Migrating a packaged database schema

Iwan Vosloo

Code & databases

```
class CObject(Base):
    __tablename__ = 'cobject'

    keyc = Column(Integer, primary_key=True)
    another_c_key = Column(Integer, ForeignKey('cobject.keyc'))
    another_c = relationship(CObject)
```

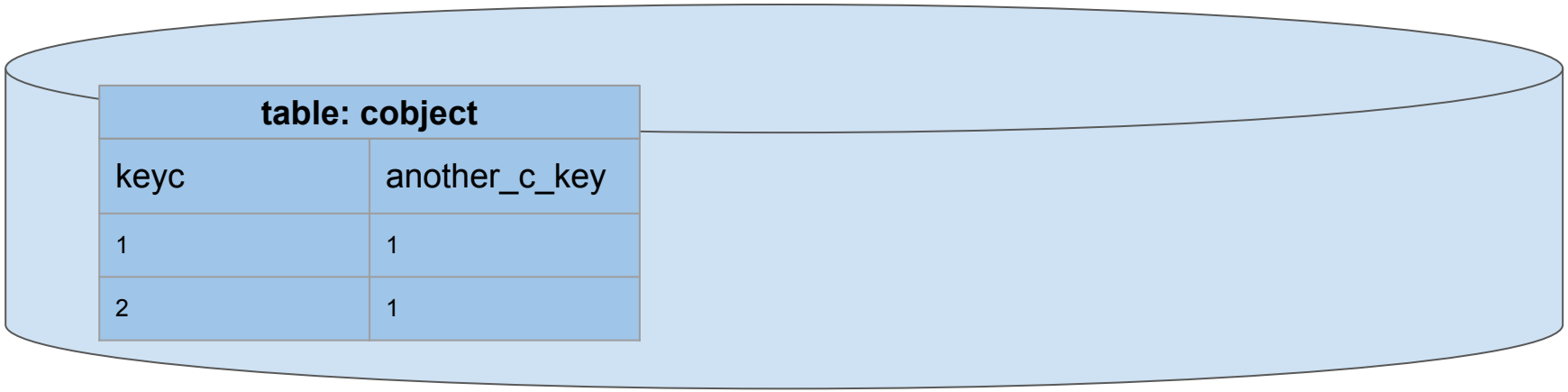


table: cobject	
keyc	another_c_key
1	1
2	1

Database schema migration

```
CREATE TABLE myobject (
    key integer NOT NULL,
    to_c_key integer
);

ALTER TABLE myobject
    ADD CONSTRAINT pk_myobject
    PRIMARY KEY (key);

ALTER TABLE myobject
    ADD CONSTRAINT
        fk_myobject_to_c_key_cobject
        FOREIGN KEY (to_c_key)
        REFERENCES cobject(~~keye~~);
```

```
CREATE TABLE myobject (
    key integer NOT NULL,
    to_c_key integer,
    email text
);

ALTER TABLE myobject
    ADD CONSTRAINT pk_myobject
        PRIMARY KEY (key);

ALTER TABLE myobject
    ADD CONSTRAINT
        fk_myobject_to_c_key_cobject
        FOREIGN KEY (to_c_key)
        REFERENCES cobject(keyc3);
```

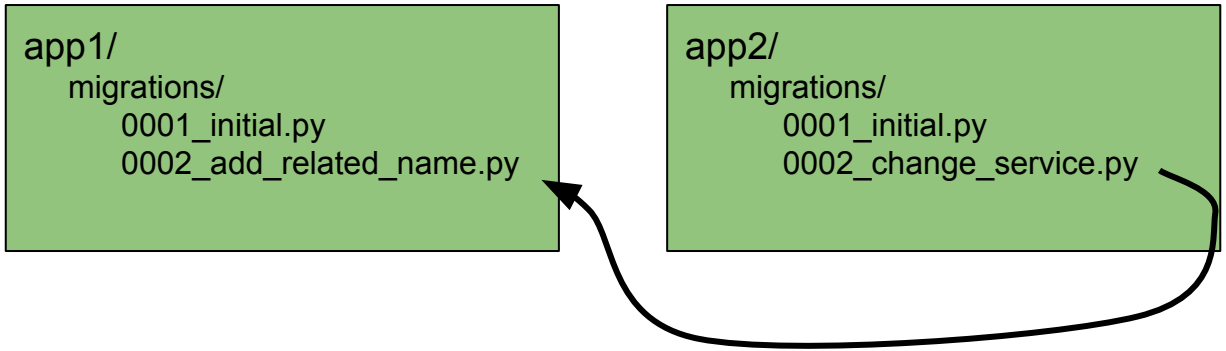
SqlAlchemy Alembic

```
yourproject/
    alembic/
        env.py
        README
        script.py.mako
        versions/
            3512b954651e_add_account.py
            2b1ae634e5cd_add_order_id.py
            3adcc9a56557_rename_username_field.py
```

```
mydb=> select * from
alembic_version;
 version_num
 ------------
 2b1ae634e5cd

(1 row)
```

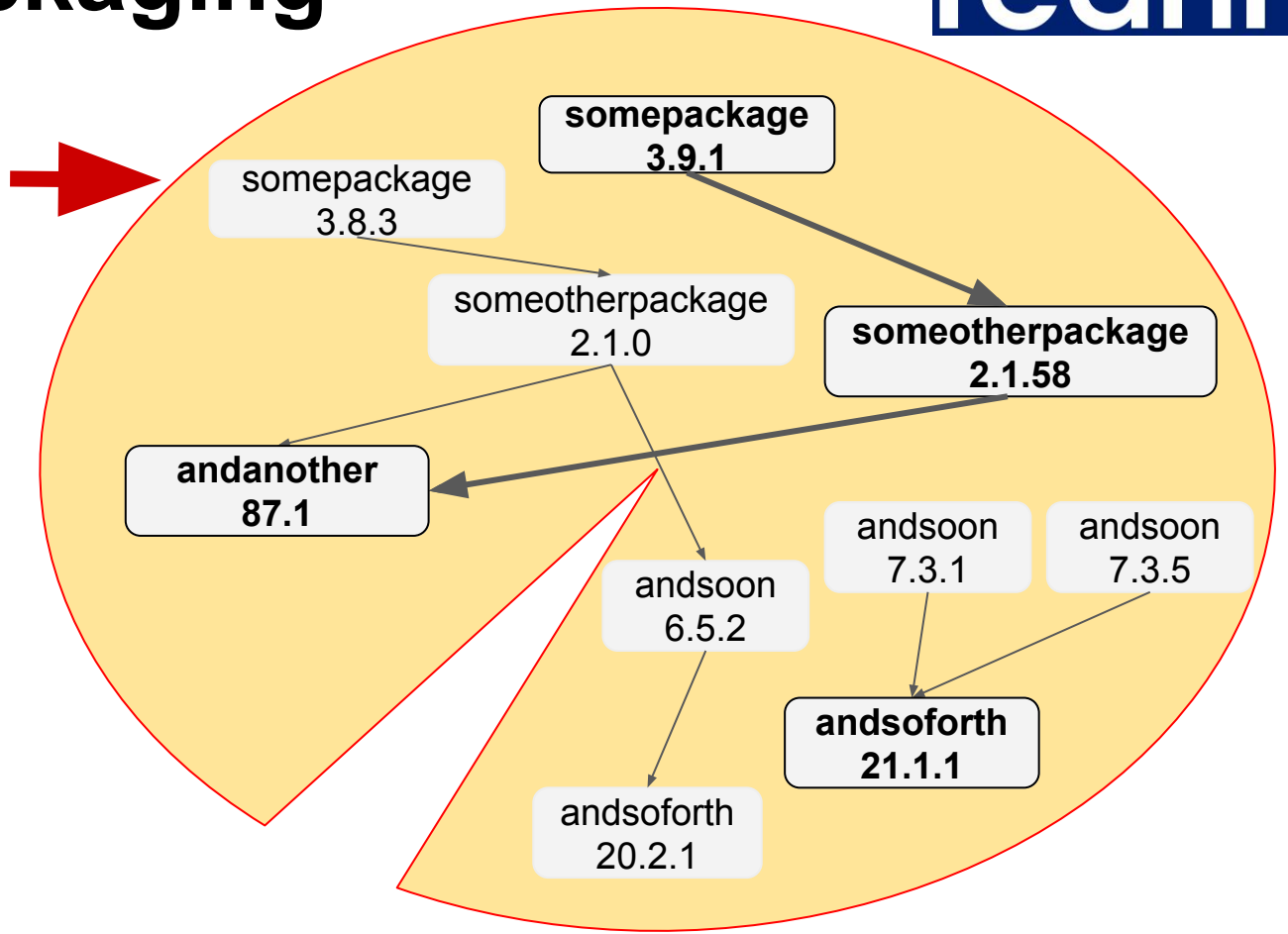
```
                            -- ae1027a6acf -->
                           /                  \
<base> --> 1975ea83b712 -->                    --> mergepoint
                           \                  /
                            -- 27c6a30d7c24 -->
```

