

جلسه هشتم تراكنش ها (قسمت دوم)

محمد جواد آکوچکیان و محمود فرجی سال تحصيلي 1401-1402



### تراکنش(Transactions)



#### • تراكنش:

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 تراكنش



- **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 -**





- · 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.





- · 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.



### بادآوری



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

دستور WAITFOR:

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

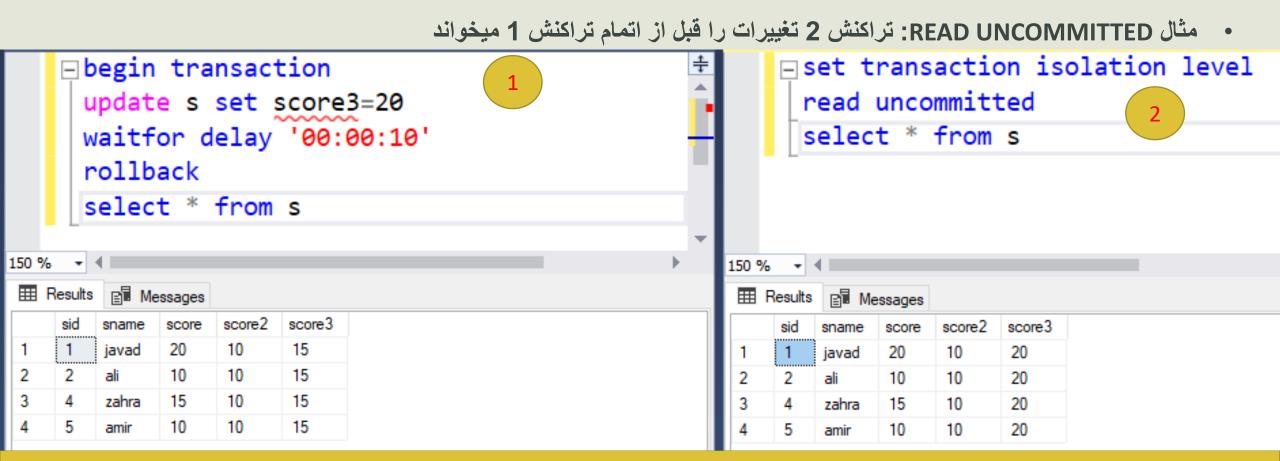
PRINT 'Done';

