Database Version Control with Alembic: Best Practices and Techniques

Pycon APAC 2025

2025-03-01

Agenda

- What is alembic and why we need it?
- How to perform db migrations
- Microservices with shared DB

AXYON.AI

What is alembic and why we need it?

What is a Database Migration?

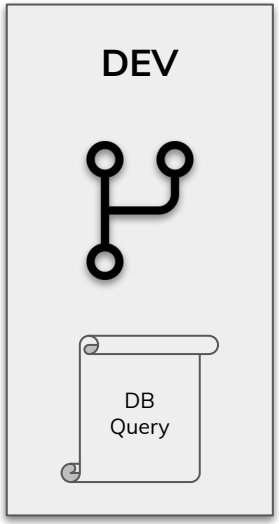
It's a way to manage and track schema changes in your database.

Examples:

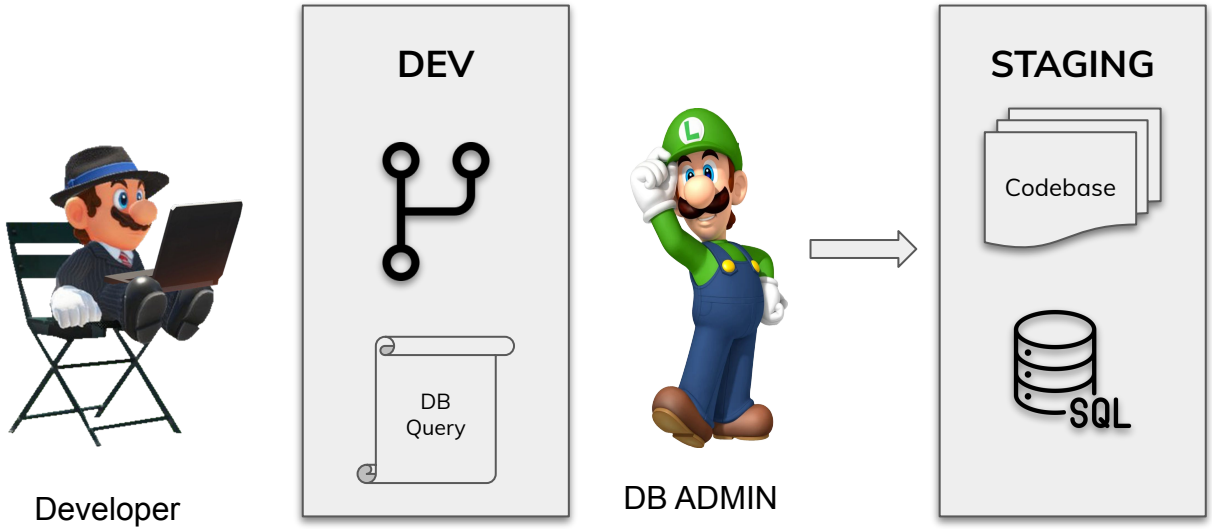
- Create/rename/drop table.
- Add column.
- Relationship definitions.

