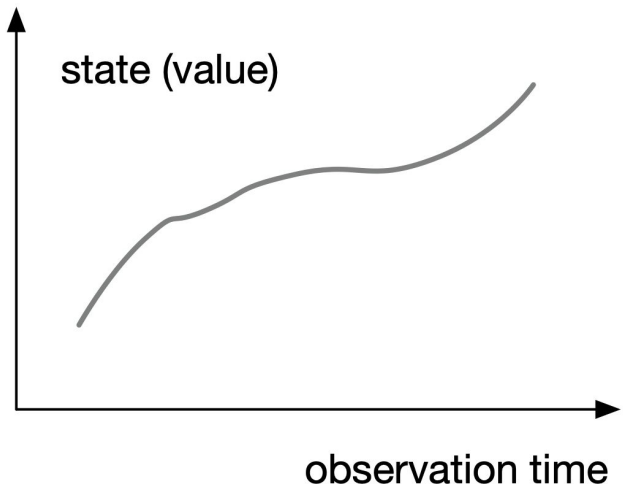
Unstructured Conditions Database (UConDB)

Igor Mandrichenko

DUNE Databases Meeting

April 13, 2022

What is Conditions Database ?

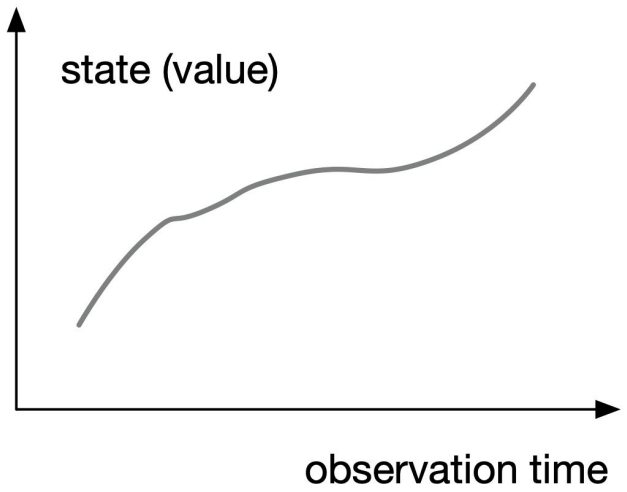


Generally, a conditions database is a record of an object state changes as a function of the observation time (a.k.a validity time)

Object can be:

- Scalar
- Tuple
- Array of tuples indexed by “channel”
- BLOB

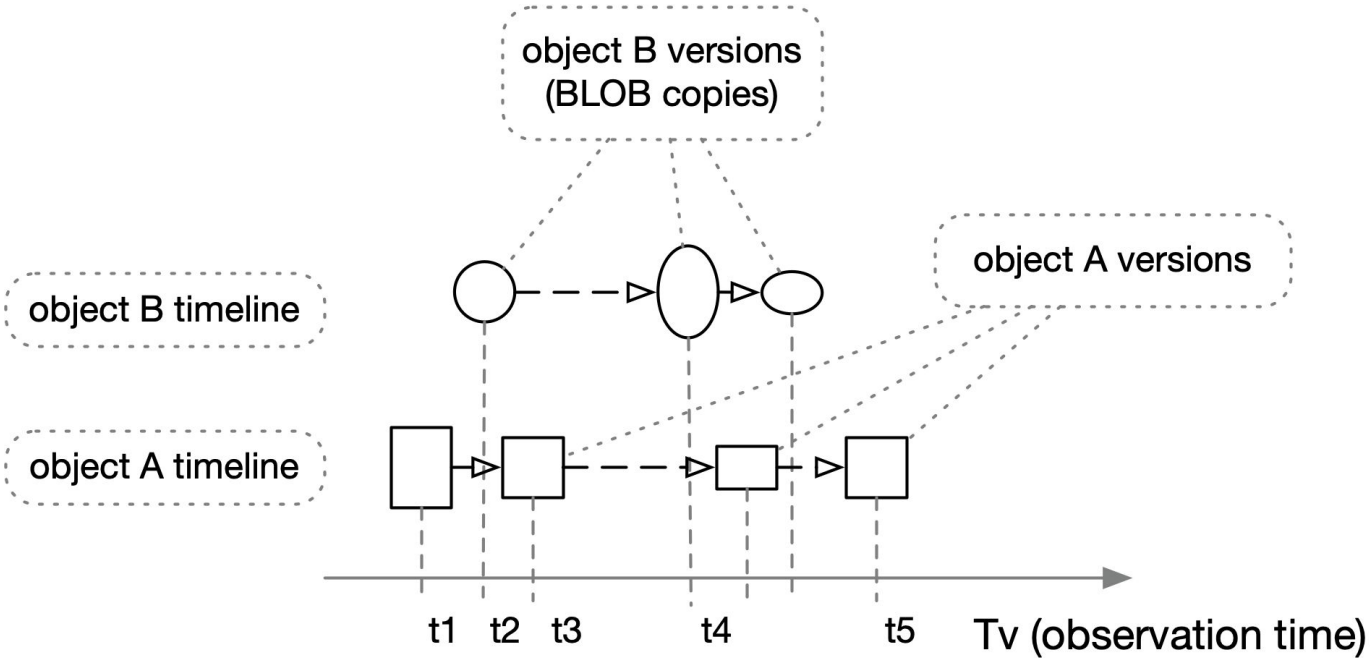
What is UConDB ?



