In [ ]:
```
지능정보SW아카데미 -> SW(프로그래밍) + 지능정보(AI)
                    코딩          모델링(데이터가 필수!)
                                    빅데이터(많은, 다양한, 신속한...)
INISW => INI(Intelligence and Informatics)
데이터 수집 - 데이터 관리 - 분석(모델링) - 응용(시각화/SW) => python
1.            (R)DB(MS) - SQLite : 서비스(비즈니스로직) != 코드산출물(*)
                        ORM(SQLAlchemy - Core / ORM)
    Structured Data
2. 수집:Crawler(Crawling+Scrapping) IR≠텍스트마이닝
    Unstructured + Semi(Document)
    Image+Audio+Video...    SGML/XML/HTML/JSON... => 형식(Parsing)
                            RE, DOM, CSS Selctor, XPath, ...
3. 분석:텍스트(한글) 전처리 + ML(Classification+Clustering/Topic)
4. Web, App, => Server(C/S) + UI/UX
   REST(ful) API
```

In [1]:
```
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.schema import MetaData, Table, Column, ForeignKey
from sqlalchemy.types import Integer, Text, String
from sqlalchemy import create_engine
```

In [2]:
```
# ORM -> Object / Data Mapping (engine)
base = declarative_base()
```

In [4]:
```
class TableA(base):
    __tablename__ = 'T_A'

    PK = Column('pk', Integer, primary_key=True)
    NAME = Column('name', Text, nullable=False) # NotNull
```

In [5]:
```
TableA.__table__
```

Out[5]:
```
Table('T_A', MetaData(), Column('pk', Integer(), table=<T_A>, primary_key=True, nu
llable=False), Column('name', Text(), table=<T_A>, nullable=False), schema=None)
```

In [6]:
```
TableA.__tablename__
```

Out[6]:
```
'T_A'
```

In [7]:
```
a = TableA(NAME='아무나')
```

In [10]:
```
a.PK, a.NAME
```

Out[10]:
```
(None, '아무나')
```

In [11]:
```
engine = create_engine('sqlite:///:memory:', echo=True)
```

In [13]:
```
# ORM Core
base.metadata.create_all(engine)
```

```
2023-03-08 09:22:39,027 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2023-03-08 09:22:39,030 INFO sqlalchemy.engine.Engine PRAGMA main.table_info("T_
A")
2023-03-08 09:22:39,033 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-03-08 09:22:39,036 INFO sqlalchemy.engine.Engine PRAGMA temp.table_info("T_
A")
2023-03-08 09:22:39,039 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-03-08 09:22:39,041 INFO sqlalchemy.engine.Engine
CREATE TABLE "T_A" (
        pk INTEGER NOT NULL,
        name TEXT NOT NULL,
        PRIMARY KEY (pk)
)


2023-03-08 09:22:39,042 INFO sqlalchemy.engine.Engine [no key 0.00050s] ()
2023-03-08 09:22:39,042 INFO sqlalchemy.engine.Engine COMMIT
```

In [ ]:
```
Class선언할때, 오류 나는 사람들(여러번 수행해서)

base.metadata.clear()  ==> Table 객체 삭제
base.registry.dispose()  ==> Class 삭제
후 한번만 수행
```

In [15]:
```
# ORM Session
from sqlalchemy.orm import sessionmaker
```

In [16]:
```
Session = sessionmaker(engine)
session = Session()
```

In [18]:
```
session.is_modified(a), session.dirty
```

Out[18]:
```
(True, IdentitySet([]))
```

In [19]:
```
session.add(a)
```

In [20]:
```
session.commit()
```

```
2023-03-08 09:27:41,084 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2023-03-08 09:27:41,094 INFO sqlalchemy.engine.Engine INSERT INTO "T_A" (name) VAL
UES (?)
2023-03-08 09:27:41,094 INFO sqlalchemy.engine.Engine [generated in 0.00101s] ('아
무나',)
2023-03-08 09:27:41,096 INFO sqlalchemy.engine.Engine COMMIT
```

