



transactions



م. عبد البديع مراد

قواعد المعطيات (2)

تكلمنا في المحاضرة السابقة عن ال *fragmentation* وتحدثنا أيضاً عن ال *join* وأنواعها وستحدث في هذه المحاضرة عن كيفية حفاظ ال *database* على المعلومات لتجنب حدوث أي نوع من أنواع التقارب عند القيام بأكثر من عملية على ال *database* في نفس الوقت.

فلو قمنا بدايةً ببناء نفس الـ *tables* الذين قمنا ببنائهم في المحاضرة السابقة:

The screenshot shows the Microsoft SQL Server Management Studio interface. The title bar reads "SQLQuery2.sql - DESKTOP-J6IKHT7.lecture8 (DESKTOP-J6IKHT7\HP (59)) - Microsoft SQL Server Management Studio". The Object Explorer sidebar shows a connection to "DESKTOP-J6IKHT7 (SQL Server 14.0.1000.169 - DESKTOP-J6IKHT7\HP (59))" with databases like "master", "tempdb", "model", "msdb", "lecture8", "pubs", and "masterdb" listed. The main query editor window has two tabs: "SQLQuery2.sql - DE...P-J6IKHT7\HP (59)*" and "SQLQuery1.sql - DE...P-J6IKHT7\HP (59)*". The SQLQuery2 tab contains the following script:

```
create table Department  
depid int identity (1,1) primary key,  
dname varchar(50) not null,  
);  
  
insert into Department values('admin');  
insert into Department values('sales');  
insert into Department values('archives');  
  
select * from Department
```

The Results tab shows the execution results:

	depid	dname
1	1	admin
2	2	sales
3	3	archives

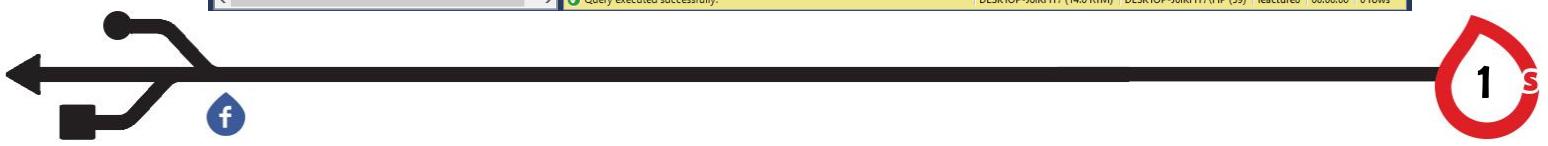
The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left shows the database structure for 'lecture8'. The central pane displays the following T-SQL script:

```
create table Employees(
    empid int identity (1,1) primary key,
    empname varchar(50) not null,
    gender char(1) not null,
    salary int not null,
    depid int foreign key references Department (depid),
    constraint tbemp_ch1 check (salary > 0)
);

insert into Employees values('maher','M',8000,1);
insert into Employees values('sara','F',5800,2);
insert into Employees values('ahmad','M',9400,1);
insert into Employees values('maha','F',8500,1);
insert into Employees values('fares','M',7000,2);
```

The 'Messages' tab at the bottom shows the results of the execution:

```
(1 row affected)  
(1 row affected)  
(1 row affected)  
(1 row affected)
```



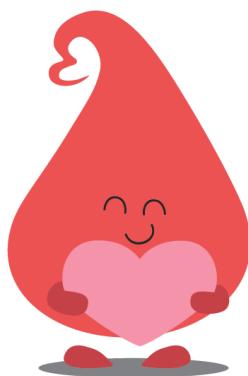


لو قمنا بعد ذلك بتنفيذ ال Query التالية وعرض ال execution plan :

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left lists the database structure for 'DESKTOP-J6IKHT7\lecture8'. The 'Employees' table is selected. The main pane displays a query result window for 'SQLQuery2.sql' containing the following SQL code:

```
select * from Employees
order by depid
```

Below the results, the execution plan shows a single step: a Clustered Index Scan (Clustered) on the [Employees].[PK_Employee] index. The cost of this operation is 78%, while the total batch cost is 100%.



