برای کار با داده‌های JSON در MySQL و PostgreSQL در پایتون، باید از کتابخانه‌های مخصوص این پایگاه داده‌ها استفاده کنید. در اینجا توضیح می‌دهم که چطور می‌توانید داده‌های JSON را در این پایگاه داده‌ها ذخیره، بازیابی و پردازش کنید.

1. کار با داده‌های JSON در MySQL با پایتون

برای اتصال به MySQL و کار با داده‌های JSON، باید از کتابخانه `mysql-connector-python` یا `PyMySQL` استفاده کنید.

1.1. نصب کتابخانه‌های مورد نیاز

```
pip install mysql-connector-python
```

1.2. اتصال به پایگاه داده MySQL

```
import mysql.connector

# اتصال به پایگاه داده
conn = mysql.connector.connect(
    host="localhost",
    user="root",
    password="password",
    database="your_database"
)

cursor = conn.cursor()
```

1.3. ذخیره داده‌های JSON در MySQL

برای ذخیره داده‌های JSON در MySQL، باید از نوع داده `JSON` در MySQL استفاده کنید. می‌توانید داده‌ها را به‌صورت یک رشته JSON وارد کنید.

```
import json

data = {
    "brand": "Dell",
    "price": 1000,
    "specs": {"cpu": "Intel i7", "ram": "16GB"}
}

json_data = json.dumps(data)  # تبدیل به JSON string

# دستور SQL برای درج داده
cursor.execute("INSERT INTO products (name, details) VALUES (%s, %s)", ('Laptop', json_data))

conn.commit()
```

1.4. بازیابی داده‌های JSON از MySQL

برای بازیابی داده‌های JSON، از تابع `()fetchone` یا `()fetchall` استفاده می‌کنید.

```
cursor.execute("SELECT name, details FROM products")
result = cursor.fetchall()

for row in result:
    name = row[0]
    details = json.loads(row[1])  # تبدیل JSON string به دیکشنری
    print(name, details)
```

1.5. به‌روزرسانی داده‌های JSON در MySQL

برای به‌روزرسانی داده‌های JSON، از دستور `()JSON_SET` می‌توانید استفاده کنید.

```
new_data = {"price": 1200}
new_json_data = json.dumps(new_data)

cursor.execute("UPDATE products SET details = JSON_SET(details, '$.price', %s) WHERE name = %s",
(new_json_data, 'Laptop'))
conn.commit()
```

1.6. حذف داده‌های JSON در MySQL

برای حذف مقادیر از داخل داده JSON، از تابع `()JSON_REMOVE` استفاده می‌کنید:

```
cursor.execute("UPDATE products SET details = JSON_REMOVE(details, '$.specs') WHERE name = %s",
('Laptop',))
conn.commit()
```

---

2. کار با داده‌های JSON در PostgreSQL با پایتون

برای اتصال به PostgreSQL و کار با داده‌های JSON یا `JSONB`، باید از کتابخانه `psycopg2` استفاده کنید.

2.1. نصب کتابخانه‌های مورد نیاز

```
pip install psycopg2
```

2.2. اتصال به پایگاه داده PostgreSQL

```
import psycopg2

# اتصال به پایگاه داده
conn = psycopg2.connect(
    dbname="your_database",
    user="postgres",
    password="password",
    host="localhost"
)

cursor = conn.cursor()
```

2.3. ذخیره داده‌های JSONB در PostgreSQL

برای ذخیره داده‌های JSONB، باید از نوع داده `JSONB` در PostgreSQL استفاده کنید. می‌توانید داده‌ها را به‌صورت یک دیکشنری Python و سپس به فرمت JSON تبدیل کنید.

```
import json

data = {
    "brand": "Dell",
    "price": 1000,
    "specs": {"cpu": "Intel i7", "ram": "16GB"}
}

# دستور SQL برای درج داده
cursor.execute("INSERT INTO products (name, details) VALUES (%s, %s)", ('Laptop', json.dumps(data)))

conn.commit()
```

2.4. بازیابی داده‌های JSONB از PostgreSQL

برای بازیابی داده‌های JSONB، می‌توانید از تابع `()fetchone` یا `()fetchall` استفاده کنید و سپس داده‌ها را به دیکشنری Python تبدیل کنید.

```
cursor.execute("SELECT name, details FROM products")
result = cursor.fetchall()

for row in result:
    name = row[0]
    details = json.loads(row[1])  # تبدیل JSONB به دیکشنری
    print(name, details)
```

2.5. به‌روزرسانی داده‌های JSONB در PostgreSQL

برای به‌روزرسانی داده‌های JSONB، می‌توانید از تابع `()jsonb_set` استفاده کنید.

```
new_data = {"price": 1200}

cursor.execute("UPDATE products SET details = jsonb_set(details, '{price}', %s) WHERE name = %s",
(json.dumps(new_data), 'Laptop'))
conn.commit()
```

2.6. حذف داده‌های JSONB در PostgreSQL

برای حذف مقادیر از داخل داده JSONB، می‌توانید از عملگر `-` استفاده کنید:

```
cursor.execute("UPDATE products SET details = details - 'specs' WHERE name = %s", ('Laptop',))
conn.commit()
```

---

3. مقایسه JSON در MySQL و PostgreSQL از نظر پایتون

ویژگی	MySQL (JSON)	PostgreSQL (JSONB)
نوع داده	`JSON`	`JSON`, `JSONB`
ذخیره‌سازی	متنی	باینری
کارایی جستجو	پایین‌تر از `JSONB`	سریع‌تر از `JSON`
پشتیبانی از ایندکس	ندارد (در نسخه‌های قبلی)	پشتیبانی از ایندکس‌های GIN و GiST
پردازش داده‌ها	توابع JSON مشابه PostgreSQL	پردازش بهینه‌تر از داده‌های JSON

---

4. نتیجه‌گیری

با استفاده از کتابخانه‌های `mysql-connector-python` برای MySQL و `psycopg2` برای PostgreSQL، می‌توانید به‌راحتی داده‌های JSON را در پایگاه داده‌های MySQL و PostgreSQL ذخیره، بازیابی و به‌روزرسانی کنید. برای ذخیره داده‌ها، از نوع داده‌های `JSON` در MySQL و `JSONB` در PostgreSQL استفاده کنید تا بهترین کارایی را داشته باشید.