Unstructured Conditions Database (UConDB)

- UConDB object is a BLOB
  - document (PDF, JSON, XML, CSV, text, FHICL, HDF5, …)
  - image
  - anything
- The database is unaware of BLOB's internal structure

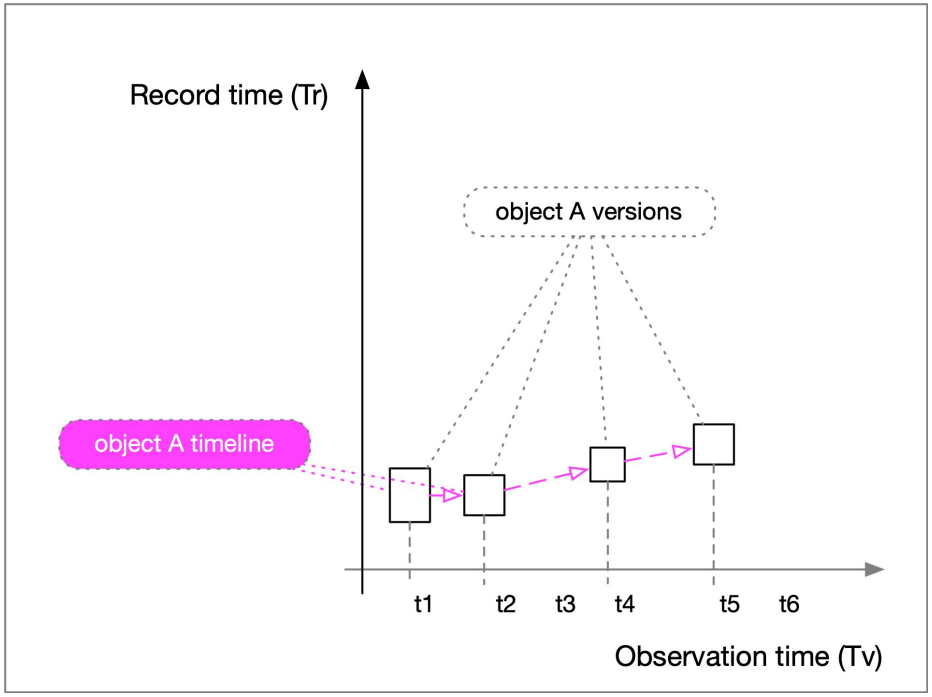
Timeline



A sequence of timestamped (observation time) BLOB *versions* (observations) of the *object* is a *Timeline*

