ΠΕΡΙΕΧΟΜΕΝΑ:

Ευάγγελος Μ.

Σμαραγδένιος Χορηγός Μαθήματος

Απόστολος Σ.

Ασημένιος Χορηγός Μαθήματος

Inner Join

- Κατασκευάζουμε μία εσωτερική σύνδεση ως εξής:

SQL

```
SELECT cn.name AS country, ct.name AS city
FROM city ct
        JOIN country cn ON ct.CountryCode = cn.Code
ORDER BY country, city
```

SQL Alchemy

```
select([country.c["Name"].label("country"),
        city.c["Name"].label("city")]).\
select_from(country.
            join(city, country.c["Code"]==city.c["CountryCode"])).\
order_by(country.c["Name"], city.c["Name"])
```

Left (Outer) Join

- Κατασκευάζουμε μία εξωτερική σύνδεση ως εξής:

SQL

```
SELECT cn.name AS country, count(ct.name) AS cities
FROM country cn
        LEFT JOIN city ct ON cn.Code = ct.CountryCode
GROUP BY cn.Name
```

SQL Alchemy

```
select([country.c["Name"].label("country"),
        func.count(city.c["Name"]).label("cities")]).\
select_from(country.
    outer_join(city, country.c["Code"] == city.c["CountryCode"])).\
group_by(country.c["Name"])
```

Σημείωση:

- Μπορούμε να εφαρμόσουμε join επί του αποτελέσματος ενός join, για να έχουμε σύνδεση με 3+ πίνακες.

- Δεν υποστηρίζεται Right Outer Join
- Δεν υποστηρίζεται Full Outer Join (με ΒΔ MySQL)
- Δεν υποστηρίζεται Equi-Join (με μικρή επιφύλαξη, βλ. βίντεο)

Παράδειγμα 1: inner_join (βλ. βίντεο)

Άσκηση 1:

Μετατρέψτε σε SQLAlchemy το ερώτημα:

```
SELECT ct.name AS city, ct.population as population,
        COUNT(*) AS languages_spoken
FROM city ct JOIN country cn ON ct.countrycode = cn.code
             JOIN countrylanguage cl ON cn.code = cl.countrycode
WHERE cn.name = 'Greece'
GROUP BY ct.name
ORDER BY population DESC
LIMIT 2
```

Natural Join

- Παραλείπουμε τη σχέση κλειδιών:

SQL

```
SELECT offices.city, employees.lastName
FROM employees NATURAL JOIN offices;
```

SQL Alchemy

```
select([offices.c["city"], employees.c["lastName"]]).\
select_from(offices.join(employees))
```

Cross Join:

SQLALC

```
select([offices, employees])
```

Παράδειγμα 2-3: left_join και natural_join (βλ. βίντεο)

Υποερωτήματα

- Η σύνθεση υποερωτημάτων απαιτεί να γράψουμε πρώτα το υποερώτημα και να καλέσουμε επί αυτού την subquery().
- Έπειτα μπορούμε να το χρησιμοποιήσουμε στο κυρίως ερώτημα.

SQL

```
SELECT name AS country, population
FROM country
WHERE population > (
        SELECT AVG(population)
        FROM country
      )
```

SQL Alchemy

```
sub = select([func.avg(country.c["Population"])]).subquery()

query = select([country.c["Name"].label("country"),
                country.c["Population"]]).\
        where(country.c["Population"] > sub)
```

UNION

SQL

```
SELECT contactLastName AS lastName FROM customers
UNION
SELECT lastName FROM employees
```

SQL Alchemy

```
union(
  select([customers.c["contactLastName"].label("lastName")]),
  select([employees.c["lastName"]])
)
```

Αντίστοιχα χρησιμοποιούμε τη union_all() (αντί της union())

Παράδειγμα 4-5: subquery και union (βλ. βίντεο)

Απ' ευθείας εκτέλεση ερωτημάτων SQL

- Προφανώς υπάρχει και η δυνατότητα να εκτελέσουμε απ' ευθείας ερωτήματα SQL.
- Χρησιμοποιούμε την συνάρτηση text():

```
query = text(...)
```

στην οποία θέτουμε ως όρισμα συμβολοσειρά με το ερώτημα SQL.

