LatteDB *(in development)*

— Database oriented workflow management (for LQCD)

In collaboration with Jason Chang and

Christopher Körber | UC Berkeley | Feodor Lynen Fellow

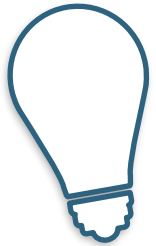
Unterstützt von / Supported by

Alexander von Humboldt
Stiftung/Foundation

Berkeley
UNIVERSITY OF CALIFORNIA

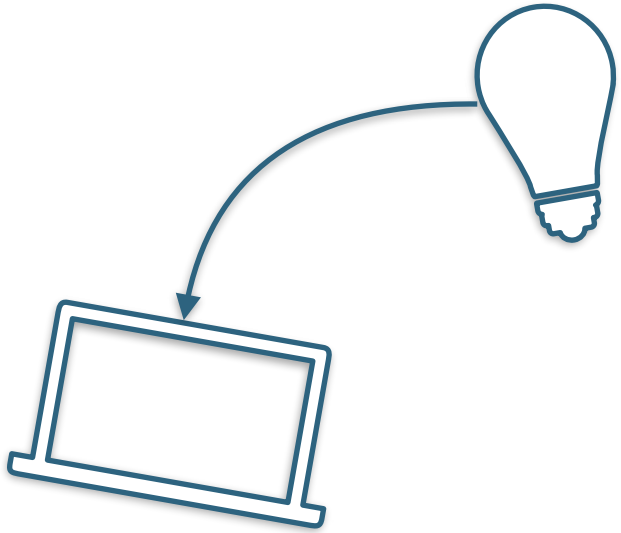
Workflow

- **Scientific Idea**
  - Specification of problem
  - Feasibility studies



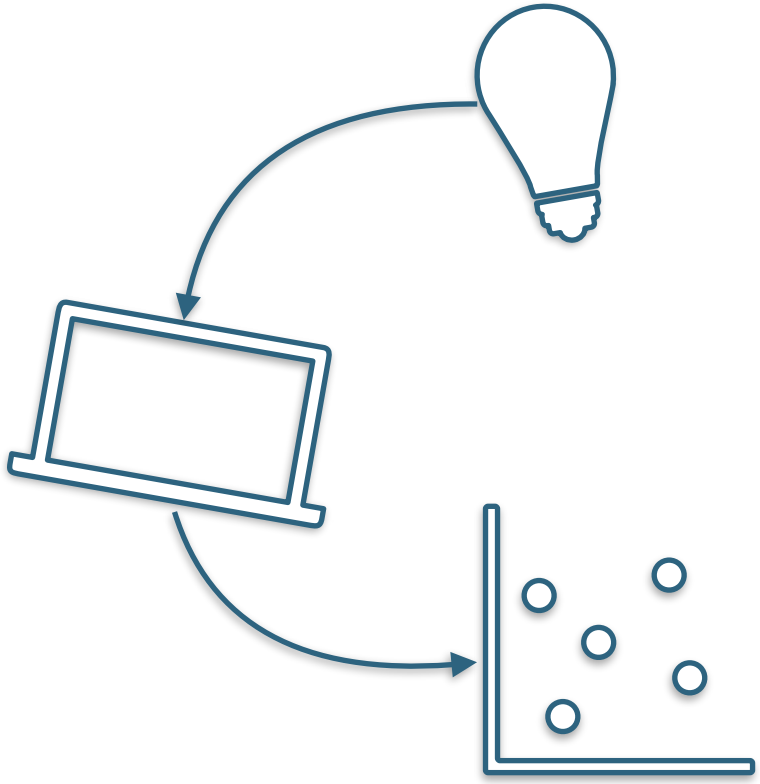
Workflow

- **Scientific Idea**
  - Specification of problem
  - Feasibility studies
- **Code Development**
  - Creating new code
  - Optimizing old code
- **Computation**
  - Job allocation & monitoring



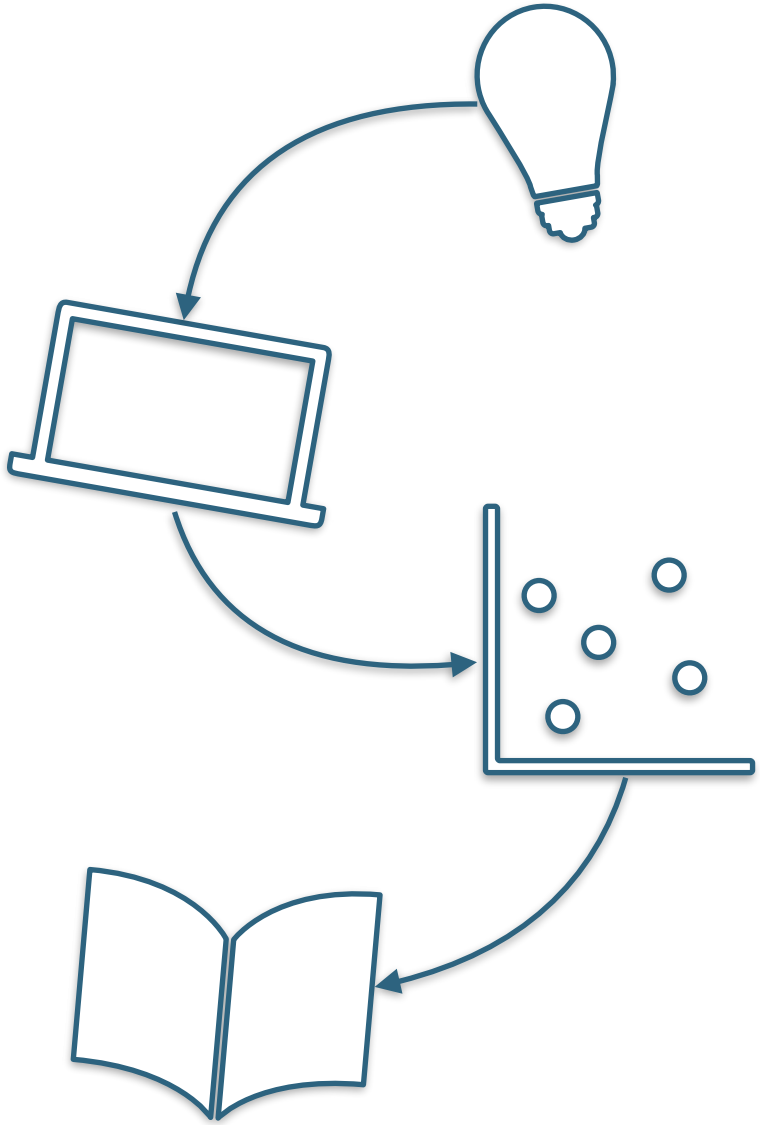
Workflow

- **Scientific Idea**
  - Specification of problem
  - Feasibility studies
- **Code Development**
  - Creating new code
  - Optimizing old code
- **Computation**
  - Job allocation & monitoring
- **Data Post-Processing**
  - Collecting files
  - Extracting relevant information



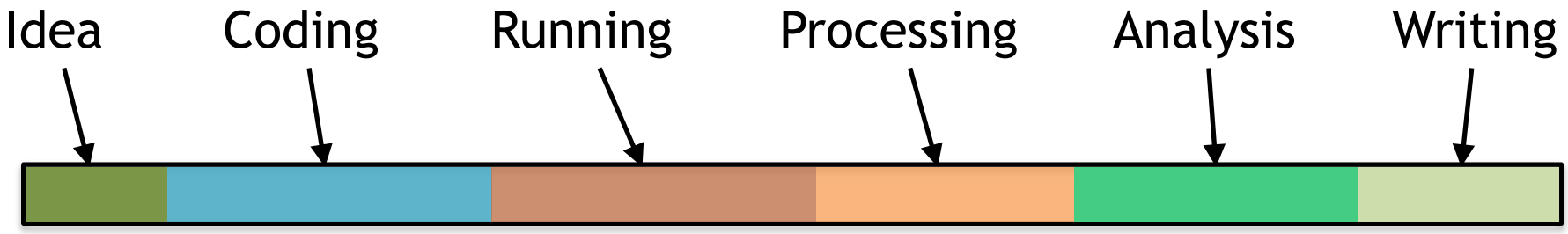
Workflow

- **Scientific Idea**
  - Specification of problem
  - Feasibility studies
- **Code Development**
  - Creating new code
  - Optimizing old code
- **Computation**
  - Job allocation & monitoring
- **Data Post-Processing**
  - Collecting files
  - Extracting relevant information
- **Analysis**
- **Writing Publication**