In [21]:
```
a.PK
```

```
2023-03-08 09:27:56,627 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2023-03-08 09:27:56,635 INFO sqlalchemy.engine.Engine SELECT "T_A".pk AS "T_A_pk",
"T_A".name AS "T_A_name"
FROM "T_A"
WHERE "T_A".pk = ?
2023-03-08 09:27:56,638 INFO sqlalchemy.engine.Engine [generated in 0.00357s] (1,)
```

Out[21]:
```
1
```

In [24]:
```
b = session.query(TableA).one()
```

```
2023-03-08 09:31:31,379 INFO sqlalchemy.engine.Engine SELECT "T_A".pk AS "T_A_pk",
"T_A".name AS "T_A_name"
FROM "T_A"
2023-03-08 09:31:31,384 INFO sqlalchemy.engine.Engine [cached since 22.34s ago] ()
```

In [25]:
```
b.PK # 로그인 회원정보
```

Out[25]:
```
1
```

In [27]:
```
b.NAME = '??????????'
```

In [28]:
```
session.dirty
```

Out[28]:
```
IdentitySet([<__main__.TableA object at 0x107f7f460>])
```

In [30]:
```
session.commit()
```

```
2023-03-08 09:32:51,495 INFO sqlalchemy.engine.Engine UPDATE "T_A" SET name=? WHER
E "T_A".pk = ?
2023-03-08 09:32:51,499 INFO sqlalchemy.engine.Engine [generated in 0.00457s]
('??????????', 1)
2023-03-08 09:32:51,501 INFO sqlalchemy.engine.Engine COMMIT
```

In [31]:
```
session.query(TableA).where(TableA.PK == 1).one().NAME
```

```
2023-03-08 09:34:51,839 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2023-03-08 09:34:51,843 INFO sqlalchemy.engine.Engine SELECT "T_A".pk AS "T_A_pk",
"T_A".name AS "T_A_name"
FROM "T_A"
WHERE "T_A".pk = ?
2023-03-08 09:34:51,844 INFO sqlalchemy.engine.Engine [generated in 0.00088s] (1,)
```

Out[31]:
```
'??????????'
```

In [32]:
```
session.add_all([TableA(NAME='2'), TableA(NAME='3'), TableA(NAME='4')])
session.commit()
```

```
2023-03-08 09:36:01,161 INFO sqlalchemy.engine.Engine INSERT INTO "T_A" (name) VAL
UES (?)
2023-03-08 09:36:01,166 INFO sqlalchemy.engine.Engine [cached since 500.1s ago]
('2',)
2023-03-08 09:36:01,169 INFO sqlalchemy.engine.Engine INSERT INTO "T_A" (name) VAL
UES (?)
2023-03-08 09:36:01,175 INFO sqlalchemy.engine.Engine [cached since 500.1s ago]
('3',)
2023-03-08 09:36:01,180 INFO sqlalchemy.engine.Engine INSERT INTO "T_A" (name) VAL
UES (?)
2023-03-08 09:36:01,180 INFO sqlalchemy.engine.Engine [cached since 500.1s ago]
('4',)
2023-03-08 09:36:01,182 INFO sqlalchemy.engine.Engine COMMIT
```

In [33]:
```
inst = session.query(TableA).all()
```

```
2023-03-08 09:36:25,699 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2023-03-08 09:36:25,701 INFO sqlalchemy.engine.Engine SELECT "T_A".pk AS "T_A_pk",
"T_A".name AS "T_A_name"
FROM "T_A"
2023-03-08 09:36:25,704 INFO sqlalchemy.engine.Engine [cached since 316.7s ago] ()
```

