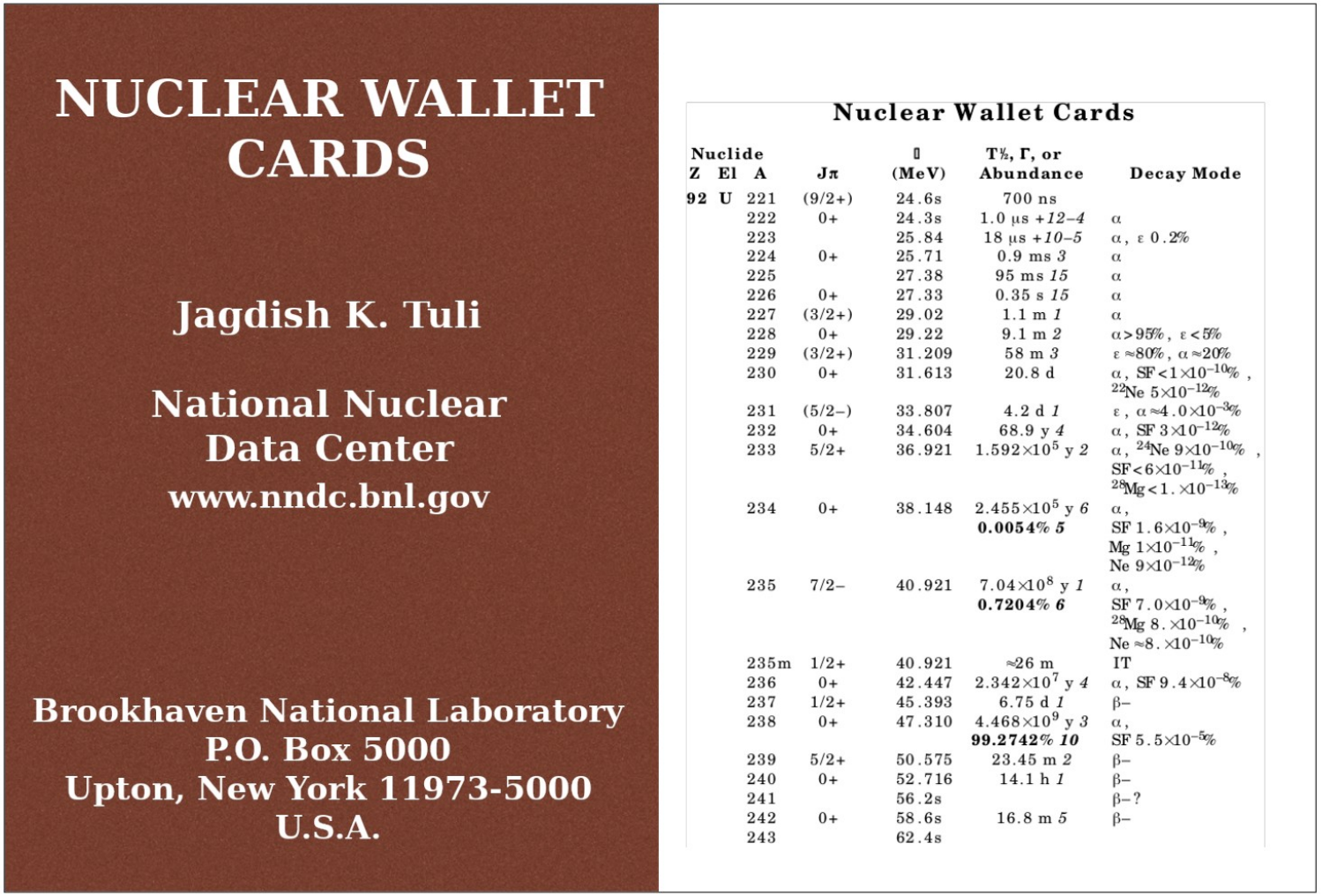
“WalletCraft” - Object-oriented databasing for nuclear data

Adam Hayes, Benjamin Shu, Libby McCutchan,
Shaofei Zhu, Alejandro Sonzogni

National Nuclear Data Center
Brookhaven National Laboratory

WPEC EG-GNDS
12 May 2020

Nuclear Wallet Cards (print version)



NUCLEAR WALLET CARDS

Jagdish K. Tuli

National Nuclear Data Center
www.nndc.bnl.gov

Brookhaven National Laboratory
P.O. Box 5000
Upton, New York 11973-5000
U.S.A.