Development workflow

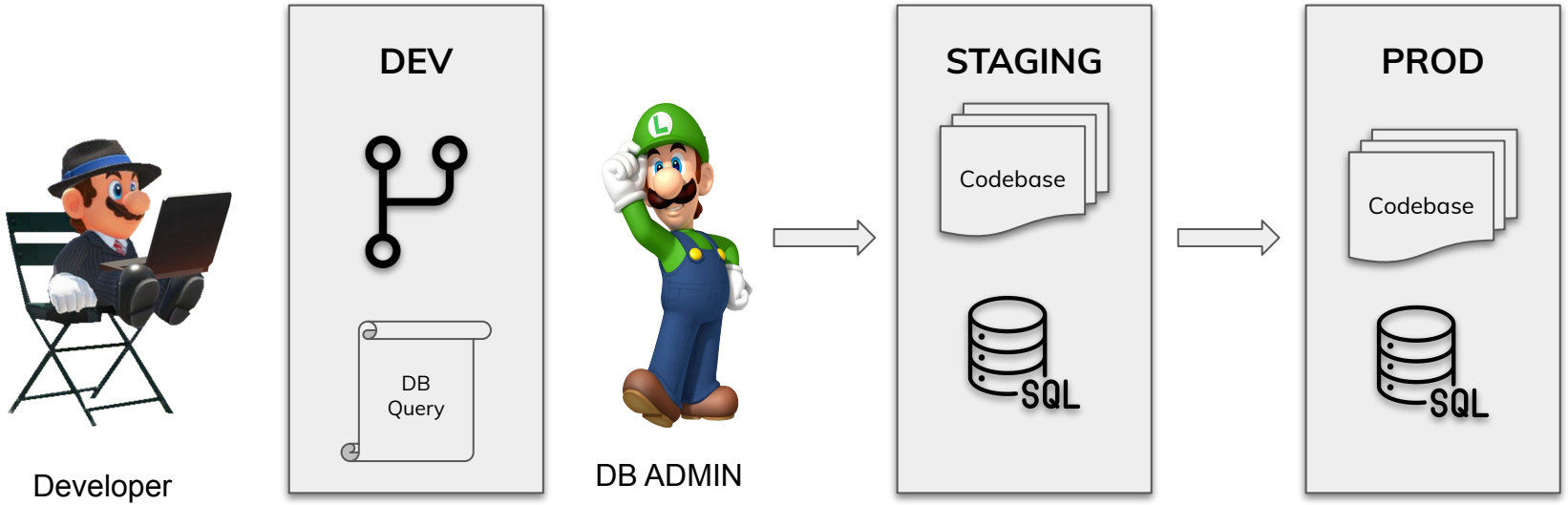
Developer



Development workflow - Open PR

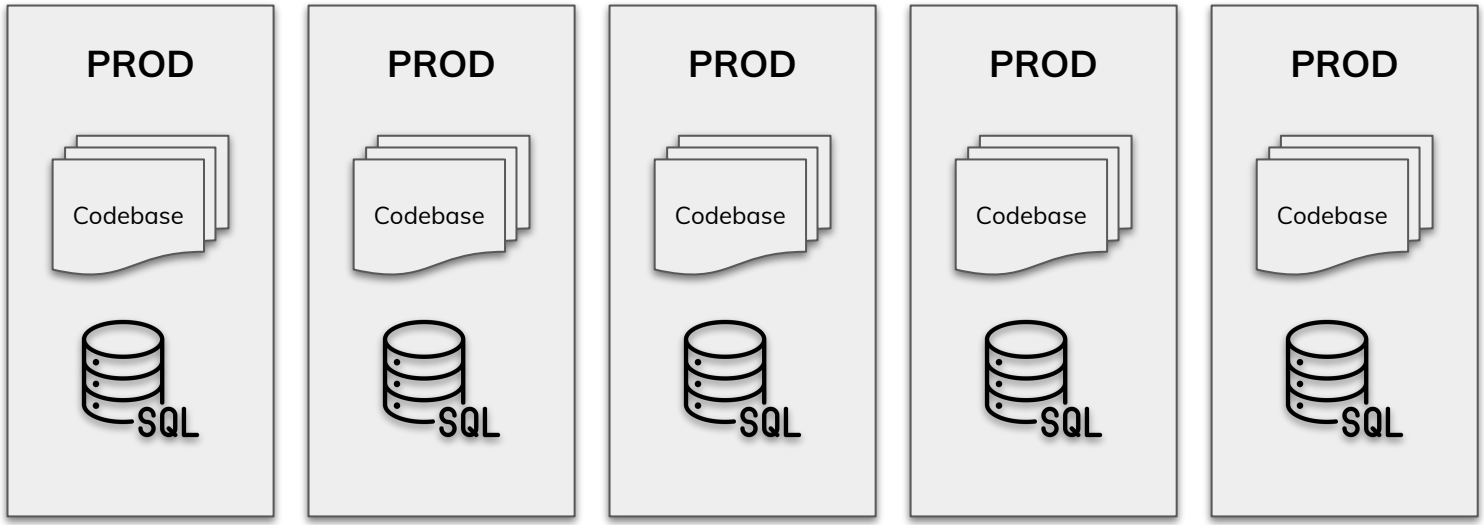


Development workflow - Deploy



Development workflow

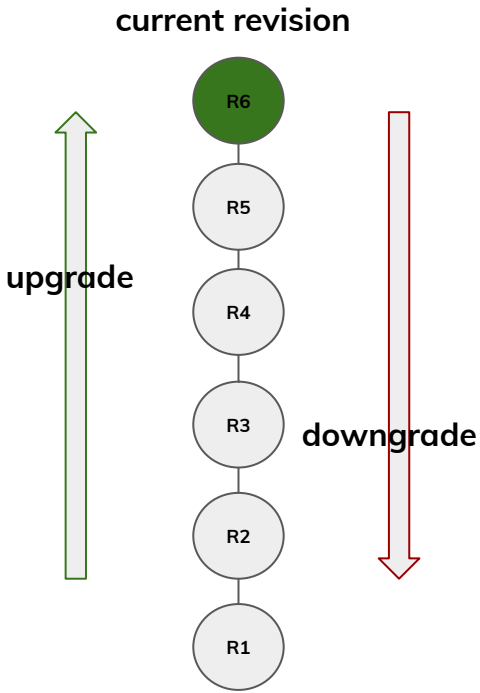
What if your clients ask you to seclude their data?



Enter Alembic!

a lightweight migration tool for SQLAlchemy.

- Track schema changes over time.
- Move between different revision. (upgrade/downgrade)
- Keep your database consistent across multiple environments.



pip install alembic

After running:

```
~/pycon-apac$ alembic init alembic
```

We get the following structure:

```
📂 alembic/
├── env.py          # Main configuration file
├── script.py.mako  # Template for new migrations
├── 📂versions/     # Stores migration scripts
├── README          # Basic documentation
└── 📄 alembic.ini  # Global settings
```

Object-Relational Mapping (ORM)