نلاحظ أنها استخدمت الـ *clustered index* كالمعتاد.

لنقم بتوسيع فكرة التضارب بشكل أكبر من خلال المثال التالي:

لدينا آل *table* التالى:

table 1

<i>ID</i>	<i>Name</i>	<i>Salary</i>
1	<i>A</i>	1000
2	<i>B</i>	2000
3	<i>C</i>	3000

وقام شخص بالاتصال لهذه الـ `database` وقام بعمل `connect` ومن ثم قام بتنفيذ تعليمة الـ `select` على هذا الـ `table`.

وبالتأكيد فإن ذلك لن يحدث أي تغيير على ال `table` لأن تأثير ال `select` يكون على ال `view` فقط. ولكن ال `insert, update, delete` تبدأ في حماية نفسها عندما نبدأ تنفيذ تعليمات `dml` والتي هي `database` فلو قمنا الآن بتنفيذ هذه ال `query` على ال `table` السابق:

```
update table1  
set Name ='D'  
where ID = 1
```

عند تنفيذ هذه التعليمة سيصبح هنالك *lock* على ال *row* ذو ال *ID = 1* أي يتم قفله ولكن لم ذلك؟ لنفترض في هذه الأثناء هنالك شخص آخر يقوم بتنفيذ ال *query* التالية على ال *table* السابق:

```
select *\nfrom table1\nwhere ID = 1;
```

فمادا سيكون ال *name* في ال *result set* المعادة هل سيكون 'A' أم 'D'?
 الجواب انه إذا كان في هذه ال *query* يستخدم *clustered scan* فعندها لن يكون قادراً على قراءة ال *data* لأن ال *lock* يكون على كامل ال *table*.

بينما لو أردنا الوصول إلى *row* معين باستخدام *query* تستخدم ال *seek* فذلك ممكن.

إذاً فإن ال *select* التي قام بها الشخص الثاني في مثالنا تبقى في حالة *waiting* وبالطبع ذلك لا يعني بالضرورة وجود مشاكل في ال *database* أو ال *query* المنفذة. ولكن ال *query* تكون في حالة انتظار لل *transaction* حتى ينتهي..... وال *data* التي يتم التعديل عليها في هذه الأثناء يطلق عليها *dirty data* وال *database* بشكل عام تكون في حالة *dirty*.

ملاحظة:

عملية ال *update* السابقة التي قمنا بها هي التي تقابل بدء عملية ال *transaction*.

وعند التطبيق العملي فإن ال *transaction* يعبر عن بدايته ب *begin transaction* وكل *transaction* ينتهي إما ب: لحفظ التغييرات التي طرأت. *commit*. للتراجع عن التغييرات لحد آخر حالة *stable* لل *database* إذاً عندما يحصل تغيير على ال *database* فإن هذا التغيير سيكون أثره واضحًا ضمن أي *query* تنفذ *session* على هذه ال *database*.

ملاحظة:

بالنسبة لعمليات ال *insert* الاعتيادية التي نقوم بها على سبيل المثال فإن *sql server* يحوي *feature* يطلق عليها *auto commit* تعني أن أي تعلیمة *dml* تنفذ فإنه مباشرةً بعد تنفيذها يقوم بعمل *commit*.

ولكن هذا ليس دائمًا صحيح إذ إننا في الحياة العملية من الممكن أن نقوم بتنفيذ عدة تعليمات *dml* ككتلة واحدة فإذاً أن ينجح التنفيذ أو يفشل.

فنحن في مثل هذه الحالة بحاجة لتخزين كتلة التعليمات هذه ضمن *transaction* وفي نهايته إما يكون *commit* في حال نجاح التنفيذ لهم ككل أو ان يفشل تنفيذ إحدى التعليمات على الأقل فيقوم بعمل *roll back* حينها.

