آزمایشگاه پایگاه داده

جلسه هفتم
تراکنش ها

محمد جواد آکوچکیان و محمود فرجی
سال تحصیلی ۱۴۰۱-۱۴۰۲

تراکنش(Transactions)

- تراکنش :

عبارت است از یک واحد کاری که اگر اجرای کل آن موفقیت آمیز باشد تمامی اطلاعاتی که در ضمن آن تغییر یافته اند به طور دائمی در دیتابیس ذخیره میشوند (Commit) و اگر دچار اشکار شود تمامی تغییرات انجام شده به قبل (Rollback) باز خواهند گشت.

خواص Transactions

- Atomicity : اتمیک باشند بطوریکه در یک واحد عملیاتی یا همه دستورات به درستی انجام شود یا هیچکدام انجام نشود. اگر در اجرا مشکلی پیش آمد تمامی تغییرات به قبل بازخواهند گشت.

- Consistency: سازگاری یه این معنا است که هر تراکنش پایگاه داده را از یک وضعیت صحیح و معتبر به وضعیت معتبر درگری میبرد. مثلا این که مجموع پول تمام حساب ها در یک بانک بعد تراکنش ها مقدار ثابتی دارد.

- Isolation: تغییرات ایجاد شده توسط یک تراکنش باید از تغییرات ناشی از تراکنش های همزمان دیگر ایزوله و مجزا باشد. یک تراکنش اطلاعات را هم در وضعیت قبل از تغییر توسط تراکنش دیگر می بیند و هم در وضعیت بعد از تراکنش دیگ ولی اطلاعات را در وضعیت مابین اجرای تراکنش دیگر نمی بیند.

- Durability: پایداری یا دوام به این معنا است که هنگامی که یک تراکنش کامل شد تغییرات ناشی از آن بطور دائمی در پایگاه داده نوشته میشوند یعنی هنگامی که یک تراکنش بطور کامل انجام شد. حتی اگر SQL Server دچار مشکل شود و سرویس restart شود اطمینان داریم که تغییرات ناشی از تراکنش در اطلاعات جداول لحاظ شده است. در مجموع این خاصیت تضمین میکند که تراکنش کامل شده حتما در اطلاعات جدول لحاظ خواهد شد.

انواع تراکنش

- Autocommit transaction
- Implicit Transaction
- Explicit Transaction
- Control Of Flow
- ISOLATION LEVEL

Autocommit transaction

- Autocommit transaction :
  - هر عبارت T SQL به عنوان یک تراکنش است
  - هر دستور T SQL هنگامی که کامل شود امکان commit یا rollback را دارد.
  - اگر کامل و بدون خطا انجام شود در نهایت تغییرات commit خواهند شد اما اگر خطایی رخ دهد کل تغییرات rollback میشود
  - تراکنش های SQL Server به صورت پیشفرض از این نوع تراکنش است.

Autocommit transaction

```
--Autocommit transaction
insert into s values (1,'javad',20)
insert into s values (2,'ali',10)
insert into s valuess (3,'ahmad',16)
insert into s values (4,'zahra',15)
```

150 %

Messages

```
Msg 102, Level 15, State 1, Line 8
Incorrect syntax near 'valuess'.
```

- مثال Autocommit transaction :
- برای دستورات دارای خطای syntax

```
select * from s
```

150 %

Results | Messages

sid	sname	score

Autocommit transaction

```sql
--Autocommit transaction
insert into s values (1,'javad',20)
insert into s values (2,'ali',10)
insert into s values (2,'ahmad',16)
insert into s values (4,'zahra',15)
```

150 %

Messages

```
(1 row affected)

(1 row affected)
Msg 2627, Level 14, State 1, Line 12
Violation of PRIMARY KEY constraint 'PK__s__DDDFDD36ABA7A39D'. Cannot insert duplicate key
The statement has been terminated.

(1 row affected)

Completion time: 2023-05-09T04:59:49.0870770-07:00
```

- مثال Autocommit transaction :
- برای دستورات دارای خطای run time

	sid	sname	score
1	1	javad	20
2	2	ali	10
3	4	zahra	15

Implicit Transaction

- با استفاده از مد Implicit Transaction باید commit یا rollback شدن دستورات را خودمان کنترل کنیم.
- در این مد ما شروع تراکنش ها را تعیین نمیکنیم اما در انتهای تراکنش میتوان وضعیت commit یا rollback را مشخص کرد.
- به صورت خودکار پس از هر تراکنش SQL Server یک تراکنش جدید را آغاز خواهد کرد.
- زنجیره پیوسته ای از تراکنش ها به وجود می آید که با دستور set implicit_transactions on شروع شده و با دستور set implicit_transactions off پایان میابد