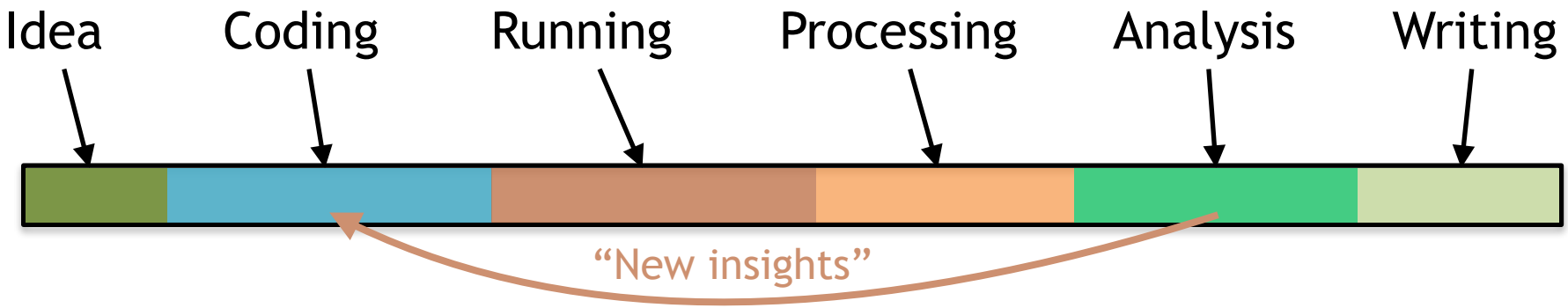
Time scales

~ Rough average estimation



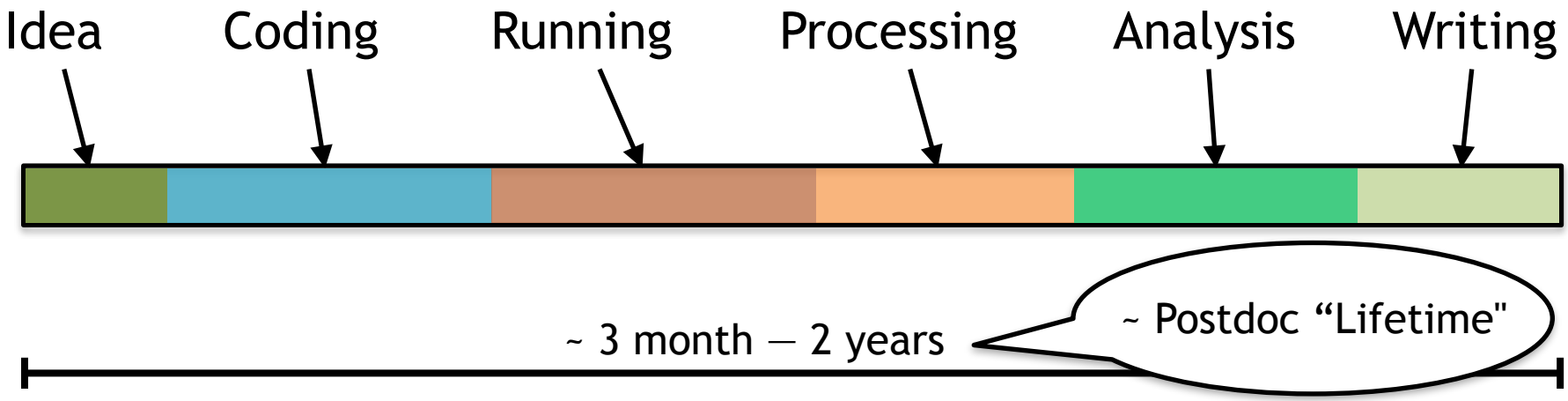
Time scales

~ Rough average estimation



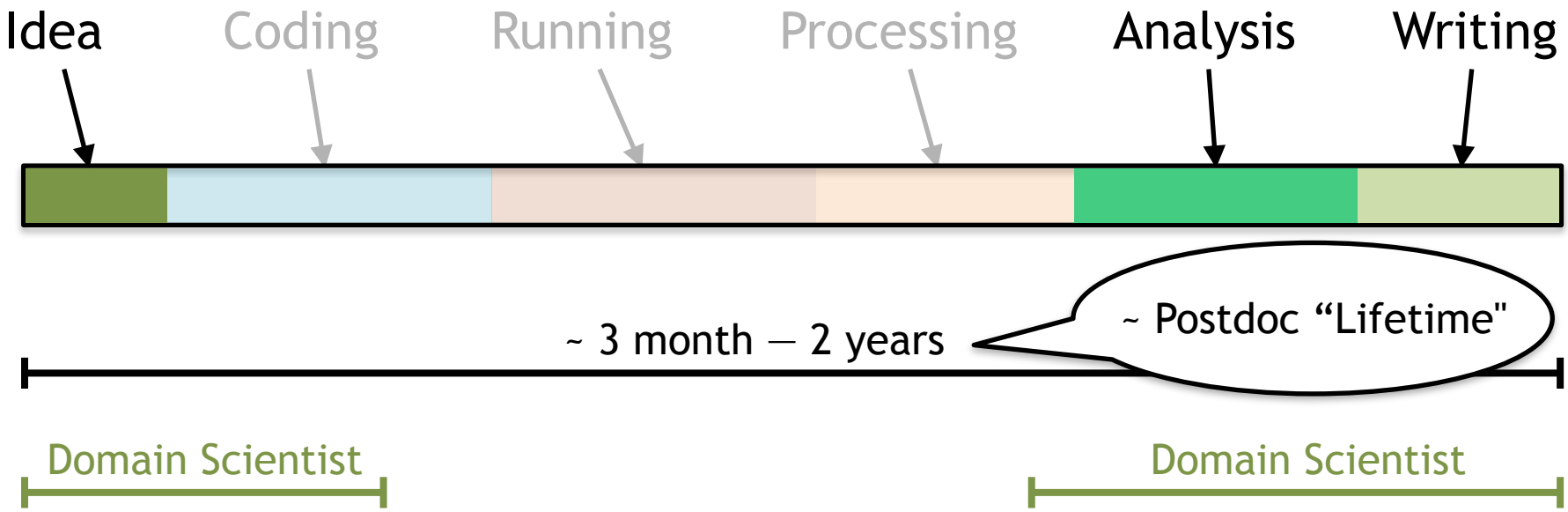
Time scales

~ Rough average estimation



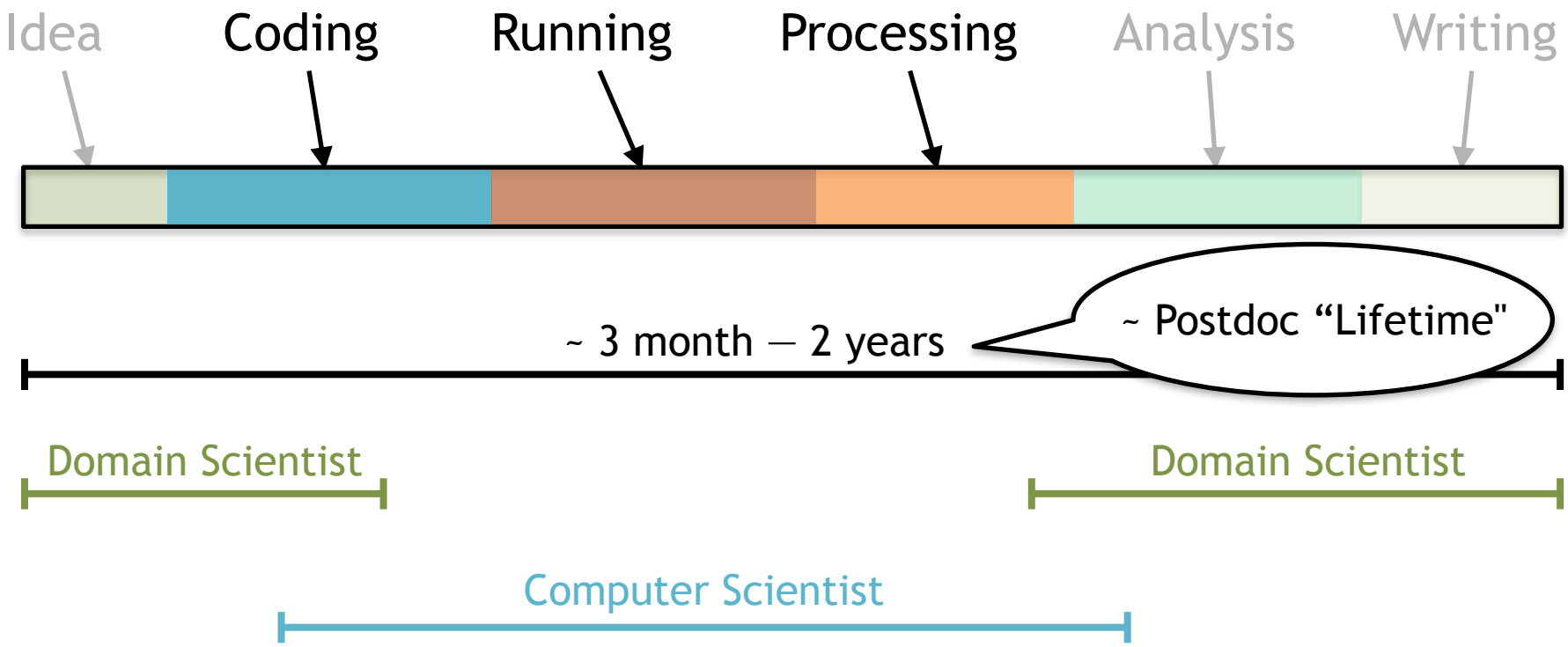
Time scales

~ Rough average estimation



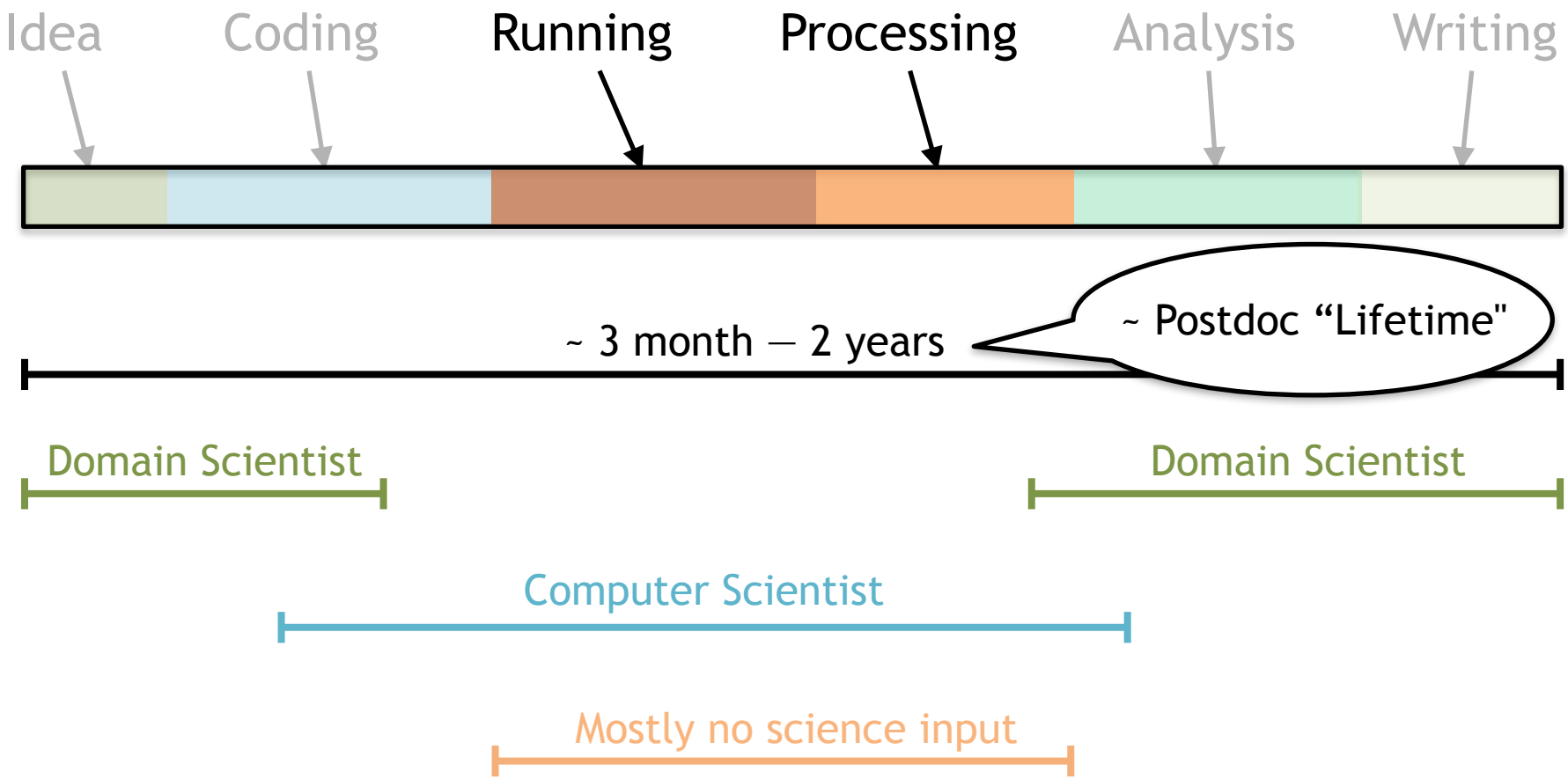
Time scales

~ Rough average estimation



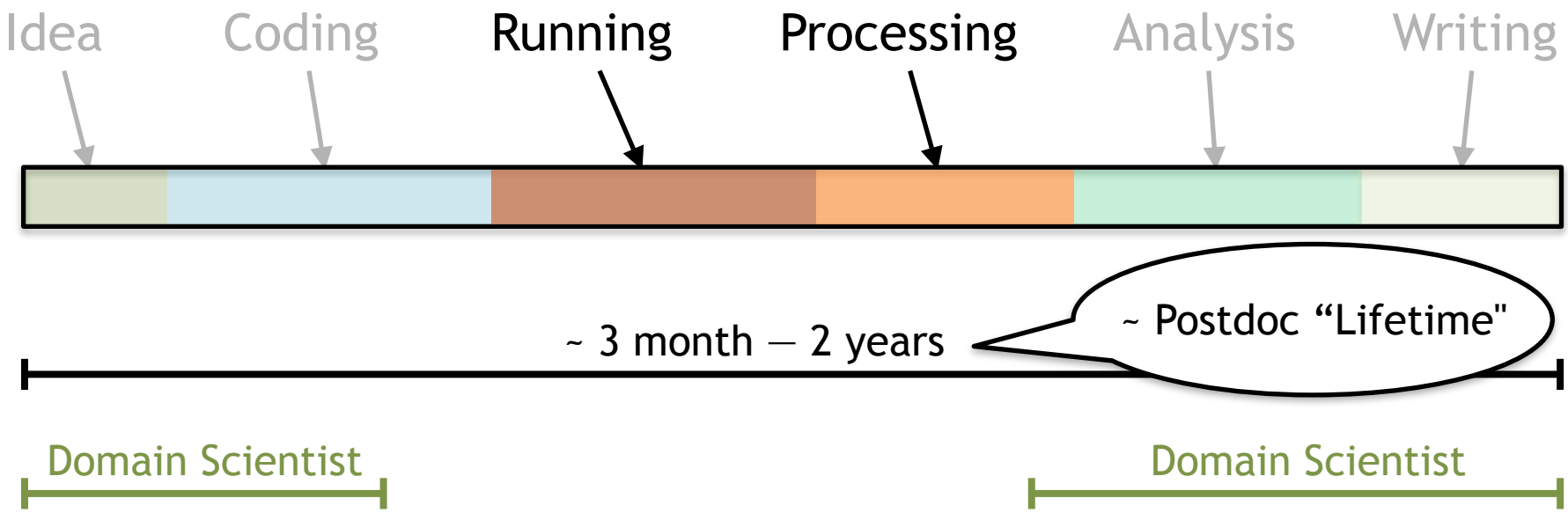
Time scales

~ Rough average estimation



Time scales

~ Rough average estimation



Overview
→ We have many files…

[1] github.com/evanberkowitz/metaq
[2] github.com/kenmcelvain/mpi_jm (public soon)
[3] github.com/callat-qcd/nucleon_elastic_FF
[4] github.com/callat-qcd/project_gA

- **Computation [1,2]**
  - Job generation → large physics & computation parameter space
  - Job scheduling → bundle jobs to large allocation
    - Computation requirements (GPU vs CPU)
    - Dependencies requirements (A → B)
    - Job manager based on (many) job files

Overview
→ We have many files…

[1] github.com/evanberkowitz/metaq
[2] github.com/kenmcelvain/mpi_jm (public soon)
[3] github.com/callat-qcd/nucleon_elastic_FF
[4] github.com/callat-qcd/project_gA

- **Computation [1,2]**
  - Job generation → large physics & computation parameter space
  - Job scheduling → bundle jobs to large allocation
    - Computation requirements (GPU vs CPU)
    - Dependencies requirements (A → B)
    - Job manager based on (many) job files
- **Data Post-Processing [3]**
  - Raw data ~ several hundred terabytes (many data files)
  - Transform data to analysis
    - Averaging, Fourier transforming, concatenating, resampling, …

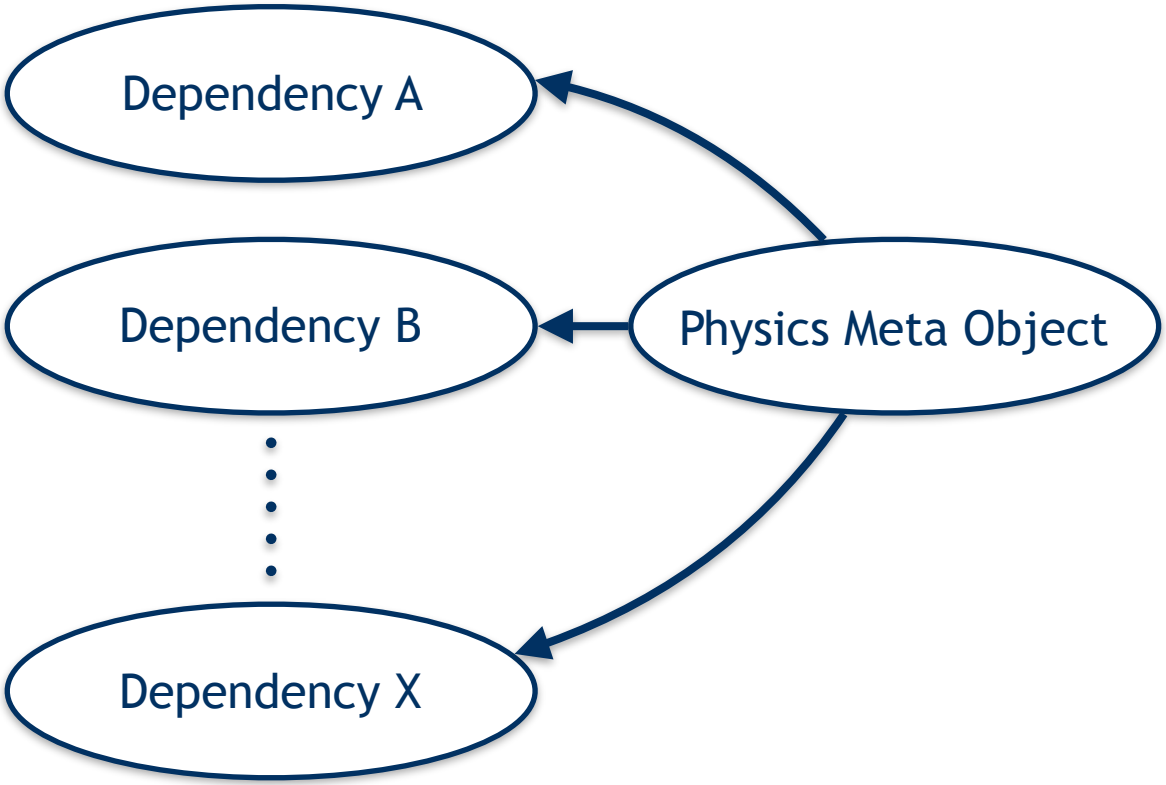
Overview
→ We have many files…

[1] github.com/evanberkowitz/metaq
[2] github.com/kenmcelvain/mpi_jm (public soon)
[3] github.com/callat-qcd/nucleon_elastic_FF
[4] github.com/callat-qcd/project_gA

- **Computation [1,2]**
  - Job generation → large physics & computation parameter space
  - Job scheduling → bundle jobs to large allocation
    - Computation requirements (GPU vs CPU)
    - Dependencies requirements (A → B)
    - Job manager based on (many) job files
- **Data Post-Processing [3]**
  - Raw data ~ several hundred terabytes (many data files)
  - Transform data to analysis
    - Averaging, Fourier transforming, concatenating, resampling, …
- **Analysis [4]**
  - Statistical data → Physical parameters
    - Input data, fit models → results & uncertainties

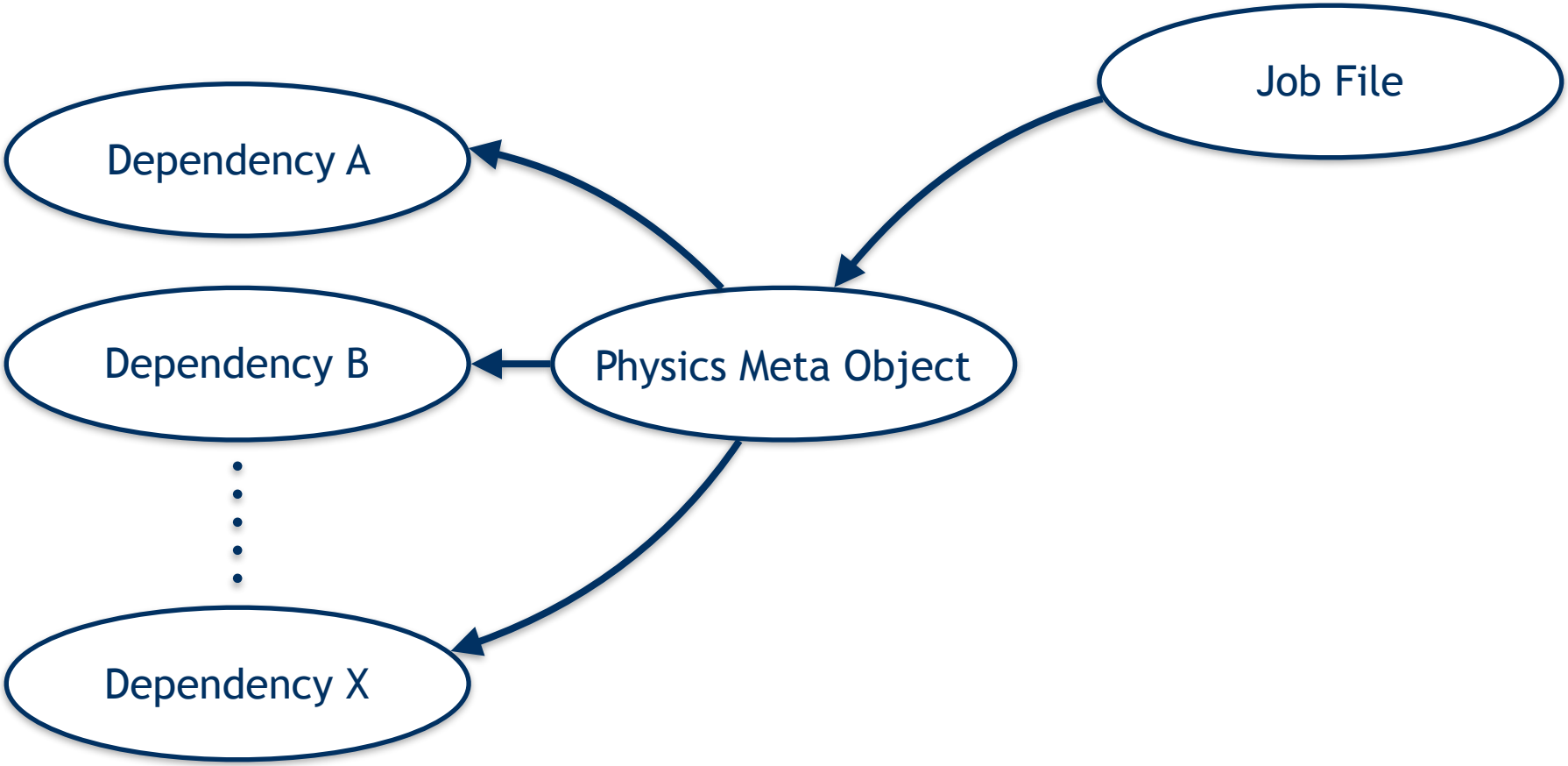
Abstraction:

→ Workflow controlled by physics meta objects



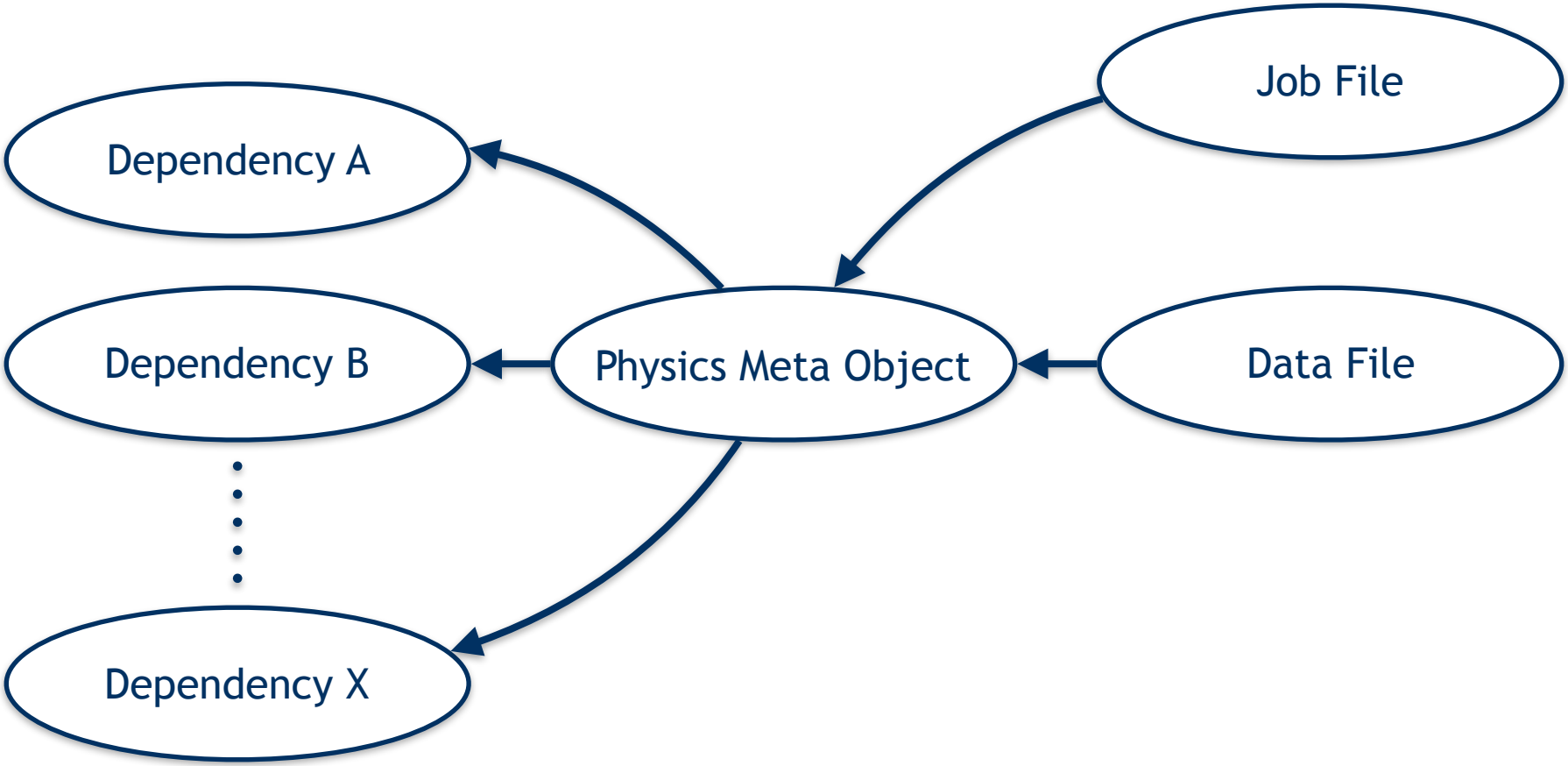
Abstraction:

→ Workflow controlled by physics meta objects



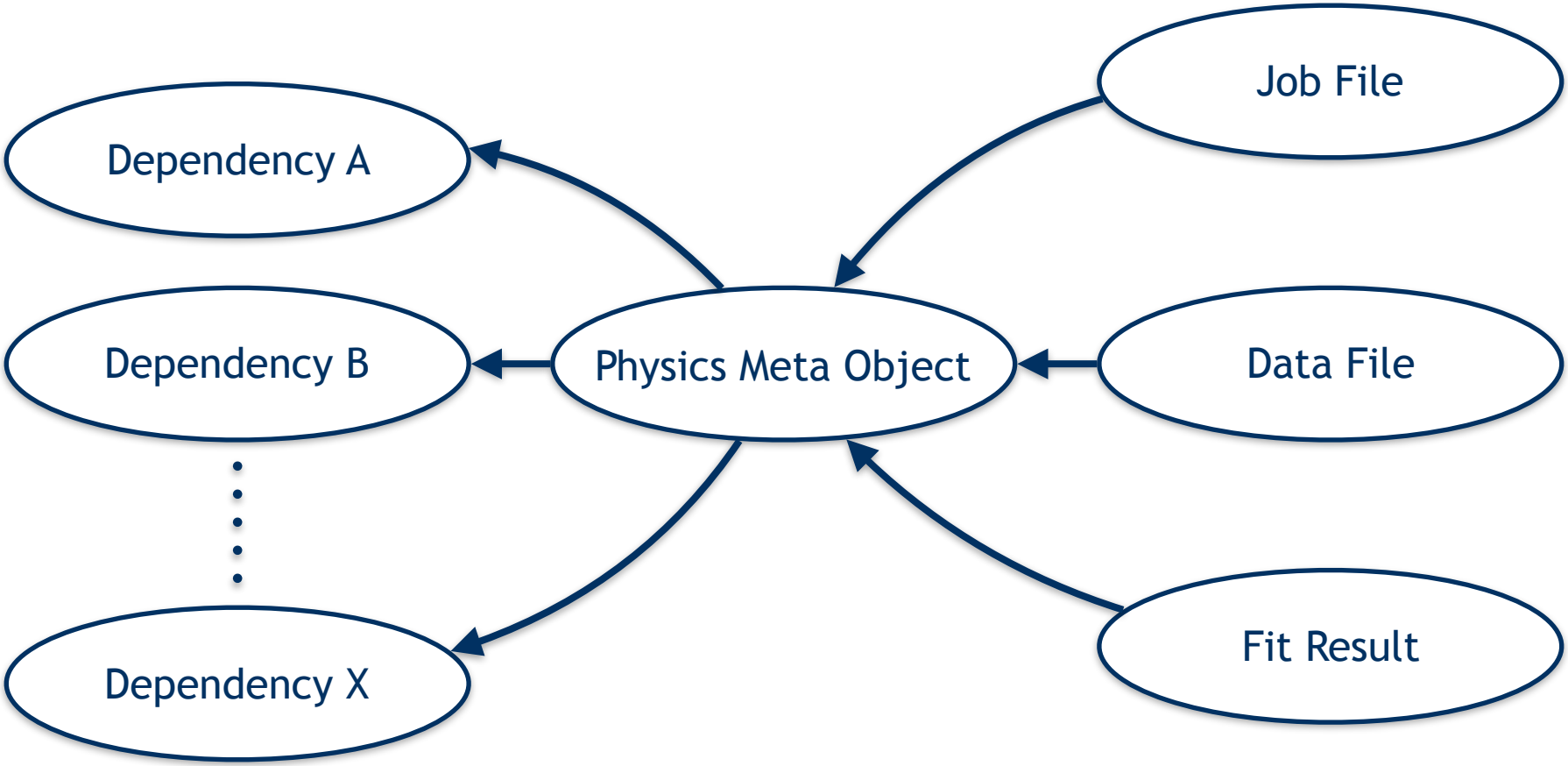
Abstraction:

→ Workflow controlled by physics meta objects



Abstraction:

→ Workflow controlled by physics meta objects



Idea

→ Replace: file based approach → DataBase*d* approach

- **First Order Advantages of Database**
  - Systemic storage of data
    - no string parsing required (regex)

```
300/prop_a09m310_e_300_gf1.0_w3.5…_x3y3z19t62.lime
```

Idea

→ Replace: file based approach → DataBase*d* approach

- **First Order Advantages of Database**
  - Systemic storage of data
    - no string parsing required (regex)
  - Relationship between data
    - A → B

```
300/prop_a09m310_e_300_gf1.0_w3.5…_x3y3z19t62.lime
```

vs

propagator_ptr_id	origin_x	origin_y	origin_z	origin_t	fermionaction_id
1	2	2	14	31	4
2	14	14	2	31	4
3	16	23	18	7	4

Idea

→ Replace: file based approach → DataBase*d* approach

- **First Order Advantages of Database**
  - Systemic storage of data
    - no string parsing required (regex)
  - Relationship between data
    - A → B
  - Filtering of data
    - Fast & automatic status repo

```
300/prop_a09m310_e_300_gf1.0_w3.5…_x3y3z19t62.lime
```

vs

propagator_ptr_id	origin_x	origin_y	origin_z	origin_t	fermionaction_id
1	2	2	14	31	4
2	14	14	2	31	4
3	16	23	18	7	4

```
pattern = r"…_x(?P<x>[0-9]+)….lime"
for file in propagator_files:
      if re.match(pattern, file) and …:
            filterd_files.append(file)
```

vs

```
SELECT * FROM Propagator WHERE origin_x > 3;
```

Idea

→ Replace: file based approach → DataBase*d* approach

- **First Order Advantages of Database**
  - Systemic storage of data
    - no string parsing required (regex)
  - Relationship between data
    - A → B
  - Filtering of data
    - Fast & automatic status repo
- **First Order Challenges**
  - Convince group to use database
    - “Learning of language SQL”

```
300/prop_a09m310_e_300_gf1.0_w3.5…_x3y3z19t62.lime
```

vs

propagator_ptr_id	origin_x	origin_y	origin_z	origin_t	fermionaction_id
1	2	2	14	31	4
2	14	14	2	31	4
3	16	23	18	7	4

```
pattern = r"…_x(?P<x>[0-9]+)….lime"
for file in propagator_files:
      if re.match(pattern, file) and …:
            filterd_files.append(file)
```

vs

```
SELECT * FROM Propagator WHERE origin_x > 3;
```

Idea

→ Replace: file based approach → DataBase*d* approach

- **First Order Advantages of Database**
  - Systemic storage of data
    - no string parsing required (regex)
  - Relationship between data
    - A → B
  - Filtering of data
    - Fast & automatic status repo
- **First Order Challenges**
  - Convince group to use database
    - “Learning of language SQL”
  - Map existing system to new structure
    - “Never change a running system”

```
300/prop_a09m310_e_300_gf1.0_w3.5…_x3y3z19t62.lime
```

vs

propagator_ptr_id	origin_x	origin_y	origin_z	origin_t	fermionaction_id
1	2	2	14	31	4
2	14	14	2	31	4
3	16	23	18	7	4

```
pattern = r"…_x(?P<x>[0-9]+)….lime"
for file in propagator_files:
      if re.match(pattern, file) and …:
            filterd_files.append(file)
```

vs

```
SELECT * FROM Propagator WHERE origin_x > 3;
```

Framework

[1] www.djangoproject.com

→ **Django: "The web framework for perfectionists with deadlines" [1]**

Object–Relational Mapper (ORM)

- Maps database tables to Python objects
  - Users don’t have to learn SQL
  - Integrates easily in existing code

Framework

[1] www.djangoproject.com

→ **Django: "The web framework for perfectionists with deadlines" [1]**

Object–Relational Mapper (ORM)

- Maps database tables to Python objects
  - Users don’t have to learn SQL
  - Integrates easily in existing code

Django

- Well maintained
  - Used by: Pinterest, Instagram, Bitbucket, Mozilla Support, …
- Well documented
  - Tutorials, multiple version history, multiple languages, …
- Open Source
  - Active development, many extensions, …

Python Classes ↔ Tables

Nf211(GaugeConfig) Table

SHORT TAG	STREAM	CONFIG	NX	NY	NZ	NT	MPI
a09m310	e	396	32	32	32	96	310.000000
a09m310	e	390	32	32	32	96	310.000000
a09m310	e	384	32	32	32	96	310.000000
a09m310	e	378	32	32	32	96	310.000000

Python Classes ↔ Tables

Nf211(GaugeConfig) Table

SHORT TAG	STREAM	CONFIG	NX	NY	NZ	NT	MPI
a09m310	e	396	32	32	32	96	310.000000
a09m310	e	390	32	32	32	96	310.000000
a09m310	e	384	32	32	32	96	310.000000
a09m310	e	378	32	32	32	96	310.000000

Nf211(GaugeConfig) Class

```
1    from lattedb.gaugeconfig.models import Nf211
2
3    gaugeconfigs = Nf211.objects.all()
4    gaugeconfig = gaugeconfigs[0]
5
6    gaugeconfig.stream == "e"
7    gaugeconfig.config += 1
8
9    gaugeconfig.save()
```

Python Classes ↔ Tables

Nf211(GaugeConfig) Table

SHORT TAG	STREAM	CONFIG	NX	NY	NZ	NT	MPI
a09m310	e	396	32	32	32	96	310.000000
a09m310	e	390	32	32	32	96	310.000000
a09m310	e	384	32	32	32	96	310.000000
a09m310	e	378	32	32	32	96	310.000000

Nf211(GaugeConfig) Class

```
1   from lattedb.gaugeconfig.models import Nf211
2
3   gaugeconfigs = Nf211.objects.all()
4   gaugeconfig = gaugeconfigs[0]
5
6   gaugeconfig.stream == "e"
7   gaugeconfig.config += 1
8
9   gaugeconfig.save()
```

Object Relational Manager converts SQL to class

Python Classes ↔ Tables

Nf211(GaugeConfig) Table

SHORT TAG	STREAM	CONFIG	NX	NY	NZ	NT	MPI
a09m310	e	396	32	32	32	96	310.000000
a09m310	e	390	32	32	32	96	310.000000
a09m310	e	384	32	32	32	96	310.000000
a09m310	e	378	32	32	32	96	310.000000

Nf211(GaugeConfig) Class

```
1    from lattedb.gaugeconfig.models import Nf211
2
3    gaugeconfigs = Nf211.objects.all()
4    gaugeconfig = gaugeconfigs[0]
5
6    gaugeconfig.stream == "e"
7    gaugeconfig.config += 1
8
9    gaugeconfig.save()
```

Object Relational Manager converts SQL to class

Convert all rows (lazy)

Python Classes ↔ Tables

Nf211(GaugeConfig) Table

SHORT TAG	STREAM	CONFIG	NX	NY	NZ	NT	MPI
a09m310	e	396	32	32	32	96	310.000000
a09m310	e	390	32	32	32	96	310.000000
a09m310	e	384	32	32	32	96	310.000000
a09m310	e	378	32	32	32	96	310.000000

Nf211(GaugeConfig) Class

```
1    from lattedb.gaugeconfig.models import Nf211
2
3    gaugeconfigs = Nf211.objects.all()
4    gaugeconfig = gaugeconfigs[0]
5
6    gaugeconfig.stream == "e"
7    gaugeconfig.config += 1
8
9    gaugeconfig.save()
```

Object Relational Manager converts SQL to class

Convert all rows (lazy)