ملاحظة:

إن ال *session* التي نقوم بتنفيذ تعليمات ال *dml* ضمنها بإمكانها رؤية التغييرات التي تحدث ضمن هذه ال *session*.

أي بالنسبة لعملية ال *update* التي قمنا بتنفيذها على سبيل المثال فلو تمت ضمن *transaction* معين وبدون أن ننهي ب *commit* أو ان نقوم بـ *roll back* وقمنا في نفس هذه ال *session* بعمل *select* فإن ال *result* العائد ستكون حاملة لل 'D'.

بينما لا يمكن رؤية هذه التغييرات في أي *session* أخرى حتى يتم الانتهاء من هذه ال *transaction*.
 الآن نذهب للتطبيق على ال *tables* الذين قمنا ببنائهم سابقاً:



نقوم بتنفيذ تعليمة ال *insert* التالية على ال *Employees table*:

The screenshot shows the Microsoft SQL Server Management Studio interface. The title bar reads "SQLQuery1.sql - DESKTOP-J6IKHT7.master (DESKTOP-J6IKHT7\HP (57)) - Microsoft SQL Server Management Studio". The Object Explorer sidebar on the left lists the database structure for "DESKTOP-J6IKHT7 (SQL Server 14.0.1000.169 - DESKTOP-J6IKHT7)". The main pane displays a query window titled "SQLQuery1.sql - DE...P-J6IKHT7\HP (57)*". The query code is as follows:

```
select * from Employees  
order by depid  
  
insert into Employees values('pppppp' , 'M' , 8000 , 1);
```

The results pane shows the following data:

	empid	ename	gender	salary	depid
1	1	maher	M	8000	1
2	3	ahmad	M	9400	1
3	4	maha	F	8500	1
4	6	pppppp	M	8000	1
5	2	sara	F	5800	2
6	5	fares	M	7000	2

The status bar at the bottom indicates "Query executed successfully." and "DESKTOP-J6IKHT7 (14.0 RTM) DESKTOP-J6IKHT7\HP (57) master 00:00:00 6 rows".



كما تكلمنا فإن هذه التعليمية يحصل لها *commit* مباشرة.
الآن نقوم بفتح *transaction* جديدة وبده *session* جديد:
Begin transaction

ملاحظة:

فور كتابة `begin transaction` فإن الـ `auto commit` الموجود *by default* يتم إلغاؤه.

و ضمن هذا ال transaction سنقوم بعملية insert أخرى ل record و نعطيه Name و ليكن 'xxxx'

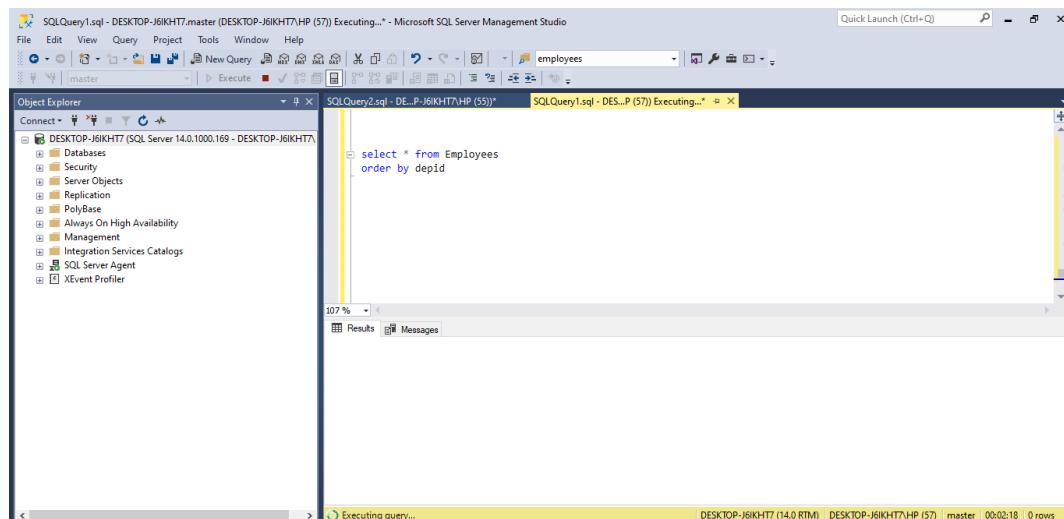
The screenshot shows the Microsoft SQL Server Management Studio interface. The title bar reads "SQLQuery2.sql - DESKTOP-J6IKHT7.master (DESKTOP-J6IKHT7\HP (55)) - Microsoft SQL Server Management Studio". The Object Explorer sidebar on the left lists the database structure for "DESKTOP-J6IKHT7 (SQL Server 14.0.1000.169 - DESKTOP-J6IKHT7)". The main pane displays a T-SQL script in the "employees" database:

```
begin transaction  
insert into Employees values ('xxxxx', 'M', '8000', 1);  
  
commit transaction ;
```