Django migration

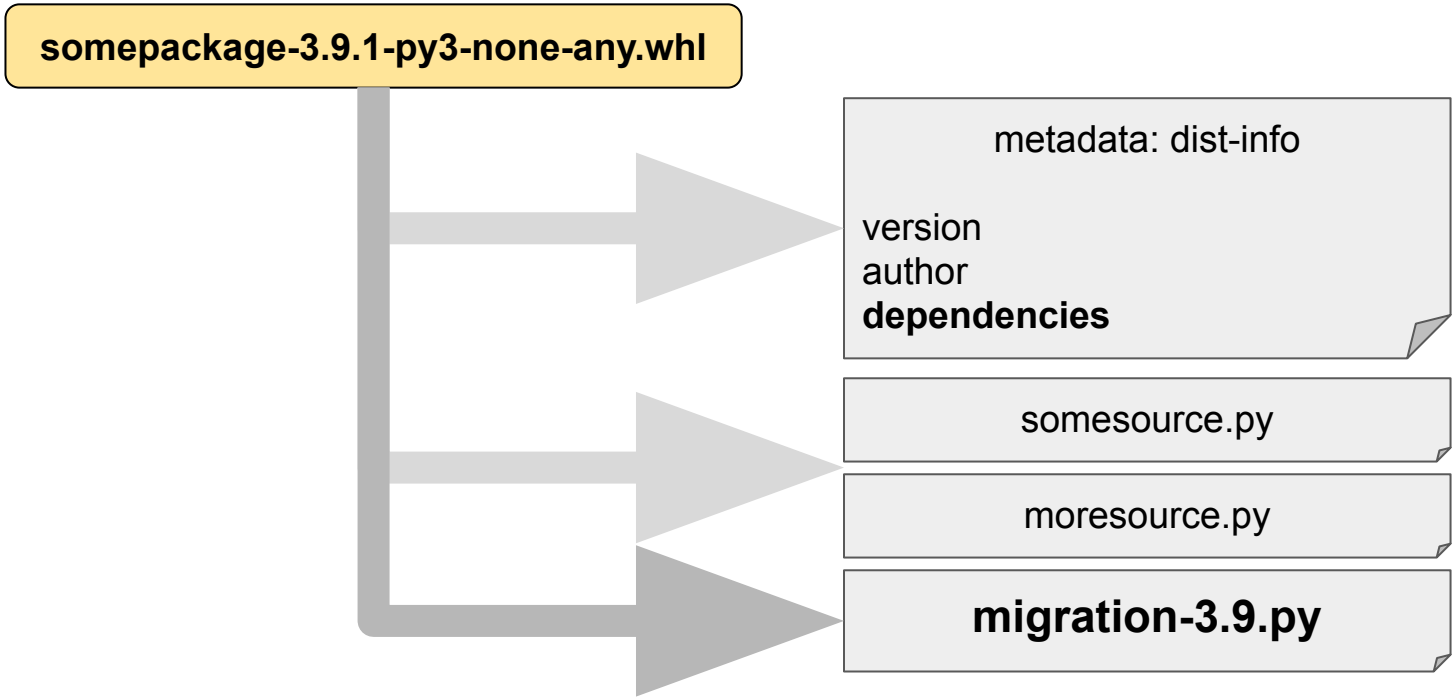


Python packaging

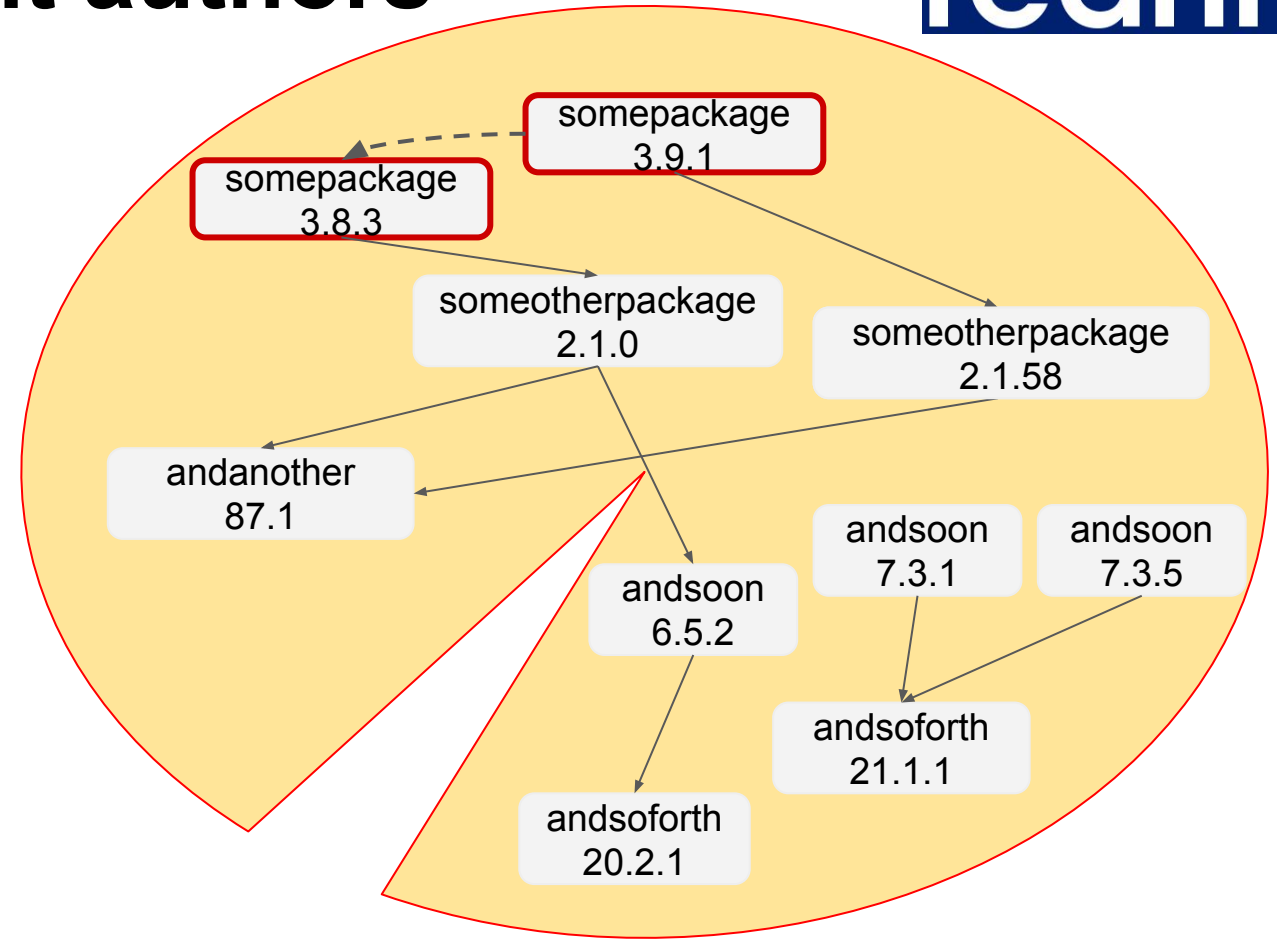
pip install somepackage



What is in a package?