Implicit Transaction

- مثال از Implicit Transaction

```
set implicit_transactions on
insert into s values (5,'amir',10)
insert into s values (6,'tina',16)
insert into s values (7,'jafar',15)
select * from s
rollback
select * from s
```

Results | Messages

	sid	sname	score
1	1	javad	20
2	2	ali	10
3	4	zahra	15
4	5	amir	10
5	6	tina	16
6	7	jafar	15

	sid	sname	score
1	1	javad	20
2	2	ali	10
3	4	zahra	15

Explicit Transaction

- با استفاده از Explicit Transaction ها به طور صریح شروع و پایان تراکنش را مشخص میکنیم.
- برای شروع تراکنش از دستور BEGIN TRANSACTION استفاده میشود.
- برای کامیت کردن تراکنش از COMMIT TRANSACTION یا COMMIT WORK استفاده میشود.
- برای رول بک کردن تراکنش و بازگشت به حالت قبل از تراکنش از دستور ROLLBACK TRANSACTION یا ROLLBACK WORK استفاده میشود.

Explicit Transaction

```
select * from s
begin transaction
insert into s values (5,'amir',10)
insert into s values (6,'tina',16)
insert into s values (7,'jafar',15)
select * from s
rollback
select * from s
insert into s values (8,'tina',16)
insert into s values (9,'jafar',15)
rollback
select * from s
```

Results | Messages

	sid	sname	score
1	1	javad	20
2	2	ali	10
3	4	zahra	15

	sid	sname	score
1	1	javad	20
2	2	ali	10
3	4	zahra	15
4	5	amir	10
5	6	tina	16
6	7	jafar	15

	sid	sname	score
1	1	javad	20
2	2	ali	10
3	4	zahra	15

	sid	sname	score
1	1	javad	20
2	2	ali	10
3	4	zahra	15
4	8	tina	16
5	9	jafar	15

- مثال از Explicit Transaction

تابع @@ERROR

- این تابع کد اروری که رخ داده است را در خود نگه میدارد و مانند یک متغیر بعد از بروز ارور میتوان به این کد دسترسی داشت.

- در صورت نبود خطا در اجرای تراکنش مقدار این تابع ۰ خواهد بود.

```
select * from s
print @@error
```

150 %

Results | Messages

```
(5 rows affected)
0
```

```
insert into s values (10,'sobhan',14)
insert into s values (10,'sobhan',14)
print @@error
```

150 %

Messages

```
(1 row affected)
Msg 2627, Level 14, State 1, Line 45
Violation of PRIMARY KEY constraint 'PK__s__DDDFDD36ABA7A39D'. Cannot insert duplicate key i
The statement has been terminated.
2627
```

تابع @@ERROR

- مثال

```
select * from s
begin transaction
insert into s values (5,'amir',10)
insert into s values (6,'tina',16)
insert into s values (5,'jafar',15)
if @@error>0
rollback transaction
else
commit transaction
select * from s
```

Results | Messages

	sid	sname	score
1	1	javad	20
2	2	ali	10
3	4	zahra	15

	sid	sname	score
1	1	javad	20
2	2	ali	10
3	4	zahra	15

تابع @@ERROR

- نکته: در صورت وجود فاصله هنگام ارور و اجرای آن ممکن است کد ارور در تابع ثبت نشود.

```
select * from s
begin transaction
insert into s values (5,'amir',10)
insert into s values (5,'tina',16)
insert into s values (6,'jafar',15)
if @@error>0
rollback transaction
else
commit transaction
select * from s
```

Results | Messages

	sid	sname	score
1	1	javad	20
2	2	ali	10
3	4	zahra	15

	sid	sname	score
1	1	javad	20
2	2	ali	10
3	4	zahra	15
4	5	amir	10
5	6	jafar	15

Control flow try … catch

- برای کنترل خطا در SQL میتوان از ساختار زیر استفاده کرد.

```
BEGIN TRY
    { sql_statement | statement_block }
END TRY
BEGIN CATCH
    [ { sql_statement | statement_block } ]
END CATCH
[ ; ]
```