- ORM abstracts SQL → No longer raw queries.
- Table becomes Python classes.
- SQLAlchemy manages connections, queries, transactions.

Example:

model.py > ...

```python
from sqlalchemy import Column, String, Float, DateTime
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import validates

Base = declarative_base()

class Prices(Base):
    __tablename__ = 'prices'

    date = Column(DateTime, primary_key=True)
    equity_id = Column(String(32), primary_key=True)
    price = Column(Float, nullable=False)
    volume = Column(Float, nullable=False)
```

AXYON.AI

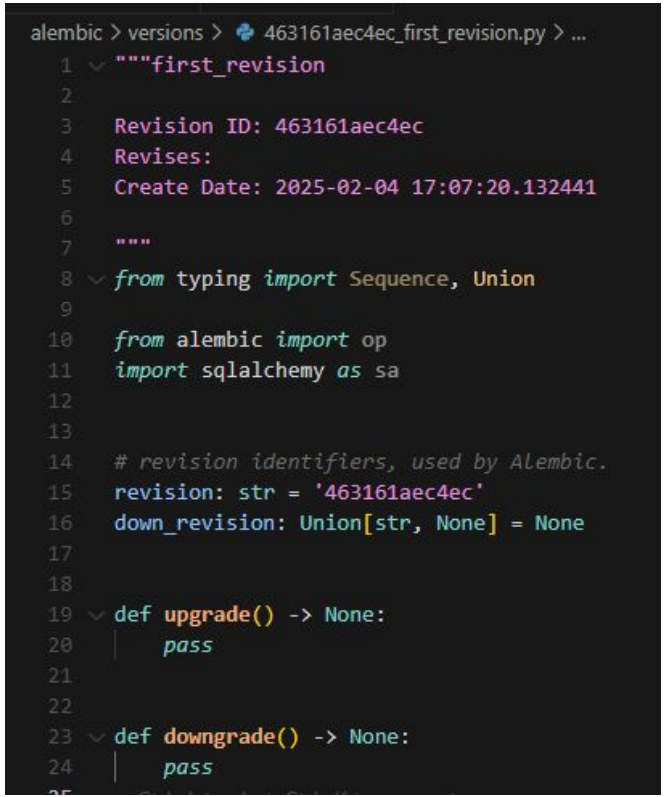
How to perform db migrations

Create first revision

```
roberto-landi@l-100074-sh19:~/pycon-apac$ alembic revision -m "first_revision"
  Generating /home/roberto-landi/pycon-apac/alembic/versions/463161aec4ec_first_revision.py ...  done
roberto-landi@l-100074-sh19:~/pycon-apac$
```

