



آزمایشگاه پایگاه داده

جلسه پنجم
آغازگرها

محمد جواد آکوچکیان و محمود فرجی

سال تحصیلی ۱۴۰۱-۱۴۰۲



آغازگرها (Triggers)

- مجموعه ای از کد است که با رخ دادن رویدادی خاص فعال و اجرا می شود.

- دو نوع Trigger در Server SQL وجود دارد :

۱- آغازگرهای (Data Manipulation Language):DML

با تغییراتی که در داده ها با اعمال درج، حذف و تغییر اعمال می شوند.

۲- آغازگرهای (Data Definition Language):DDL

که با تغییرات دیگر در پایگاه داده مانند تعریف و حذف جدول فعال می شوند.

آغازگرها ابزارهای مناسبی برای واکنش اتوماتیک به برخی از رویدادها هستند.



آغازگرها (Triggers)

۱- آغازگرهای (DML(Data Manipulation Language):

- SELECT واکنشی اطلاعات از دیتابیس
- UPDATE ویرایش اطلاعات دیتابیس
- DELETE پاک کردن اطلاعات از دیتابیس
- INSERT INTO اضافه کردن اطلاعات جدید به دیتابیس

۲- آغازگرهای (DDL (Data Definition Language):

- CREATE DATABASE ایجاد یک دیتابیس جدید
- ALTER DATABASE ایجاد تغییرات در دیتابیس
- CREATE TABLE ایجاد یک table جدید
- ALTER TABLE اعمال تغییرات در table
- DROP TABLE پاک کردن یک table
- CREATE INDEX ایجاد یک شاخص
- DROP INDEX حذف یک شاخص



آغازگرهای DML

- این نوع از آغازگرها برای اعمال حذف، درج و تغییر داده های جداول و viewها قابل تعریف هستند. در تعریف آغازگر می توان تعریف کرد که برای کدام عمل باید فعال شود. بدنه ی آغازگر مانند روال ها از تعدادی دستور T-SQL تعریف می شود.

- این نوع آغازگرها بر دو نوع هستند:

۱- نوع AFTER که کد آغازگر بعد از اعمال تغییر اجرا می شود.

۲- نوع INSTEAD OF که آغازگر به جای عملیات تغییر اصلی اجرا می شود.



آغازگرهای DML

- تعریف این آغازگر به صورت زیر است:

```
CREATE TRIGGER trigger_name  
ON table  
[with <dml_trigger_option>, ...]  
AFTER [INSERT] [,] [UPDATE] [,] [DELETE]  
AS <sql_statement> ...
```

- برای تغییر یک آغازگر از دستور مقابل استفاده می گردد:

```
ALTER TRIGGER trigger_name ...
```

- برای حذف یک آغازگر :

```
DROP TRIGGER trigger_name
```

- اگر بخواهیم یک آغازگر خاص فعال و یا غیرفعال شود از دستورات زیر استفاده می گردد:

```
ENABLE TRIGGER trigger_name ON object_name  
DISABLE TRIGGER trigger_name ON object_name
```



آغازگرهای DML



• مثال تعریف آغازگر

```
create trigger trigger1 on book
after insert as
print 'ok'

insert into book(bookname,BSD) values ('HHH', 33);
```

ok

(1 row(s) affected)

```
create trigger trigger3 on book
after update as
insert into bookaudit(bookname,BSD) values ('newbook', 43);
```



آغازگرهای DML



• مثال تعریف آغازگر

```
create trigger t1 on boat  
instead of insert as  
print 'you can not inserted ;)'
```

```
insert into boat values (164, '50', 'red', 50)
```

100 %

Messages

you can not inserted ;)



آغازگرهای DML



- مثال تغییر در یک آغازگر

```
alter trigger t1 on boat
after insert as
print 'new trigger inserted row'

insert into boat values (114, 'R11', 'red', 20)
```

100 %

Messages

new trigger inserted row



آغازگرهای DML



- مثال فعال و غیر فعال کردن یک آغازگر

```
alter trigger t1 on boat  
after insert as  
print 'new trigger inserted row'
```

```
disable trigger t1 on boat
```

```
insert into boat values (115, 'R11', 'red', 20)
```

100 %
Messages

(1 row affected)

Completion time: 2023-04-26T07:49:22.5919302+04:30

```
enable trigger t1 on boat
```

```
insert into boat values (116, 'R16', 'red', 26)
```

100 %

Messages

new trigger inserted row

(1 row affected)

Completion time: 2023-04-26T07:51:17.4445975+04:30



آغازگرهای DML



- مثال حذف یک آغازگر

```
drop trigger t1
```



آغازگرهای DML

آغازگر دو جدول مجازی با نام های INSERTED و DELETED را در اختیار آغازگر قرار می دهد.