In [36]:
```
a is b, b is inst[0], a is inst[2]
```

Out[36]:
```
(True, True, False)
```

In [37]:
```
inst[0].PK, inst[1].PK, inst[2].PK
```

Out[37]:
```
(1, 2, 3)
```

In [39]:
```
session.delete(a)
session.commit()
```

```
2023-03-08 09:37:44,091 INFO sqlalchemy.engine.Engine DELETE FROM "T_A" WHERE "T_
A".pk = ?
2023-03-08 09:37:44,094 INFO sqlalchemy.engine.Engine [generated in 0.00473s] (1,)
2023-03-08 09:37:44,096 INFO sqlalchemy.engine.Engine COMMIT
```

In [41]:
```
session.dirty
```

Out[41]:
```
IdentitySet([])
```

In [42]:
```
session.is_modified(a)
```

Out[42]:
```
False
```

In [43]:
```
a.NAME = 'dd'
```

In [44]:
```
session.is_modified(a)
# 객체가 수정됐는지(in memory -> Class의 인스턴스 ; 물리적 DB X)
```

Out[44]:
```
True
```

In [45]:
```
session.dirty
# Database와 mapped 인스턴스 사이의 변화가 있는지를
```

Out[45]:
```
IdentitySet([])
```

In [46]:
```
session.commit()
```

In [47]:
```
type(inst)
```

Out[47]:
```
list
```

In [48]:
```
len(inst)
```

Out[48]:
```
4
```

In [49]:
```
inst.pop(0), len(inst) # a 객체 삭제 in-memory
```

Out[49]:
```
(<__main__.TableA at 0x107f7f460>, 3)
```

In [ ]:
```
try:
    session.begin()    # Transaction Begin
    session 작업들 ...
    session.commit()
except:
    session.rollback()
```

In [50]:
```
session.close()
```

In [51]:
```
engine.dispose()
```

In [52]:
```
base.metadata.clear()
```

In [53]:
```
base.metadata.tables
```

Out[53]:
```
FacadeDict({})
```

In [ ]:
```
# 실습 - Playlist (ORM 버전) 25분
```

In [ ]:
```
in-memory                              DBMS(SQLite)
            session(instance-data)
1. base상속받은 Class 선언
2. base.metadata.tables (...)
3. base.create_all()   ----------------> TABLE 생성 (SQLX, Core알아서)
4. session.bind = engine (객체-data)
5. Clss로부터 instance 생성
6. session.add(instance) : pending
7.                                  session.commit() (INSERT)
instance
```

In [82]:
```
session.close()
engine.dispose()
base.metadata.clear()
base.registry.dispose()
```

