آزمایشگاه پایگاه داده

جلسه هشتم

تراکنش ها (قسمت دوم)

محمد جواد آکوچکیان و محمود فرجی

سال تحصیلی 1402-1401

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 دیگر 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

ISOLATION LEVEL

- ISOLATION LEVEL:

- تراکنش ها این امکان را دارند که در میان تراکنش های همروند قابلیت ISOLATION LEVEL خود را تغییر دهند.
- منابعی که یک تراکنش میخواهد استفاده کند میتواند در برابر تراکنش های دیگر کاملا ایزوله شود
- یک تراکنش زمانی که از یک قلم داده استفاده میکند بر روی آن یک قفل ایجاد میکند و پس از تکمیل تراکنش آن قفل را آزاد میکند.
- گرفتن و آزاد شدن قفل ها از ISOLATION LEVEL تاثیر نمی پذیرد

ISOLATION LEVEL

- انواع ISOLATION LEVEL:

READ UNCOMMITTED -

READ COMMITTED -

REPEATABLE READ -

SNAPSHOT -

SERIALIZABLE -

READ UNCOMMITTED

- در این حالت دستورات TSQL دیگر میتوانند اطلاعات تغییر یافته یک تراکنش دیگر را حتی اگر کامل نشده باشد مشاهده کنند.
- این حالت میتواند دو حالت دیگر پیدا کند:
  - Dirty Read: زمانی رخ میدهد که ما اطلاعاتی را از یک تراکنش که هنوز CMMIT نشده است میخوانیم.
  - PHANTOM READS: هنگامی رخ میدهد که بعد از این که شما اطلاعاتی را خواندید و بر روی آن درحال کار هستید یک تراکنش دیگر آنها را تغییر دهد که در این صورت اگر دوباره اطلاعات را بخوانیم نتایج متفاوتی دریافت میکنیم.

READ UNCOMMITTED

- در این حالت دستورات TSQL دیگر میتوانند اطلاعات تغییر یافته یک تراکنش دیگر را حتی اگر کامل نشده باشد مشاهده کنند.
- این حالت میتواند دو حالت دیگر پیدا کند:
  - Dirty Read: زمانی رخ میدهد که ما اطلاعاتی را از یک تراکنش که هنوز CMMIT نشده است میخوانیم.
  - PHANTOM READS: هنگامی رخ میدهد که بعد از این که شما اطلاعاتی را خواندید و بر روی آن درحال کار هستید یک تراکنش دیگر آنها را تغییر دهد که در این صورت اگر دوباره اطلاعات را بخوانیم نتایج متفاوتی دریافت میکنیم.

یادآوری

دستور WAITFOR:

```
WAITFOR {DELAY 'time' | TIME 'time'}
```

```
--Pause for ten seconds
WAITFOR delay '000:00:10';

PRINT 'Done';

--Pause until a certain time
WAITFOR time '12:00:00';

PRINT 'It is noon';
```

READ UNCOMMITTED

- مثال READ UNCOMMITTED: تراکنش 2 تغییرات را قبل از اتمام تراکنش 1 میخواند

```
begin transaction
update s set score3=20
waitfor delay '00:00:10'
rollback
select * from s
```

1

150 %

Results | Messages

	sid	sname	score	score2	score3
1	1	javad	20	10	15
2	2	ali	10	10	15
3	4	zahra	15	10	15
4	5	amir	10	10	15

```
set transaction isolation level
read uncommitted
select * from s
```

2

150 %

Results | Messages

	sid	sname	score	score2	score3
1	1	javad	20	10	20
2	2	ali	10	10	20
3	4	zahra	15	10	20
4	5	amir	10	10	20

READ COMMITTED

- حالت پیش فرض خود SQL Server این حالت است و اگر به صورت دستی خودمان حالت ISOLATION LEVEL را مشخص نکنیم SQL Server تراکنش ها را در این حالت اجرا میکند.
- هر تراکنشی تنها میتواند اطلاعات commit شده را بخواند.