نوع تغییر	اطلاعات جدول <i>inserted</i>	اطلاعات جدول <i>deleted</i>
INSERT	ردیفهای اضافه شده	
UPDATE	ردیفهای جدید (پس از تغییر)	ردیفهای قدیمی (پیش از تغییر)
DELETE		ردیفهای حذف شده
جدول ۱: محتوای جدولهای مجازی		



آغازگرهای DML



• مثال

```
create trigger trigger4 on book
after insert as
insert into bookaudit (bookname, BSD)
select inserted.bookname, inserted.BSD
from inserted;
```

```
insert into book (bookname, BSD) values ('11111', 12);
```

	id	bookname	BSD
	1	kkk	43
	2	kkk	43
	3	newbook	43
	4	kkk	43
	5	IIII	12



آغازگرهای DML



• مثال

```
= alter trigger t2 on boat
after insert
as
declare @temp int
select @temp=inserted.boatID from inserted
set @temp=@temp+1
= insert into boat(boatID,boatName,boatRank)
select @temp,inserted.boatName, inserted.boatRank
from inserted

insert into boat values (150,'50','red',50)
select * from boat
```

	boatID	boatName	boatColor	boatRank
12	115	R11	red	20
13	116	R16	red	26
14	117	R17	red	27
15	121	R18	NULL	38
16	130	R18	red	38
17	131	R31	red	31
18	132	R32	red	32
19	140	R31	NULL	31
20	150	50	red	50
21	151	50	NULL	50



آغازگرهای DML

اگر بخواهیم تعداد سطرهای تحت تأثیر قرار گرفته نمایش داده نشود:

- در آغازگرها با استفاده از تابع UPDATE (که متفاوت از دستور UPDATE است) می توان کنترل کرد که آیا مقادیر یک ستون خاص جدول تغییر یافته است یا نه (در صورتی که اعمال update و یا insert بر روی مقادیر این ستون اعمال شود)



غیر فعال کردن فراخوانی تو در توی آغازگرها

- اگر آغازگر t1 تغییراتی را در جدولی اعمال کند، این تغییرات ممکن است به فعال شدن آغازگر دیگری مانند t2 منجر شود و این زنجیره می تواند همین طور ادامه یابد. برای جلوگیری از این زنجیره ها با طول نامحدود در SQL SERVER حداکثر طول این زنجیره برابر با ۳۲ در نظر گرفته شده است.
- با دستور زیر می توان فعال شدن تو در توی triggerها را غیر فعال کرد.

```
EXEC sp_configure 'nested triggers', 0  
RECONFIGURE WITH OVERRIDE
```

- اگر به جای عدد صفر، یک قرار دهیم، فراخوانی تو در تو دوباره فعال می شود.
- برای مشاهده ی trigger ها، table مورد نظر را expand کرده و در گزینه ی trigger لیستی از Trigger ها قابل مشاهده است.



آغازگرهای DML



• مثال


```
= create trigger t1 on boat
  after insert as
  print 't1'

= create trigger t2 on boat
  after insert as
  print 't2'

= alter trigger t3 on boat
  after insert as
  print 't3'
  insert into customer values (218,'akbar',110,'R10')

= create trigger ct1 on customer
  after insert as
  print 'customer t1'

insert into boat values (161,'50','red',50)
```

 Messages

t1
t2
t3
customer t1



آغازگرهای DML



• مثال

```
exec sp_configure 'nested triggers', 0  
reconfigure with override  
  
insert into boat values (163, '50', 'red', 50)
```

Messages
t1
t2
t3



DDL Triggers



آغازگرهای DDL با عبارات ALTER، CREATE، DROP استفاده می شود.

- این triggerها بر روی یک دیتابیس تعریف می شوند.
- این triggerها در بخش programmability - Database triggers قابل مشاهده هستند.



دستور کار

دستور کار جلسه ی پنجم

جدول زیر را ایجاد کنید:

Book:

(ID(int, identity,primary key), Bookname(varchar(30)), yearpublish(int), authorname(varchar(40)), QTY(int))

Triggerهای زیر را ایجاد کنید. در هر مرحله عکسی از trigger و صحت آن درج کنید.

۱- trigger بنویسید که در زمان insert کردن داده‌ها، به جای عملیات insert، پیام 'No change was done' را نشان دهد.

۲- trigger بنویسید که پس از عملیات insert و delete، تمامی مقادیر insert شده و delete شده را در جدول دیگری با نام Book_Audit وارد کند(در جدول Book_Audit فیلد دیگری با نام Ins_or_del اضافه کنید که در هنگام del مقدار صفر و در زمان

insert مقدار ۱ بگیرد). [برای عکس گرفتن از نتیجه هم با insert و هم با Delete امتحان کنید]

توجه: trigger1 در زمان اجرای این trigger غرر فعال کنید.



دستور کار

۳- trigger بنویسید که مانع از update فیلد Bookname شود. [راهنمایی: از rollback و تابع Update استفاده کنید]

۴-view را تعریف کنید که نام هر کتاب را به همراه نام نویسنده‌ی آن کتاب و تعداد چاپ (QTY) را نمایش دهد.

بر روی این view، trigger تعریف کنید که در هنگام insert، کتاب‌هایی را که $QTY < 1000$ دارند را به $QTY = 1000$ تغییر دهد.

۵- trigger بنویسید که امکان حذف جدول Book را از ما بگیرد.