Σημείωση:

- Η text() μπορεί να χρησιμοποιηθεί και μέσα σε ερωτήματα που ακολουθούν το τυπικό συντακτικό της SQLAlchemy

Παράδειγμα 6: text (βλ. βίντεο)

Άσκηση 2:

Μετατρέψτε σε SQLAlchemy το ερώτημα:

```
SELECT CONCAT(firstName, ' ', lastName) AS supervisedSF
FROM employees
WHERE reportsTo IN
        (
          SELECT employeeNumber
          FROM offices o JOIN employees e
                ON o.officeCode = e.officeCode
          WHERE o.city = 'San Francisco'
        )
```

Εισαγωγή

- Η εισαγωγή γίνεται ως insert(table) ή table.insert() που έχει ως μέθοδο τη values() η οποία παίρνει ως ορίσματα με λέξεις κλειδιά τις τιμές της νέας εγγραφής:

SQL
```
INSERT INTO actor(first_name, last_name)
VALUES ('Woody', 'McConaughey');
```

SQL Alchemy
```
insert(actor).values(
        first_name = "Woody",
        last_name = "McConaughey")
```

  - Εκτελούμε το παραπάνω με την connection.execute()
  - Παίρνουμε το κλειδί της νέας εγγραφής με την result.inserted_primary_key
- Μπορούμε να εισάγουμε πολλαπλές εγγραφές με το συντακτικό:

```
result = conn.execute(insert(actor), [
    {"first_name": "Matthew", "last_name":"Harelson"},
    {"first_name": "Colin", "last_name": "McAdams"}
])
```

- τέλος, μπορούμε να εισάγουμε εγγραφές με SELECT (INSERT INTO… SELECT) {βλ. βίντεο}

Σημαντικό:

- Οι τιμές ελέγχονται και αποφεύγονται εμπλοκές (π.χ SQL Injection)

Παράδειγμα 7: insert (βλ. βίντεο)

Ενημέρωση

- Η ενημέρωση δεδομένων γίνεται ως εξής:

SQL
```
UPDATE actor
SET last_name = 'Ali'
WHERE last_name = 'Harelson';
```

SQL Alchemy
```
update(actor).\
where(actor.c["last_name"] == "Harelson").\
values(last_name="Ali")
```

Διαγραφή

- Η διαγραφή δεδομένων γίνεται ως εξής:

SQL
```
DELETE FROM actor
WHERE actor_id >= 201;
```

SQL Alchemy
```
delete(actor).\
where(actor.c["actor_id"] >= 201)
```

Παράδειγμα 8: update_delete (βλ. βίντεο)

Άσκηση 3:

Μετατρέψτε σε SQLAlchemy το ερώτημα:

```
UPDATE country
SET headOfState = 'Boris Johnson'
WHERE headofState = 'Elisabeth II'
      OR headofState = 'Jacques Chirac';
```

ΜΑΘΗΜΑ 2: CORE: Δημιουργία ΒΔ

3. Δημιουργία Βάσης Δεδομένων

SQLAlchemy psounis

- Παρέχεται πλήρης δυνατότητα να ορίσουμε την βάση δεδομένων μας μέσω της SQLAlchemy.
- Βλέπουμε ένα ολοκληρωμένο παράδειγμα (με την επισήμανση ότι το συντακτικό είναι ακόμη πιο πλούσιο από όσα παρουσιάζονται εδώ)

SQL

```
CREATE TABLE employees (
  emp_no      INT             AUTO_INCREMENT,
  birth_date  DATE,
  first_name  VARCHAR(30)     NOT NULL,
  last_name   VARCHAR(50)     NOT NULL,
  gender      ENUM('M','F')   NOT NULL DEFAULT 'M',
  hire_date   DATE            NOT NULL,
  CONSTRAINT emp_pk PRIMARY KEY (emp_no));
```

SQL Alchemy

```
employees = Table('employees', metadata,
    Column('emp_no', Integer(), autoincrement=True),
    Column('birth_date', Date()),
    Column('first_name', String(30),  nullable=False),
    Column('last_name', String(50),  nullable=False),
    Column('gender', Enum('M', 'F'),  nullable=False, default='M'),
    Column('hire_date', Date(), nullable=False),
    PrimaryKeyConstraint('emp_no', name='emp_pk'))
```