Check/adjust status of configs

Python Classes ↔ Tables

Nf211(GaugeConfig) Table

SHORT TAG	STREAM	CONFIG	NX	NY	NZ	NT	MPI
a09m310	e	396	32	32	32	96	310.000000
a09m310	e	390	32	32	32	96	310.000000
a09m310	e	384	32	32	32	96	310.000000
a09m310	e	378	32	32	32	96	310.000000

Nf211(GaugeConfig) Class

```
1    from lattedb.gaugeconfig.models import Nf211
2
3    gaugeconfigs = Nf211.objects.all()
4    gaugeconfig = gaugeconfigs[0]
5
6    gaugeconfig.stream == "e"
7    gaugeconfig.config += 1
8
9    gaugeconfig.save()
```

Object Relational Manager converts SQL to class

Convert all rows (lazy)

Check/adjust status of configs

Insert/Update database

Python Classes ↔ Tables

Nf211(GaugeConfig) Table

SHORT TAG	STREAM	CONFIG	NX	NY	NZ	NT	MPI
a09m310	e	396	32	32	32	96	310.000000
a09m310	e	390	32	32	32	96	310.000000
a09m310	e	384	32	32	32	96	310.000000
a09m310	e	378	32	32	32	96	310.000000

2nd Order Advantages:

- Consistency checks
- Auto-documentation
- High-level API support

Nf211(GaugeConfig) Class

```
1    from lattedb.gaugeconfig.models import Nf211
2
3    gaugeconfigs = Nf211.objects.all()
4    gaugeconfig = gaugeconfigs[0]
5
6    gaugeconfig.stream == "e"
7    gaugeconfig.config += 1
8
9    gaugeconfig.save()
```

Object Relational Manager converts SQL to class

Convert all rows (lazy)

Check/adjust status of configs

Insert/Update database

Relationships & Updates

Regular Columns

- `short_tag`
- `stream`
- `config`
- …

Relationships & Updates

SHORT TAG	STREAM	CONFIG	NX	NY	NZ	NT	LIGHT	MPI
a09m310	e	396	32	32	32	96	Hisq[FermionAction](quark_mass=0.007400, quark_tag=light)	310.000000
a09m310	e	390	32	32	32	96	Hisq[FermionAction](quark_mass=0.007400, quark_tag=light)	310.000000
a09m310	e	384	32	32	32	96	Hisq[FermionAction](quark_mass=0.007400, quark_tag=light)	310.000000
a09m310	e	378	32	32	32	96	Hisq[FermionAction](quark_mass=0.007400, quark_tag=light)	310.000000

Regular Columns

- `short_tag`
- `stream`
- `config`
- …

Relationships & Updates

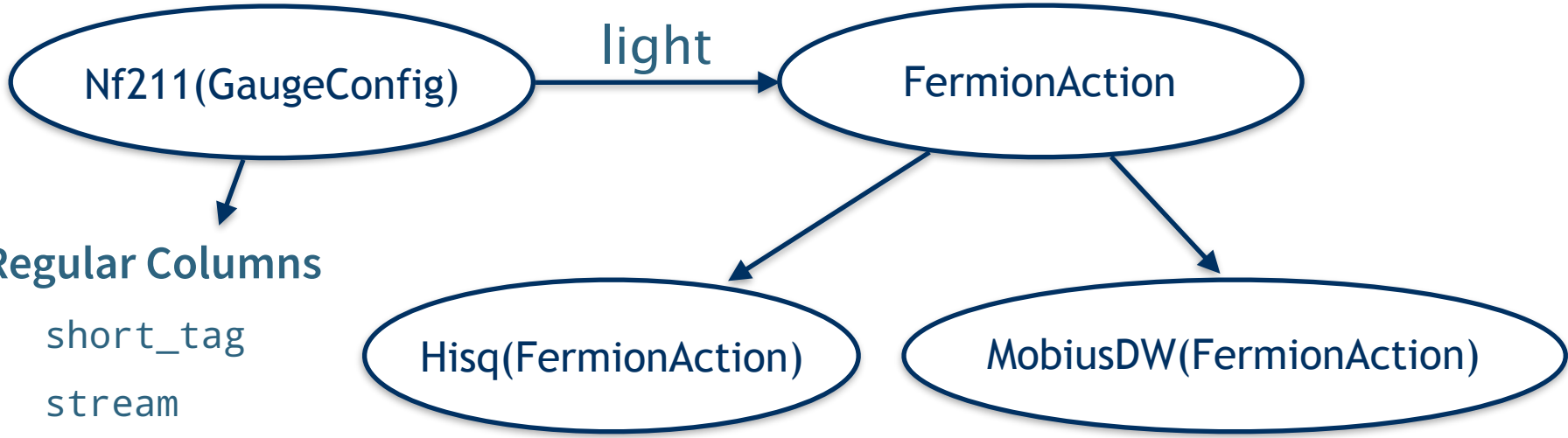
SHORT TAG	STREAM	CONFIG	NX	NY	NZ	NT	LIGHT	MPI
a09m310	e	396	32	32	32	96	Hisq[FermionAction](quark_mass=0.007400, quark_tag=light)	310.000000
a09m310	e	390	32	32	32	96	Hisq[FermionAction](quark_mass=0.007400, quark_tag=light)	310.000000
a09m310	e	384	32	32	32	96	Hisq[FermionAction](quark_mass=0.007400, quark_tag=light)	310.000000
a09m310	e	378	32	32	32	96	Hisq[FermionAction](quark_mass=0.007400, quark_tag=light)	310.000000

Regular Columns

- short_tag
- stream
- config
- …

Relationships & Updates

SHORT TAG	STREAM	CONFIG	NX	NY	NZ	NT	LIGHT	MPI
a09m310	e	396	32	32	32	96	Hisq[FermionAction](quark_mass=0.007400, quark_tag=light)	310.000000
a09m310	e	390	32	32	32	96	Hisq[FermionAction](quark_mass=0.007400, quark_tag=light)	310.000000
a09m310	e	384	32	32	32	96	Hisq[FermionAction](quark_mass=0.007400, quark_tag=light)	310.000000
a09m310	e	378	32	32	32	96	Hisq[FermionAction](quark_mass=0.007400, quark_tag=light)	310.000000

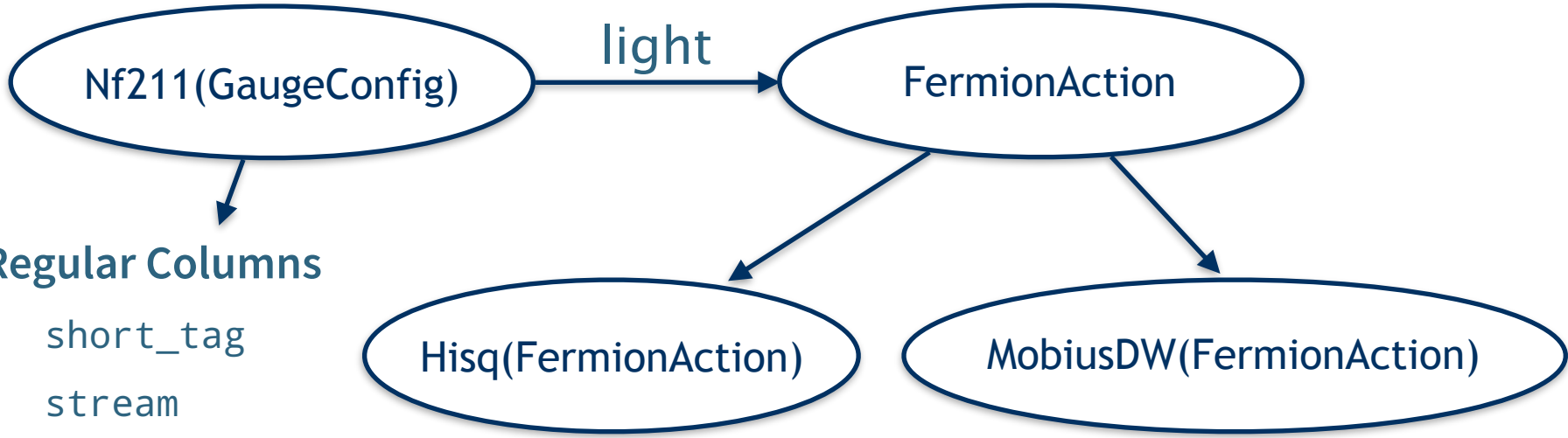


Regular Columns

- `short_tag`
- `stream`
- `config`
- …

Relationships & Updates

SHORT TAG	STREAM	CONFIG	NX	NY	NZ	NT	LIGHT	MPI
a09m310	e	396	32	32	32	96	Hisq[FermionAction](quark_mass=0.007400, quark_tag=light)	310.000000
a09m310	e	390	32	32	32	96	Hisq[FermionAction](quark_mass=0.007400, quark_tag=light)	310.000000
a09m310	e	384	32	32	32	96	Hisq[FermionAction](quark_mass=0.007400, quark_tag=light)	310.000000
a09m310	e	378	32	32	32	96	Hisq[FermionAction](quark_mass=0.007400, quark_tag=light)	310.000000



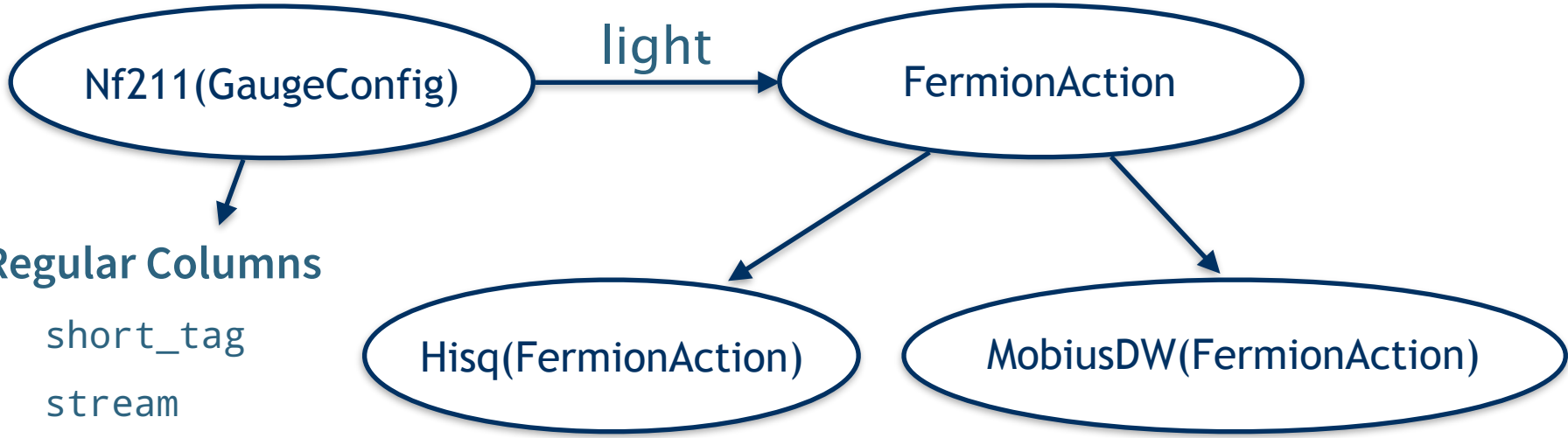
Regular Columns

- short_tag
- stream
- config
- …

```
1  from lattedb.gaugeconfig.models import Nf211
2  from lattedb.fermionaction.models import Hisq
3
4  gaugeconfig = Nf211.objects.first()
5
6  gaugeconfig.light = Hisq(...)
```

Relationships & Updates

SHORT TAG	STREAM	CONFIG	NX	NY	NZ	NT	LIGHT	MPI
a09m310	e	396	32	32	32	96	Hisq[FermionAction](quark_mass=0.007400, quark_tag=light)	310.000000
a09m310	e	390	32	32	32	96	Hisq[FermionAction](quark_mass=0.007400, quark_tag=light)	310.000000
a09m310	e	384	32	32	32	96	Hisq[FermionAction](quark_mass=0.007400, quark_tag=light)	310.000000
a09m310	e	378	32	32	32	96	Hisq[FermionAction](quark_mass=0.007400, quark_tag=light)	310.000000



Regular Columns

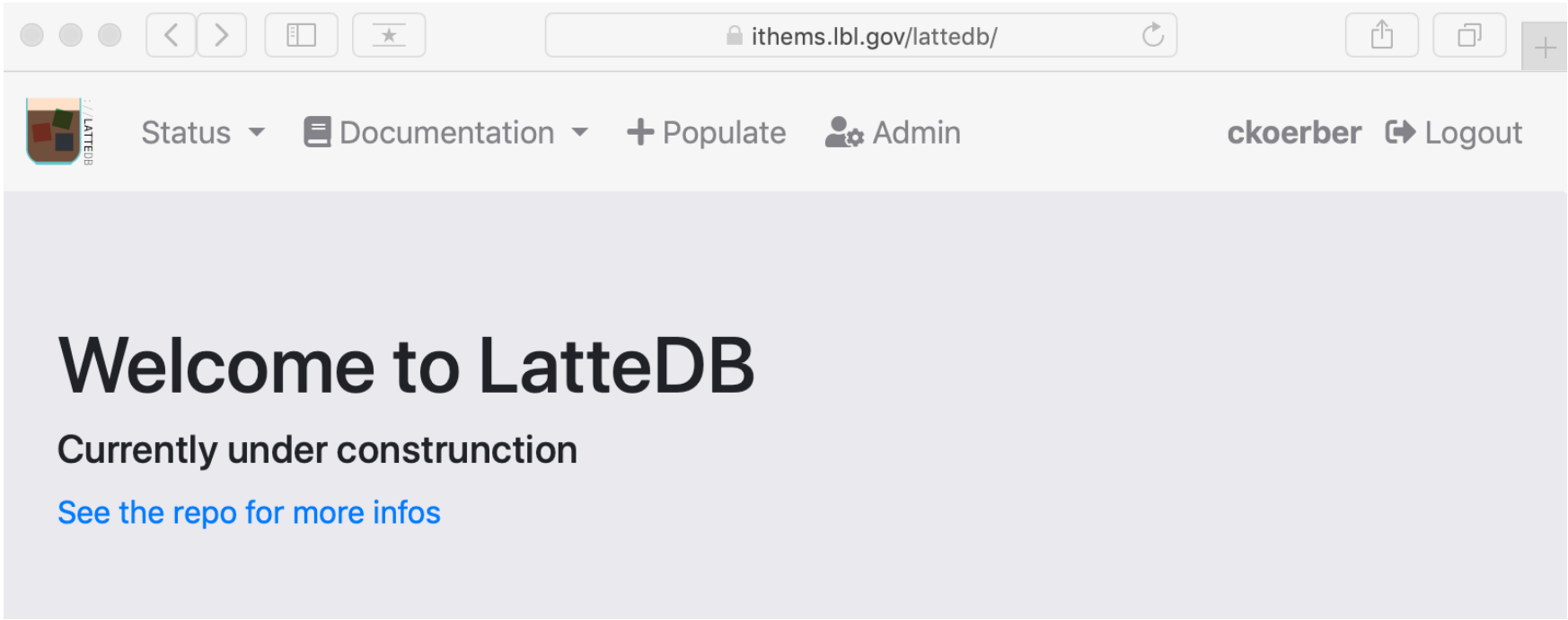
- short_tag
- stream
- config
- …

```
1   from lattedb.gaugeconfig.models import Nf211
2   from lattedb.fermionaction.models import Hisq
3
4   gaugeconfig = Nf211.objects.first()
5
6   gaugeconfig.light = Hisq(...)
```