In [57]:
```
engine = create_engine('sqlite:///playlist.db', echo=True)
```

In [60]:
```
base.metadata.reflect(engine)
```

```
2023-03-08 11:30:59,918 INFO sqlalchemy.engine.Engine SELECT name FROM sqlite_mast
er WHERE type='table' ORDER BY name
2023-03-08 11:30:59,927 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-03-08 11:30:59,930 INFO sqlalchemy.engine.Engine PRAGMA main.table_xinfo("alb
um")
2023-03-08 11:30:59,931 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-03-08 11:30:59,933 INFO sqlalchemy.engine.Engine SELECT sql FROM  (SELECT * F
ROM sqlite_master UNION ALL   SELECT * FROM sqlite_temp_master) WHERE name = ? AND
type = 'table'
2023-03-08 11:30:59,933 INFO sqlalchemy.engine.Engine [raw sql] ('album',)
2023-03-08 11:30:59,935 INFO sqlalchemy.engine.Engine PRAGMA main.foreign_key_list
("album")
2023-03-08 11:30:59,935 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-03-08 11:30:59,937 INFO sqlalchemy.engine.Engine PRAGMA temp.foreign_key_list
("album")
2023-03-08 11:30:59,937 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-03-08 11:30:59,939 INFO sqlalchemy.engine.Engine SELECT sql FROM  (SELECT * F
ROM sqlite_master UNION ALL   SELECT * FROM sqlite_temp_master) WHERE name = ? AND
type = 'table'
2023-03-08 11:30:59,941 INFO sqlalchemy.engine.Engine [raw sql] ('album',)
2023-03-08 11:30:59,943 INFO sqlalchemy.engine.Engine PRAGMA main.index_list("albu
m")
2023-03-08 11:30:59,944 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-03-08 11:30:59,945 INFO sqlalchemy.engine.Engine PRAGMA temp.index_list("albu
m")
2023-03-08 11:30:59,945 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-03-08 11:30:59,946 INFO sqlalchemy.engine.Engine PRAGMA main.index_list("albu
m")
2023-03-08 11:30:59,946 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-03-08 11:30:59,946 INFO sqlalchemy.engine.Engine PRAGMA temp.index_list("albu
m")
2023-03-08 11:30:59,947 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-03-08 11:30:59,948 INFO sqlalchemy.engine.Engine SELECT sql FROM  (SELECT * F
ROM sqlite_master UNION ALL   SELECT * FROM sqlite_temp_master) WHERE name = ? AND
type = 'table'
2023-03-08 11:30:59,948 INFO sqlalchemy.engine.Engine [raw sql] ('album',)
2023-03-08 11:30:59,949 INFO sqlalchemy.engine.Engine PRAGMA main.table_xinfo("art
ist")
2023-03-08 11:30:59,950 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-03-08 11:30:59,950 INFO sqlalchemy.engine.Engine SELECT sql FROM  (SELECT * F
ROM sqlite_master UNION ALL   SELECT * FROM sqlite_temp_master) WHERE name = ? AND
type = 'table'
2023-03-08 11:30:59,950 INFO sqlalchemy.engine.Engine [raw sql] ('artist',)
2023-03-08 11:30:59,951 INFO sqlalchemy.engine.Engine PRAGMA main.foreign_key_list
("artist")
2023-03-08 11:30:59,951 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-03-08 11:30:59,951 INFO sqlalchemy.engine.Engine PRAGMA temp.foreign_key_list
("artist")
2023-03-08 11:30:59,951 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-03-08 11:30:59,952 INFO sqlalchemy.engine.Engine SELECT sql FROM  (SELECT * F
ROM sqlite_master UNION ALL   SELECT * FROM sqlite_temp_master) WHERE name = ? AND
type = 'table'
2023-03-08 11:30:59,952 INFO sqlalchemy.engine.Engine [raw sql] ('artist',)
2023-03-08 11:30:59,952 INFO sqlalchemy.engine.Engine PRAGMA main.index_list("arti
st")
2023-03-08 11:30:59,953 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-03-08 11:30:59,954 INFO sqlalchemy.engine.Engine PRAGMA temp.index_list("arti
st")
2023-03-08 11:30:59,954 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-03-08 11:30:59,954 INFO sqlalchemy.engine.Engine PRAGMA main.index_list("arti
st")
```

```
2023-03-08 11:30:59,955 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-03-08 11:30:59,955 INFO sqlalchemy.engine.Engine PRAGMA temp.index_list("arti
st")
2023-03-08 11:30:59,955 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-03-08 11:30:59,956 INFO sqlalchemy.engine.Engine SELECT sql FROM  (SELECT * F