Nuclear Wallet Cards

Nuclide Z	El	A	Jπ	Δ (MeV)	T½, Γ, or Abundance	Decay Mode
92	U	221	(9/2+)	24.6s	700 ns	
		222	0+	24.3s	1.0 μs *+12–4*	α
		223		25.84	18 μs *+10–5*	α, ε 0.2%
		224	0+	25.71	0.9 ms *3*	α
		225		27.38	95 ms *15*	α
		226	0+	27.33	0.35 s *15*	α
		227	(3/2+)	29.02	1.1 m *1*	α
		228	0+	29.22	9.1 m *2*	α>95%, ε<5%
		229	(3/2+)	31.209	58 m *3*	ε≈80%, α≈20%
		230	0+	31.613	20.8 d	α, SF<1×10^{-10}%, ^{22}Ne 5×10^{-12}%
		231	(5/2–)	33.807	4.2 d *1*	ε, α≈4.0×10^{-3}%
		232	0+	34.604	68.9 y *4*	α, SF 3×10^{-12}%
		233	5/2+	36.921	1.592×10^{5} y *2*	α, ^{24}Ne 9×10^{-10}%, SF<6×10^{-11}%, ^{28}Mg<1.×10^{-13}%
		234	0+	38.148	2.455×10^{5} y *6* **0.0054% *5***	α, SF 1.6×10^{-9}%, Mg 1×10^{-11}%, Ne 9×10^{-12}%
		235	7/2–	40.921	7.04×10^{8} y *1* **0.7204% *6***	α, SF 7.0×10^{-9}%, ^{28}Mg 8.×10^{-10}%, Ne ≈8.×10^{-10}%
		235m	1/2+	40.921	≈26 m	IT
		236	0+	42.447	2.342×10^{7} y *4*	α, SF 9.4×10^{-8}%
		237	1/2+	45.393	6.75 d *1*	β–
		238	0+	47.310	4.468×10^{9} y *3* **99.2742% *10***	α, SF 5.5×10^{-5}%
		239	5/2+	50.575	23.45 m *2*	β–
		240	0+	52.716	14.1 h *1*	β–
		241		56.2s		β–?
		242	0+	58.6s	16.8 m *5*	β–
		243		62.4s		

Nuclear Wallet Cards (online)

www.nndc.bnl.gov/wallet

Wallet Card Lookup by Element

0 n	1 H																2 He
	3 Li	4 Be											5 B	6 C	7 N	8 O	9 F	10 Ne
	11 Na	12 Mg											13 Al	14 Si	15 P	16 S	17 Cl	18 Ar
	19 K	20 Ca	21 Sc	22 Ti	23 V	24 Cr	25 Mn	26 Fe	27 Co	28 Ni	29 Cu	30 Zn	31 Ga	32 Ge	33 As	34 Se	35 Br	36 Kr
	37 Rb	38 Sr	39 Y	40 Zr	41 Nb	42 Mo	43 Tc	44 Ru	45 Rh	46 Pd	47 Ag	48 Cd	49 In	50 Sn	51 Sb	52 Te	53 I	54 Xe

WalletCraft: Motivation

- Need system to speed & streamline updates of Nuclear Wallet Cards
- Synchronize ENSDF evaluation & Wallet Cards updates
- Uniform calculation of mean half-lives, etc.
- Store
  - Source data: measured values from publications
  - Evaluated quantities
  - Comments, notes, etc. separated from numerical data; all numerical data in numerical fields
- Version tracking
  - Freeze “evaluation” versions
  - Tag “publication” versions
- Shorten time to publish a version
- Test for future new ENSDF database