AXYON.AI

Migration file structure



Write first revision

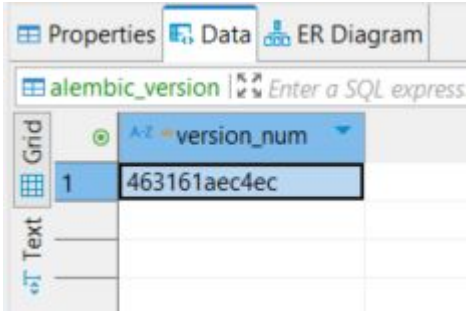
```
18  def upgrade() -> None:
19      op.create_table('prices',
20      sa.Column('date', sa.DateTime(), nullable=False),
21      sa.Column('equity_id', sa.String(length=32), nullable=False),
22      sa.Column('price', sa.Float(), nullable=False),
23      sa.Column('volume', sa.Float(), nullable=False),
24      sa.PrimaryKeyConstraint('date', 'equity_id')
25      )
26
27  def downgrade() -> None:
28      op.drop_table('prices')
29
```

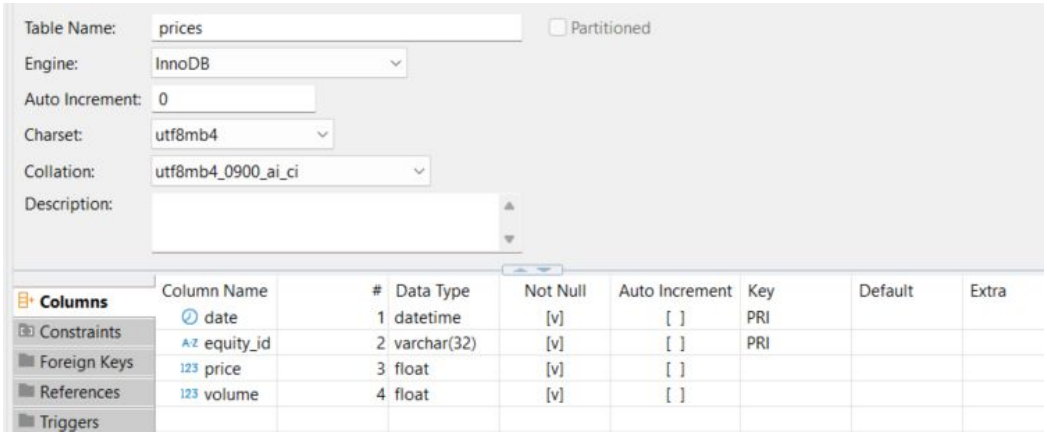
Database view

🌐workshop/

├── alembic_version

├── prices





AXYON.AI

Add data quality checks

```
5
6   class Prices(Base):
7       __tablename__ = 'prices'
8       __table_args__ = (UniqueConstraint('date', 'equity_id', 'market', name='unique_date_equity_id_market'),)
9
10      date = Column(DateTime, primary_key=True)
11      equity_id = Column(String(32), primary_key=True)
12      market = Column(String(32), nullable=True)
13      price = Column(Float, nullable=False)
14      volume = Column(Float, nullable=False)
15      Ctrl+L to chat, Ctrl+K to generate
```

AXYON.AI

Alembic revision –autogenerate

*Alembic can **automatically detect** schema changes in your SQLAlchemy models and generate migration scripts for you.*

```
INFO  [alembic.runtime.migration] Running downgrade b587ee11a882 -> 4051d1aec4ec, data_quality
(venv) roberto-landi@l-100074-sh19:~/pycon-apac$ alembic revision --autogenerate -m "quality_layer"
INFO  [alembic.runtime.migration] Context impl MySQLImpl.
INFO  [alembic.runtime.migration] Will assume non-transactional DDL.
INFO  [alembic.autogenerate.compare] Detected added column 'prices.market'
INFO  [alembic.autogenerate.compare] Detected added unique constraint 'unique_date_equity_id_market' on '('date', 'equity_id', 'market')'
  Generating /home/roberto-landi/pycon-apac/alembic/versions/9b84540a7c1c_quality_layer.py ...  done
(venv) roberto-landi@l-100074-sh19:~/pycon-apac$
```