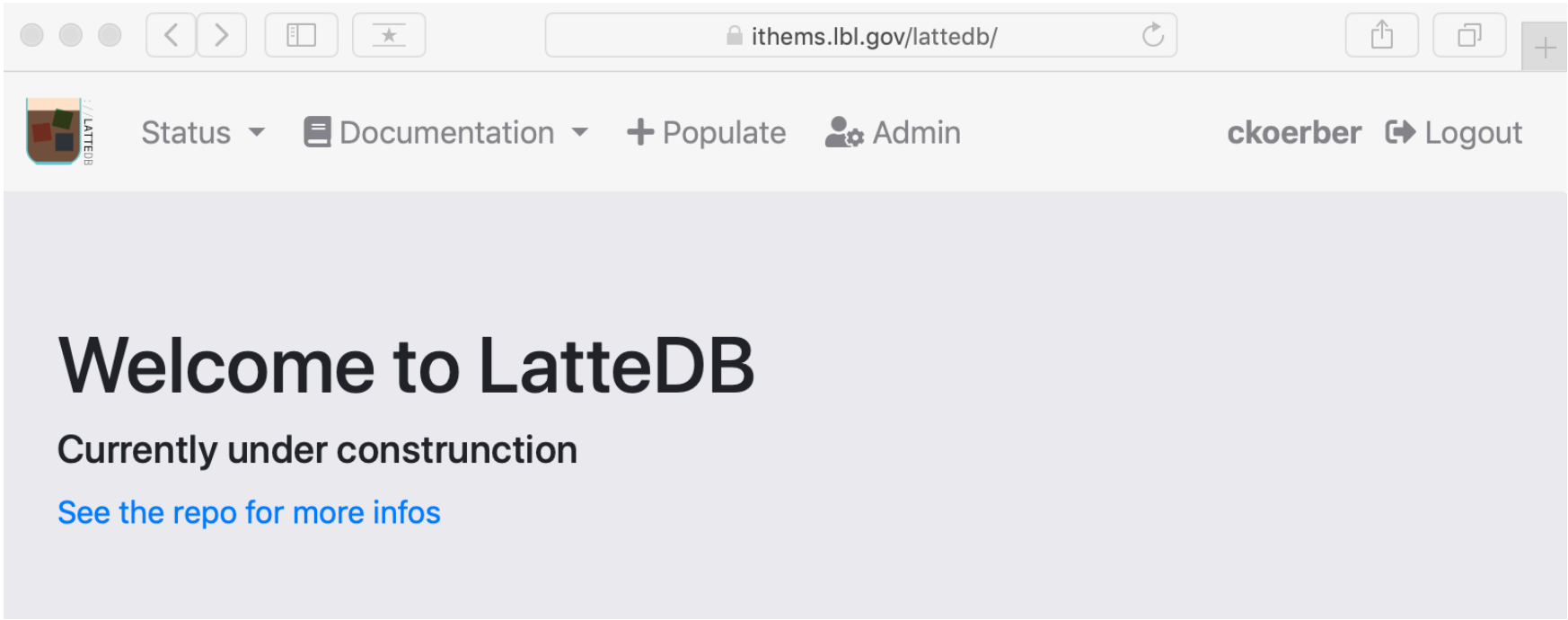
2nd Order Advantages:

- Information reuse
- Flexibility

Features: Webpage



Features: Webpage



2nd Order Advantages:

- Automated progress reports
- Centralized source for information
- Easy to set up open data platform

Features: Table views

Real view – fake data

Home › Correlator › Baryon2pts

Select baryon2pt to view

Action: -------- Go 0 of 100 selected

ID	ENSEMBLE	CONFIG	TAG	ORIGIN (X, Y, Z, T)	PARITY	SPIN X2	SPIN Z X2	ISOSPIN X2	ISOSPIN Z X2
1888	a09m310	396	proton	(7, 29, 23, 41)	1	1	1	1	1
1887	a09m310	396	proton	(7, 29, 23, 41)	1	1	-1	1	1
1886	a09m310	396	proton	(7, 29, 23, 41)	-1	1	1	1	1
1885	a09m310	396	proton	(7, 29, 23, 41)	-1	1	-1	1	1
1884	a09m310	396	proton	(23, 13, 7, 41)	1	1	1	1	1
1883	a09m310	396	proton	(23, 13, 7, 41)	1	1	-1	1	1
1882	a09m310	396	proton	(23, 13, 7, 41)	-1	1	1	1	1
1881	a09m310	396	proton	(23, 13, 7, 41)	-1	1	-1	1	1
1880	a09m310	396	proton	(18, 21, 25, 89)	1	1	1	1	1
1879	a09m310	396	proton	(18, 21, 25, 89)	1	1	-1	1	1

FILTER

By tag

All

proton

By ensemble

All

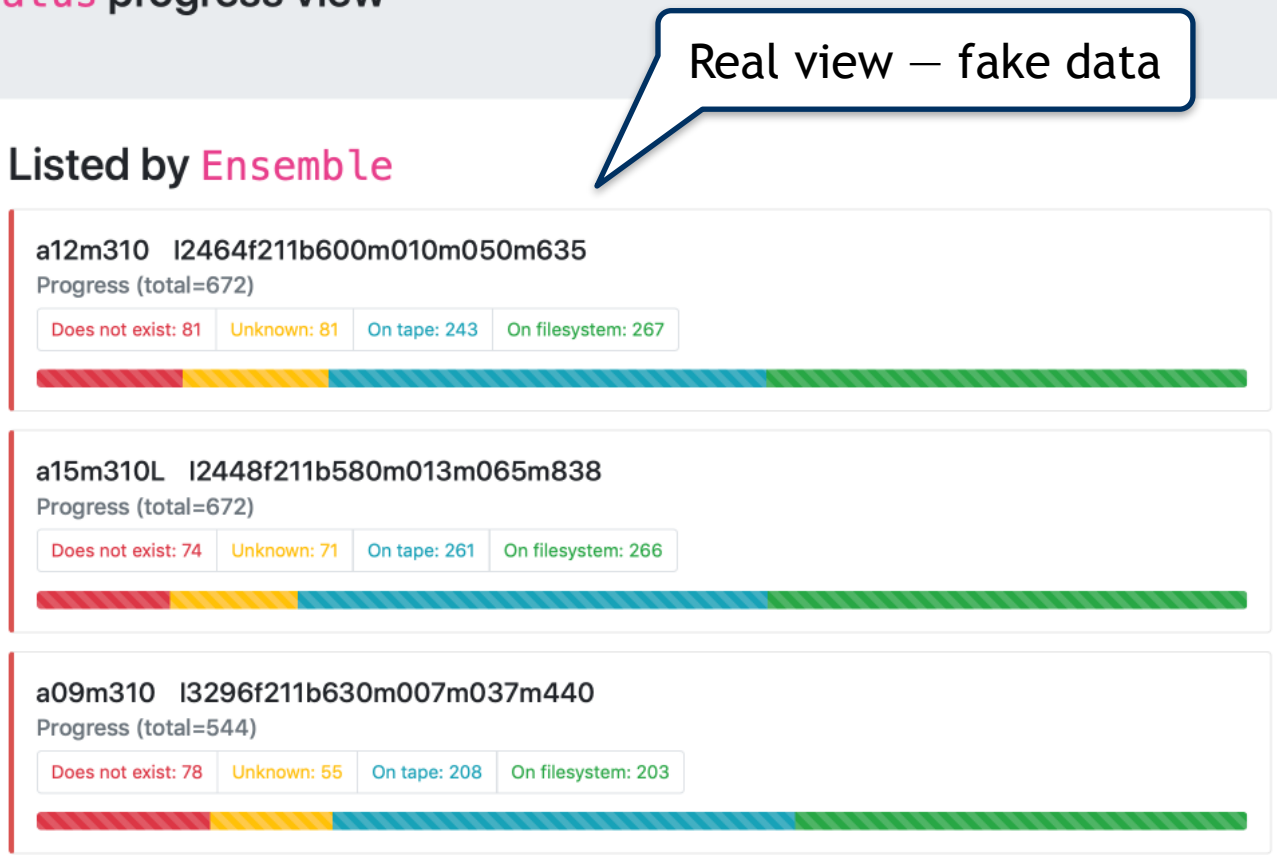
a15m310L

a09m310

a12m310

Features: Automatic Progress Report

`Baryon2pt Status` progress view



Features: Automatic Documentation

Documentation of `lattedb.propagator`

See table properties

`OneToAll`

Module: `lattedb.propagator.models`

Base: `Propagator[Base]`

Columns

Name	Type	Help
`tag`	`CharField` (Optional)	(Optional) Char(20): User defined tag for easy searches
`misc`	`JSONField` (Optional)	(Optional) JSON: {'anything': 'you want'}
`gaugeconfig`	`ForeignKey`: GaugeConfig	ForeignKey pointing to specific gauge configuration inverted on
`fermionaction`	`ForeignKey`: FermionAction	ForeignKey pointing to valence lattice fermion action
`origin_x`	`PositiveSmallIntegerField`	PositiveSmallInt: x-coordinate origin location of the propagator
`origin_y`	`PositiveSmallIntegerField`	PositiveSmallInt: y-coordinate origin location of the propagator
`origin_z`	`PositiveSmallIntegerField`	PositiveSmallInt: z-coordinate origin location of the propagator
`origin_t`	`PositiveSmallIntegerField`	PositiveSmallInt: t-coordinate origin location of the propagator

Install

1. clone the repository (not public yet)

```
> git clone https://github.com/callat-qcd/lattedb.git
```

Install

1. clone the repository (not public yet)

```
> git clone https://github.com/callat-qcd/lattedb.git
```

2. pip install it (including dependencies)

```
> cd lattedb
> pip install -e .
```

Install

1. clone the repository (not public yet)

```
> git clone https://github.com/callat-qcd/lattedb.git
```

2. pip install it (including dependencies)

```
> cd lattedb
> pip install -e .
```

3. establish a database connection

```
> touch db-config.yaml
```



```
ENGINE: django.db.backends.sqlite3
NAME: mydatabase
```

Install

1. clone the repository (not public yet)

```
> git clone https://github.com/callat-qcd/lattedb.git
```

2. pip install it (including dependencies)

```
> cd lattedb
> pip install -e .
```

3. establish a database connection

```
> touch db-config.yaml
```



```
ENGINE: django.db.backends.sqlite3
NAME: mydatabase
```

4. use existing tables

```
> lattedb migrate
```

Install

1. clone the repository (not public yet)

```
> git clone https://github.com/callat-qcd/lattedb.git
```

2. pip install it (including dependencies)

```
> cd lattedb
> pip install -e .
```

3. establish a database connection

```
> touch db-config.yaml
```



```
ENGINE: django.db.backends.sqlite3
NAME: mydatabase
```

4. use existing tables

```
> lattedb migrate
```

5. or add new

```
> touch new_app/models.py
> lattedb makemigrations
> lattedb migrate
```

```
from lattedb.base.models import Base


class Propagator(Base):
    """Base table for propagators
    """
```

Install

1. clone the repository (not public yet)

```
> git clone https://github.com/callat-qcd/lattedb.git
```

2. pip install it (including dependencies)

```
> cd lattedb
> pip install -e .
```

3. establish a database connection

```
> touch db-config.yaml
```



```
ENGINE: django.db.backends.sqlite3
NAME: mydatabase
```

4. use existing tables

```
> lattedb migrate
```

5. or add new

```
> touch new_app/models.py
```

```
from lattedb.base.models import Base


class Propagator(Base):
    """Base table for propagators
    """
```

```
> lattedb makemigrations
> lattedb migrate
```

6. import in your Python code

```
from lattedb.propagator.models import OneToAll
```

Install

1. clone the repository (not public yet)

```
> git clone https://github.com/callat-qcd/lattedb.git
```

2. pip install it (including dependencies)

```
> cd lattedb
> pip install -e .
```

3. establish a database connection

```
> touch db-config.yaml
```

```
ENGINE: django.db.backends.sqlite3
NAME: mydatabase
```

4. use existing tables

```
> lattedb migrate
```

5. or add new

```
> touch new_app/models.py
> lattedb makemigrations
> lattedb migrate
```

```
from lattedb.base.models import Base


class Propagator(Base):
    """Base table for propagators
    """
```

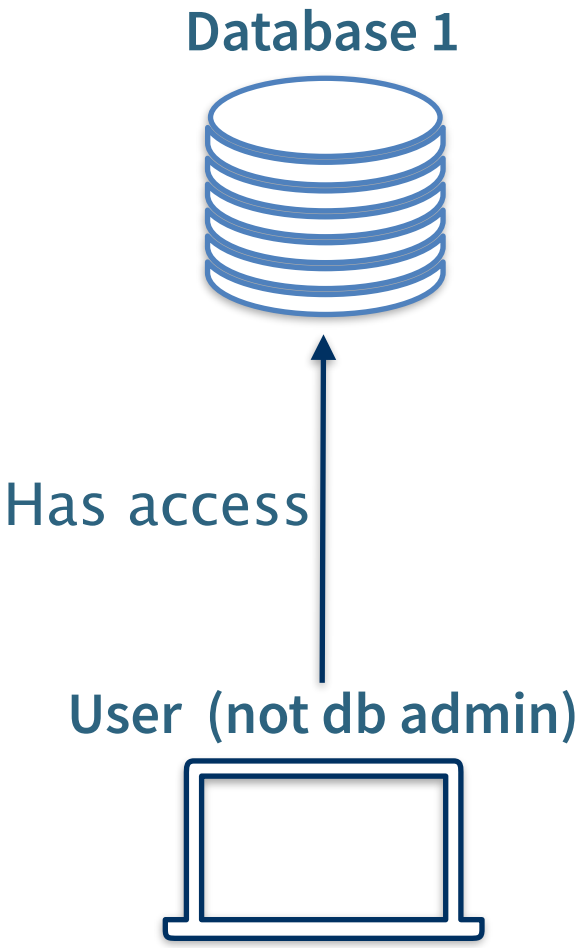
6. import in your Python code

```
from lattedb.propagator.models import OneToAll
```

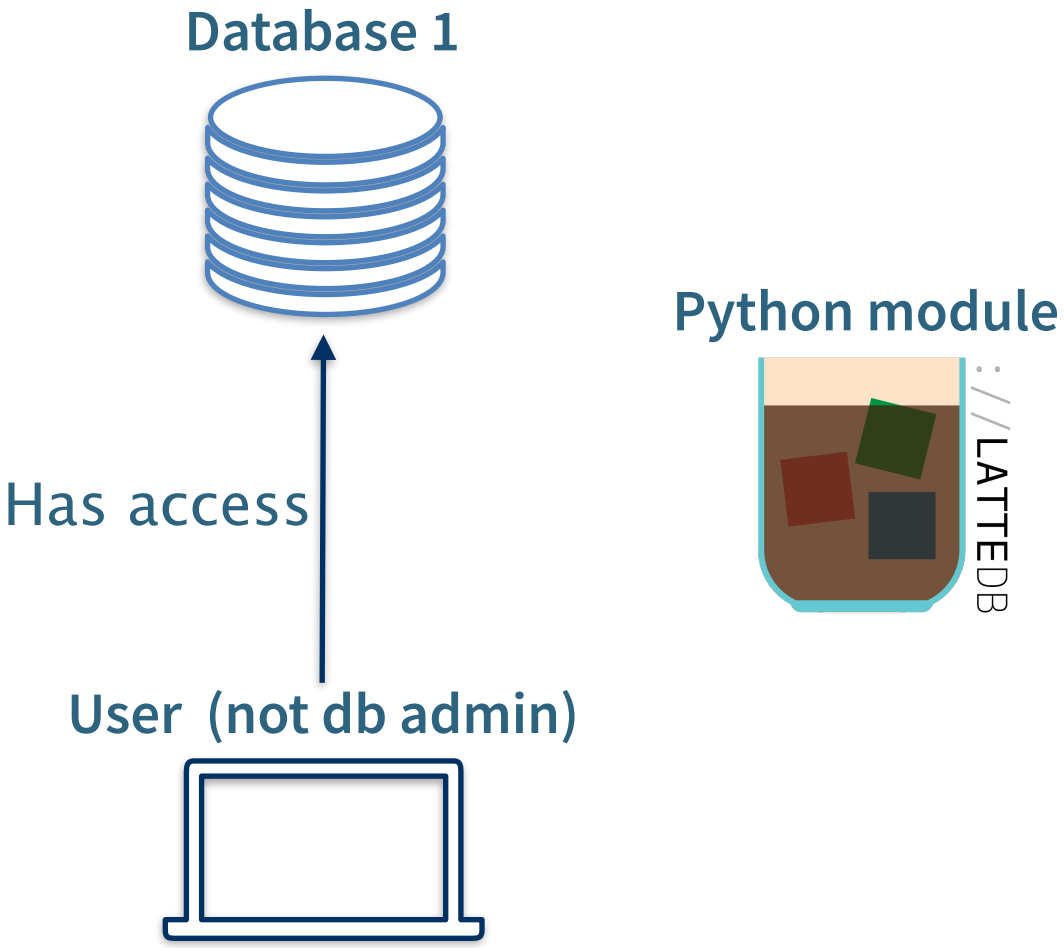
7. launch the webapp (on localhost)

```
> lattedb runserver
```

