کار با داده‌های JSON در SQLite

SQLite از JSON به عنوان یک نوع داده‌ای پشتیبانی نمی‌کند، اما می‌توان JSON را به صورت **رشته** (`TEXT`) در پایگاه داده ذخیره و هنگام خواندن آن را به **دیکشنری پایتون** تبدیل کرد.

در این بخش یاد می‌گیریم که چگونه **داده‌های JSON را در SQLite ذخیره، بازیابی و پردازش کنیم**.

---

۱. ذخیره داده‌های JSON در SQLite

برای ذخیره داده‌های JSON در SQLite، باید آن را به رشته تبدیل کنیم. این کار را با `()json.dumps` انجام می‌دهیم.

مثال: ایجاد جدول و ذخیره داده‌های JSON

```
import sqlite3
import json

# اتصال به پایگاه داده
conn = sqlite3.connect("test.db")
cursor = conn.cursor()

# ایجاد جدول با فیلد JSON
cursor.execute('''
    CREATE TABLE IF NOT EXISTS users (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        name TEXT,
        details TEXT  -- فیلد برای ذخیره JSON
    )
''')

# داده JSON
user_data = {
    "age": 25,
    "email": "ali@example.com",
    "skills": ["Python", "Django", "SQL"]
}

# تبدیل JSON به رشته
json_data = json.dumps(user_data)

# افزودن داده به پایگاه داده
cursor.execute("INSERT INTO users (name, details) VALUES (?, ?)", ("علی", json_data))

# ذخیره تغییرات
conn.commit()
conn.close()
```

✅ **نکات:**

- داده‌های JSON با `()json.dumps` به رشته تبدیل و در فیلد `TEXT` ذخیره می‌شوند.
- مقدار `skills` در قالب یک لیست (`list`) در JSON ذخیره شده است.

۲. بازیابی داده‌های JSON از SQLite

برای بازیابی داده‌های JSON از پایگاه داده، مقدار ذخیره‌شده را با `()json.loads` دوباره به دیکشنری تبدیل می‌کنیم.

مثال: خواندن و تبدیل داده‌های JSON

```
conn = sqlite3.connect("test.db")
cursor = conn.cursor()

# خواندن داده از پایگاه داده
cursor.execute("SELECT name, details FROM users WHERE name = ?", ("علی",))
user = cursor.fetchone()

if user:
    name, json_data = user
    details = json.loads(json_data)  # تبدیل رشته به دیکشنری
    print(f"نام: {name}")
    print(f"سن: {details['age']}")
    print(f"ایمیل: {details['email']}")
    print(f"مهارت‌ها: {', '.join(details['skills'])}")

conn.close()
```

✅ خروجی:

```
نام: علی
سن: 25
ایمیل: ali@example.com
مهارت‌ها: Python, Django, SQL
```

📌 نکات:

- مقدار `details` از رشته JSON به دیکشنری تبدیل شده است.
- از `()json.loads` برای تبدیل رشته JSON استفاده شده است.

۳. به‌روزرسانی داده‌های JSON در SQLite

اگر بخواهیم مقدار خاصی از JSON ذخیره‌شده را تغییر دهیم، باید ابتدا آن را بخوانیم، مقدار جدید را در دیکشنری اعمال کنیم و دوباره ذخیره کنیم.

مثال: افزودن مهارت جدید به JSON ذخیره‌شده

```
conn = sqlite3.connect("test.db")
cursor = conn.cursor()

# دریافت داده JSON
cursor.execute("SELECT details FROM users WHERE name = ?", ("علی",))
user = cursor.fetchone()
```

```
if user:
    details = json.loads(user[0])  # تبدیل رشته به دیکشنری
    details["skills"].append("Flask")  # افزودن مهارت جدید

    # و ذخیره در پایگاه داده JSON تبدیل دیکشنری به رشته
    json_data = json.dumps(details)
    cursor.execute("UPDATE users SET details = ? WHERE name = ?", (json_data, "علی"))

    conn.commit()
    print("مهارت جدید اضافه شد!")

conn.close()
```

✅ **در این مثال:**

- ابتدا داده JSON از پایگاه داده دریافت می‌شود.
- مقدار جدید (`"Flask"`) به لیست `skills` اضافه می‌شود.
- داده JSON جدید دوباره در پایگاه داده ذخیره می‌شود.

---

۴. جستجو در داده‌های JSON

از آنجایی که SQLite توابع داخلی برای پردازش JSON ندارد (برخلاف **PostgreSQL** یا **MySQL**)، باید از روش‌های جایگزین استفاده کنیم.
یکی از روش‌ها استفاده از **LIKE** در کوئری‌ها برای جستجو درون JSON است.

مثال: پیدا کردن کاربران دارای مهارت خاص

```
skill_to_search = "Python"

conn = sqlite3.connect("test.db")
cursor = conn.cursor()

# جستجو در JSON با استفاده از LIKE
cursor.execute("SELECT name FROM users WHERE details LIKE ?", (f'%{skill_to_search}%',))

users = cursor.fetchall()
for user in users:
    print(f"کاربر: {user[0]}")

conn.close()
```

✅ **در این مثال:**

- از `LIKE` برای جستجوی وجود `Python` در `details` استفاده شده است.
- این روش کار می‌کند اما ایده‌آل نیست؛ برای جستجوی پیشرفته بهتر است از پایگاه داده‌هایی مثل **PostgreSQL** استفاده شود.

---

۵. حذف داده‌های JSON در SQLite

برای حذف فیلدهای خاص از JSON ذخیره‌شده، باید ابتدا آن را دریافت کنیم، مقدار مورد نظر را حذف کنیم، و سپس دوباره ذخیره کنیم.

مثال: حذف `email` از JSON ذخیره‌شده

```
conn = sqlite3.connect("test.db")
cursor = conn.cursor()

# دریافت داده JSON
cursor.execute("SELECT details FROM users WHERE name = ?", ("علی",))
user = cursor.fetchone()

if user:
    details = json.loads(user[0])  # تبدیل رشته به دیکشنری
    details.pop("email", None)  # حذف کلید 'email'

    # تبدیل و ذخیره دوباره در پایگاه داده JSON جدید
    json_data = json.dumps(details)
    cursor.execute("UPDATE users SET details = ? WHERE name = ?", (json_data, "علی"))

    conn.commit()
    print("ایمیل حذف شد!")

conn.close()
```

✅ **در این مثال:**

- مقدار `email` حذف شده و JSON جدید دوباره در پایگاه داده ذخیره شده است.

جمع‌بندی

✅ **ذخیره JSON** → تبدیل JSON به `TEXT` با `json.dumps()`
✅ **خواندن JSON** → تبدیل `TEXT` به دیکشنری با `json.loads()`
✅ **به‌روزرسانی JSON** → تغییر مقدار و ذخیره مجدد در پایگاه داده
✅ **جستجو در JSON** → استفاده از `LIKE` در SQL (محدودیت دارد)
✅ **حذف مقادیر خاص از JSON** → حذف مقدار و ذخیره مجدد

📌 این روش‌ها راهکارهای کاربردی برای کار با JSON در SQLite هستند، اما اگر نیاز به جستجو و پردازش پیشرفته‌تر داشته باشید، پیشنهاد می‌شود از پایگاه داده‌هایی مثل **PostgreSQL** استفاده کنید که پشتیبانی بومی از JSON دارند. 🚀