The status bar at the bottom indicates "(1 row affected)" and "Completion time: 2023-12-01T08:32:51.5316933-06:00".

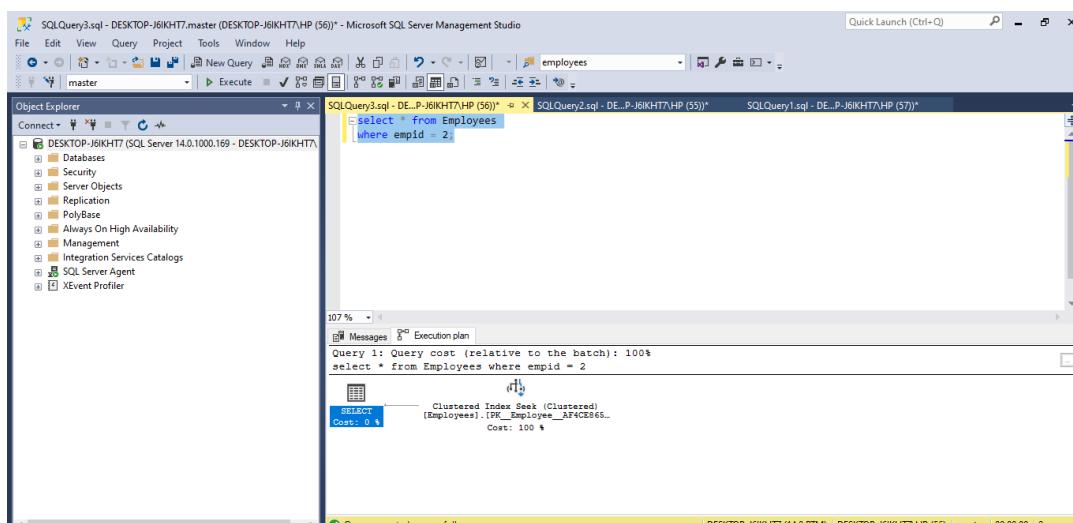
عملية ال *insert* الأخيرة أصبحت معلقة الآن وال *database* أصبحت بحالة *dirty* وأصبح هنالك *lock* على ال *Employees table* لأنه لا يستطيع توقع ما سيحصل في ال *transaction* إما *DML* أخرى أو *commit* أو *roll back*.

لذلك فلو ذهبنا إلى ال session الأخرى وقمنا بكتابة تعليمية select من هذا ال table فإن ال query ستبقى في حالة waiting وذلك يعود سببه كما تحدثنا أن ال هنا هي :
scan clustered operation