READ COMMITTED

- مثال READ COMMITTED:تراکنش 2 منتظر میماند و بعد از تراکنش 1 اجرا میشود.

```
begin transaction
update s set score3=20
waitfor delay '00:00:10'
rollback
select * from s
```

1

150 %

Results | Messages

	sid	sname	score	score2	score3
1	1	javad	20	10	15
2	2	ali	10	10	15
3	4	zahra	15	10	15
4	5	amir	10	10	15

```
set transaction isolation level
read committed
select * from s
```

2

150 %

Results | Messages

	sid	sname	score	score2	score3
1	1	javad	20	10	15
2	2	ali	10	10	15
3	4	zahra	15	10	15
4	5	amir	10	10	15

REPEATABLE READ

- مانند حالت read commited است با این تفاوت که اکر شما در متن یک تراکنش یک دستور select را دوبار اجرا کنید این حالت تضمین میکند که در هر دو اجرا خروجی های یکسانی را دریافت خواهید کرد.
- باید توجه داشت که برای عمل insert این اتفاق نمی افتد.

READ COMMITTED

```
set transaction isolation level
repeatable read
begin transaction
select * from s
waitfor delay '00:00:10'
select * from s
rollback
```

150 %

Results | Messages

	sid	sname	score	score2	score3
1	1	javad	20	10	15
2	2	ali	10	10	15
3	4	zahra	15	10	15
4	5	amir	10	10	15

	sid	sname	score	score2	score3
1	1	javad	20	10	15
2	2	ali	10	10	15
3	4	zahra	15	10	15
4	5	amir	10	10	15

```
update s set score3=12
select * from s
```

150 %

Results | Messages

	sid	sname	score	score2	score3
1	1	javad	20	10	12
2	2	ali	10	10	12
3	4	zahra	15	10	12
4	5	amir	10	10	12

- مثال REPEATABLE READ:

READ COMMITTED

```
set transaction isolation level
read committed
begin transaction
select * from s
waitfor delay '00:00:10'
select * from s
rollback
```

150 %

Results | Messages

	sid	sname	score	score2	score3
1	1	javad	20	10	15
2	2	ali	10	10	15
3	4	zahra	15	10	15
4	5	amir	10	10	15

	sid	sname	score	score2	score3
1	1	javad	20	10	12
2	2	ali	10	10	12
3	4	zahra	15	10	12
4	5	amir	10	10	12

```
update s set score3=12
select * from s
```

150 %

Results | Messages

	sid	sname	score	score2	score3
1	1	javad	20	10	12
2	2	ali	10	10	12
3	4	zahra	15	10	12
4	5	amir	10	10	12

- مثال REPEATABLE READ:

مقایسه با حالت

read committed

READ COMMITTED

```
set transaction isolation level
repeatable read
begin transaction
select * from s
waitfor delay '00:00:10'
select * from s
rollback
```

150 %

Results | Messages

	sid	sname	score	score2	score3
1	1	javad	20	10	13
2	2	ali	10	10	13
3	4	zahra	15	10	13
4	5	amir	10	10	13

	sid	sname	score	score2	score3
1	1	javad	20	10	13
2	2	ali	10	10	13
3	4	zahra	15	10	13
4	5	amir	10	10	13
5	6	matin	14	15	16
6	7	matin	14	15	16
7	8	matin	14	15	16

```
select * from s

insert into s values
(6,'matin',14,15,16),
(7,'matin',14,15,16),
(8,'matin',14,15,16)
select * from s
```

150 %

Results | Messages

	sid	sname	score	score2	score3
1	1	javad	20	10	13
2	2	ali	10	10	13
3	4	zahra	15	10	13
4	5	amir	10	10	13

	sid	sname	score	score2	score3
1	1	javad	20	10	13
2	2	ali	10	10	13
3	4	zahra	15	10	13
4	5	amir	10	10	13
5	6	matin	14	15	16
6	7	matin	14	15	16
7	8	matin	14	15	16

- مثال REPEATABLE READ:

برای حالت insert

SERIALIZABLE

- این حالت مانند repeatable read است به علاوه این که تضمین میکند که دستورات insert نیز اتفاق نیوفتد و از حالت phantom read که یعنی نمیگذارد زمانی که شما در حال خواندن و کار بر روی تعدادی قلم داده هستید تراکنش دیگیر آنها را تغییر دهند.
- در صورتی که تراکنشی سعی در تغییر یا درج تراکنش جدید داشته یاشد آن تراکنش را وادار میکند صبر کند تا کار تراکنش اول تمام شده و قفل ها آزاد شوند.