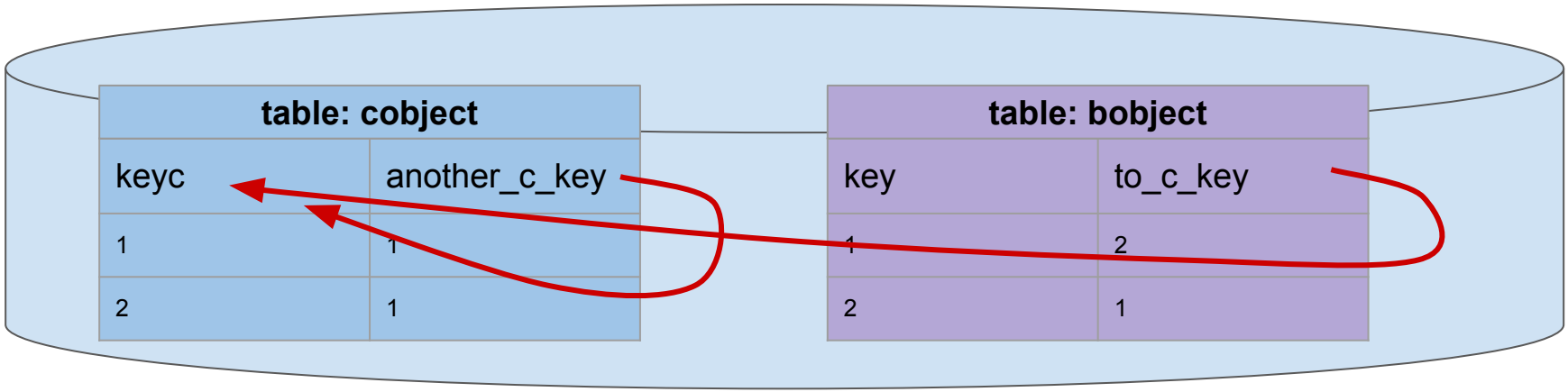
Independent authors



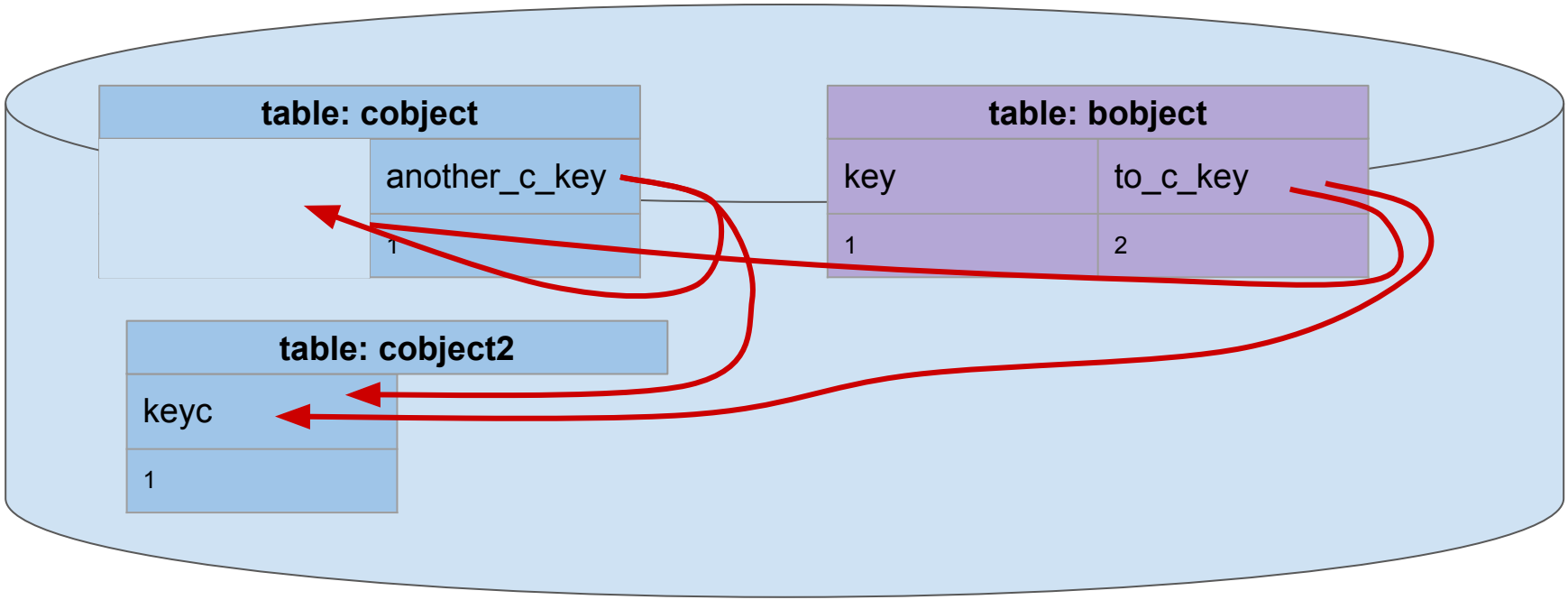
Database-level dependencies

```
class CObject(Base):
    __tablename__ = 'cobject'

    keyc = Column(Integer, primary_key=True)
    another_c_key = Column(Integer, ForeignKe
    another_c = relationship(CObject)
```

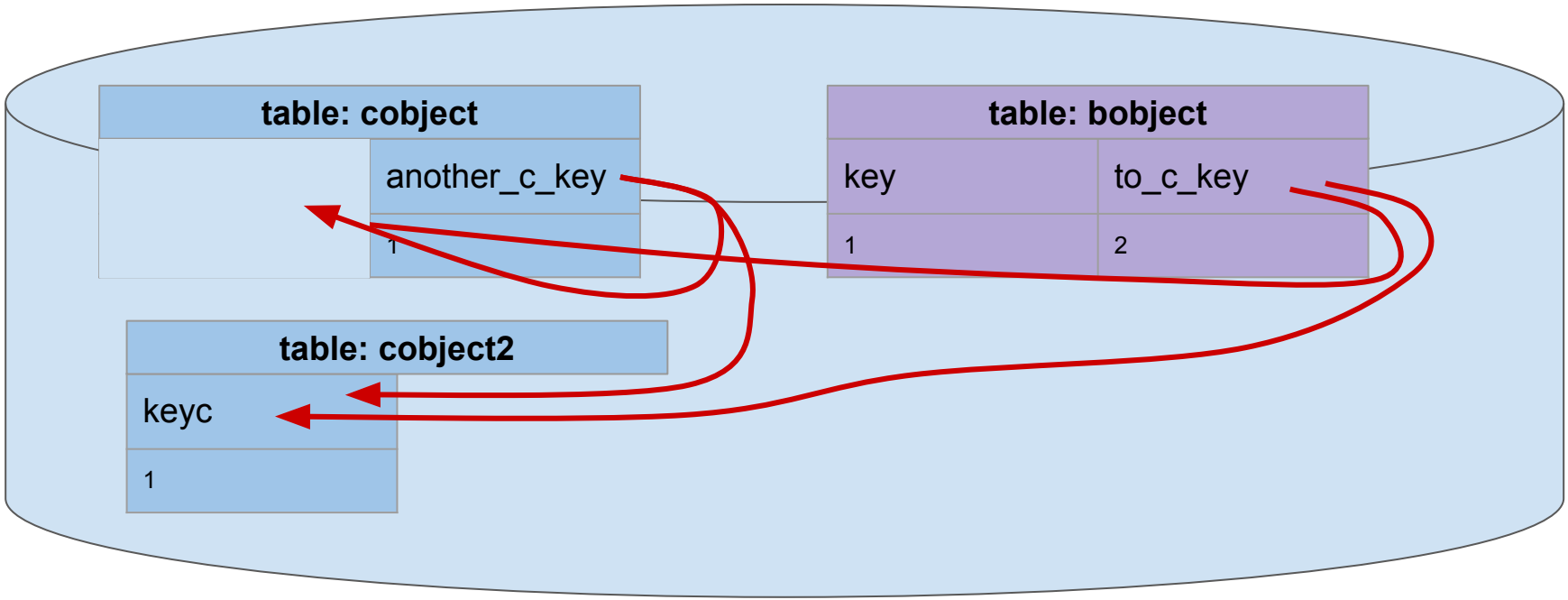
```
class BObject(Base):
    __tablename__ = 'bobject'

    key = Column(Integer, primary_key=True)
    to_c_key = Column(Integer, ForeignKey('cobject.keyc'))
    to_c = relationship(CObject)
```