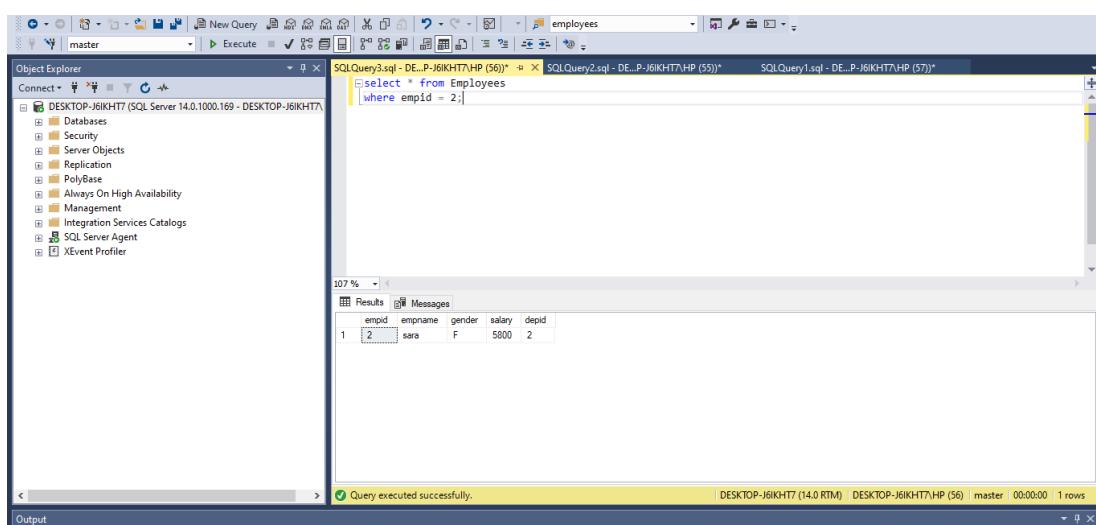
```
File Edit View Query Project Tools Window Help
New Query Execute
master Object Explorer
DESKTOP-J6IKHT7 (SQL Server 14.0.1000.169 - DESKTOP-J6IKHT7\HP (57)) - Microsoft SQL Server Management Studio
File Edit View Project Tools Window Help
New Query Execute
employees
Object Explorer
Connect master
DESKTOP-J6IKHT7 (SQL Server 14.0.1000.169 - DESKTOP-J6IKHT7\HP (57))
Databases Security Server Objects Replication PolyBase Always On High Availability Management Integration Services Catalogs SQL Server Agent XEvent Profiler
SQLQuery1.sql - DESKTOP-J6IKHT7\HP (57) Executing...
SQLQuery2.sql - DESKTOP-J6IKHT7\HP (55)
SQLQuery1.sql - DESKTOP-J6IKHT7\HP (57) Executing...
select * from Employees
order by depid
107 %
Results Messages
Executing query...
DESKTOP-J6IKHT7 (14.0 RTM) DESKTOP-J6IKHT7\HP (57) master 00:02:18 0 rows
```

في الوقت ذاته سنقوم بفتح session ثالثة وننفذ فيها ال query التالية والتي تستخدم ال seek في ال execution plan...



```
File Edit View Query Project Tools Window Help
New Query Execute
master Object Explorer
DESKTOP-J6IKHT7 (SQL Server 14.0.1000.169 - DESKTOP-J6IKHT7\HP (56)) - Microsoft SQL Server Management Studio
File Edit View Project Tools Window Help
New Query Execute
employees
Object Explorer
Connect master
DESKTOP-J6IKHT7 (SQL Server 14.0.1000.169 - DESKTOP-J6IKHT7\HP (56))
Databases Security Server Objects Replication PolyBase Always On High Availability Management Integration Services Catalogs SQL Server Agent XEvent Profiler
SQLQuery3.sql - DESKTOP-J6IKHT7\HP (56) Executing...
SQLQuery2.sql - DESKTOP-J6IKHT7\HP (55)
SQLQuery1.sql - DESKTOP-J6IKHT7\HP (57)
select * from Employees
where empid = 2;
107 %
Messages Execution plan
Query 1: Query cost (relative to the batch): 100%
select * from Employees where empid = 2
SELECT [empid] [empname] [gender] [salary] [depid]
      FROM [Employees].[PK_Employee_AF4CB865...]
      Cost: 0
      Cost: 100 %
Query executed successfully.
DESKTOP-J6IKHT7 (14.0 RTM) DESKTOP-J6IKHT7\HP (56) master 00:00:00 0 rows
```