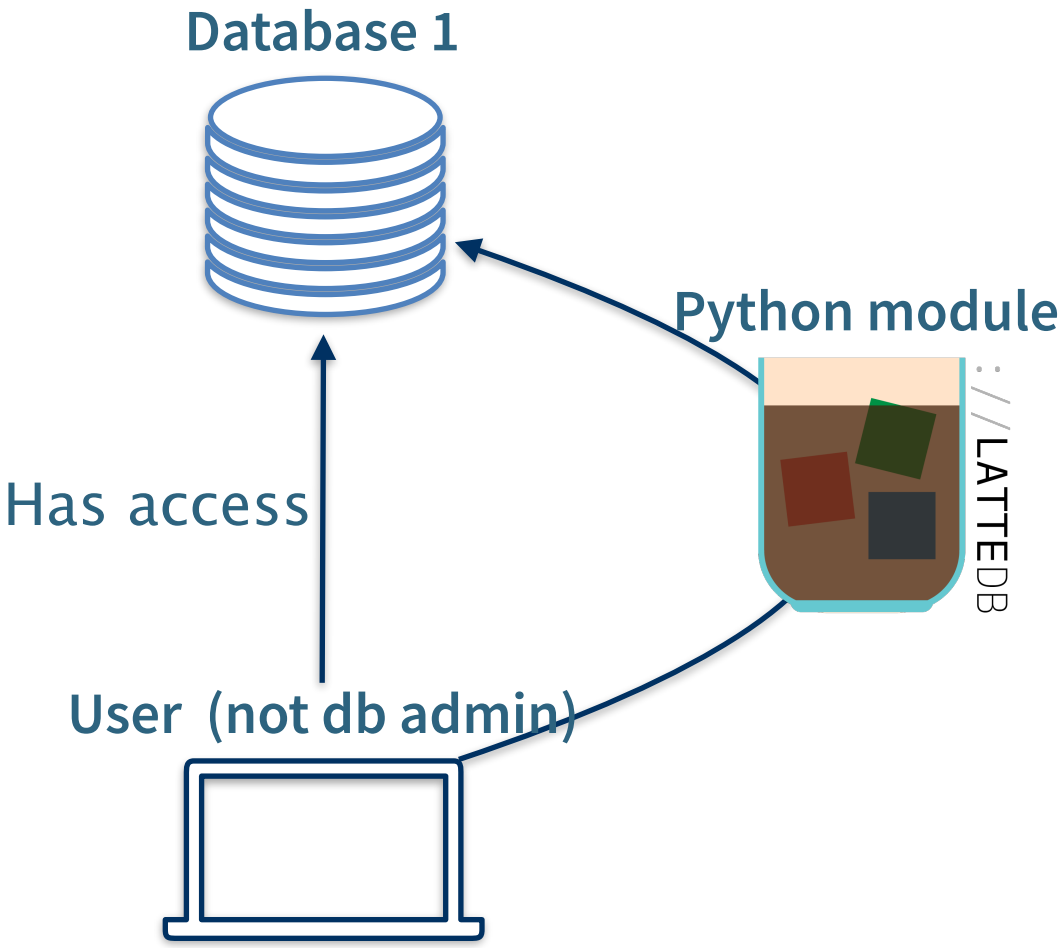
Security Layers (Python Module)



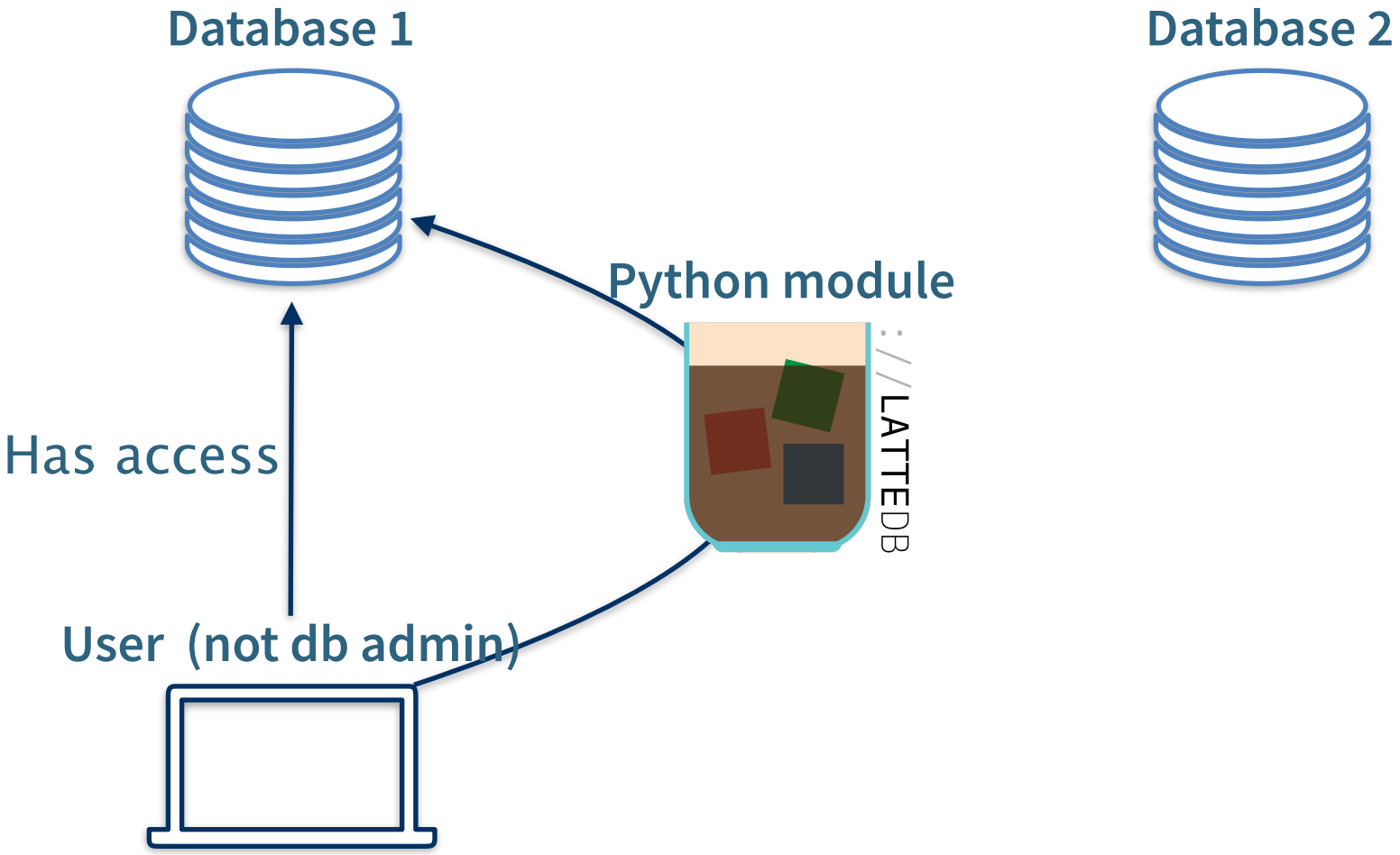
Security Layers (Python Module)



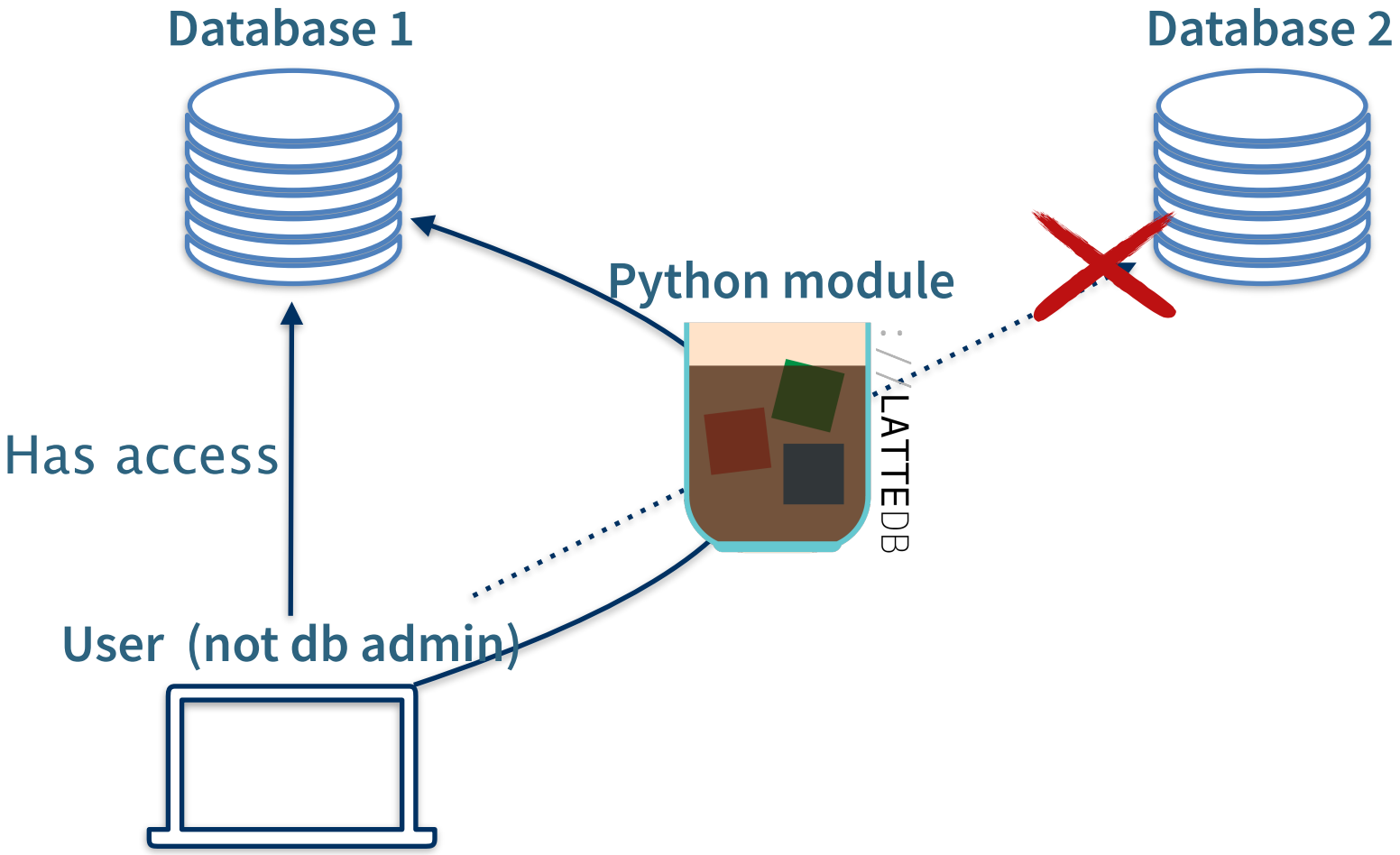
Security Layers (Python Module)



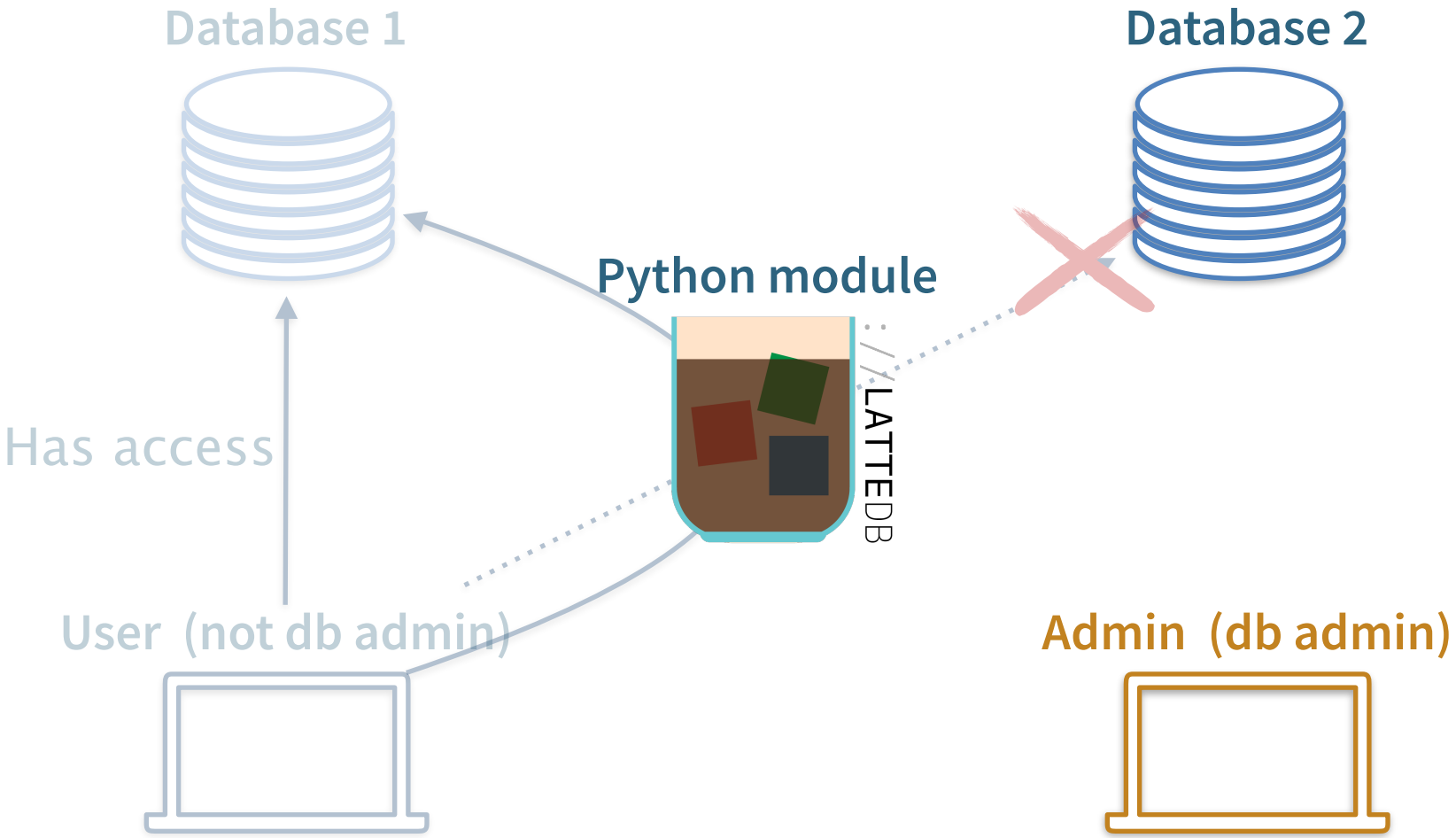
Security Layers (Python Module)