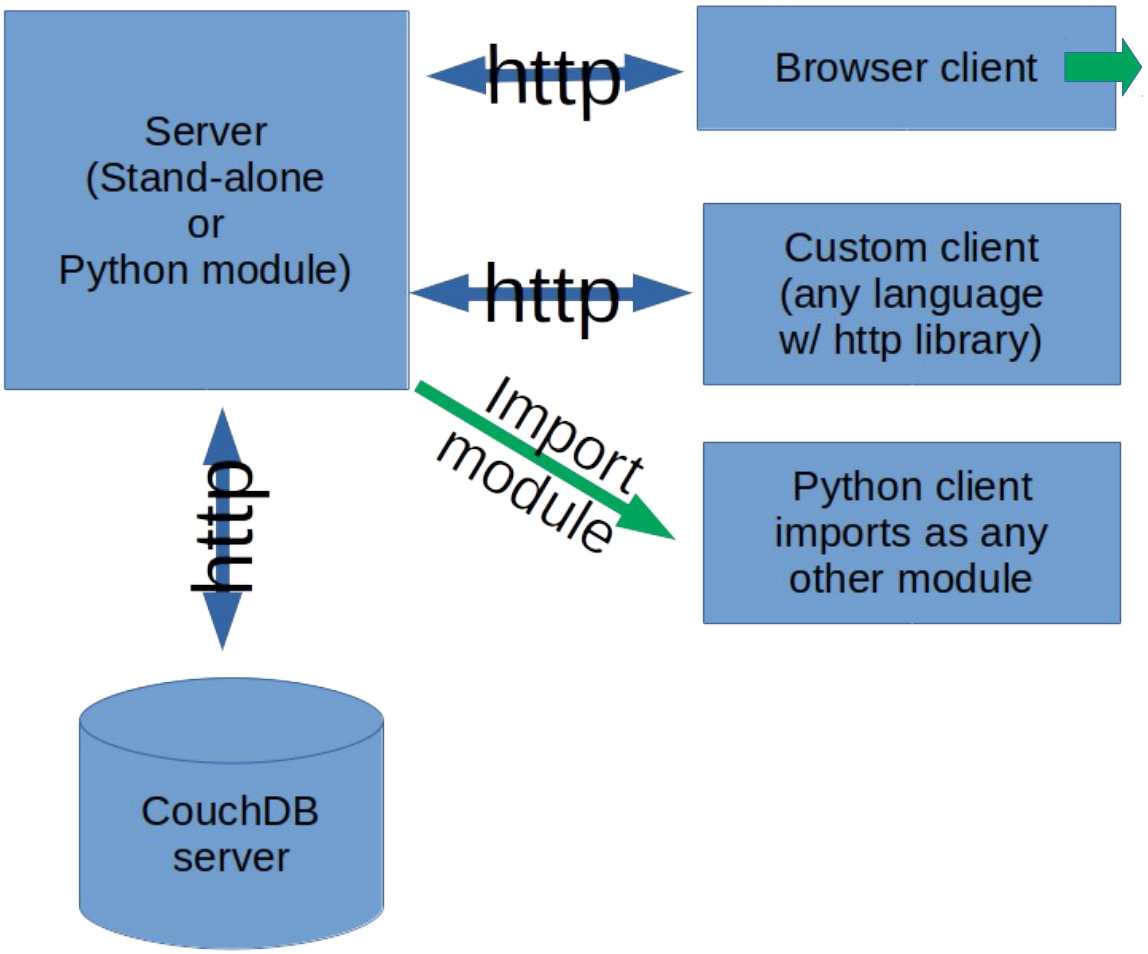
Why object-oriented database?

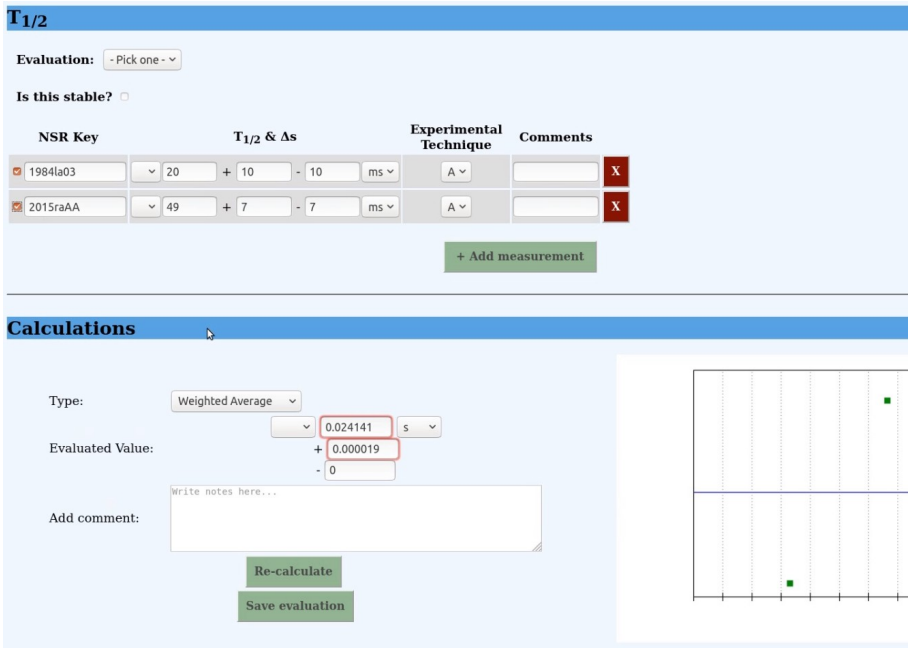
- Variety of data types
  - Float, int, etc.
  - List, dict, etc.
  - Long-form text
  - Documents
  - Images (e.g. plots)
  - Binary data
- Expansible without disruptive changes to format
- Simplicity paradigm: “Store together what you access together”

WalletCraft:
Server-side design overview

- CouchDB license is very permissive (Apache)
- Server side in Python3
- Simple http request system allows any number of different UI's
  - Returns and accepts data payloads as JSON (not html)
  - Any client or browser makes same http requests
  - *Any language* (Even Bash!)
- Handles multiple users safely
  (Don't lose your work because another user makes changes)
- Does not raise standard http errors—errors are returned in data payload
  (Simpler for developers to understand)