UConDB records (independent) timelines for multiple objects, identified by name

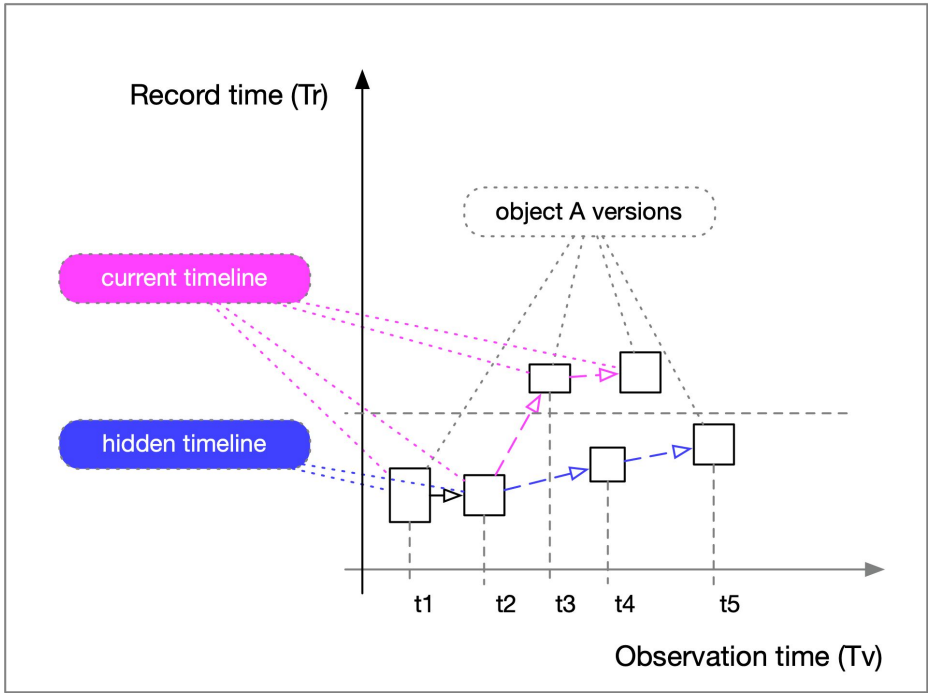
Timeline



Recording time is the time at which the version was actually recorded in the database

$Tr \neq Tv$

Timeline

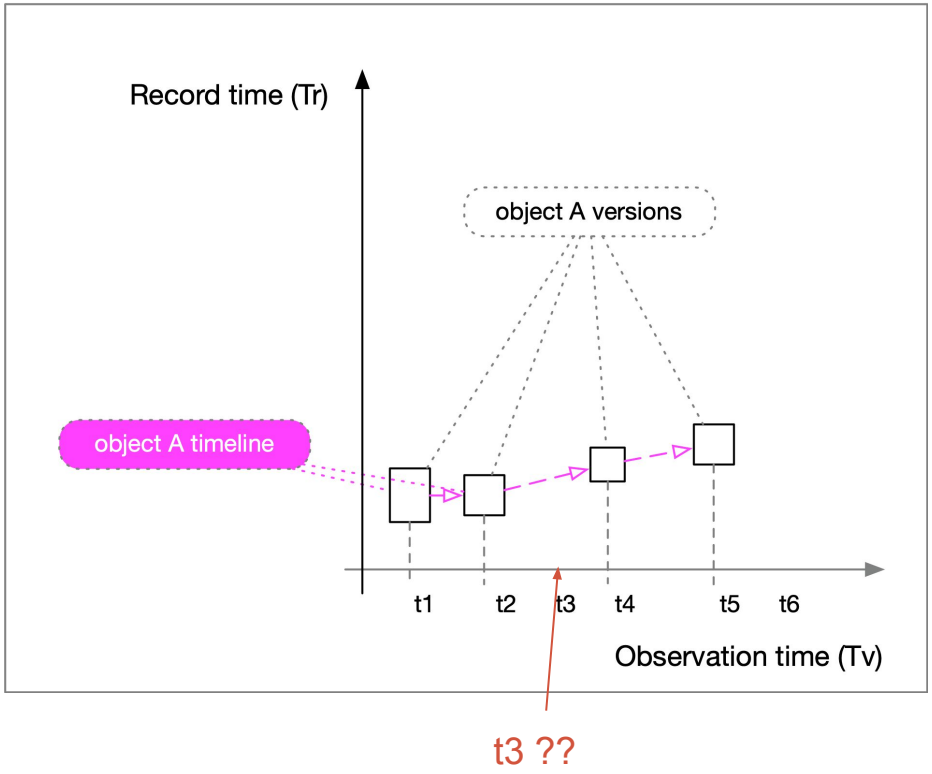


If a versions with Tv in the past w.r.t. the latest recorded version is added, that creates a *new* timeline and "hides" the current one

→ Timeline must be recorded monotonically

Old timeline is not deleted, remains in the database as *hidden* and can be retrieved