AXYON.AI

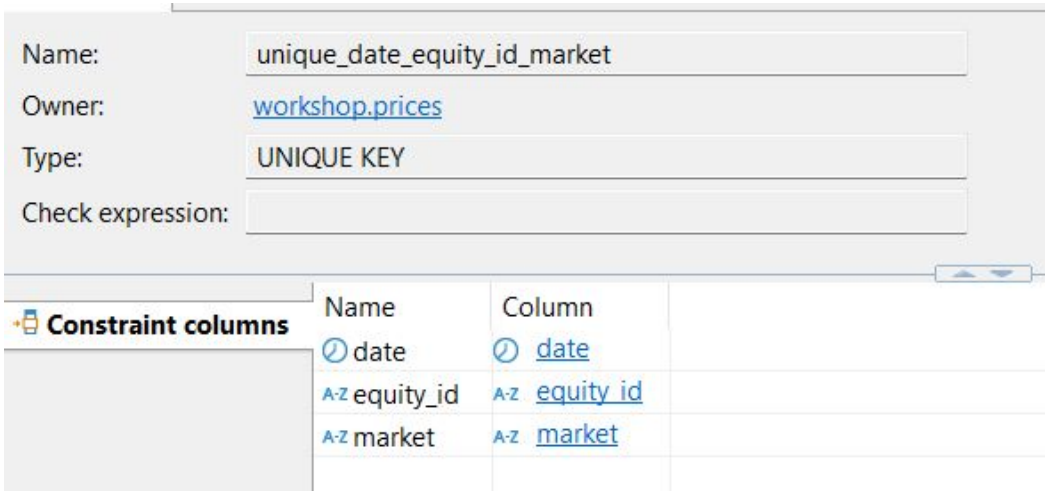
Autogenerated migration

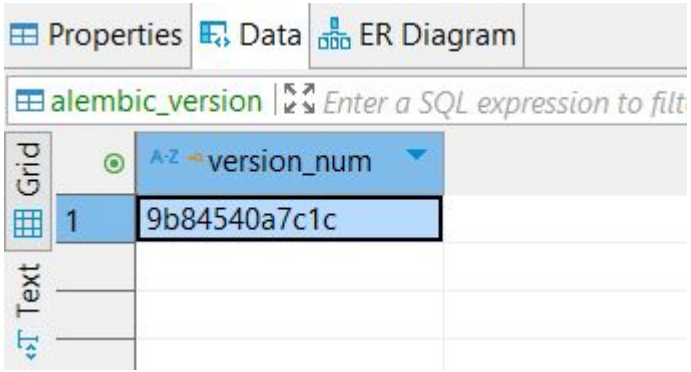
alembic > versions > 9b84540a7c1c_quality_layer.py > ...

```python

# revision identifiers, used by Alembic.
revision: str = '9b84540a7c1c'
down_revision: Union[str, None] = '463161aec4ec'
branch_labels: Union[str, Sequence[str], None] = None
depends_on: Union[str, Sequence[str], None] = None


def upgrade() -> None:
    # ### commands auto generated by Alembic - please adjust! ###
    op.add_column('prices', sa.Column('market', sa.String(length=32), nullable=True))
    op.create_unique_constraint('unique_date_equity_id_market', 'prices', ['date', 'equity_id', 'market'])
    # ### end Alembic commands ###


def downgrade() -> None:
    # ### commands auto generated by Alembic - please adjust! ###
    op.drop_constraint('unique_date_equity_id_market', 'prices', type_='unique')
    op.drop_column('prices', 'market')
    # ### end Alembic commands ###

```

Alembic upgrade head

```
(venv) roberto-landi@l-100074-sh19:~/pycon-apac$ alembic upgrade head
INFO  [alembic.runtime.migration] Context impl MySQLImpl.
INFO  [alembic.runtime.migration] Will assume non-transactional DDL.
INFO  [alembic.runtime.migration] Running upgrade 463161aec4ec -> 9b84540a7c1c, quality_layer
(venv) roberto-landi@l-100074-sh19:~/pycon-apac$
```





