مدیریت تراکنش‌ها (Transactions) در SQLite

تراکنش‌ها در پایگاه داده به شما این امکان را می‌دهند که مجموعه‌ای از عملیات را به صورت یک واحد انجام دهید. اگر همه عملیات موفق باشند، تغییرات ثبت می‌شوند، اما اگر خطایی رخ دهد، می‌توان همه تغییرات را به وضعیت قبلی برگرداند.

۱. مفهوم تراکنش در SQLite

SQLite به صورت پیش‌فرض هر دستور `INSERT`, `UPDATE`, `DELETE` را بلافاصله اجرا و ذخیره می‌کند. اما اگر می‌خواهید چند عملیات را به صورت یک تراکنش انجام دهید، باید از `BEGIN TRANSACTION`, `COMMIT`, و `ROLLBACK` استفاده کنید.

- `COMMIT()` → ذخیره تغییرات در پایگاه داده (تراکنش موفق)
- `ROLLBACK()` → بازگرداندن تغییرات به قبل از تراکنش در صورت خطا

---

۲. استفاده از `commit()` برای ذخیره تغییرات

زمانی که چندین عملیات را به صورت یک تراکنش انجام می‌دهید، باید در پایان `commit()` را فراخوانی کنید تا تغییرات ثبت شوند.

مثال: ذخیره تغییرات پس از انجام چندین عملیات

```
import sqlite3

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

try:
    # شروع تراکنش
    cursor.execute("BEGIN TRANSACTION")

    # اضافه کردن چندین کاربر
    cursor.execute("INSERT INTO users (name, age, email) VALUES (?, ?, ?)", ("احمد", 27,
"ahmad@example.com"))
    cursor.execute("INSERT INTO users (name, age, email) VALUES (?, ?, ?)", ("زهرا", 24, "zahra@example.com"))

    # ذخیره تغییرات
    conn.commit()
    print("تراکنش با موفقیت انجام شد!")

except sqlite3.Error as e:
    print("خطا در تراکنش:", e)
    conn.rollback()  # بازگرداندن تغییرات در صورت بروز خطا

finally:
    conn.close()
```

📌 **توضیح:**

- اگر هر دو `INSERT` موفق باشند، `commit()` اجرا شده و تغییرات ثبت می‌شوند.

- اگر خطایی رخ دهد، `()rollback` همه تغییرات را به حالت قبل بازمی‌گرداند.
- `finally` برای اطمینان از بسته شدن اتصال پایگاه داده استفاده شده است.

---

۳. استفاده از `()rollback` برای برگشت تغییرات در صورت خطا

در شرایطی که خطایی در طول اجرای تراکنش رخ دهد، می‌توان با `()rollback` تغییرات را لغو کرد.

مثال: برگشت تغییرات در صورت وقوع خطا

```
import sqlite3

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

try:
    cursor.execute("BEGIN TRANSACTION")

    # درج یک رکورد صحیح
    cursor.execute("INSERT INTO users (name, age, email) VALUES (?, ?, ?)", ("29 ,"مریم,
"maryam@example.com"))

    # (درج یک رکورد نامعتبر (ایمیل تکراری باعث خطا می•شود
    cursor.execute("INSERT INTO users (name, age, email) VALUES (?, ?, ?)", ("31 ,"حسن,
"maryam@example.com"))

    conn.commit()  # این خط اجرا نمی•شود اگر خطا رخ دهد
    print("تراکنش موفقیت•آمیز بود!")

except sqlite3.IntegrityError as e:
    print("خطای یکپارچگی داده•ها:", e)
    conn.rollback()  # برگرداندن تغییرات در صورت بروز خطا
    print("تراکنش به حالت قبل برگشت!")

finally:
    conn.close()
```

📌 **توضیح:**

- خطای `IntegrityError` رخ می‌دهد زیرا ایمیل "maryam@example.com" قبلاً وجود دارد (ستون `email` مقدار `UNIQUE` دارد).
- `()rollback` باعث می‌شود که هیچ‌کدام از `INSERT` ها ذخیره نشوند.

---

۴. مدیریت خطاها با استفاده از `try-except`

در کار با پایگاه داده، همیشه باید برای مدیریت خطاها از `try-except` استفاده کنیم تا در صورت بروز خطا، تراکنش را لغو کنیم.

مثال: مدیریت خطاهای عمومی SQLite

```
import sqlite3

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

try:
    cursor.execute("BEGIN TRANSACTION")

    # انجام چند عملیات
    cursor.execute("UPDATE users SET age = age + 1 WHERE name = ?", ("مریم",))
    cursor.execute("DELETE FROM users WHERE name = ?", ("علی",))

    conn.commit()  # ذخیره تغییرات
    print("تراکنش موفقیت‌آمیز بود!")

except sqlite3.Error as e:
    print("خطایی رخ داد:", e)
    conn.rollback()  # برگرداندن تغییرات
    print("تراکنش لغو شد!")

finally:
    conn.close()
```

📌 **توضیح:**

- تمام تغییرات با هم انجام می‌شوند و در صورت موفقیت `()commit` ذخیره می‌شود.
- اگر خطایی رخ دهد، `()rollback` همه تغییرات را لغو می‌کند.

---

۵. تراکنش‌های خودکار (Auto-Commit Mode)

به‌صورت پیش‌فرض، SQLite در "حالت خودکار ذخیره" (`Auto-Commit Mode`) کار می‌کند، یعنی هر تغییر بلافاصله ذخیره می‌شود. برای غیرفعال کردن این حالت، باید صراحتاً یک تراکنش را آغاز کنید.

غیرفعال کردن `Auto-Commit`

```
conn = sqlite3.connect("test.db")
conn.isolation_level = None  # غیرفعال کردن حالت Auto-Commit
cursor = conn.cursor()

cursor.execute("BEGIN TRANSACTION")

try:
    cursor.execute("INSERT INTO users (name, age, email) VALUES (?, ?, ?)", ("حسین", 33,
"hossein@example.com"))
    cursor.execute("INSERT INTO users (name, age, email) VALUES (?, ?, ?)", ("لیلا", 26, "leila@example.com"))

    conn.commit()
    print("تراکنش موفقیت‌آمیز انجام شد!")

except sqlite3.Error as e:
```

```
    print("خطا در تراکنش:", e)
    conn.rollback()

finally:
    conn.close()
```

📌 **توضیح:**

- با `conn.isolation_level = None` حالت خودکار ذخیره غیرفعال شده است.
- `BEGIN TRANSACTION` برای شروع تراکنش به‌طور دستی استفاده شده است.

جمع‌بندی

✅ `()commit` → برای ذخیره تغییرات پس از موفقیت عملیات
✅ `()rollback` → برای لغو تغییرات در صورت بروز خطا
✅ `try-except` → برای مدیریت خطاها و جلوگیری از از دست رفتن داده‌ها
✅ SQLite → `Auto-Commit` به صورت پیش‌فرض ذخیره خودکار دارد، اما می‌توان آن را غیرفعال کرد

با این روش‌ها، می‌توانید تراکنش‌های پایدار و ایمن‌تری در SQLite داشته باشید. اگر سوالی داشتی بپرس! 🚀