Interpolation within a timeline



Assuming some sort of state continuity along the timeline

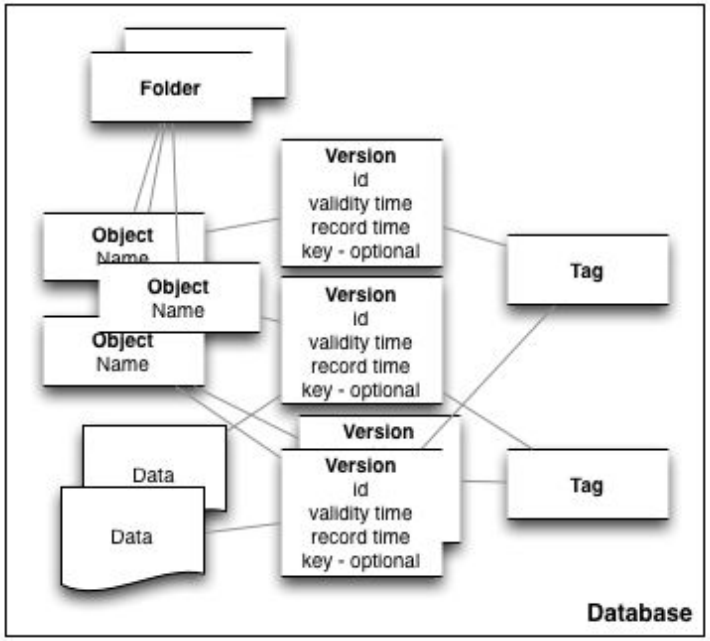
Knowing the previous and the next state of the object, intermediate state can be interpolated somehow

- Application specific
- Simplest: assume constant until next change

Interpolation is not always meaningful. Example: runs configuration

Run configuration for run #2 can not be derived from #1 and #3

UConDB Data Model



Database

- Folders (name)
  - Objects (name)
    - Versions (Tv, Tr, …)

Object name may contain slashes:

/folder/path/to/object

It's unstructured:

- Objects in the folder *do not* have to be of the same structure
- Versions of same object *do not* have to be of the same structure

Version properties

- Tv - observation time
  - Numeric, default=0
  - Assigned by the user, does not actually have to be time
- Tr - version record time
  - Auto-assigned
- Id
  - Integer, unique across the folder
  - Auto-assigned
- Key
  - Text, unique for the object
  - Assigned by the user, optional, can be moved to another version
- Tags
  - Text
  - Assigned by the user, optional
- Value - BLOB
  - Checksum (Adler32)
  - Size (bytes)

Creating a Version

When creating a version, the client specifies:

- Folder name
- Object name
- Value (BLOB)
- Optionally:
  - Key, can be moved from an existing version
  - One or more tags
  - Tv

The database returns:

- Version ID

Retrieving a Version

The client specifies:

- Folder name
- Object name
- Optionally:
  - Observation time (Tv) as a number, default = current timestamp
    - current object timeline
  - Tag - only versions with the specified tag will be returned
    - retrieve hidden timelines
  - Record time (Tr) - do not return versions created since Tr
    - retrieve hidden timelines
  - Key - return version by the key (Tv, Tr, tag ignored)
  - Version ID - return version by ID (Tv, Tr, tag ignored)

Interpolation

Interpolation: constant between updates

- find the BLOB value with Tv
  - <= the specified Tv
  - closest to the specified Tv

Interpolation enabled if version selected with:

- Tv (default = now)
- Tag + Tv (hidden timelines)
- Tr + Tv (hidden timelines)

Interpolation (and timelining) disabled:

- Key
- Version ID

REST Interface: writing

Uploading:

```
curl -T /data/file.txt \
     -X POST \
     http://.../data/folder/object?tv=123.4
```

Authentication:

- RFC2617, digest authentication
- shared password used to calculate digital signature but not sent over the wire

```
curl -T /data/file.txt \
     -X POST \
     --digest -u user:password \
     http://.../data/folder/object?tv=123.4
```

REST Interface: reading

by Tv:

```
curl -o /data/file.txt http://.../data/folder/object?tv=1234.5
```

metadata:

```
curl http://.../data/folder/object?tv=1234.5&meta_only=yes
```

returns metadata only, including version id, as JSON

by id:

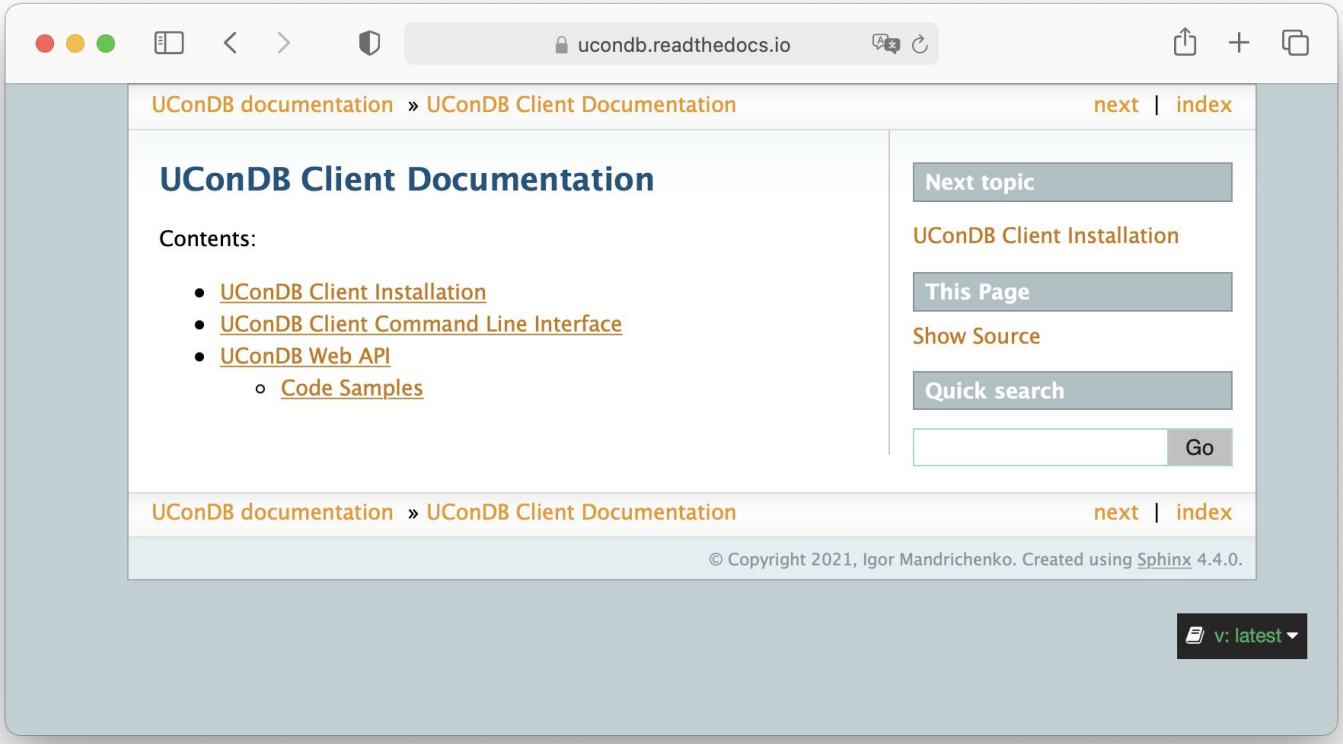
```
curl -o /data/file.txt http://.../data/folder/object?version_id=1234
```

with tag:

```
curl -o /data/file.txt http://.../data/folder/object?tv=1234.5&tag=v3_4_5
```