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

PRINT 'It is noon';
```









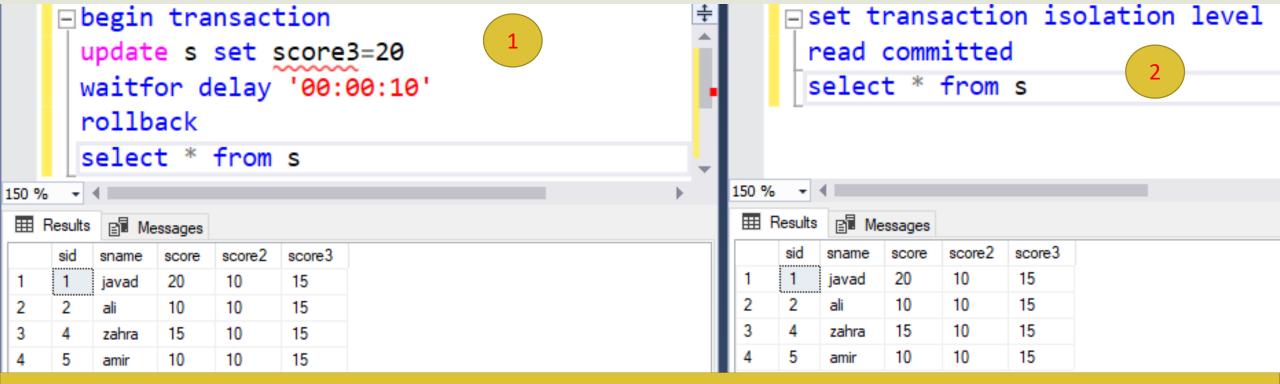


- حالت پیش فرض خود SQL Server این حالت است و اگر به صورت دستی خودمان حالت ISOLATION LEVEL را مشخص نکنیم SQL Server تراکنش ها را در این حالت اجرا میکند.
  - ه دراکنشی تنها میتواند اطلاعات commit شده را بخواند.





• مثال READ COMMITTED: تراكنش 2 منتظر ميماند و بعد از تراكنش 1 اجرا ميشود.





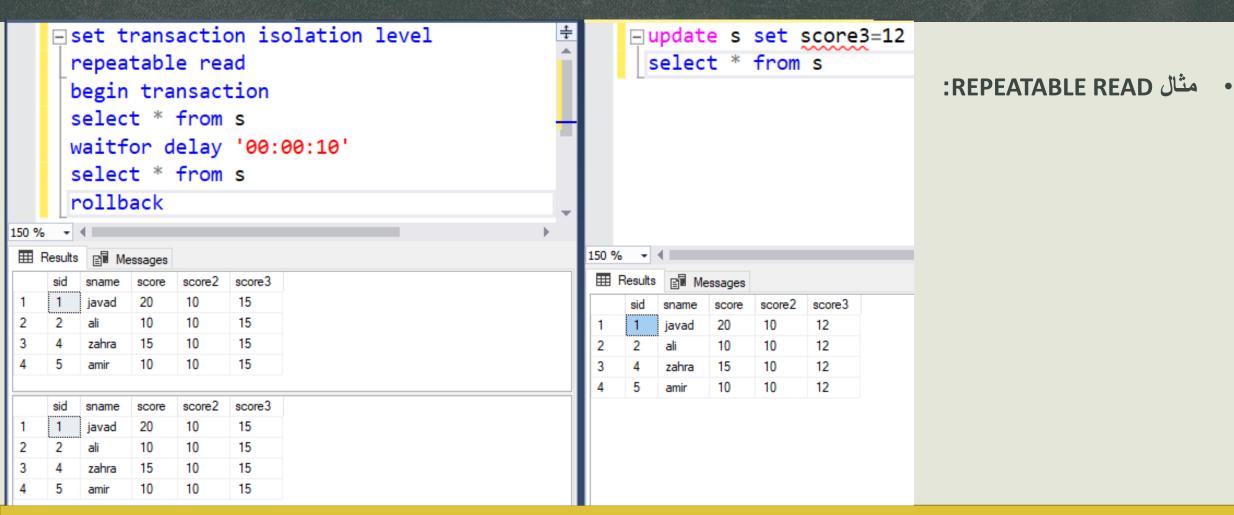
#### REPEATABLE READ



- 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.

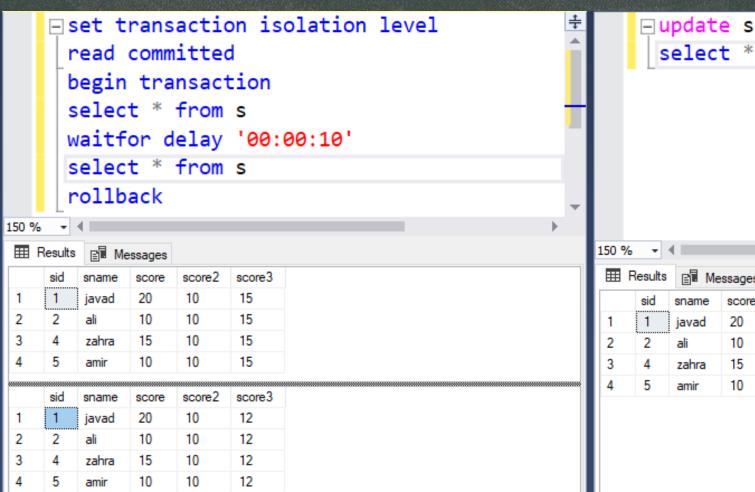


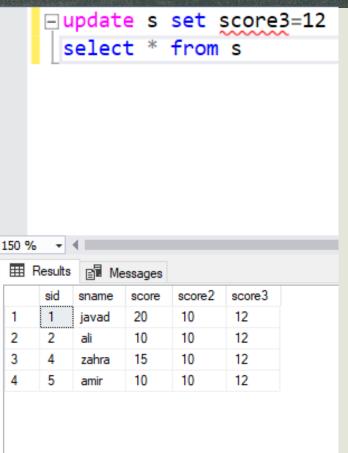












• مثال REPEATABLE READ.

مقایسه با حالت

read committed

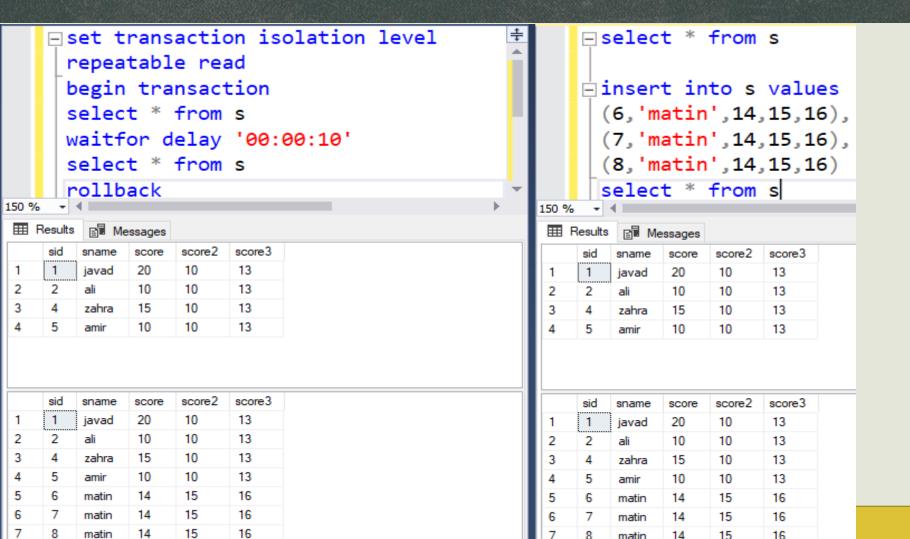


score3

score3

13





• مثال REPEATABLE READ. برای حالت insert



#### **SERIALIZABLE**

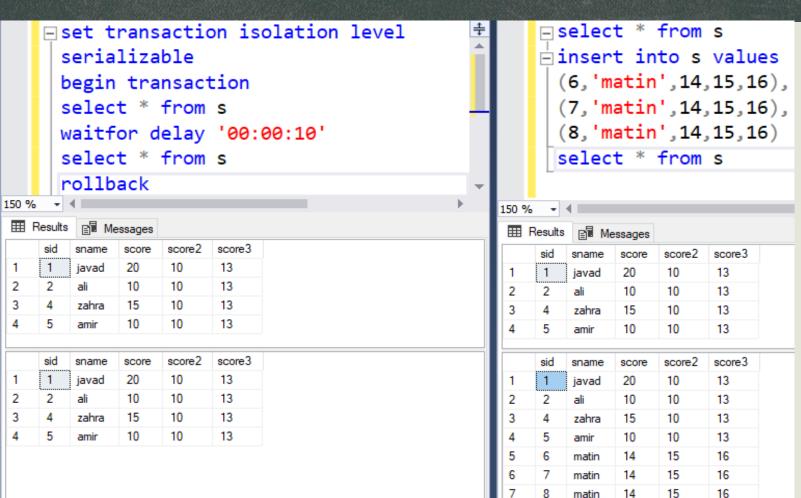