SERIALIZABLE

```
set transaction isolation level
serializable
begin transaction
select * from s
waitfor delay '00:00:10'
select * from s
rollback
```

150 %

Results | Messages

	sid	sname	score	score2	score3
1	1	javad	20	10	13
2	2	ali	10	10	13
3	4	zahra	15	10	13
4	5	amir	10	10	13

	sid	sname	score	score2	score3
1	1	javad	20	10	13
2	2	ali	10	10	13
3	4	zahra	15	10	13
4	5	amir	10	10	13

```
select * from s
insert into s values
(6,'matin',14,15,16),
(7,'matin',14,15,16),
(8,'matin',14,15,16)
select * from s
```

150 %

Results | Messages

	sid	sname	score	score2	score3
1	1	javad	20	10	13
2	2	ali	10	10	13
3	4	zahra	15	10	13
4	5	amir	10	10	13

	sid	sname	score	score2	score3
1	1	javad	20	10	13
2	2	ali	10	10	13
3	4	zahra	15	10	13
4	5	amir	10	10	13
5	6	matin	14	15	16
6	7	matin	14	15	16
7	8	matin	14	15	16

- مثال SERIALIZABLE:

SNAPSHOT

- این حالت نیز شبیه seriazable است با این تفاوت که زمانی که وقتی تراکنش ما بر روی قسمتی از داده ها در حال کار است جلو تراکنش های دیگر برای درج و به روز رسانی گرفته نمیشود. بجای آن برای هر رکورد ورژن های گوناگونی در نظر گرفته میشود.
- در این حالت هنگامی که تراکنش دیگری داده ها را تغییر دهد نسخه قدیمی در tempdb نگهداری میشود.
- پس از اتمام تمام تراکنش هایی که قبل از تغییرات شروع شده اند SQL Server نسخه های قبلی درون tempdb را پاک میکند.
- در این حالت ما باعث ایجاد وقفه برای تغییر توسط تراکنش های دیگر نمیشویم
- در این روش ورژن های مختلفی از پایگاه داده نگهداری میشود.

SNAPSHOT

- برای فعال کردن این حالت ابتدا باید کد زیر اجرا شود

```
ALTER DATABASE DatabaseName SET ALLOW_SNAPSHOT_ISOLATION ON
```

- مثال

```
alter database uni set allow_snapshot_isolation on
```

SNAPSHOT

```
set transaction isolation level
snapshot
begin transaction
select * from s
waitfor delay '00:00:10'
select * from s
rollback
```

150 %

Results | Messages

	sid	sname	score	score2	score3
1	1	javad	20	10	13
2	2	ali	10	10	13
3	4	zahra	15	10	13
4	5	amir	10	10	13

	sid	sname	score	score2	score3
1	1	javad	20	10	13
2	2	ali	10	10	13
3	4	zahra	15	10	13
4	5	amir	10	10	13

```
select * from s
insert into s values
(6,'matin',14,15,16),
(7,'matin',14,15,16),
(8,'matin',14,15,16)
select * from s
```

150 %

Results | Messages

	sid	sname	score	score2	score3
1	1	javad	20	10	13
2	2	ali	10	10	13
3	4	zahra	15	10	13
4	5	amir	10	10	13

	sid	sname	score	score2	score3
1	1	javad	20	10	13
2	2	ali	10	10	13
3	4	zahra	15	10	13
4	5	amir	10	10	13
5	6	matin	14	15	16
6	7	matin	14	15	16
7	8	matin	14	15	16

- مثال SNAPSHOT:

دستور کار

دستور کار