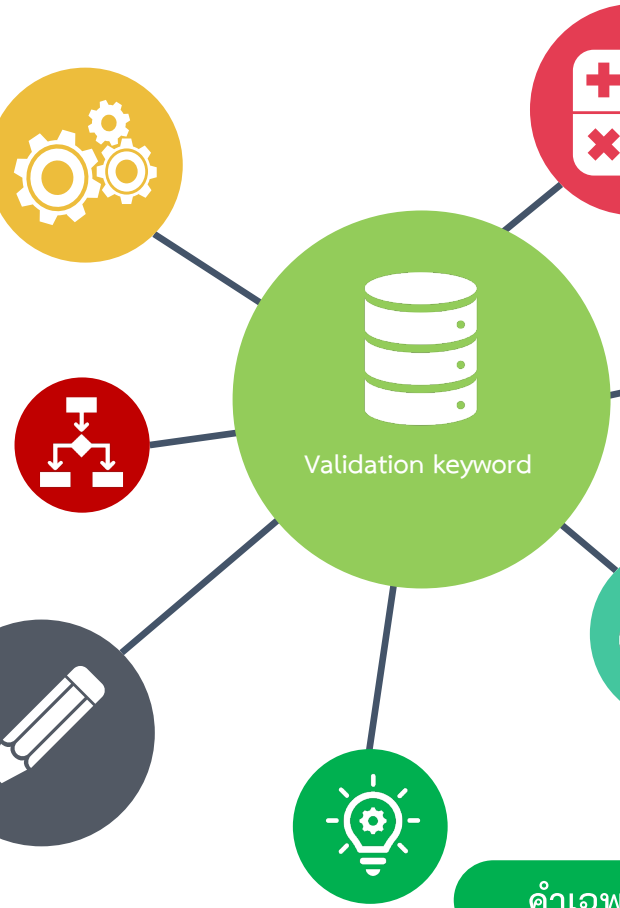


ตัวอย่างเช่น JSON Schema



JSON Validator คือ ส่วนตรวจสอบรูปแบบไวยากรณ์ (Syntactics) และยังสามารถตรวจสอบเชิงความหมาย (Semantics) ได้ อาทิเช่น ชนิดข้อมูล ขอบเขตของข้อมูล และ เงื่อนไข

ตัวอย่างเช่น JSON Schema



Any Instance Type

ได้แก่ type, enum และ const

คำเฉพาะสำหรับ Number

ได้แก่ multipleOf, maximum, exclusiveMaximum, minimum และ exclusiveMinimum

คำเฉพาะสำหรับ Object

ได้แก่ maxProperties, minProperties, required, properties, patternProperties, additionalProperties, dependencies และ propertyNames

คำเฉพาะสำหรับ Condition

ได้แก่ if, then และ else

คำเฉพาะสำหรับ Array

ได้แก่ items, additionalItems, maxItems, minItems, uniqueItems และ contains

คำเฉพาะสำหรับ String

ได้แก่ maxLength, minLength และ pattern

คำเฉพาะสำหรับ Boolean

ได้แก่ allOf, anyOf, oneOf และ not

Validation keywords ของ JSON Schema ใช้เพื่อกำหนดเงื่อนไขต่าง ๆ ชนิดข้อมูล และ ขอบเขตของช่วงข้อมูล เพื่อให้เกิดการสอดคล้องกับความต้องการในข้อมูลแต่ละกรณี (instance)