- این حالت مانند repeatable read است به علاوه این که تضمین میکند که دستورات insert نیز اتفاق نیوفتد و از حالت و این حالت مانند phantom read که یعنی نمیگذارد زمانی که شما در حال خواندن و کار بر روی تعدادی قلم داده هستید تراکنش دیگیر آنها را تغییر دهند.
- در صورتی که تراکنشی سعی در تغییر یا درج تراکنش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



#### **SERIALIZABLE**





• 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

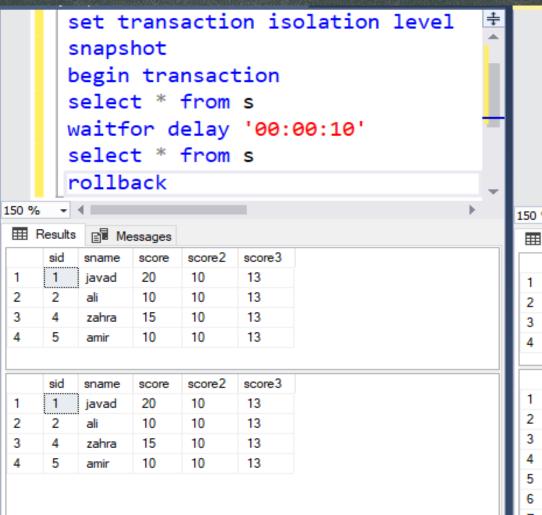
مثال

alter database uni set allow\_snapshot\_isolation on



#### **SNAPSHOT**





```
□ 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
                   score2
                         score3
                    10
                         13
        iavad
                   10
                         13
        zahra
                          13
                   score2
                         score3
              score
                   10
                         13
        javad
                         13
        matin
        matin
```

• مثال SNAPSHOT:



# دستور کار





# دستور کار