Database-level changes

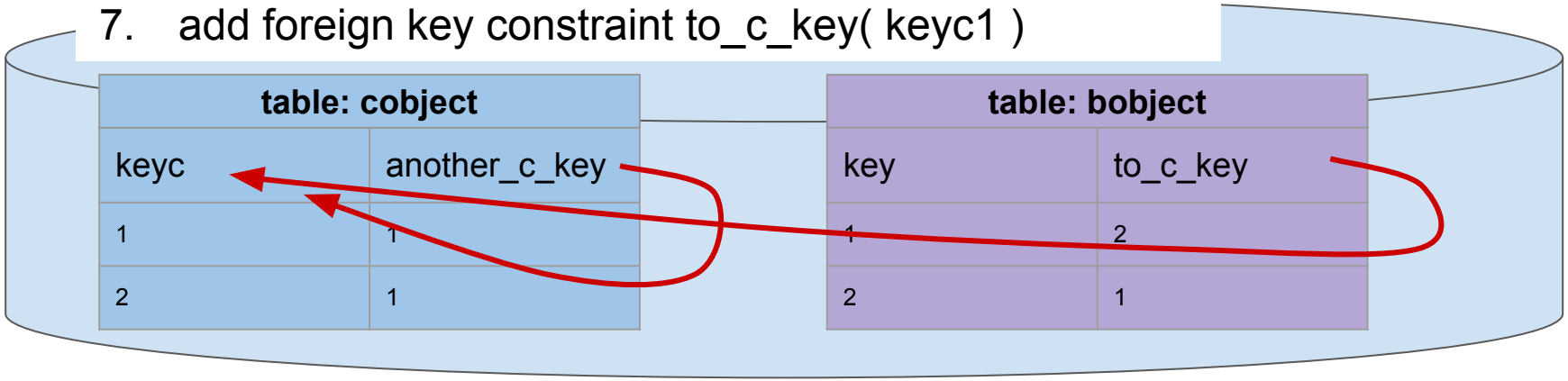


Migration and ordering

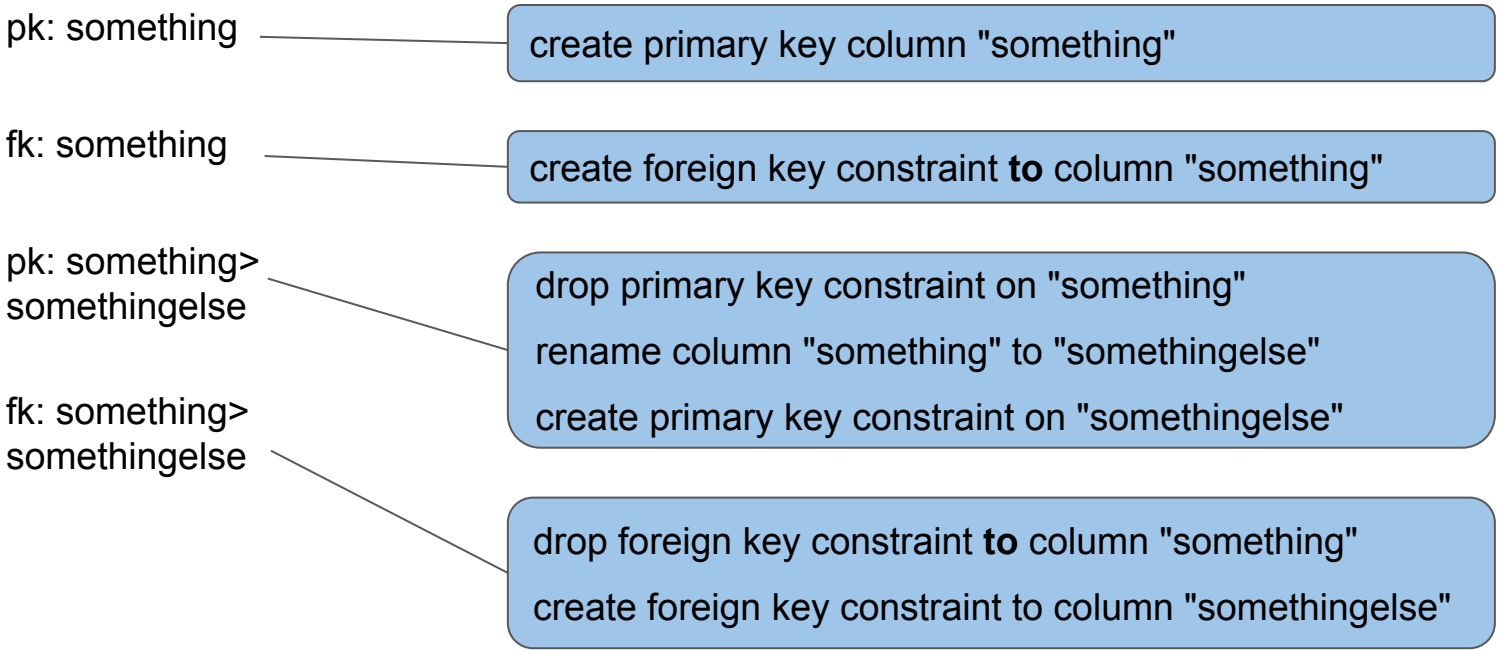


A simplification

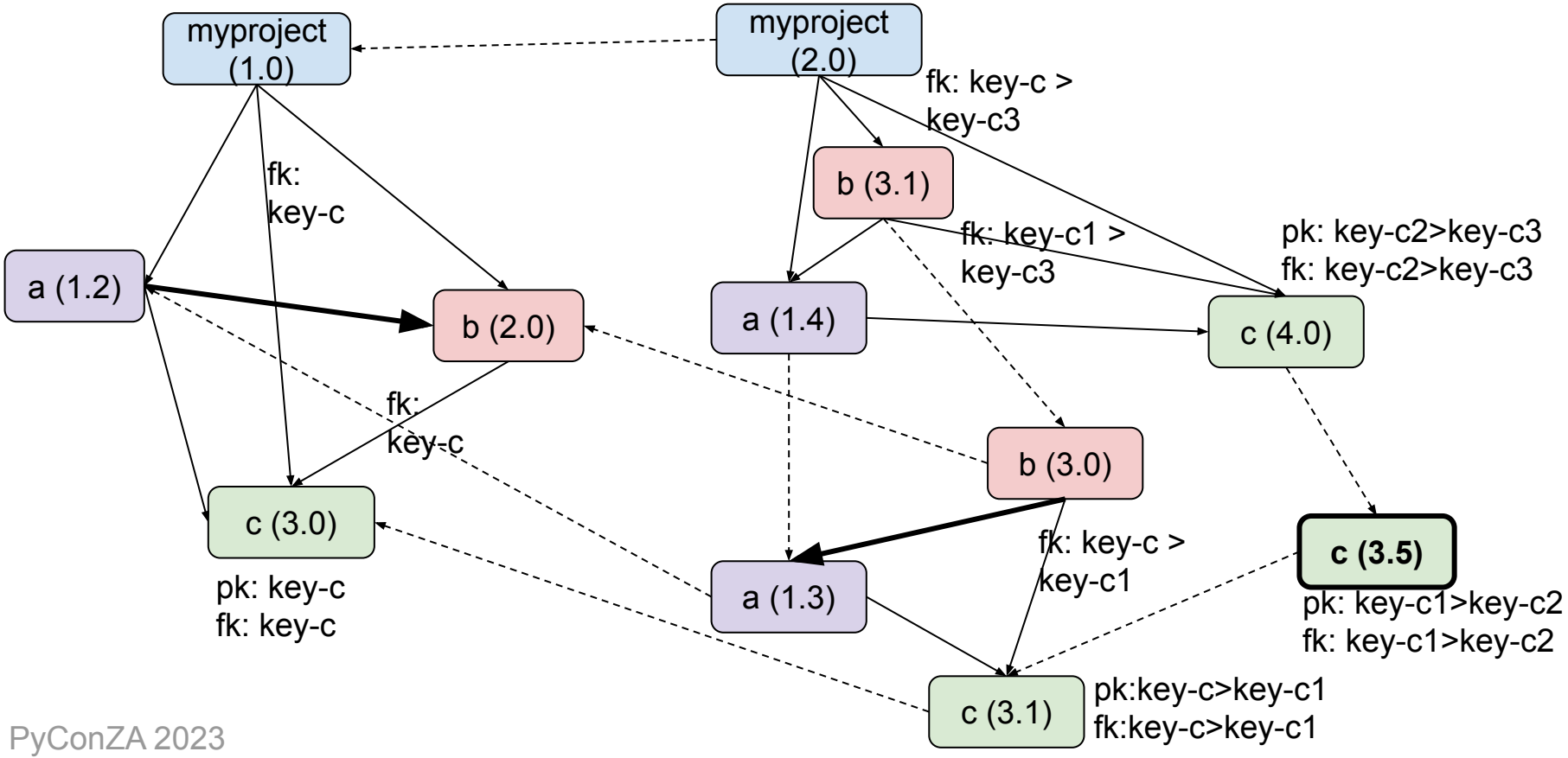
1. remove foreign key constraint to_c_key(keyc)
2. remove foreign key constraint another_c_key(keyc)
3. drop primary key constraint (keyc)
4. rename column (keyc > keyc1)
5. create primary key constraint (keyc1)
6. add foreign key constraint another_c_key(keyc1)
7. add foreign key constraint to_c_key(keyc1)



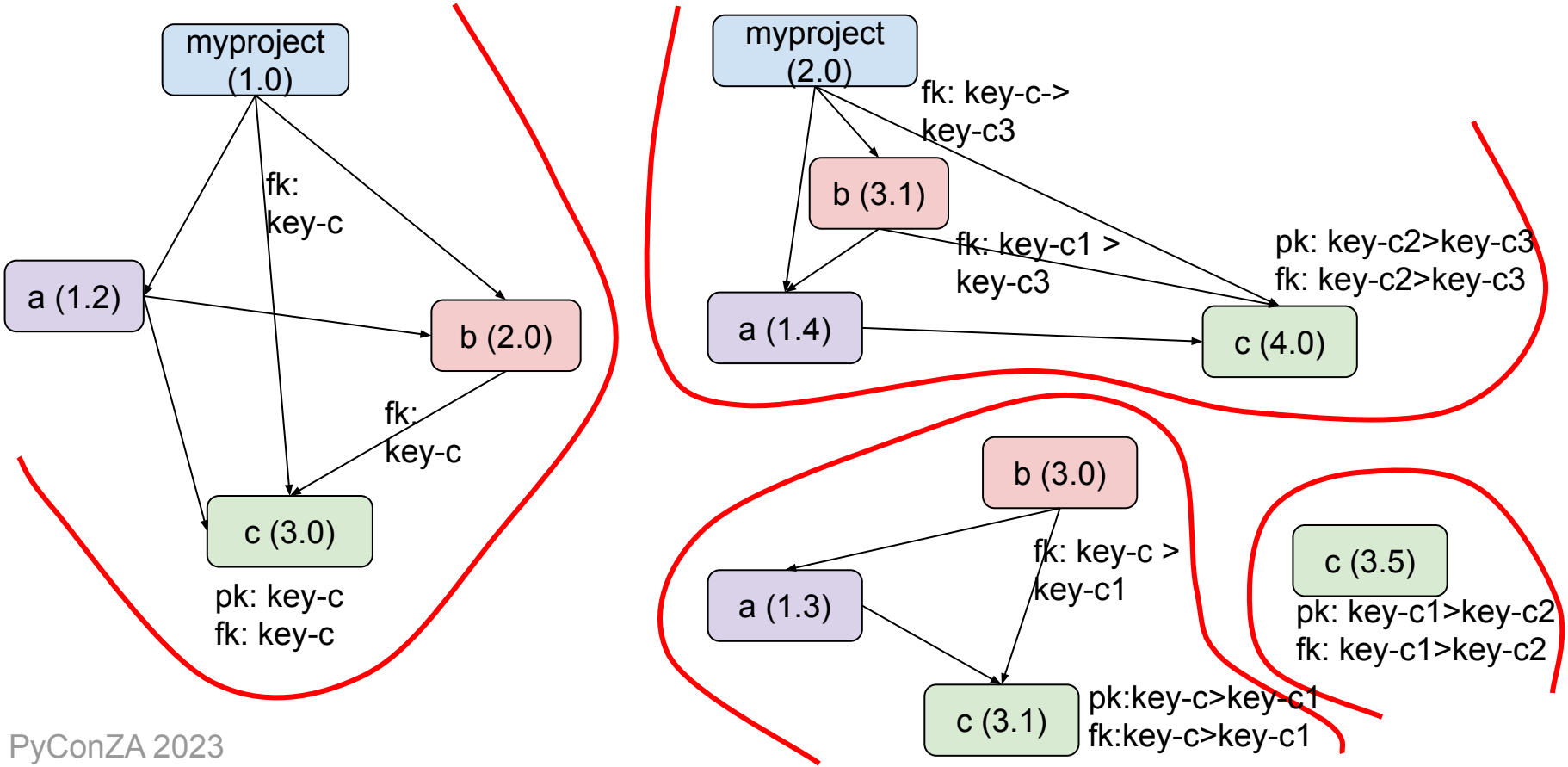
A shorthand


pk: something
create primary key column "something"
fk: something
create foreign key constraint **to** column "something"
pk: something> somethingelse
drop primary key constraint on "something"
rename column "something" to "somethingelse"
create primary key constraint on "somethingelse"
fk: something> somethingelse
drop foreign key constraint **to** column "something"
create foreign key constraint to column "somethingelse"