Renaming columns

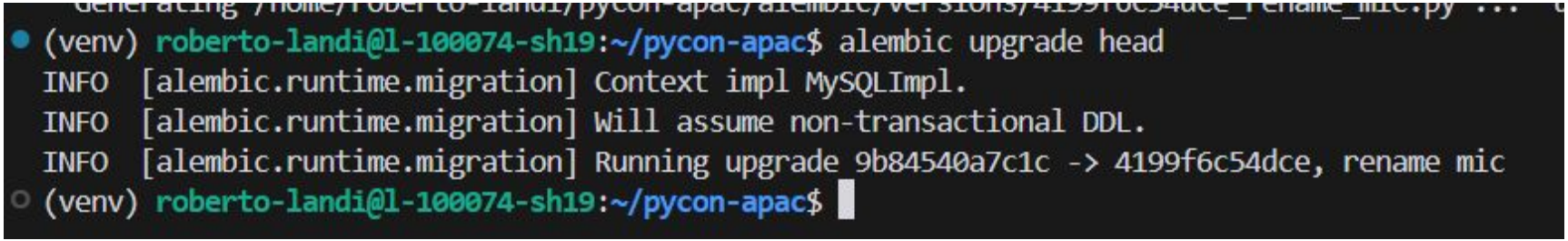
```
4   Base = declarative_base()
5
6   class Prices(Base):
7       __tablename__ = 'prices'
8       __table_args__ = (UniqueConstraint('date', 'equity_id', 'mic', name='unique_date_equity_id_market'),)
9
10      date = Column(DateTime, primary_key=True)
11      equity_id = Column(String(32), primary_key=True)
12      mic = Column(String(32), nullable=True)
13      #market = Column(String(32), nullable=True)
14      price = Column(Float, nullable=False)
15      volume = Column(Float, nullable=False)
16
```

AXYON.AI

Alembic revision –autogenerate

```
(venv) roberto-landi@l-100074-sh19:~/pycon-apac$ alembic revision --autogenerate -m "rename mic"
INFO  [alembic.runtime.migration] Context impl MySQLImpl.
INFO  [alembic.runtime.migration] Will assume non-transactional DDL.
INFO  [alembic.autogenerate.compare] Detected added column 'prices.mic'
INFO  [alembic.autogenerate.compare] Detected changed unique constraint 'unique_date_equity_id_market' on 'prices': expression ('date', 'equity_id', 'market') to ('date', 'e
quity_id', 'mic')
INFO  [alembic.autogenerate.compare] Detected removed column 'prices.market'
  Generating /home/roberto-landi/pycon-apac/alembic/versions/4199f6c54dce_rename_mic.py ...  done
(venv) roberto-landi@l-100074-sh19:~/pycon-apac$
```

Alembic upgrade head



Alembic upgrade head

Autogenerated migration

```python
revision: str = '4199f6c54dce'
down_revision: Union[str, None] = '9b84540a7c1c'
branch_labels: Union[str, Sequence[str], None] = None
depends_on: Union[str, Sequence[str], None] = None


def upgrade() -> None:
    # ### commands auto generated by Alembic - please adjust! ###
    op.add_column('prices', sa.Column('mic', sa.String(length=32), nullable=True))
    op.drop_constraint('unique_date_equity_id_market', 'prices', type_='unique')
    op.create_unique_constraint('unique_date_equity_id_market', 'prices', ['date', 'equity_id', 'mic'])
    op.drop_column('prices', 'market')
    # ### end Alembic commands ###


def downgrade() -> None:
    # ### commands auto generated by Alembic - please adjust! ###
    op.add_column('prices', sa.Column('market', mysql.VARCHAR(length=32), nullable=True))
    op.drop_constraint('unique_date_equity_id_market', 'prices', type_='unique')
    op.create_unique_constraint('unique_date_equity_id_market', 'prices', ['date', 'equity_id', 'market'])
    op.drop_column('prices', 'mic')
    # ### end Alembic commands ###
```

What does Autogenerate Detect ?

Detects ✅:

- Table/columns additions and removals.
- Column type changes.
- Simple constraints.