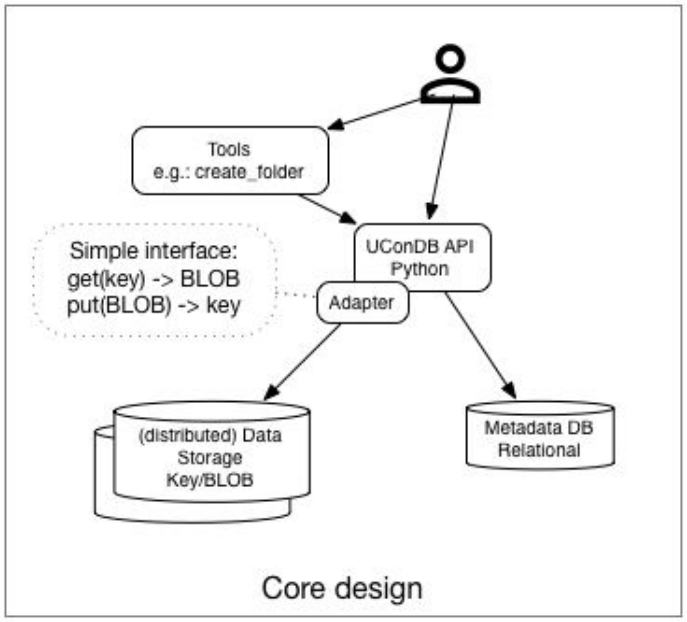
Documentation

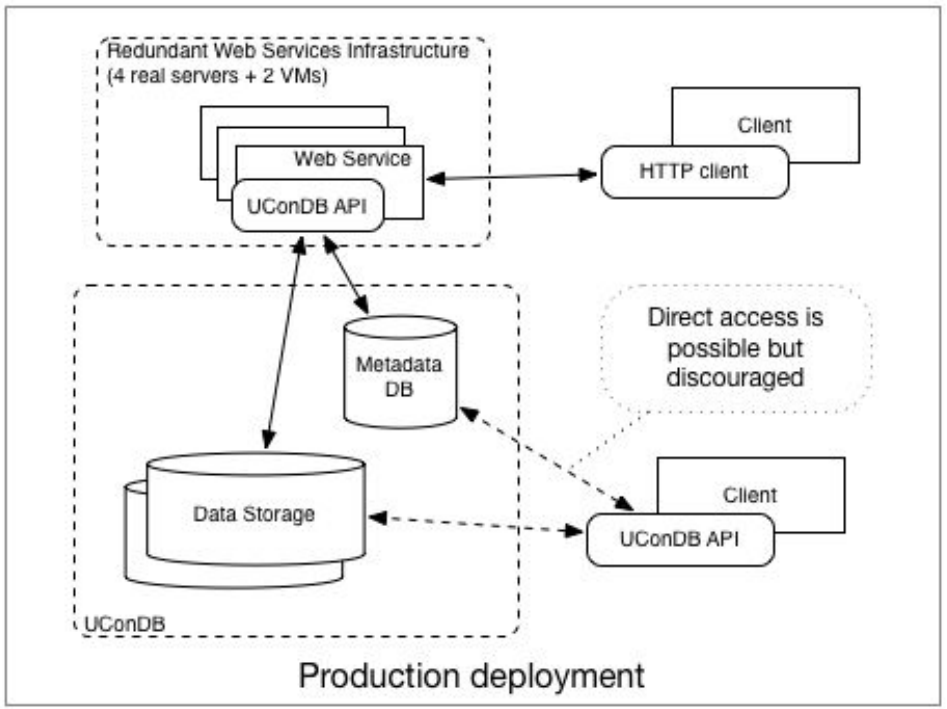
https://ucondb.readthedocs.io



BACKUP

UConDB Design and Deployment





API, tools - Python

Metadata - Postgres

Data storage - pluggable implementations

- key/BLOB interface adaptor
- Postgres
  - smaller databases
- CouchBase - distributed NoSQL product
  - larger databases, horizontal scalability, memory cache, data replication, HA

ProtoDUNE Run Configurations DB

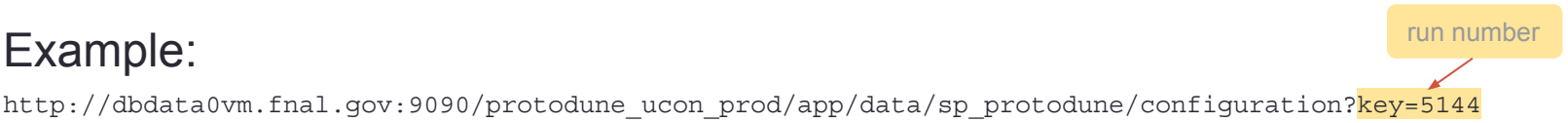
UConDB with Postgres data backend

~5000 run configurations

- FHICL file
- Average size ~3MB

~16 GB total data size

Example:



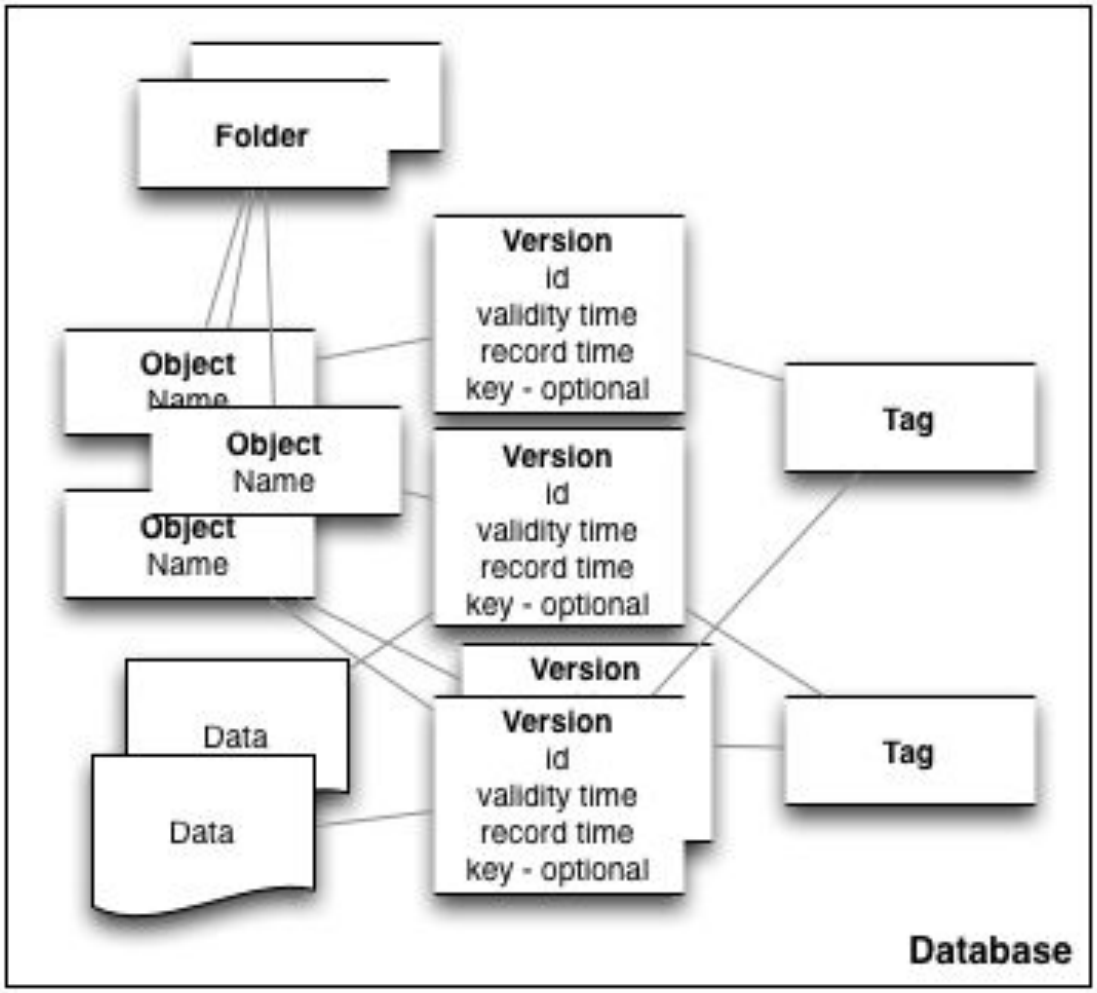
BACKUP

BACKUP

UConDB Concepts

- Database is organized as a set of named folders
- Objects (BLOBs) live inside folders
- Object has a name unique for the folder
  - Object identification: <folder name, object name>
- Object has versions (i.e. measurements, observations)
- Version properties:
  - Tv - numeric validity time (observation time)
  - Tr - version creation time (record time)
  - Value - BLOB
  - Id - an integer, automatically assigned by the DB when the version is created and returned to the user
  - Key - text - unique user defined version identifier (optional)
  - Tag(s) - text - optional - can be shared by many objects/versions

UConDB Data Model



Recording Data

To record an object version, specify:

- Folder name (required)
- Object name (required)
- Tag (optional)
- Key (optional)
- Value - BLOB

Database returns

- Generated numeric version ID (in case you care to remember it)

Reading Data

Object version can be retrieved by:

- Folder name (required)
- Object name (required)
- Tv (optional, default: current time)
- Tr - record time specification (optional)
- Tag (optional)
- Key (optional)
- ID (optional)

Metadata

- List of objects in the folder
- Versions for object
- Version metadata
- Tags for folder

Who am I ?

- Igor Mandrichenko
- 20+ years with FNAL CD
- Used to lead the Scientific DB Applications Group
  - Stephen White leads it now
- Lead development of:
  - Hardware DB, IFBeamDB
  - Stephen's group owns them now
- Developed Minerva ConDB, ConDB, UConDB
  - Still "own" and run them
  - Operations help from Stephen's group