Control flow try … catch

- مثال

```sql
select * from s
begin transaction
begin try
insert into s values (5,'amir',10)
insert into s values (5,'tina',16)
insert into s values (6,'jafar',15)
commit transaction
end try
begin catch
rollback transaction
end catch
select * from s
```

Results | Messages

	sid	sname	score
1	1	javad	20
2	2	ali	10
3	4	zahra	15

	sid	sname	score
1	1	javad	20
2	2	ali	10
3	4	zahra	15

Save transaction

- با استفاده از **savepoint** میتوانیم تنها قسمتی از یک تراکنش را **rollback** کنیم.
- هنگامی که یک **savepoint** داشته باشیم و به آن بازگردیم هنوز تراکنش ما پایان نیافته است و منابع ما آزاد نشده است.
- بعد از **rollback transaction [savepoint name]** باید الزاما تراکنش را **rollback** یا **commit** کنیم تا منابع آزاد شود.

Save transaction

```
select * from s
begin transaction
begin try
insert into s values (5,'amir',10)
save transaction point1
insert into s values (6,'tina',16)
insert into s values (5,'jafar',15)
commit transaction
end try
begin catch
rollback transaction point1
commit transaction
end catch
select * from s
```

- مثال

150 %

Results | Messages

	sid	sname	score
1	1	javad	20
2	2	ali	10
3	4	zahra	15

	sid	sname	score
1	1	javad	20
2	2	ali	10
3	4	zahra	15
4	5	amir	10

دستور کار

۱) ابتدا با استفاده از دستورات T-SQL یک جدول person با ستون‌های PersonID، FirstName، LastName، Address، City، Age تعریف کنید. (طول و نوع مقادیر ستون‌ها را متناسب با کاربرد آن‌ها به اختیار تعیین کنید)

۲) سپس تعداد ۹ عدد کوئری Insert را به صورت یک تراکنش با داده‌های دلخواه بر روی این جدول اجرا کنید.

۳) در مرحله بعد نیاز است یک Implicit Transaction را فعال و در ادامه‌ی آن یک کوئری Update و به دنبال آن یک کوئری Select اجرا کنید. دقت کنید کوئری Update می‌بایست مربوط به جدول Person (یک رکورد از جدول Person را آپدیت کنی) و کوئری Select مربوط به دریافت تعداد تراکنش‌های فعال در سیستم بوده که خروجی آن در قالب یک ستون دلخواه نمایش داده می‌شود. (راهنمایی: برای دریافت تعداد تراکنش‌های فعال می‌توانید از تابع TRANCOUNT استفاده کنید و مقدار آن را در یک ستون ریخته و نمایش دهید.)

۴) سپس نیاز است تا تراکنش را commit کرده و مجددا دستور select را برای دریافت تراکنش‌های فعال-به شیوه گفته شده در گام ۳- اجرا نمایید.

دستور کار

۵) گام ۳ و ۴ را برای Explicit Transaction ها نیز تکرار کنید.

۶) در این قسمت نیاز است یک تراکنش را شروع (Begin) کرده و در آن یک رکورد داده را در جدول Person، درج کنید. سپس با استفاده از دستور Save Transaction [Transaction_Name] تراکنش فعلی را با انتخاب نام دلخواه ذخیره کنید. (با این کار یک Save Point برای بازگشت به وضعیتی خاص از تراکنش تهیه می‌شود.)

۷) پس از آن یک رکورد داده را از جدول Person حذف کنید. (دقت کنید این مرحله نیز در ادامه تراکنشی که در گام ۶ تعریف کرده‌اید در حال تعریف و اجراست و تراکنش گام ۶ در این مرحله هنوز Commit نشده است)

۸) سپس کل رکوردهای جدول Person را Select کنید.

۹) تراکنشی را که با نام دلخواه در گام ۶ Save Point کرده‌اید را Rollback کنید.

۱۰) ابتدا تراکنش فعلی (که در گام ۶ تعریف شده است) را Commit کرده و سپس کل رکوردهای جدول Person را Select نمایید.

** دقت کنید از تمامی ۱۰ گام گفته شده، کوئری‌ها و خروجی‌ها اسکرین شات گرفته و همراه با توضیحات کامل، جامع و مانع در گزارش آزمایشگاه نمایش دهید.