Entity Attribute Value Model (EAV)

PROFESSIONAL & CONTINUING EDUCATION
UNIVERSITY *of* WASHINGTON

W

What is EAV?

Data model for efficient storage of sparse data

– Only non-empty values stored

– One entity can have many attributes

– Each attribute is stored as a key-value pair

– Very common data structure for IoT and biological data, such as temperature readouts

EAV Format

Three column format

> Entity: item or event being described
>> – Customer ID, patient number, time stamp

> Attribute: The type of value this row in the table is describing
>> – Customer Address, patient diagnosis, degrees °F

> Value: The attribute's value, stored as a string

Traditional Database Table vs EAV

Traditional Database Table

Patient ID	Visit Date	Weight	Temp
1	6/20/92	125	92
1	6/15/17	140	NA
2	8/31/04	188	90
3	1/21/13	110	102

EAV Table

Entity	Attribute	Value
1:6/20/92	Weight	125
1:6/20/92	Temp	92
1:6/15/17	Weight	140
2:8/31/04	Weight	188
2:8/31/04	Temp	90
3:1/21/13	Weight	110
3:1/21/13	Temp	102

When to use EAV

Sparse, heterogeneous data
- – When storing data in one row would leave many blank attributes
- – When there are many attributes for an entity
- – When attribute values are constantly changing

PROFESSIONAL & CONTINUING EDUCATION
UNIVERSITY *of* WASHINGTON

Translating a Table to EAV

Practice Exercise

Translate a table to EAV

Translate the following table to EAV format:

employee_id	first_name	last_name	department
1	Ren	Stimpy	IT
2	Rick	Morty	IT
3	Thelma	Louise	
4	Harold	Kumar	Marketing

Solution EAV Table

entity	attribute	value
1	first_name	Ren
1	last_name	Stimpy
1	department	IT
2	first_name	Rick
2	last_name	Morty
2	department	IT
3	first_name	Thelma
3	last_name	Louise
4	first_name	Harold
4	last_name	Kumar
4	department	Marketing

Summary

> EAV data format for storage of sparse, constantly changing information
> Three column format: entity, attribute, value
> Common format in IoT, data science and biological sciences

W

EAV Model

Introduction to JSON Format

PROFESSIONAL & CONTINUING EDUCATION
UNIVERSITY *of* WASHINGTON

W

So, what is JSON?

JavaScript Object Notation (JSON): text based structure for storing data.

– Human-readable formatting
– Language independent
– Widely used in many disciplines, such as web development, IoT and the sciences

PROFESSIONAL & CONTINUING EDUCATION
UNIVERSITY *of* WASHINGTON

When is JSON used to store information?

- –Transmitting data between web servers and apps
- –Scrape information from the internet via an API
- –Easily store nested, complex data
- –EAV data structures with key/value pairs

JSON Data Structures

JSON is built on two main data structures:
- –Objects: similar to a python dictionary
- –Array: similar to a python list

JSON Data Types

Each object is composed of values that include:

- Integers: doubles and floats
- Strings: surrounded by double quotes
- Boolean: true or false
- Null: empty

JSON Objects

- Maps to a python dictionary
- Typically collection of values related to one item
- Unordered set of key/value pairs
- Values separated by a comma
- Begins and ends with {curly brackets}

Example JSON Object

```
{ "customer":
    [
        {"name": "Steven", "city": "Seattle"},
        {"name": "David", "city": "London"}
    ]
}
```

JSON Arrays

–Maps to a python list

–Ordered collection of values

–Values separated by commas

–Begins and ends with [square brackets]

–Values accessible by index number

Example JSON Array

```
{
    "name": "Steven",
    "age": 27,
    "siblings": ["Anna", "Peter", "Lowell"]
}
```

PROFESSIONAL & CONTINUING EDUCATION
UNIVERSITY *of* WASHINGTON

Accessing JSON elements

- Objects: access by key
- Arrays: access by index
- Nested elements: key, key, index, etc.