SQL

```
CREATE TABLE departments (
    dept_no      CHAR(4),
    dept_name    VARCHAR(40)    NOT NULL UNIQUE,
    CONSTRAINT dept_pk PRIMARY KEY (dept_no));
CREATE INDEX departments_name_idx ON
                    departments(dept_name);
```

SQL Alchemy

```
departments = Table('departments', metadata,
    Column('dept_no', String(4)),
    Column('dept_name', String(40), nullable=False, unique=True),
    PrimaryKeyConstraint('dept_no', name='dept_pk'),
    Index('dept_name_idx', 'dept_name'))
```

SQL

```
CREATE TABLE dept_emp (
  emp_no INT,
  dept_no CHAR(4)
  from_date DATE NOT NULL,
  to_date   DATE NOT NULL,
  CONSTRAINT dept_emp_pk PRIMARY KEY (emp_no, dept_no),
  CONSTRAINT dept_emp_fk_emp FOREIGN KEY (emp_no) REFERENCES
                                        employees(emp_no),
  CONSTRAINT dept_emp_fk_dept FOREIGN KEY (dept_no) REFERENCES
                                        departments(dept_no),
  CONSTRAINT date_constraint CHECK (from_date<to_date));
```

SQL Alchemy

```
dept_emp = Table('dept_emp', metadata,
    Column('emp_no', Integer),
    Column('dept_no', String(4)),
    Column('from_date', Date(), nullable=False),
    Column('to_date', Date(), nullable=False),
    PrimaryKeyConstraint('emp_no', 'dept_no',
                         name='dept_emp_pk'),
    ForeignKeyConstraint(('emp_no',), ["employees.emp_no"],
                         name="fk1"),
    ForeignKeyConstraint(('dept_no',), ["departments.dept_no"],
                         name="fk2"),
    CheckConstraint('from_date < to_date', name="chk_date")
)
```

Παράδειγμα 9: createDB (βλ. βίντεο)

Τύποι Δεδομένων της SQLAlchemy:

- Η SQLAlchemy προσφέρει πληθώρα τύπων δεδομένων.
- Βλέπουμε τον πίνακα τη συσχέτιση με τους τύπους δεδομένων της MySQL και της Python:

Python	MySQL	SQL Alchemy
bool	BOOLEAN	Boolean
int	INT	Integer
int	BIGINT	BigInteger
float	FLOAT, DOUBLE	Float
decimal.Decimal	DECIMAL	Numeric
str	CHAR(N)	String
str	VARCHAR(N)	String
datetime.date	DATE	Date
datetime.time	TIME	Time
datetime.datetime	DATETIME	Datetime
byte	BLOB	LargeBinary
str	TEXT	Text

- Περισσότερα και εδώ:
  - https://docs.sqlalchemy.org/en/14/core/type_basics.html

Παράδειγμα 10: images.py και dates.py (βλ. βίντεο)

- Μία stored procedure μπορεί να κληθεί ως conn.execute(call), όπου call: “CALL proc_name(params)”

Συναλλαγές (Transactions):

- Δίνεται η δυνατότητα να προγραμματίσουμε την αλληλεπίδρασή μας με τη βάση, σε συναλλαγές (οπότε όταν χρησιμοποιούμε την SQLAlchemy, δεν απαιτείται ουσιαστικά να έχουμε stored procedures)
  - Ξεκινάμε τη συναλλαγή με τη connection.begin()
    - Επιστρέφει ένα αντικείμενο που περιτυλίσσει τη συναλλαγή (έστω transaction)
  - Εφόσον πήγαν όλα καλά και ολοκληρώσαμε τις ενέργειές μας, εκτελούμε την transaction.commit()
  - Αλλιώς (έχει προκύψει κάποια εξαίρεση), καλούμε την transaction.rollback()

```
transaction = conn.begin()

try:
  # some db work

  transaction.commit()

except Exception:
  transaction.rollback()

  print("An error occurred: The transaction was not completed.")
```

Παράδειγμα 11: transactions.py (βλ. βίντεο)