ROM sqlite_master UNION ALL   SELECT * FROM sqlite_temp_master) WHERE name = ? AND
type = 'table'
2023-03-08 11:30:59,956 INFO sqlalchemy.engine.Engine [raw sql] ('artist',)
2023-03-08 11:30:59,956 INFO sqlalchemy.engine.Engine PRAGMA main.table_xinfo("gen
re")
2023-03-08 11:30:59,957 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-03-08 11:30:59,957 INFO sqlalchemy.engine.Engine SELECT sql FROM  (SELECT * F
ROM sqlite_master UNION ALL   SELECT * FROM sqlite_temp_master) WHERE name = ? AND
type = 'table'
2023-03-08 11:30:59,958 INFO sqlalchemy.engine.Engine [raw sql] ('genre',)
2023-03-08 11:30:59,958 INFO sqlalchemy.engine.Engine PRAGMA main.foreign_key_list
("genre")
2023-03-08 11:30:59,959 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-03-08 11:30:59,959 INFO sqlalchemy.engine.Engine PRAGMA temp.foreign_key_list
("genre")
2023-03-08 11:30:59,959 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-03-08 11:30:59,959 INFO sqlalchemy.engine.Engine SELECT sql FROM  (SELECT * F
ROM sqlite_master UNION ALL   SELECT * FROM sqlite_temp_master) WHERE name = ? AND
type = 'table'
2023-03-08 11:30:59,960 INFO sqlalchemy.engine.Engine [raw sql] ('genre',)
2023-03-08 11:30:59,960 INFO sqlalchemy.engine.Engine PRAGMA main.index_list("genr
e")
2023-03-08 11:30:59,961 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-03-08 11:30:59,961 INFO sqlalchemy.engine.Engine PRAGMA temp.index_list("genr
e")
2023-03-08 11:30:59,961 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-03-08 11:30:59,962 INFO sqlalchemy.engine.Engine PRAGMA main.index_list("genr
e")
2023-03-08 11:30:59,962 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-03-08 11:30:59,963 INFO sqlalchemy.engine.Engine PRAGMA temp.index_list("genr
e")
2023-03-08 11:30:59,963 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-03-08 11:30:59,964 INFO sqlalchemy.engine.Engine SELECT sql FROM  (SELECT * F
ROM sqlite_master UNION ALL   SELECT * FROM sqlite_temp_master) WHERE name = ? AND
type = 'table'
2023-03-08 11:30:59,964 INFO sqlalchemy.engine.Engine [raw sql] ('genre',)
2023-03-08 11:30:59,965 INFO sqlalchemy.engine.Engine PRAGMA main.table_xinfo("tra
ck")
2023-03-08 11:30:59,965 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-03-08 11:30:59,966 INFO sqlalchemy.engine.Engine SELECT sql FROM  (SELECT * F
ROM sqlite_master UNION ALL   SELECT * FROM sqlite_temp_master) WHERE name = ? AND
type = 'table'
2023-03-08 11:30:59,966 INFO sqlalchemy.engine.Engine [raw sql] ('track',)
2023-03-08 11:30:59,967 INFO sqlalchemy.engine.Engine PRAGMA main.foreign_key_list
("track")
2023-03-08 11:30:59,967 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-03-08 11:30:59,967 INFO sqlalchemy.engine.Engine PRAGMA temp.foreign_key_list
("track")
2023-03-08 11:30:59,967 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-03-08 11:30:59,968 INFO sqlalchemy.engine.Engine SELECT sql FROM  (SELECT * F
ROM sqlite_master UNION ALL   SELECT * FROM sqlite_temp_master) WHERE name = ? AND
type = 'table'
2023-03-08 11:30:59,968 INFO sqlalchemy.engine.Engine [raw sql] ('track',)
2023-03-08 11:30:59,968 INFO sqlalchemy.engine.Engine PRAGMA main.index_list("trac
k")
2023-03-08 11:30:59,968 INFO sqlalchemy.engine.Engine [raw sql] ()
```

```
2023-03-08 11:30:59,969 INFO sqlalchemy.engine.Engine PRAGMA temp.index_list("trac
k")
2023-03-08 11:30:59,969 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-03-08 11:30:59,970 INFO sqlalchemy.engine.Engine PRAGMA main.index_list("trac
k")
2023-03-08 11:30:59,970 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-03-08 11:30:59,971 INFO sqlalchemy.engine.Engine PRAGMA temp.index_list("trac
k")
2023-03-08 11:30:59,971 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-03-08 11:30:59,971 INFO sqlalchemy.engine.Engine SELECT sql FROM  (SELECT * F
ROM sqlite_master UNION ALL   SELECT * FROM sqlite_temp_master) WHERE name = ? AND
type = 'table'
2023-03-08 11:30:59,971 INFO sqlalchemy.engine.Engine [raw sql] ('track',)
```