PROFESSIONAL & CONTINUING EDUCATION
UNIVERSITY *of* WASHINGTON

Access JSON Objects – Demo

```python
import json

json_obj = {
        "name": "Steven",
        "age": 27,
        "siblings": ["Anna", "Peter", "Lowell"],
        "cars": {
                "Toyota":["Tercel", "Forerunner"],
                "Nissan":["Versa", "Sentra"]
                }
        }

#access object by key
print(json_obj["name"])

#access list elements by index
print(json_obj["siblings"][1])

#access nested elements by key and index
print(json_obj["cars"]["Toyota"][0])
```

PROFESSIONAL & CONTINUING EDUCATION
UNIVERSITY *of* WASHINGTON

JSON Convenience Functions

Working with data objects:

– dumps()

> python object to a JSON string

– loads()

> JSON string/object to python object

Working with file objects:

– dump()

> python object to a JSON file object/stream

– load()

> JSON file object/stream to a python object

PROFESSIONAL & CONTINUING EDUCATION
UNIVERSITY *of* WASHINGTON

JSON Demos

JSON Object to a Python Dictionary

JSON Array to a Python List

Python Dictionary to JSON

Python List to JSON

Output JSON:

– Pretty Print JSON

– Writing JSON to a file

PROFESSIONAL & CONTINUING EDUCATION
UNIVERSITY *of* WASHINGTON

JSON Object to Python Dictionary

```
import json

# create JSON object
json_data = '{"name":"Steven", "city":"Seattle"}'

# convert JSON object to python dictionary with
json.loads()
python_obj = json.loads(json_data)

# print dictionary values by keys
print(python_obj["name"])

print(python_obj["city"])
```

PROFESSIONAL & CONTINUING EDUCATION
UNIVERSITY *of* WASHINGTON

JSON Array to Python List

```
import json

# create JSON array
json_array = '{"drinks": ["coffee", "tea", "water"]}'

# convert JSON array to python list with json.loads()
data = json.loads(json_array )

# loop through list items
for element in data['drinks']:
    print(element)
```

Python Dictionary to JSON

```
import json

# create a python dictionary
d= {}
d["Name"] = "Steve"
d["Country"] = "Merica"

# convert python dictionary to JSON using json.dumps()
json_obj = json.dumps(d)

# view JSON objectp
print(json_obj)
```

Python List to JSON

```
import json

# create a python dictionary
py_list = ["abc", 123]

# convert python list to JSON array using json.dumps()
json_array = json.dumps(py_list)

# view list structure and data type
print(py_list)
print(type(py_list))

print(json_array)
print(type(json_array)) # note that JSON arrays are strings
```

UNIVERSITY *of* WASHINGTON

Pretty Print JSON Objects

Printing is controlled by arguments to json.dumps():

- sort_keys – dictionary output is sorted by key
  - sort_keys=True
- indent – number of spaces to indent levels
  - indent=4

PROFESSIONAL & CONTINUING EDUCATION
UNIVERSITY *of* WASHINGTON

Pretty Print JSON Example

```python
import json

# create JSON object
json_data = '{"name":"Steven", "city":"Seattle"}'

# convert to python
python_obj = json.loads(json_data)

# print python dictionary before prettiness
print(json.dumps(python_obj))

# print using sorted keys and indentation
print(json.dumps(python_obj, sort_keys=True, indent=4))
```

PROFESSIONAL & CONTINUING EDUCATION
UNIVERSITY *of* WASHINGTON

Writing JSON to a file

To output python data in JSON format:
- Store your data in a python dictionary or list
- Convert the object to a JSON file format using dump()
- Write the object to a file using python

PROFESSIONAL & CONTINUING EDUCATION
UNIVERSITY *of* WASHINGTON

Example Writing JSON to a File

```python
import json

# create python dictionary
data = {}

# create list object to append entries to dictdata['people'] = []

# append entries
data['people'].append({'name': 'Steven',
    'website': 'uw.edu',
    'city': 'Seattle'})
data['people'].append({'name': 'Annie',
    'website': 'ford.com',
    'city': 'Detroit'})

# use json.dump() to convert and write to file
with open("test_file.txt", 'w') as outfile:
    json.dump(data, outfile)
```

Summary

> JSON uses and syntax
> Converting between python and JSON
> Exploring JSON objects
> Saving JSON to a file

W

Introduction to JSON Format

End of Presentation