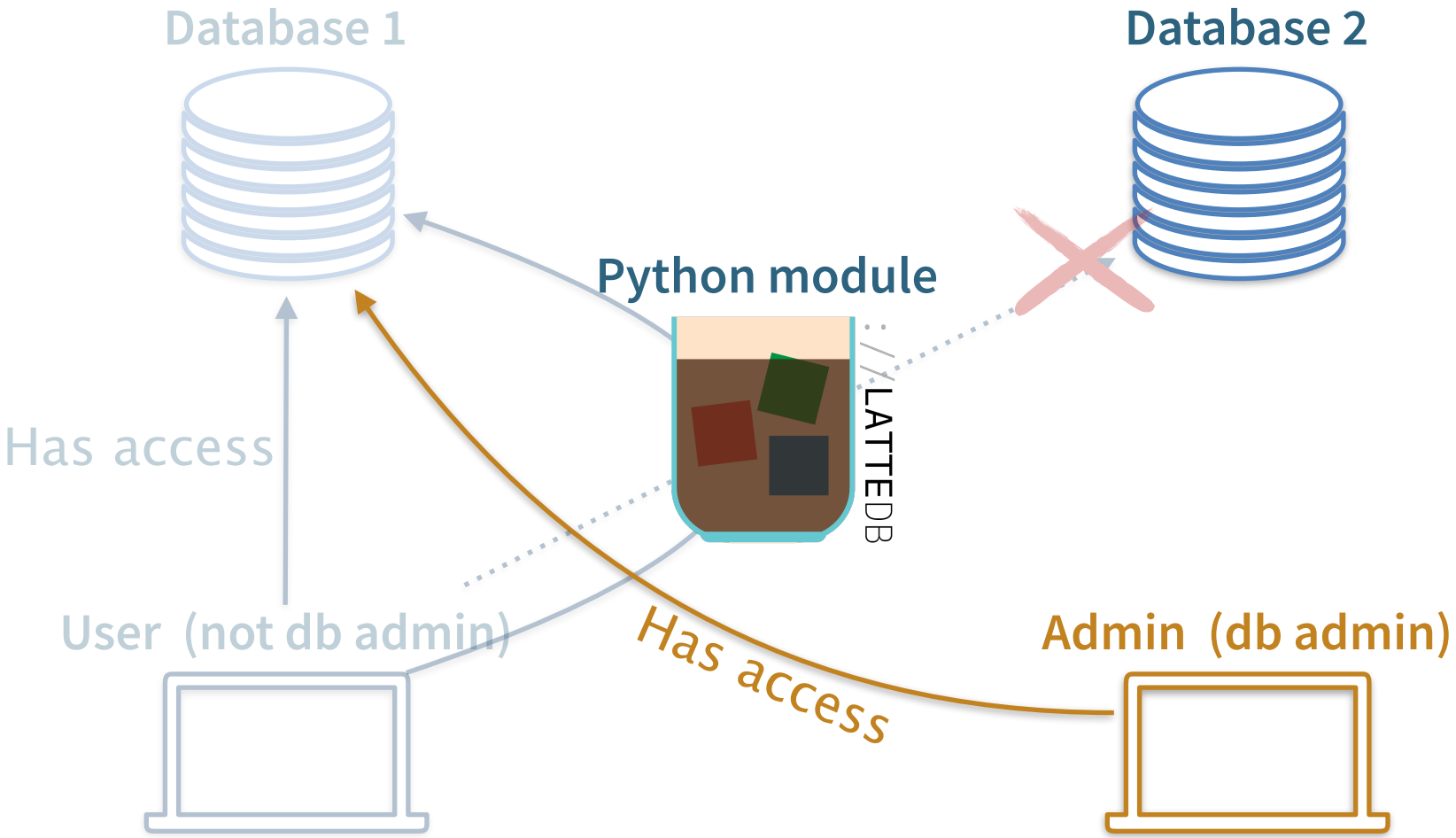
Security Layers (Python Module)



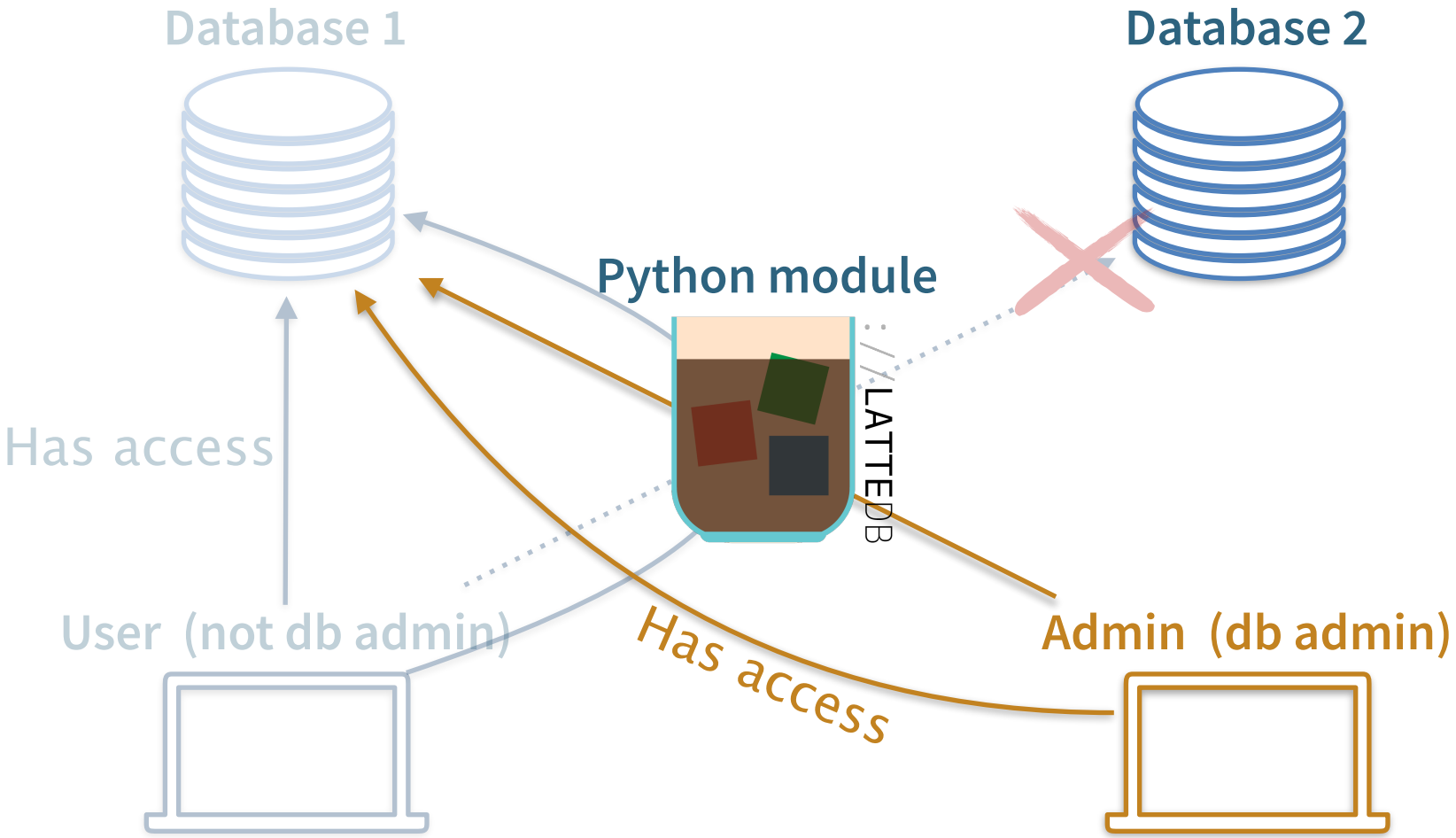
Security Layers (Python Module)



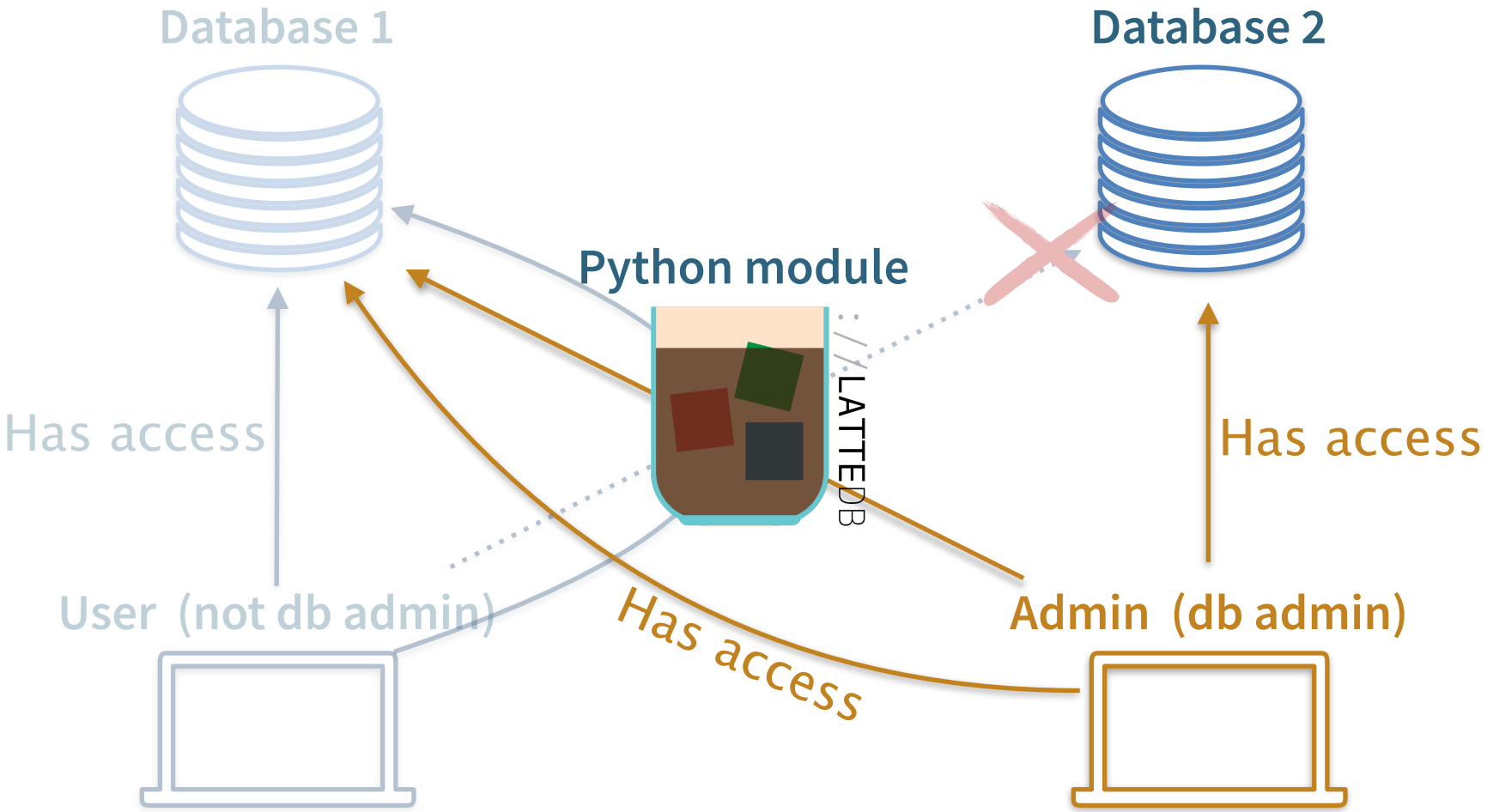
Security Layers (Python Module)



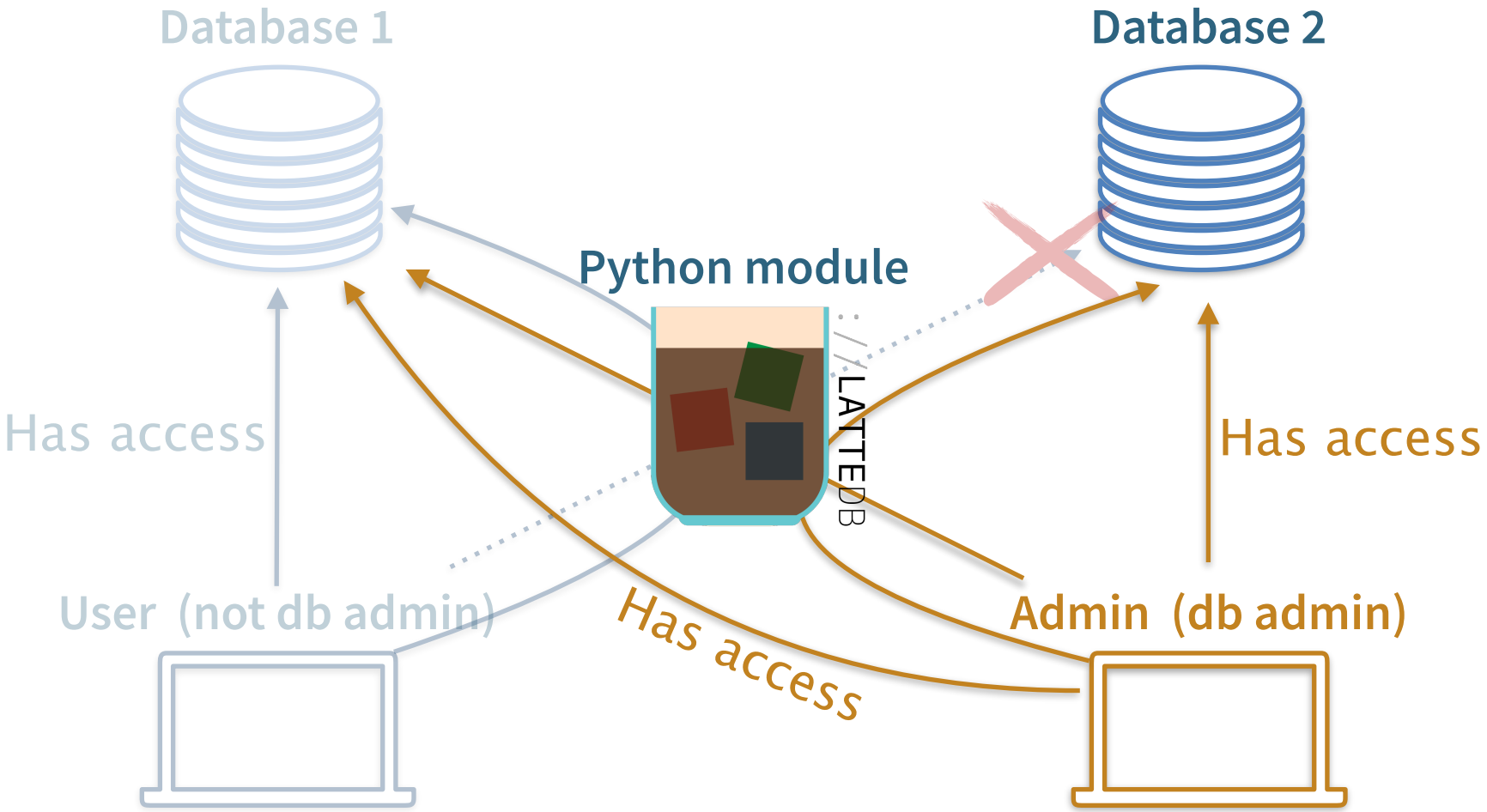
Security Layers (Python Module)



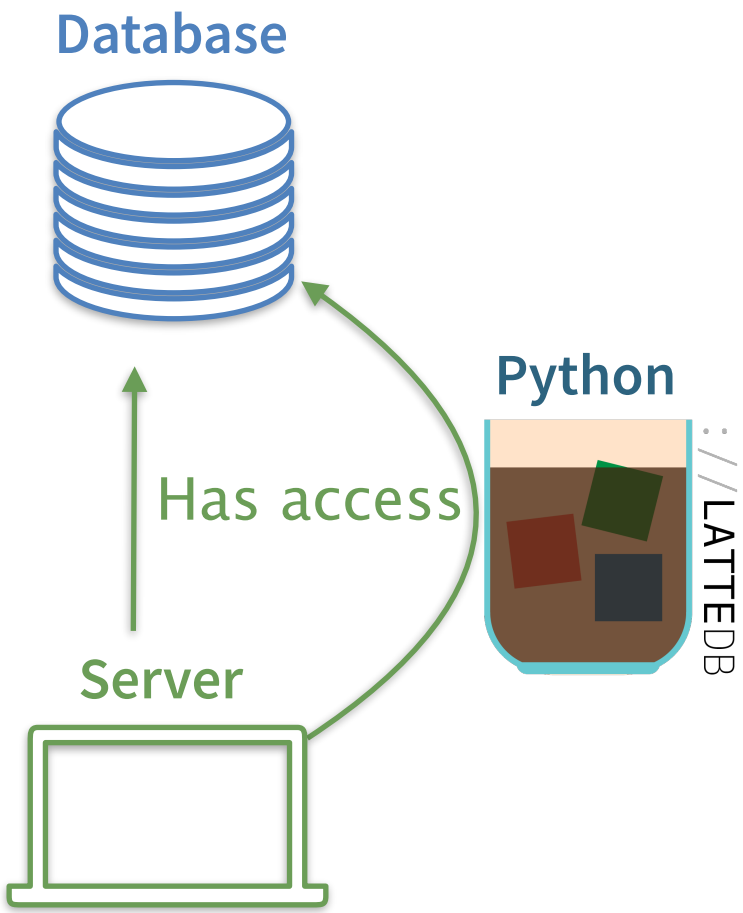
Security Layers (Python Module)