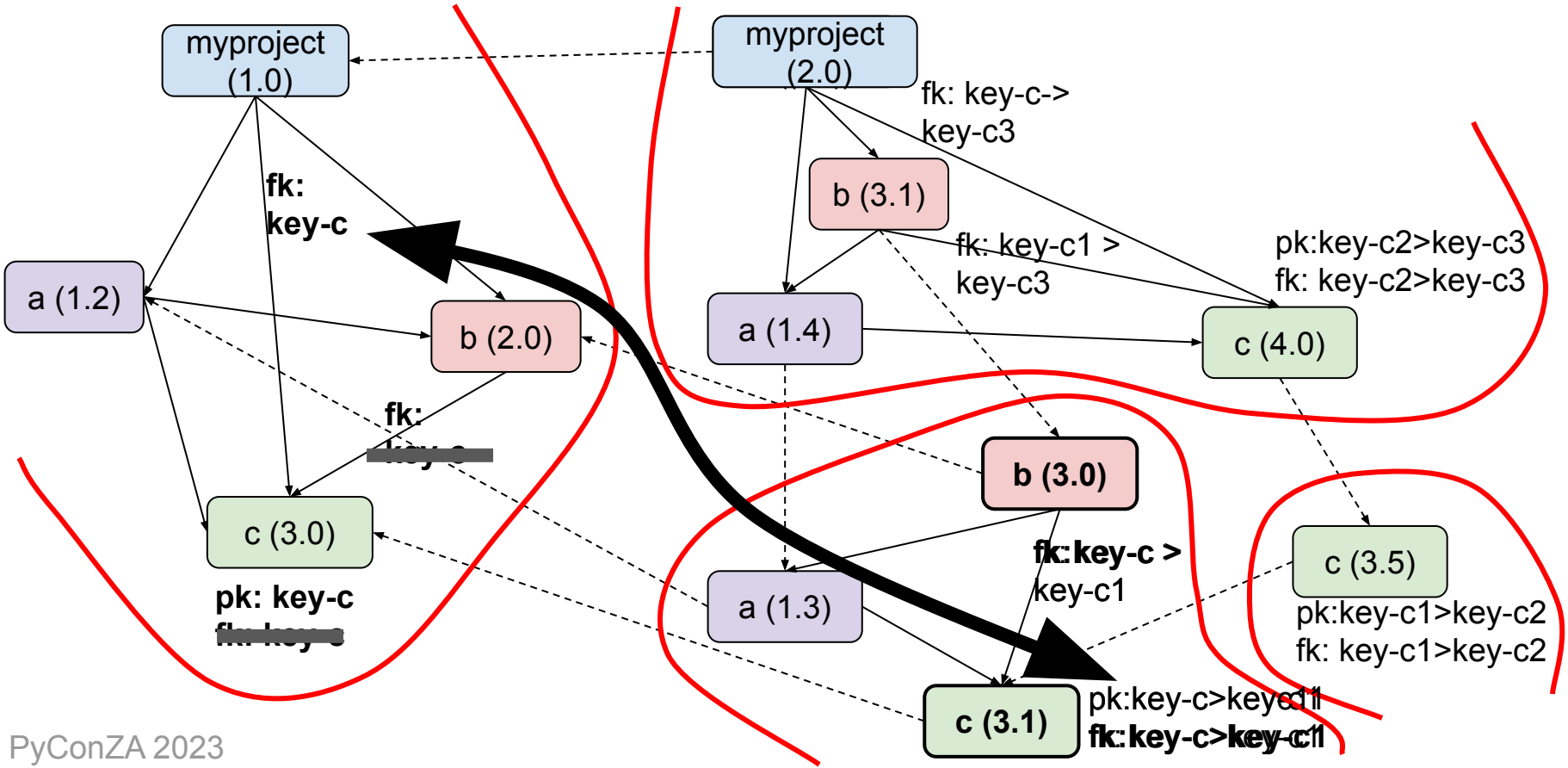
A dependency graph



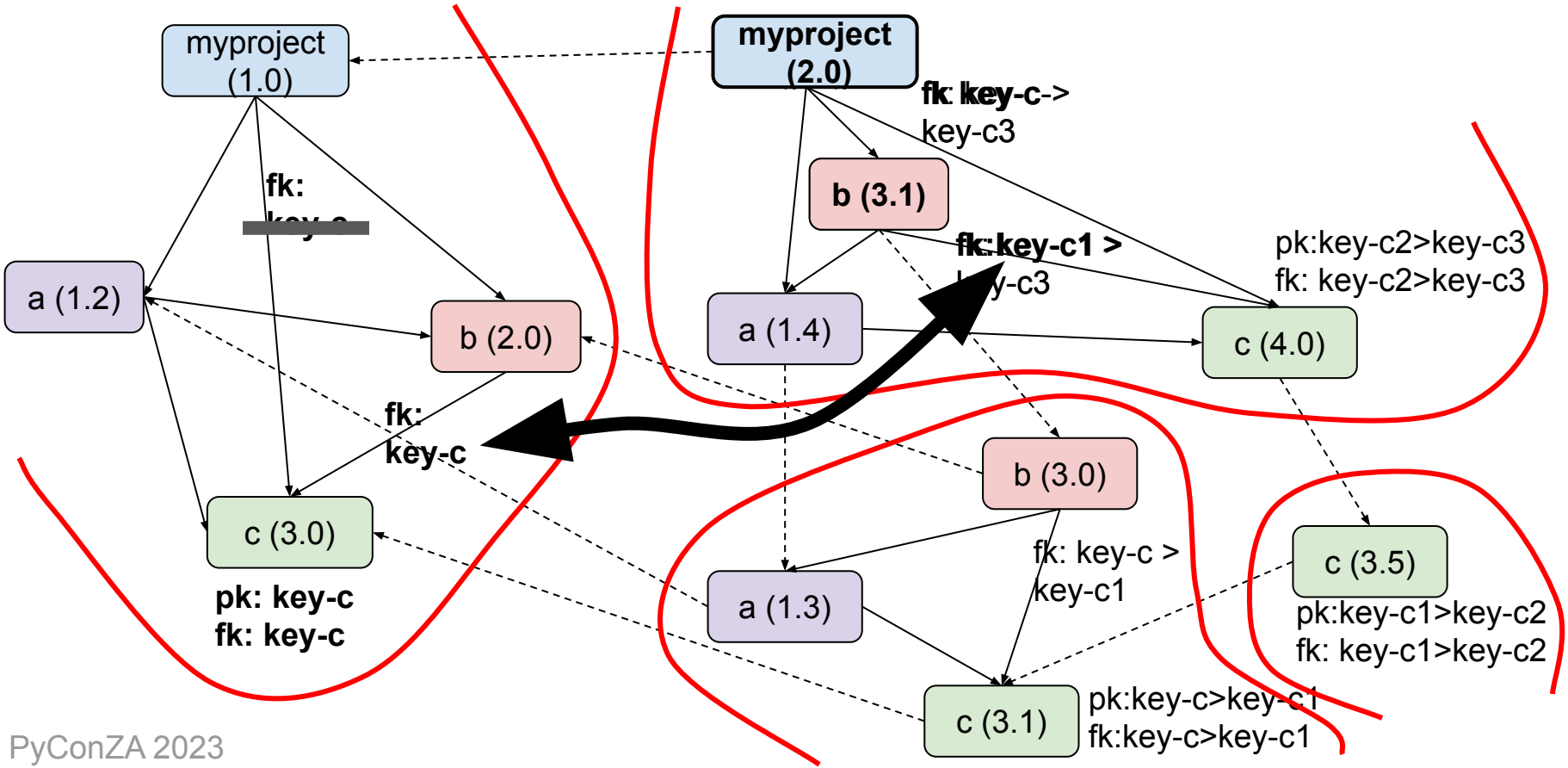
Migration clusters



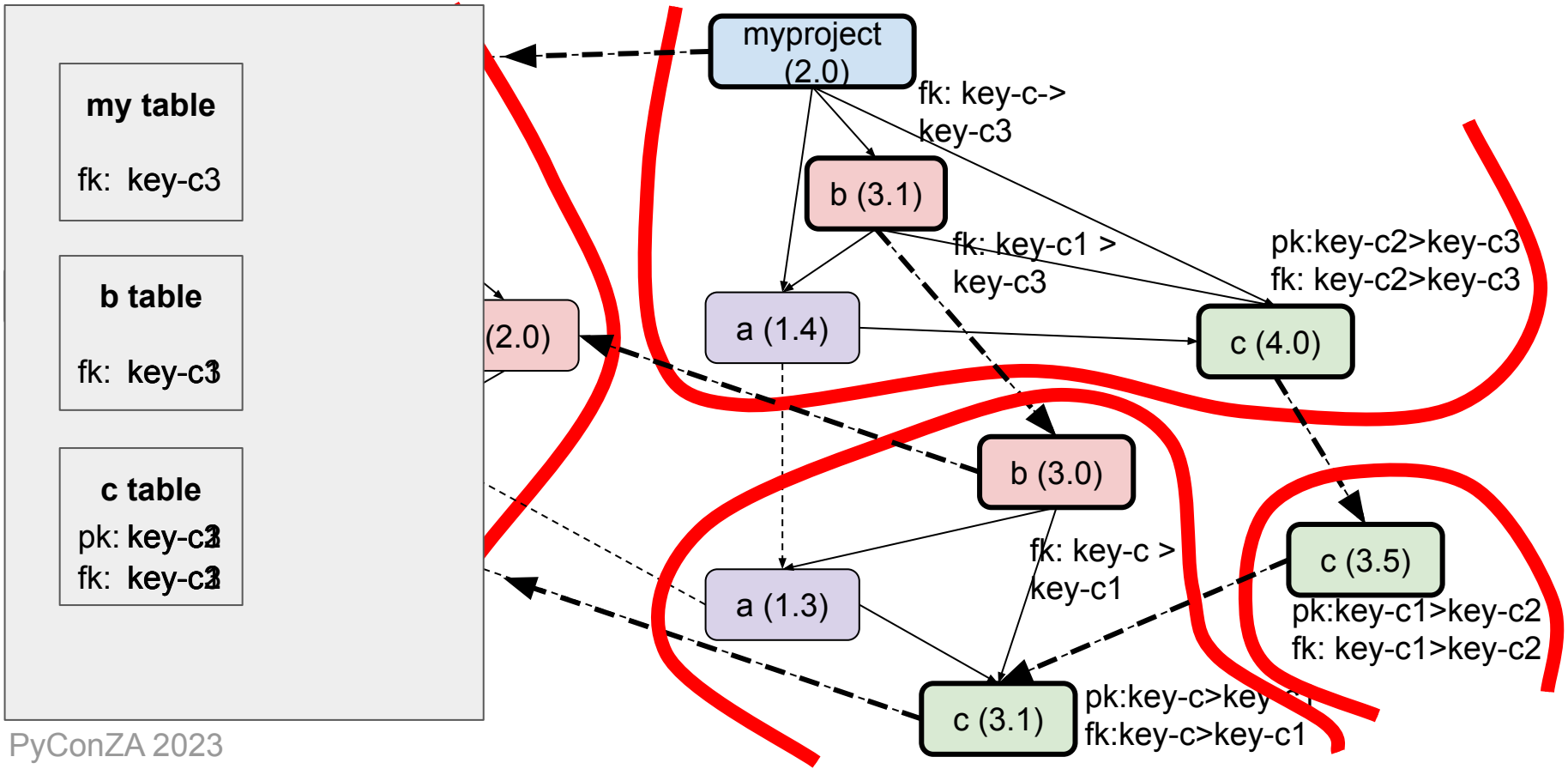
A naïve migration attempt



Another migration attempt



It's not complicated

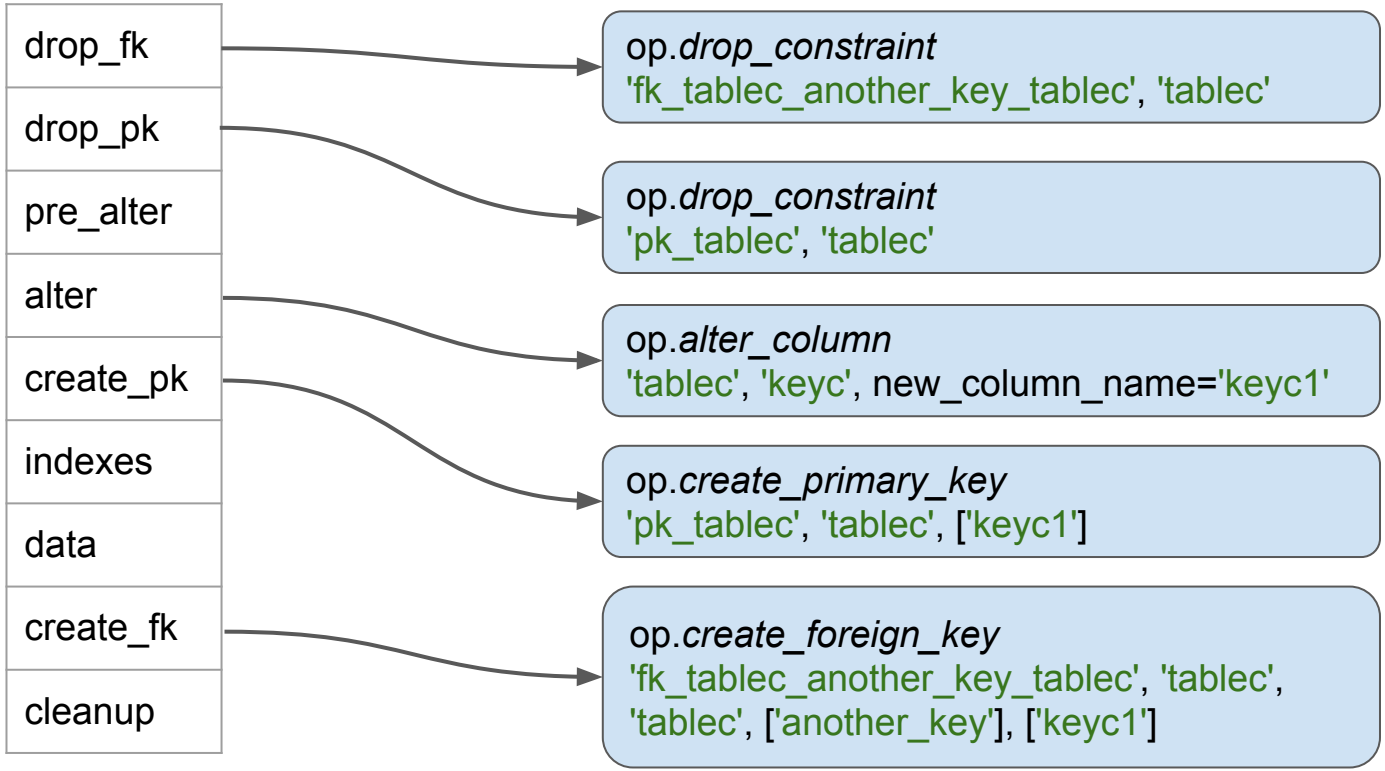


Coding Migrations


```
op.drop_constraint('fk_tablec_another_key_tablec', 'tablec')

op.create_foreign_key('fk_tablec_another_key_tablec', 'tablec', 'tablec', ['another_key'], ['keyc1'])


op.drop_constraint('pk_tablec', 'tablec')

op.alter_column('tablec', 'keyc', new_column_name='keyc1')

op.create_primary_key('pk_tablec', 'tablec', ['keyc1'])
```