In [62]:
```
base.metadata.tables.keys()
```

Out[62]:
```
dict_keys(['album', 'artist', 'genre', 'track'])
```

In [63]:
```
class Album(base):
    __table__ = base.metadata.tables['album']
    프로퍼티 = 컬럼객체생성
```

In [72]:
```
class Artist(base):
    __table__ = base.metadata.tables['artist']

# class Artist(base):
#     __table__ = base.metadata.tables['artist']
```

In [64]:
```
Session = sessionmaker(engine)
session = Session()
```

In [66]:
```
albumList = session.query(Album).all()
```

```
2023-03-08 11:33:07,835 INFO sqlalchemy.engine.Engine SELECT album.pk AS album_pk,
album.name AS album_name, album.fk AS album_fk
FROM album
2023-03-08 11:33:07,844 INFO sqlalchemy.engine.Engine [cached since 10.34s ago] ()
```

In [69]:
```
albumList[0].pk, albumList[0].name
```

Out[69]:
```
(1, '아무개')
```

In [71]:
```
type(base.metadata.tables['album'].c.pk)
```

Out[71]:
```
sqlalchemy.sql.schema.Column
```

In [75]:
```
session.query(Artist.name, Album.name).select_from(Artist)\
.join(Album, Album.fk==Artist.pk).all()
```

```
2023-03-08 11:38:51,007 INFO sqlalchemy.engine.Engine SELECT artist.name AS artist
_name, album.name AS album_name
FROM artist JOIN album ON album.fk = artist.pk
2023-03-08 11:38:51,014 INFO sqlalchemy.engine.Engine [generated in 0.00668s] ()
```

Out[75]:
```
[('가수10', '아무개'), ('가수10', '아무개1'), ('가수1', '아무개2')]
```

In [ ]:
```
Album.artist = Artist
```

In [77]:
```
base.metadata.tables
```

Out[77]:
```
FacadeDict({})
```

In [83]:
```
class User(base):
    __tablename__ = 'T_USER'

    pk = Column('PK', Integer, primary_key=True)
    name = Column('NAME', Text, nullable=False)

    def __repr__(self):
        return 'PK:{}, NAME:{}'.format(self.pk, self.name)

class Address(base):
    __tablename__ = 'T_ADDRESS'

    pk = Column('PK', Integer, primary_key=True)
    name = Column('NAME', Text, nullable=False)
    fk = Column('FK', Integer, ForeignKey(User.pk))

    def __repr__(self):
        return 'PK:{}, NAME:{}'.format(self.pk, self.name, self.fk)
```

In [84]:
```
engine = create_engine('sqlite:///:memory:', echo=True)
```

In [85]:
```
base.metadata.create_all(engine)
```

```
2023-03-08 11:53:11,267 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2023-03-08 11:53:11,270 INFO sqlalchemy.engine.Engine PRAGMA main.table_info("T_US
ER")
2023-03-08 11:53:11,271 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-03-08 11:53:11,272 INFO sqlalchemy.engine.Engine PRAGMA temp.table_info("T_US
ER")
2023-03-08 11:53:11,273 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-03-08 11:53:11,274 INFO sqlalchemy.engine.Engine PRAGMA main.table_info("T_AD
DRESS")
2023-03-08 11:53:11,275 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-03-08 11:53:11,277 INFO sqlalchemy.engine.Engine PRAGMA temp.table_info("T_AD
DRESS")
2023-03-08 11:53:11,278 INFO sqlalchemy.engine.Engine [raw sql] ()
2023-03-08 11:53:11,280 INFO sqlalchemy.engine.Engine
CREATE TABLE "T_USER" (
        "PK" INTEGER NOT NULL,
        "NAME" TEXT NOT NULL,
        PRIMARY KEY ("PK")
)


2023-03-08 11:53:11,280 INFO sqlalchemy.engine.Engine [no key 0.00085s] ()
2023-03-08 11:53:11,281 INFO sqlalchemy.engine.Engine
CREATE TABLE "T_ADDRESS" (
        "PK" INTEGER NOT NULL,
        "NAME" TEXT NOT NULL,
        "FK" INTEGER,
        PRIMARY KEY ("PK"),
        FOREIGN KEY("FK") REFERENCES "T_USER" ("PK")
)


2023-03-08 11:53:11,282 INFO sqlalchemy.engine.Engine [no key 0.00040s] ()
2023-03-08 11:53:11,282 INFO sqlalchemy.engine.Engine COMMIT
```