Flexible design





Planning ahead for expansion

- "Everything is a dict"
- Dict = map ~ JSON, but can store *objects in database*
- Expand "schema" without need to change client, e.g.
  - **OK**: (Hard to maintain, disruptive changes)
    `{"key":value}`
    Need to change data type to expand:
    `{"key":[value1,value2]}`
  - **BETTER**: (Expandible, but confusing)
    `{"key":[value1,value2,…]}`
  - **BEST**: (Expand without disrupting users & developers)
    `{"key":{"value1Name":{"attribute1":value1}}}`

    Easily add an attribute:
    `{"key":{"value1Name":`
    `{"attribute1":value1,"attribute2":value2…}}}`

Database “schema”
Full nuclide entry


```
{
    "_id": "62,163",
    "_rev": "1-4b5a0b7904644dab711ce77c6cbb20ea",
    "Z": 62,
    "A": 163,
    "symbol": "163Sm",
    "levels": {↔},
    "rawLines": ["163  SM   62   101  201704    0.0
    "documentType": "nuclide",
    "debug": "Inserted with add-ensdf-to-wc.py"
}
```

Database “schema”
Levels section


```
{
    "_id": "62,163",
    "_rev": "1-4b5a0b7904644dab711ce77c6cbb20ea",
    "Z": 62,
    "A": 163,
    "symbol": "163Sm",
    "levels": {
        "GS": {↔}
    },
    "rawLines": ["163  SM   62   101  201704    0.0
    "documentType": "nuclide",
    "debug": "Inserted with add-ensdf-to-wc.py"
}
```

Database “schema”
Ground-state level

```
"levels": {
    "GS": {
        "tHalf": {↔},
        "decayWidth": {↔},
        "massExcess": {↔},
        "abundance": {↔},
        "decayModes": {↔},
        "Jpi": {↔},
        "energy": {↔}
    }
},
```

Database “schema”
Half-life of ground state

```
4    "Z": 62,
5    "A": 163,
6    "symbol": "163Sm",
7    "levels": {
8        "GS": {
9            "tHalf": {
10               "published": {},
11               "evaluations": {↔},
27               "measurements": {}
28           },
```

Database “schema” Evaluations

```
    "Z": 62,
    "A": 163,
    "symbol": "163Sm",
    "levels": {
        "GS": {
            "tHalf": {
                "published": {},
                "evaluations": {
                    "2017, 04, 01, 00, 00, 00, 000000": {↔}
                },
                "measurements": {}
            },
```

Database “schema”
Individual evaluation (example only)


```
"evaluations": {
    "2017, 04, 01, 00, 00, 00, 000000": {
        "tHalfIsLowerLimit": false,
        "tHalfIsUpperLimit": false,
        "tHalfIsUpperEqual": false,
        "tHalfIsLowerEqual": false,
        "tHalfIsApproximate": false,
        "tHalfIsUnknown": false,
        "tHalf": 1.23,
        "tHalfUnit": "s",
        "tHalfInSeconds": 1.23,
        "dtHalf": [0.51, 0.47],
        "stable": false,
        "measurementsCited": {}
    }
```

All functions by simple http requests

- Retrieve a nuclide document with an http GET request

  https://someserver.org/?nucleus=137Cs

- Delete a document with an http DELETE request

  https://someserver.org/?nucleus=137Cs

- Request edit permission (“edit lock”) with an http GET request

  https://someserver.org/?nucleus=137Cs&edit=true

- Add a nuclide with an http POST request

```
{
  “function” : ”addnew”,
  “nuclide”  : { nuclide document },
}
```

Responses

- Foreseeable errors do **not** use http error codes (404, 500, etc.)
- Response to **valid** request

```
{"status": {"error": false, "message": ""},
 "data"  : {"_id": "137,55", …
```

- Response to **invalid** request

```
{"status"       : {
                    "error"  : true,
                    "message": "InvalidQuery"
                   },
 "data"         : {},
 "supplemental": "The keys or key combination of
                  keys in [NSRKey,A] is invalid"
}
```

Handling multiple users with “edit locks”

- Use case: few simultaneous editors, editors likely know each other
- Think: buying event tickets online
- Request “edit lock” on a nuclide
- Browser or other client “pings” to keep lock
- Setting a large ping timeout allows evaluator to safely work on a nuclide for a long time (e.g. days if necessary).
- Sessions are extremely robust, even after server reboot!

Summary

- This project is focusing on databases / storage, as opposed to a particular file format
- Produce approved output file format on demand
- Object-oriented database (CouchDB)
  - All data types stored together (even images, binary data…)
  - “Everything is a dict”--easily expanded
  - “Store together what you want to access together”
- History, version tracking
- Robust handling of multiple users
- **Testbed for future new ENSDF database**

END

END

END