SQLAlchemy
alembic

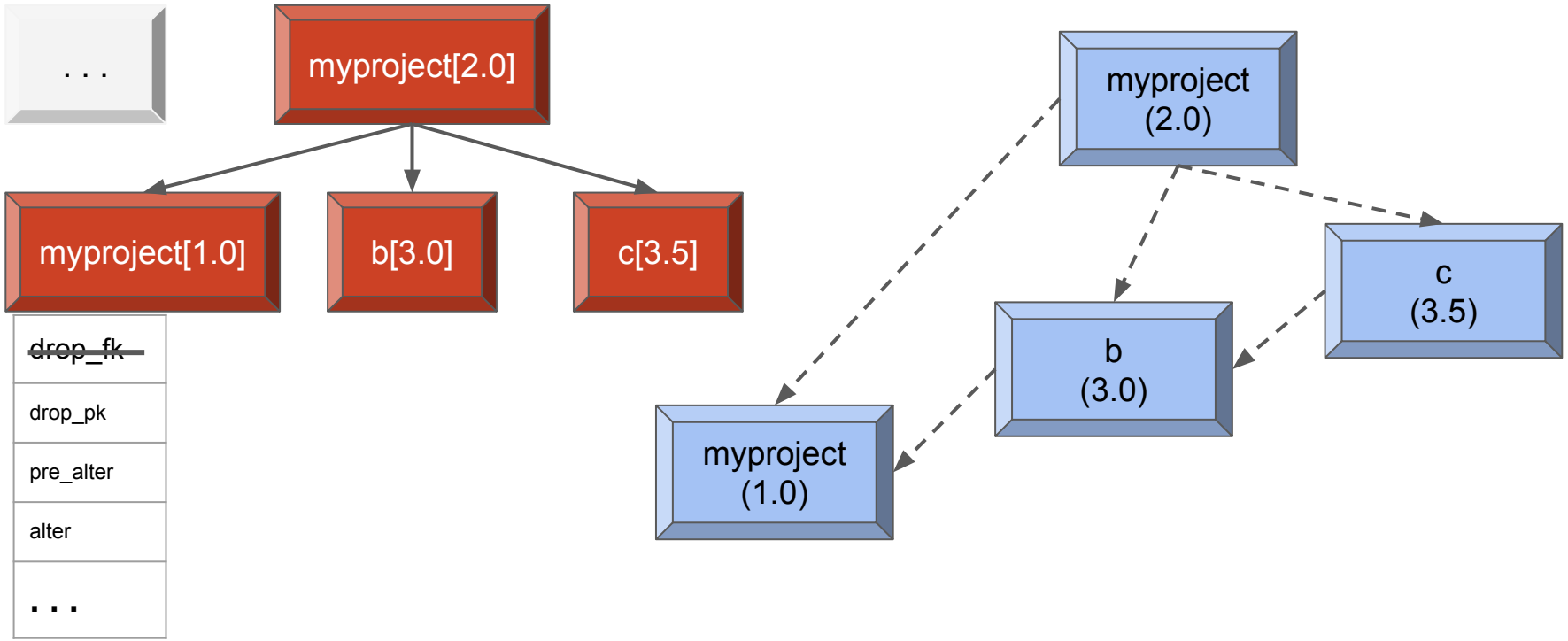
Migration phases



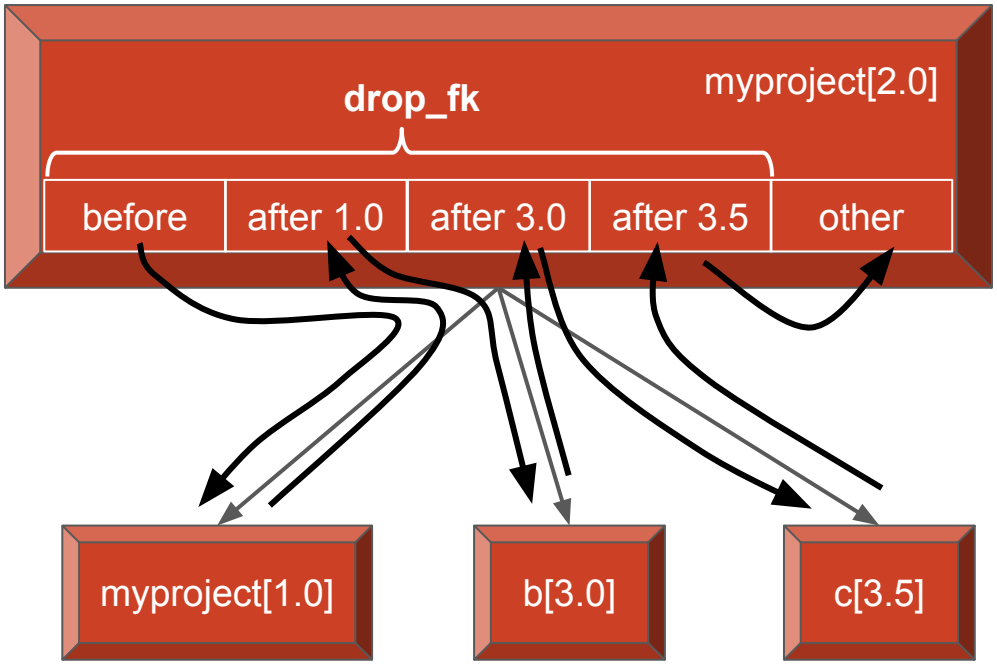
Abstracting Migrations

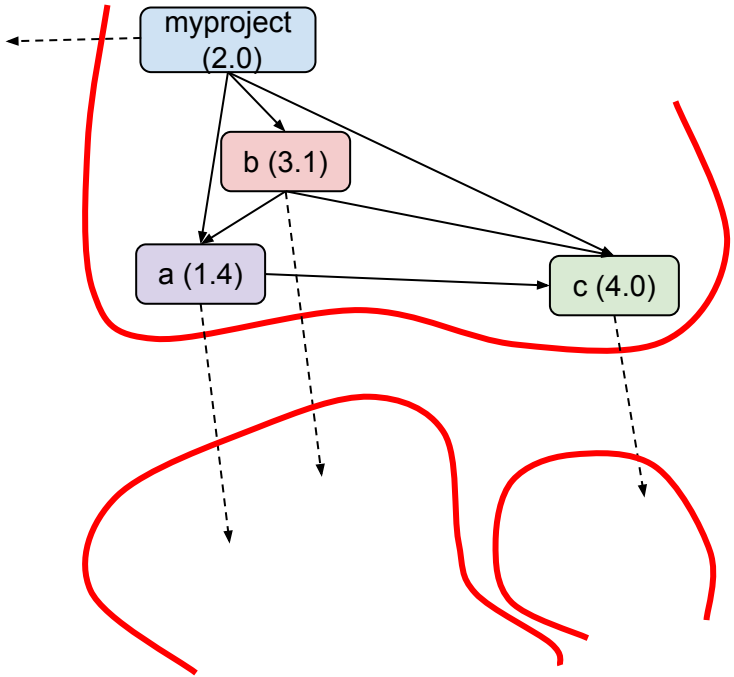

```
class To31(Migration):
    def schedule_upgrades(self):

        self.schedule('drop_fk', op.drop_constraint, 'fk_cobject_another_c_key_cobject', 'tablec')

        self.schedule('create_fk', op.create_foreign_key, 'fk_cobject_another_c_key_cobject',
                      'tablec', 'tablec', ['another_key'], ['keyc1'])


        self.schedule('drop_pk', op.drop_constraint, 'pk_tablec', 'tablec')

        self.schedule('alter', op.alter_column, 'tablec', 'keyc', new_column_name='keyc1')

        self.schedule('create_pk', op.create_primary_key, 'pk_tablec', 'tablec', ['keyc1'])
```

Migration Schedules



Migration Schedules





Writing Migrations


```
class ToNewC31(Migration):
    def schedule_upgrades(self):

        self.schedule('drop_fk', op.drop_constraint, 'fk_bobject_to_c_key_cobject', 'bobject')

        self.schedule('create_fk', op.create_foreign_key, 'fk_bobject_to_c_key_cobject',
                      'bobject', 'cobject', ['to_c_key'], ['keyc1'])
```

Specifying meta information

reahl™

pyproject.toml

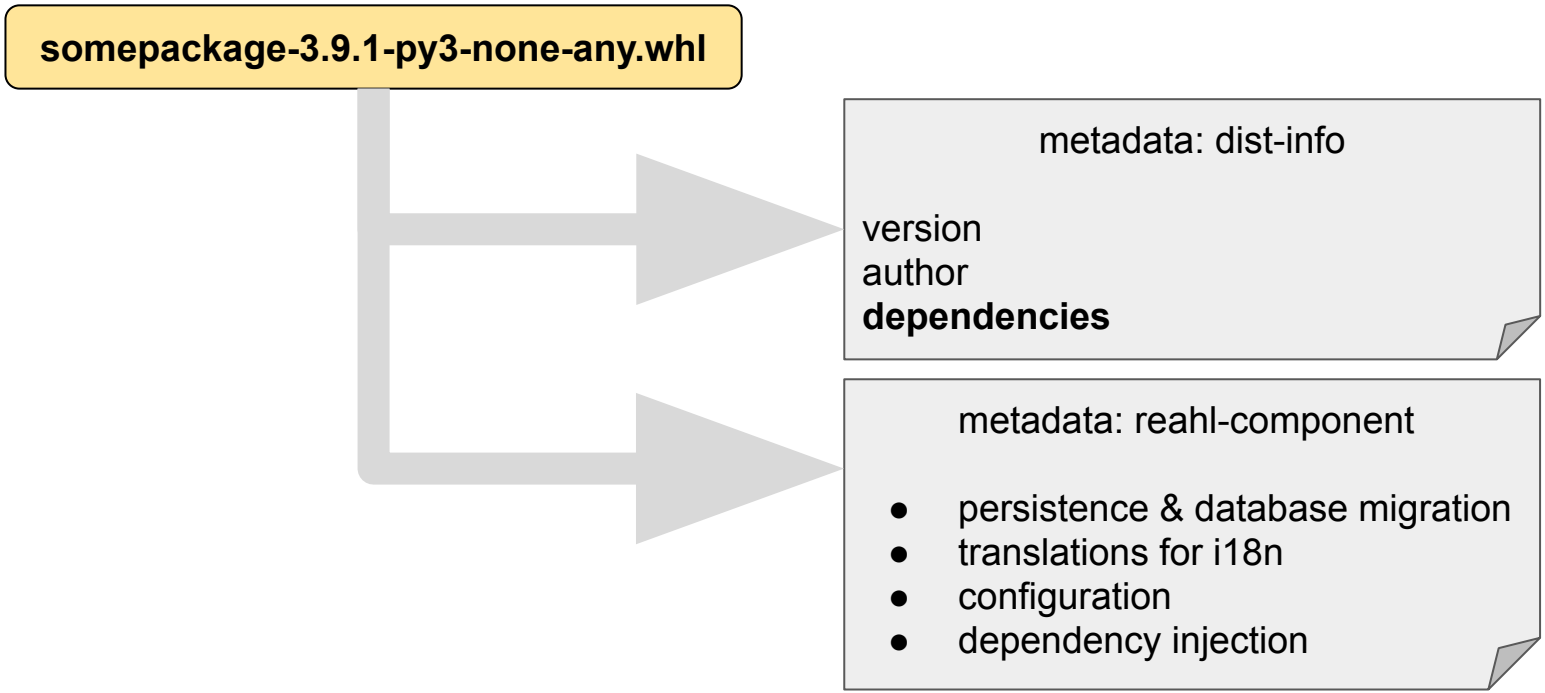
```
[build-system]
 requires = [
   "setuptools >= 68",
   "reahl-component-metadata >= 7.0.0"
 ]
 build-backend = "setuptools.build_meta"

[project]
 name = "myproject"
 version = "2.0"
 dependencies = [
   "project-b>=3.1,<3.2",
   "project-a>=1.4,<1.5",
   "project-c>=4.0,<4.1",
   "reahl-sqlalchemysupport>=7.0,<7.1"
 ]
```