In [86]:
```
User(name='아무개')
```

Out[86]:
```
PK:None, NAME:아무개
```

In [89]:
```
Session = sessionmaker(engine)
session = Session()
```

In [90]:
```
session.add(User(name='아무개'))
session.commit()
```

```
2023-03-08 11:55:00,711 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2023-03-08 11:55:00,717 INFO sqlalchemy.engine.Engine INSERT INTO "T_USER" ("NAM
E") VALUES (?)
2023-03-08 11:55:00,718 INFO sqlalchemy.engine.Engine [generated in 0.00151s] ('아
무개',)
2023-03-08 11:55:00,720 INFO sqlalchemy.engine.Engine COMMIT
```

In [91]:
```
userA = session.query(User).one()
```

```
2023-03-08 11:55:26,619 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2023-03-08 11:55:26,632 INFO sqlalchemy.engine.Engine SELECT "T_USER"."PK" AS "T_U
SER_PK", "T_USER"."NAME" AS "T_USER_NAME"
FROM "T_USER"
2023-03-08 11:55:26,634 INFO sqlalchemy.engine.Engine [generated in 0.00167s] ()
```

In [92]:
```
userA
```

Out[92]:
```
PK:1, NAME:아무개
```

In [93]:
```
userB = User(name='아무개2')
```

In [94]:
```
session.add(userB)
session.commit()
```

```
2023-03-08 11:56:26,313 INFO sqlalchemy.engine.Engine INSERT INTO "T_USER" ("NAM
E") VALUES (?)
2023-03-08 11:56:26,317 INFO sqlalchemy.engine.Engine [cached since 85.6s ago]
('아무개2',)
2023-03-08 11:56:26,320 INFO sqlalchemy.engine.Engine COMMIT
```

In [95]:
```
session.add_all([Address(name='주소1', fk=userA.pk),
                 Address(name='주소2', fk=userB.pk)])
```

```
2023-03-08 11:57:52,677 INFO sqlalchemy.engine.Engine BEGIN (implicit)
2023-03-08 11:57:52,683 INFO sqlalchemy.engine.Engine SELECT "T_USER"."PK" AS "T_U
SER_PK", "T_USER"."NAME" AS "T_USER_NAME"
FROM "T_USER"
WHERE "T_USER"."PK" = ?
2023-03-08 11:57:52,693 INFO sqlalchemy.engine.Engine [generated in 0.01064s] (1,)
2023-03-08 11:57:52,695 INFO sqlalchemy.engine.Engine SELECT "T_USER"."PK" AS "T_U
SER_PK", "T_USER"."NAME" AS "T_USER_NAME"
FROM "T_USER"
WHERE "T_USER"."PK" = ?
2023-03-08 11:57:52,696 INFO sqlalchemy.engine.Engine [cached since 0.01357s ago]
(2,)
```

In [96]:
```
session.commit()
```

```
2023-03-08 11:58:34,826 INFO sqlalchemy.engine.Engine INSERT INTO "T_ADDRESS" ("NA
ME", "FK") VALUES (?, ?)
2023-03-08 11:58:34,836 INFO sqlalchemy.engine.Engine [generated in 0.00959s] ('주
소1', 1)
2023-03-08 11:58:34,837 INFO sqlalchemy.engine.Engine INSERT INTO "T_ADDRESS" ("NA
ME", "FK") VALUES (?, ?)
2023-03-08 11:58:34,838 INFO sqlalchemy.engine.Engine [cached since 0.01174s ago]
('주소2', 2)
2023-03-08 11:58:34,839 INFO sqlalchemy.engine.Engine COMMIT
```