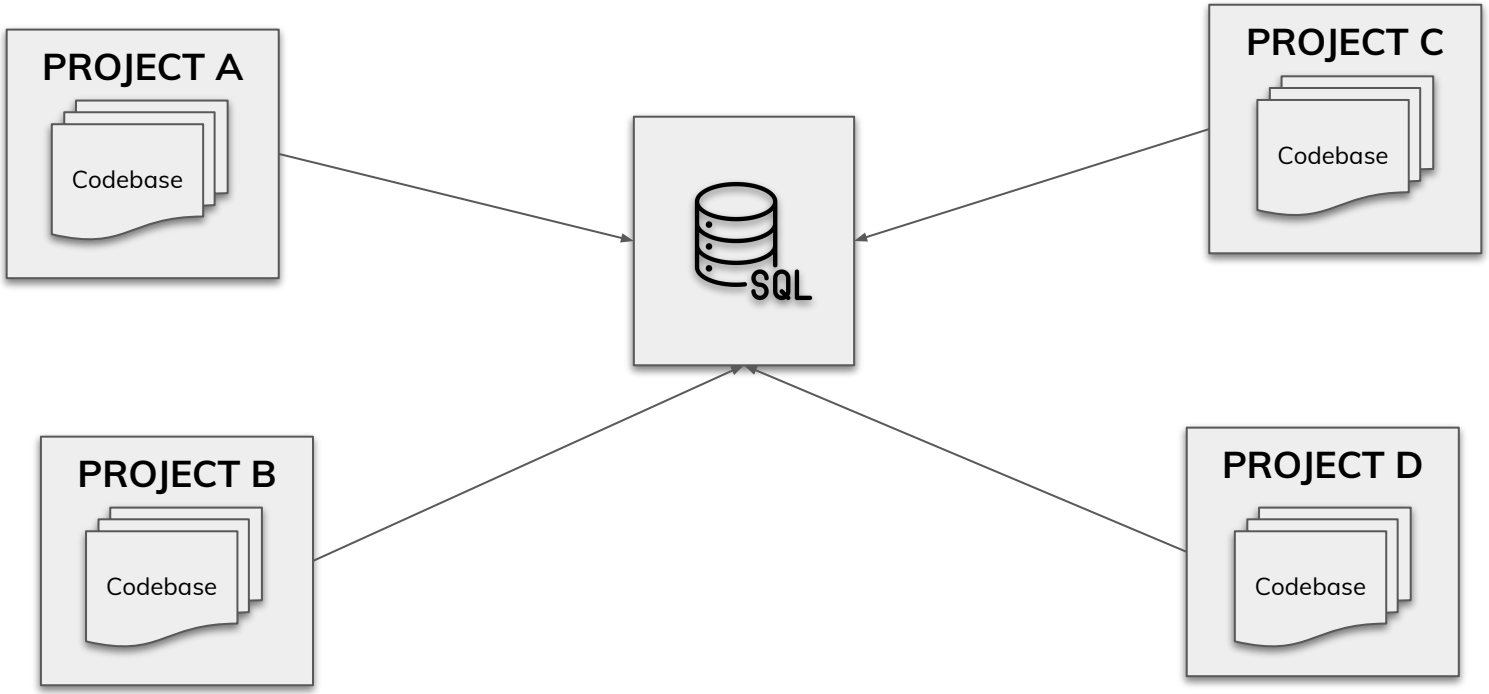
Does NOT Detect ❌:

- Change of column/table names.
- Complex constraints.
- Data migrations.

AXYON.AI

Microservices with shared DB

Microservices with shared DB



Microservices with shared DB - What Goes Wrong?

- How do I keep revision versions aligned?
- How do I prevent table deletions and data loss?

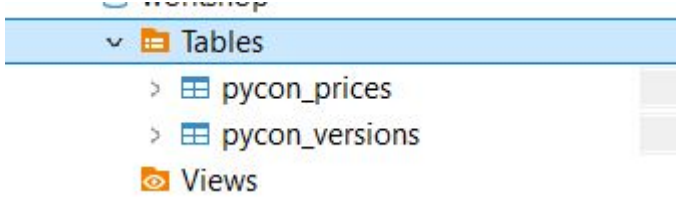
Microservices with shared DB - env.py

```python
def run_migrations_online() -> None:
    """Run migrations in 'online' mode.

    In this scenario we need to create an Engine
    and associate a connection with the context.

    """
    connectable = engine_from_config(
        config.get_section(config.config_ini_section, {}),
        prefix="sqlalchemy.",
        poolclass=pool.NullPool,
    )

    with connectable.connect() as connection:
        context.configure(
            connection=connection,
            target_metadata=target_metadata,
            version_table="pycon_versions",
            include_object=include_object,
        )

        with context.begin_transaction():
            context.run_migrations()
```