Per-version metadata

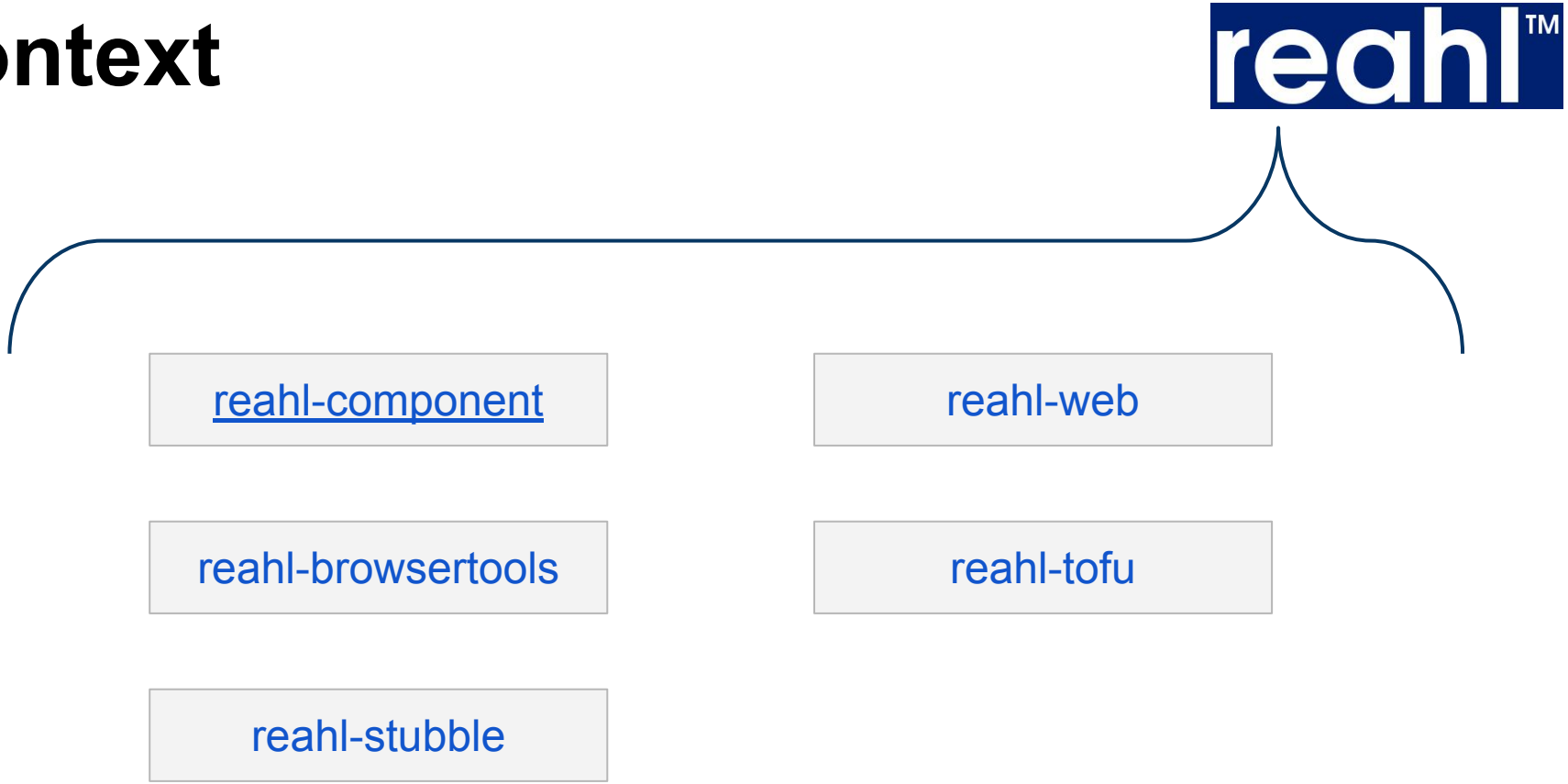
pyproject.toml

```
[tool.reahl-component]

[tool.reahl-component.versions.'1.0']
dependencies = [
  "project-b>=2.0,<2.1",
  "project-a>=1.2,<1.3",
  "project-c>=3.0,<3.1",
  "reahl-sqlalchemysupport>=7.0,<7.1"
]
migrations = [
  "myproject.migrations:CreateDB"
]

[tool.reahl-component.versions.'2.0']
migrations = [
  "myproject.migrations:ToNewC"
]
```

There's more…


somepackage-3.9.1-py3-none-any.whl
metadata: dist-info
version
author
dependencies
metadata: reahl-component
persistence & database migration
translations for i18n
configuration
dependency injection

Context


reahl™
reahl-component
reahl-web
reahl-browsertools
reahl-tofu
reahl-stubble

Questions?

pip install
'reahl-component>=7.0' 'reahl-postgresqlsupport' 'reahl-sqlalchemysupport'

https://github.com/IwanVosloo/migrationplanningexample

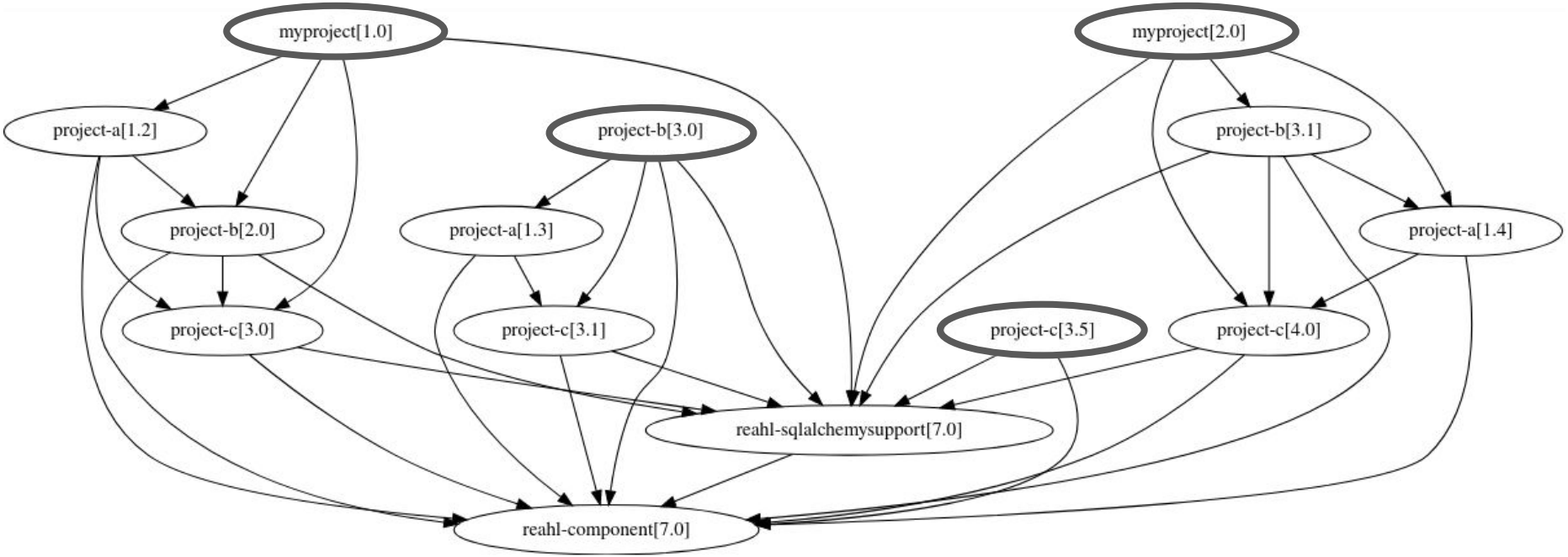
https://reahl.org/pycon2023

www.reahl.org

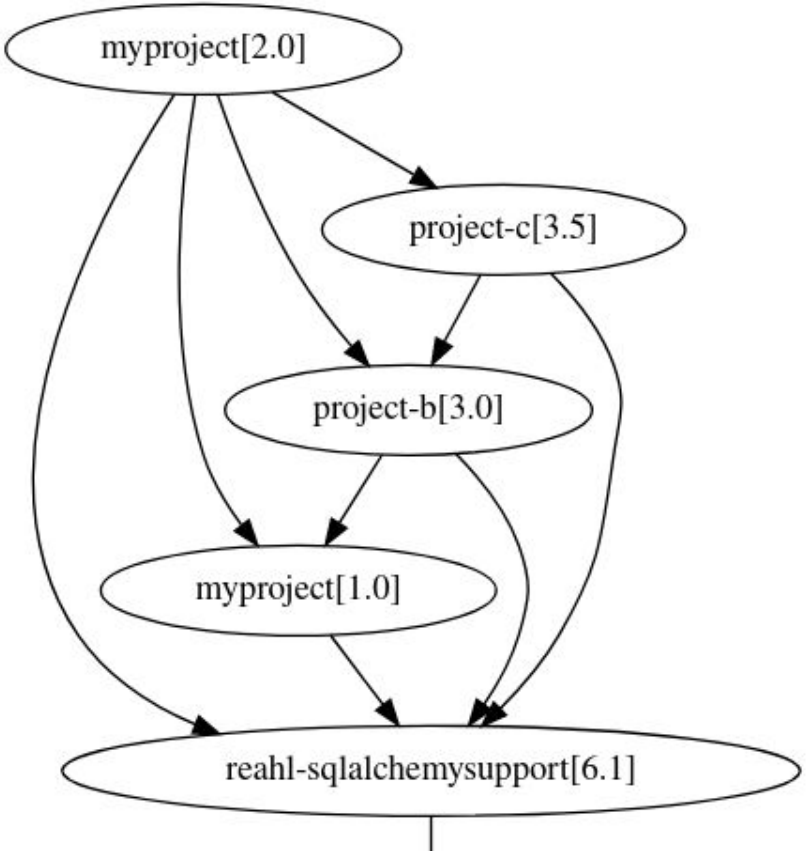
Running Migrations


```
my.ToNewC  in myproject[2.0]: drop_constraint(('fk_myobject_to_c_cobject', ...
b.ToNewC31 in project-b[3.0]: drop_constraint(('fk_bobject_to_c_cobject', ...
c.To31     in project-b[3.0]: drop_constraint(('fk_cobject_another_c_cobject', ...
c.To31     in project-b[3.0]: drop_constraint(('pk_cobject', ...
c.To31     in project-b[3.0]: alter_column(('cobject', 'keyc'), {'new_column_name': 'keyc1'})
c.To31     in project-b[3.0]: create_primary_key(('pk_cobject', ... ['keyc1'] ...
b.ToNewC31 in project-b[3.0]: create_foreign_key(('fk_bobject_to_c_cobject', ... ['keyc1'] ...
c.To31     in project-b[3.0]: create_foreign_key(('fk_cobject_another_c_cobject', ... ['keyc1'] ...
b.ToNewC40 in myproject[2.0]: drop_constraint(('fk_bobject_to_c_cobject', ...

c.To40     in myproject[2.0]: drop_constraint(('fk_cobject_another_c_cobject', ...
c.To40     in myproject[2.0]: drop_constraint(('pk_cobject', ...
c.To40     in myproject[2.0]: alter_column(('cobject', 'keyc2'), {'new_column_name': 'keyc3'})
c.To40     in myproject[2.0]: create_primary_key(('pk_cobject', ... ['keyc3'] ...
my.ToNewC  in myproject[2.0]: create_foreign_key(('fk_myobject_to_c_cobject', ... ['keyc3'] ...
b.ToNewC40 in myproject[2.0]: create_foreign_key(('fk_bobject_to_c_cobject', ...  ['keyc3'] ...
c.To40     in myproject[2.0]: create_foreign_key(('fk_cobject_another_c_cobject', ... ['keyc3'] ...
```

Planning - versions



Planning - clusters



Planning - schedules