In [99]:
```
session.query(Address).all()[1]
```

```
2023-03-08 11:59:23,376 INFO sqlalchemy.engine.Engine SELECT "T_ADDRESS"."PK" AS
"T_ADDRESS_PK", "T_ADDRESS"."NAME" AS "T_ADDRESS_NAME", "T_ADDRESS"."FK" AS "T_ADD
RESS_FK"
FROM "T_ADDRESS"
2023-03-08 11:59:23,382 INFO sqlalchemy.engine.Engine [cached since 18.11s ago] ()
```

Out[99]:
```
__main__.Address
```

In [103…
```
base.metadata.clear()
base.registry.dispose()
```

In [101…
```
from sqlalchemy.orm import relationship
```

In [104…
```
class User(base):
    __tablename__ = 'T_USER'

    pk = Column('PK', Integer, primary_key=True)
    name = Column('NAME', Text, nullable=False)
    addresses = relationship('Address', back_populates='user')

    def __repr__(self):
        return 'PK:{}, NAME:{}'.format(self.pk, self.name)

class Address(base):
    __tablename__ = 'T_ADDRESS'

    pk = Column('PK', Integer, primary_key=True)
    name = Column('NAME', Text, nullable=False)
    fk = Column('FK', Integer, ForeignKey(User.pk))
    user = relationship('User', back_populates='addresses', uselist=False)

    def __repr__(self):
        return 'PK:{}, NAME:{}, FK:{}'.format(
            self.pk, self.name, self.fk)
```

In [108…
```
type(session.query(User).all()[0]) # User
```

```
2023-03-08 12:04:42,544 INFO sqlalchemy.engine.Engine SELECT "T_USER"."PK" AS "T_U
SER_PK", "T_USER"."NAME" AS "T_USER_NAME"
FROM "T_USER"
2023-03-08 12:04:42,547 INFO sqlalchemy.engine.Engine [cached since 26.05s ago] ()
```

Out[108]:
```
__main__.User
```

In [110…
```
type(session.query(User).all()[0].addresses[0])
```

```
2023-03-08 12:04:52,274 INFO sqlalchemy.engine.Engine SELECT "T_USER"."PK" AS "T_U
SER_PK", "T_USER"."NAME" AS "T_USER_NAME"
FROM "T_USER"
2023-03-08 12:04:52,283 INFO sqlalchemy.engine.Engine [cached since 35.78s ago] ()
```

Out[110]:
```
__main__.Address
```

In [111…
```
userA = session.query(User).all()[0]
```

```
2023-03-08 12:05:05,452 INFO sqlalchemy.engine.Engine SELECT "T_USER"."PK" AS "T_U
SER_PK", "T_USER"."NAME" AS "T_USER_NAME"
FROM "T_USER"
2023-03-08 12:05:05,463 INFO sqlalchemy.engine.Engine [cached since 48.96s ago] ()
```

In [114…
```
userA.addresses[0] is session.query(Address).all()[0]
```

```
2023-03-08 12:05:38,497 INFO sqlalchemy.engine.Engine SELECT "T_ADDRESS"."PK" AS
"T_ADDRESS_PK", "T_ADDRESS"."NAME" AS "T_ADDRESS_NAME", "T_ADDRESS"."FK" AS "T_ADD
RESS_FK"
FROM "T_ADDRESS"
2023-03-08 12:05:38,499 INFO sqlalchemy.engine.Engine [generated in 0.00217s] ()
```

Out[114]:
```
True
```

In [116…
```
userA.addresses[0].user is userA
```

Out[116]:
```
True
```