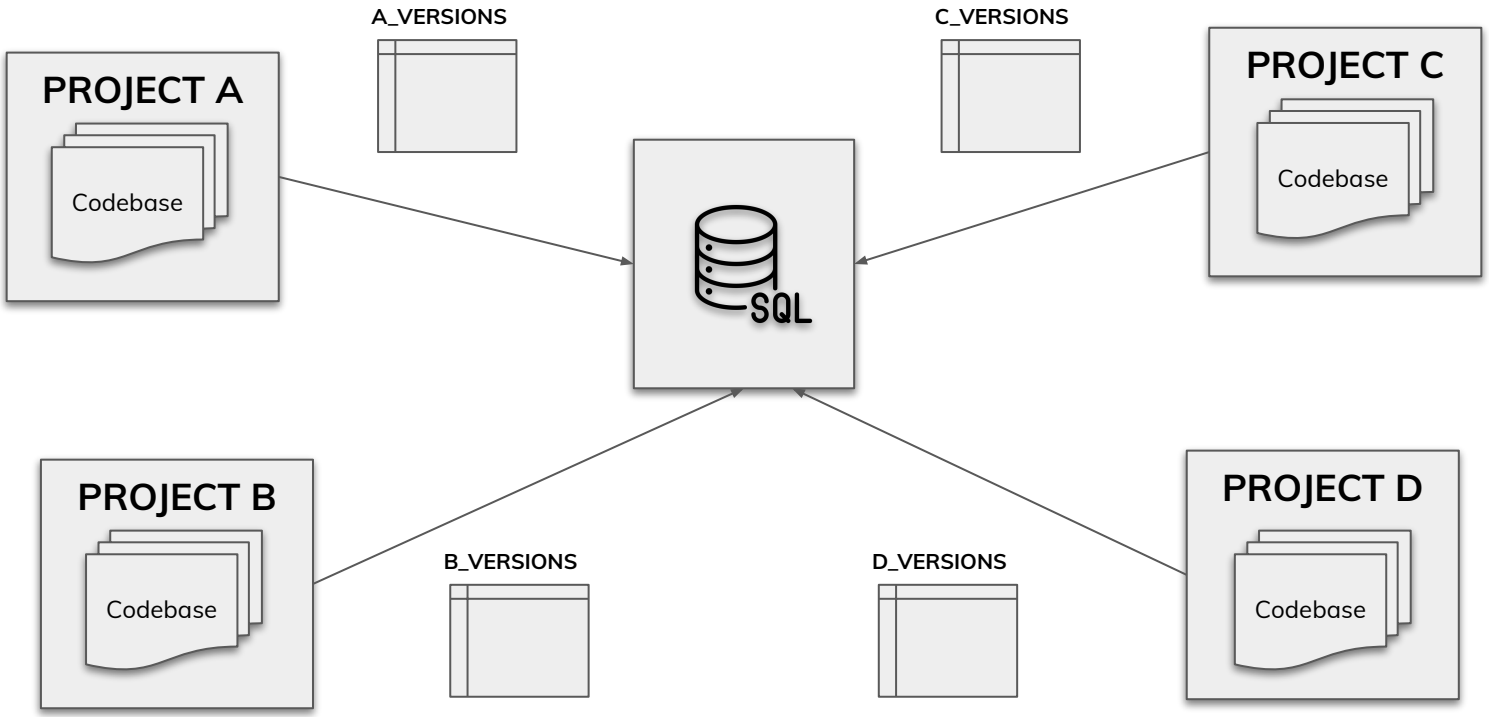
Microservices with shared DB - env.py

```python
def include_object(object, name, type_, reflected, compare_to):
    if type_ == "table" and name.startswith("pycon"):
        return True
    elif type_ == "table":
        return False
    else:
        return True
```

```python
class Prices(Base):
    __tablename__ = 'pycon_prices'
    __table_args__ = (UniqueConstraint('da
    Ctrl+L to chat, Ctrl+K to generate
    date = Column(DateTime, primary_key=Tr
```



Microservices with shared DB - Solution



Key takeaways

- Database migrations ensure schema changes are applied incrementally and consistently.
- Alembic provides version-controlled database migrations, like Git for databases.
- Autogenerate is helpful, but always review migrations before applying them.
- Alembic is best for schema changes, not data migrations.
- Handling multiple projects with a shared DB requires custom migration strategies.

Salamat Po!

Roberto Landi, Head of data engineering.

roberto.landi@axyon.ai

https://axyon.ai