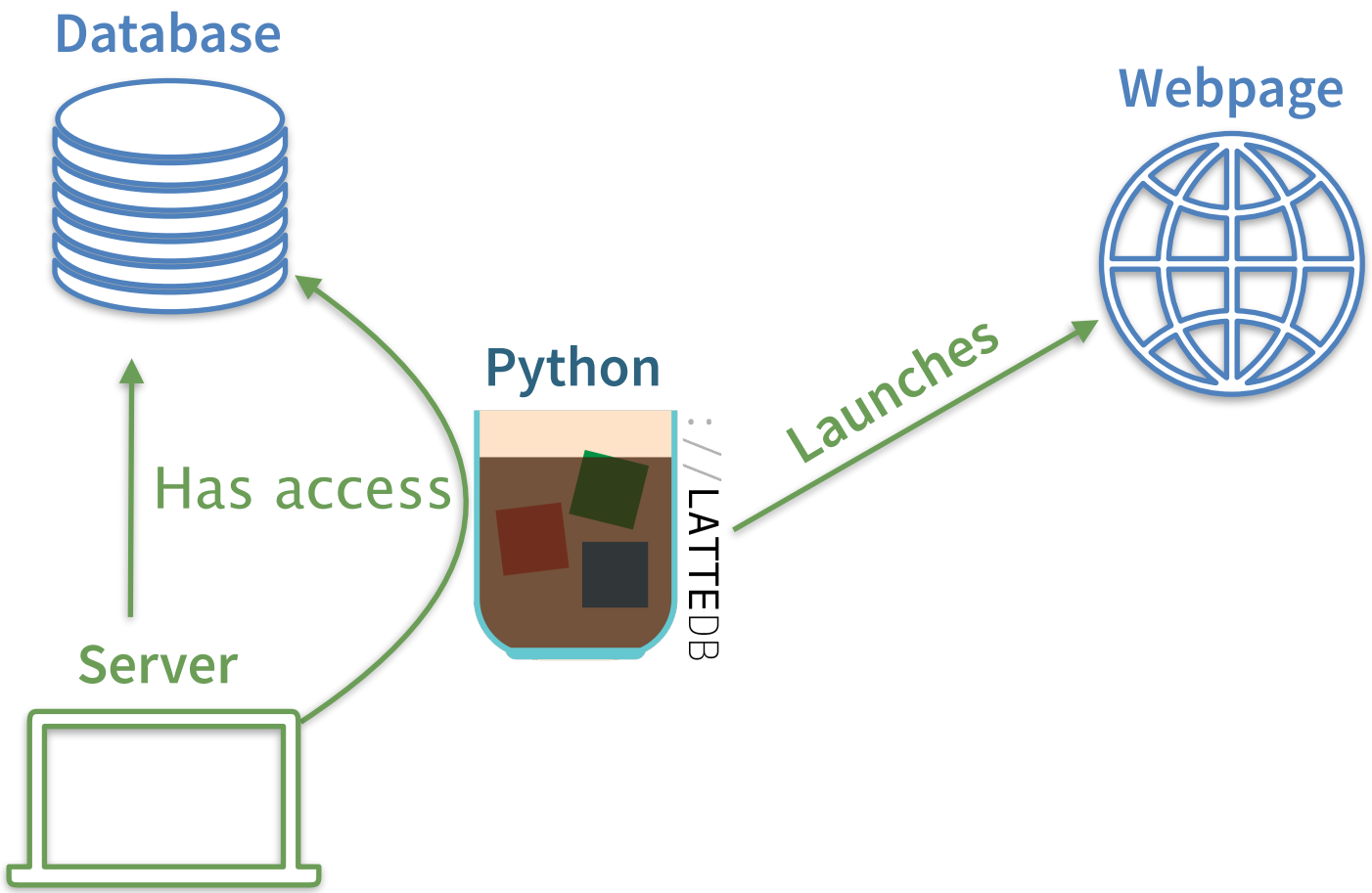
Security Layers (Python Module)



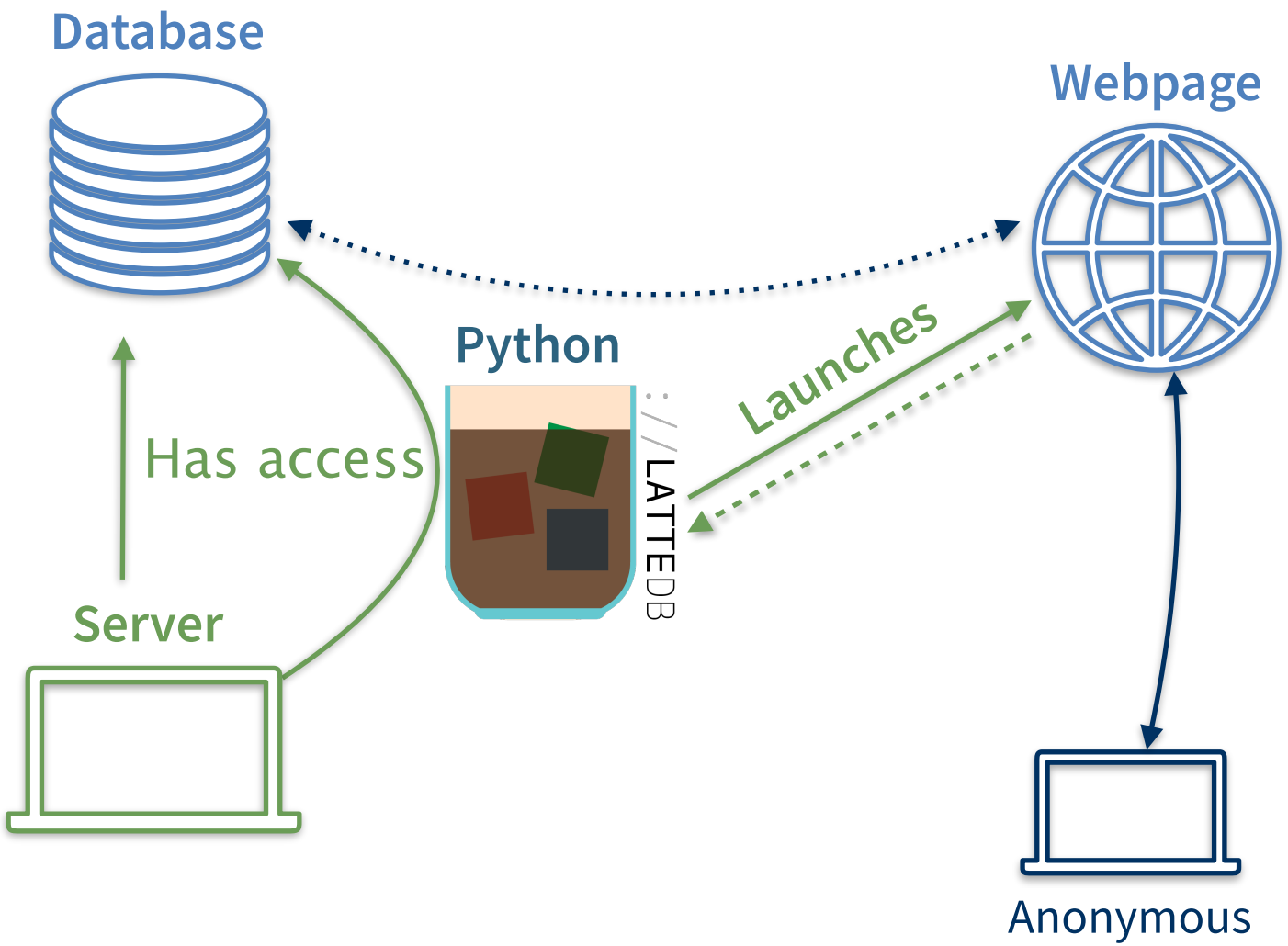
Security Layers (Webpage)



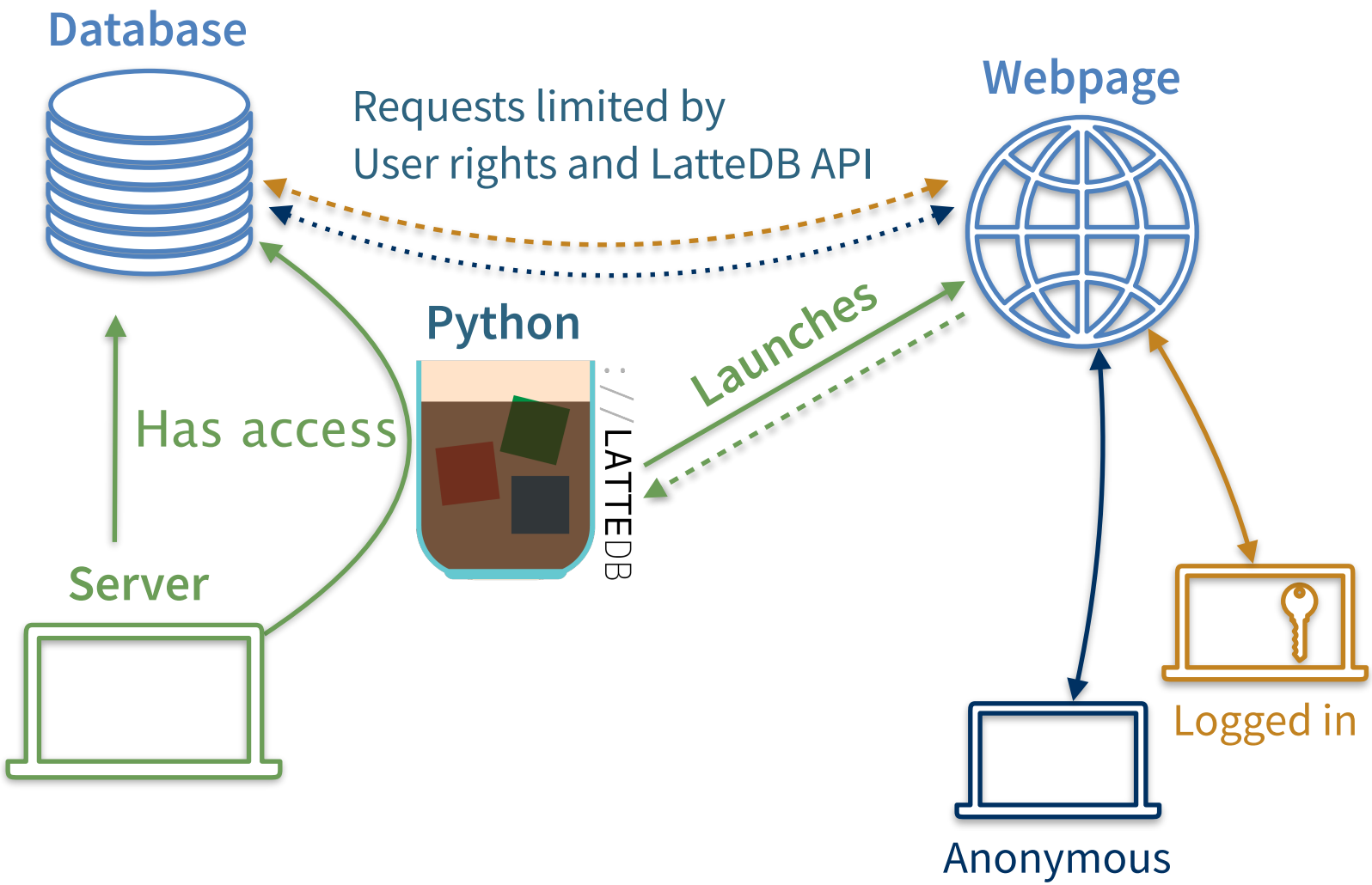
Security Layers (Webpage)



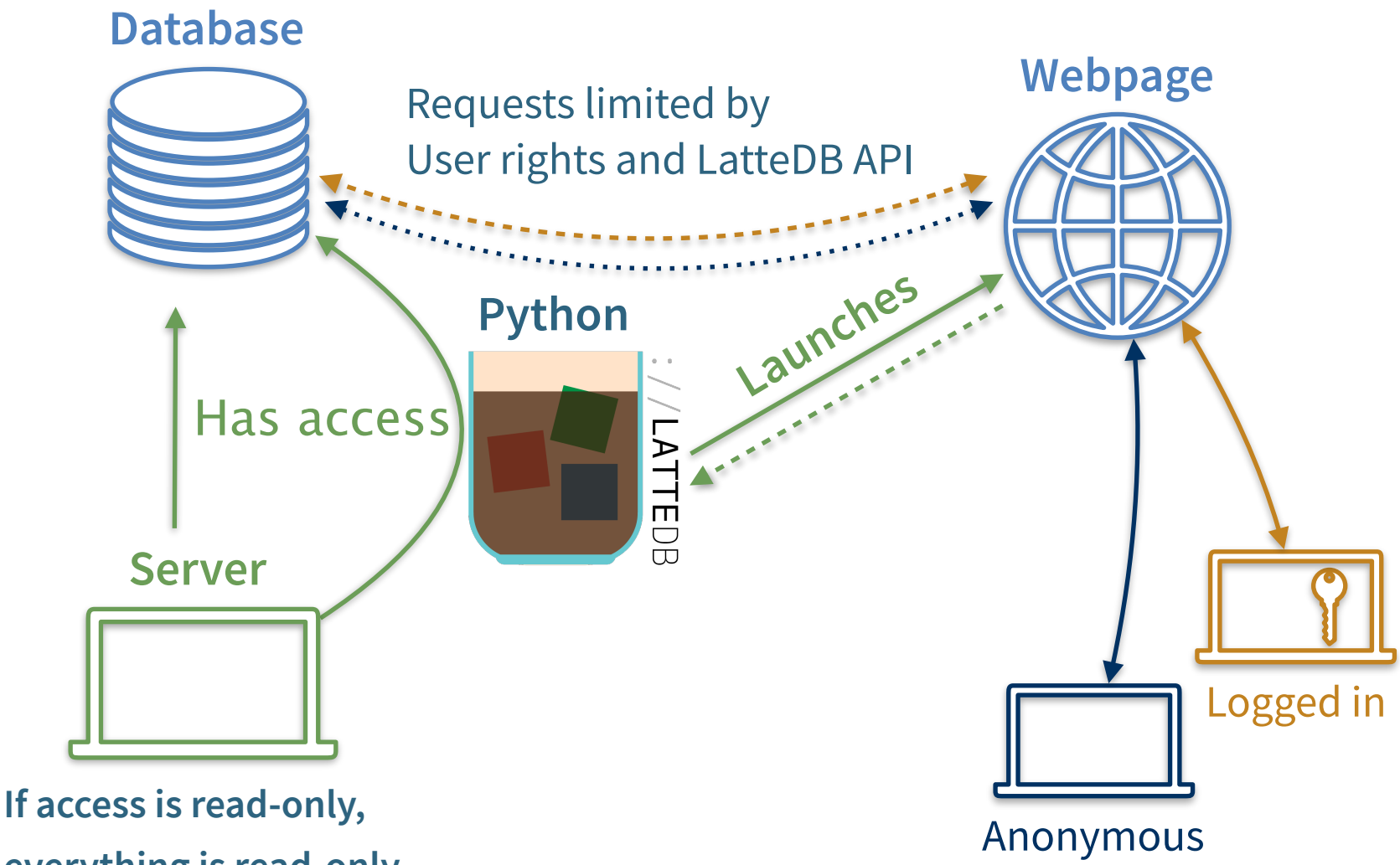
Security Layers (Webpage)



Security Layers (Webpage)



Security Layers (Webpage)



Future

- **Publish Lattice App/Tables**
  - Implement consistency checks for insertions
  - Test job management production on tiny example
  - Production runs: analysis & job management
  - Data connections (on remote machines)
  - Continuous Integration:
    - Interface enhancements
    - Table additions

Future

- **Publish Lattice App/Tables**
  - Implement consistency checks for insertions
  - Test job management production on tiny example
  - Production runs: analysis & job management
  - Data connections (on remote machines)
  - Continuous Integration:
    - Interface enhancements
    - Table additions
- **Extract & Publish Stand-Alone Mini-App**
  - Contains general framework
  - Documentation and usage guide
  - Unittests & module dock

Backup

Table Layout