نقوم بتنفيذ هذه ال query للتأكد من أنها تنفذ وتحضر :
result set



```
File Edit View Query Project Tools Window Help
New Query Execute
master Object Explorer
DESKTOP-J6IKHT7 (SQL Server 14.0.1000.169 - DESKTOP-J6IKHT7\HP (56)) - Microsoft SQL Server Management Studio
File Edit View Project Tools Window Help
New Query Execute
employees
Object Explorer
Connect master
DESKTOP-J6IKHT7 (SQL Server 14.0.1000.169 - DESKTOP-J6IKHT7\HP (56))
Databases Security Server Objects Replication PolyBase Always On High Availability Management Integration Services Catalogs SQL Server Agent XEvent Profiler
SQLQuery3.sql - DESKTOP-J6IKHT7\HP (56) Executing...
SQLQuery2.sql - DESKTOP-J6IKHT7\HP (55)
SQLQuery1.sql - DESKTOP-J6IKHT7\HP (57)
Select * from Employees
where empid = 2;
107 %
Results Messages
empid empname gender salary depid
1 2 sara F 5800 2
Query executed successfully.
DESKTOP-J6IKHT7 (14.0 RTM) DESKTOP-J6IKHT7\HP (56) master 00:00:00 1 rows
```



الآن نعود لل session التي تحوي ال transaction ونقوم بإنهائه إما ب commit transaction وفي مثالنا نستخدم roll back transaction وبعد ذلك نعود لل select التي كانت في حالة waiting فنرى أنها تتفدت وأحضرت data:

```
SQLQuery1.sql - DESKTOP-J6IKHT7.master (DESKTOP-J6IKHT7\HP (57)) - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
Object Explorer
Connect DESKTOP-J6IKHT7 (SQL Server 14.0.1000.169 - DESKTOP-J6IKHT7\)
Databases Security Server Objects Replication PolyBase Always On High Availability Management Integration Services Catalogs SQL Server Agent XEvent Profiler
master New Query Execute
SQLQuery3.sql - DE...P-J6IKHT7\HP (56)* SQLQuery2.sql - DE...P-J6IKHT7\HP (55)* SQLQuery1.sql - DE...P-J6IKHT7\HP (57)*
select * from Employees
order by depid
Results Messages
empid empname gender salary depid
1 maher M 8000 1
2 ahmad M 9400 1
3 maha F 8500 1
4 pppppp M 8000 1
5 xxxx M 8000 1
6 xxxx M 8000 1
7 sara F 5800 2
8 fares M 7000 2
```

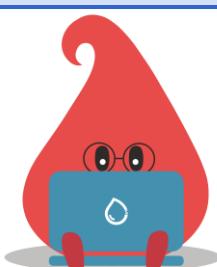
وكما تحدثنا أن التغييرات هي مرئية بالنسبة لل session الواحدة نفسها ونرى مثالنا على ذلك:

```
SQLQuery2.sql - DESKTOP-J6IKHT7.master (DESKTOP-J6IKHT7\HP (55)) - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
Object Explorer
Connect DESKTOP-J6IKHT7 (SQL Server 14.0.1000.169 - DESKTOP-J6IKHT7\)
Databases Security Server Objects Replication PolyBase Always On High Availability Management Integration Services Catalogs SQL Server Agent XEvent Profiler
master New Query Execute
SQLQuery3.sql - DE...P-J6IKHT7\HP (56)* SQLQuery2.sql - DE...P-J6IKHT7\HP (55)* SQLQuery1.sql - DE...P-J6IKHT7\HP (57)*
begin transaction
insert into Employees values ('ooooo', 'M', '8000', 1);
select * from Employees
order by depid
commit transaction ;|
```

```
Results Messages
empid empname gender salary depid
1 maher M 8000 1
2 ahmad M 9400 1
3 maha F 8500 1
4 pppppp M 8000 1
5 xxxx M 8000 1
6 xxxx M 8000 1
7 ooooo M 8000 1
8 sara F 5800 2
9 fares M 7000 2
```

ملاحظة:

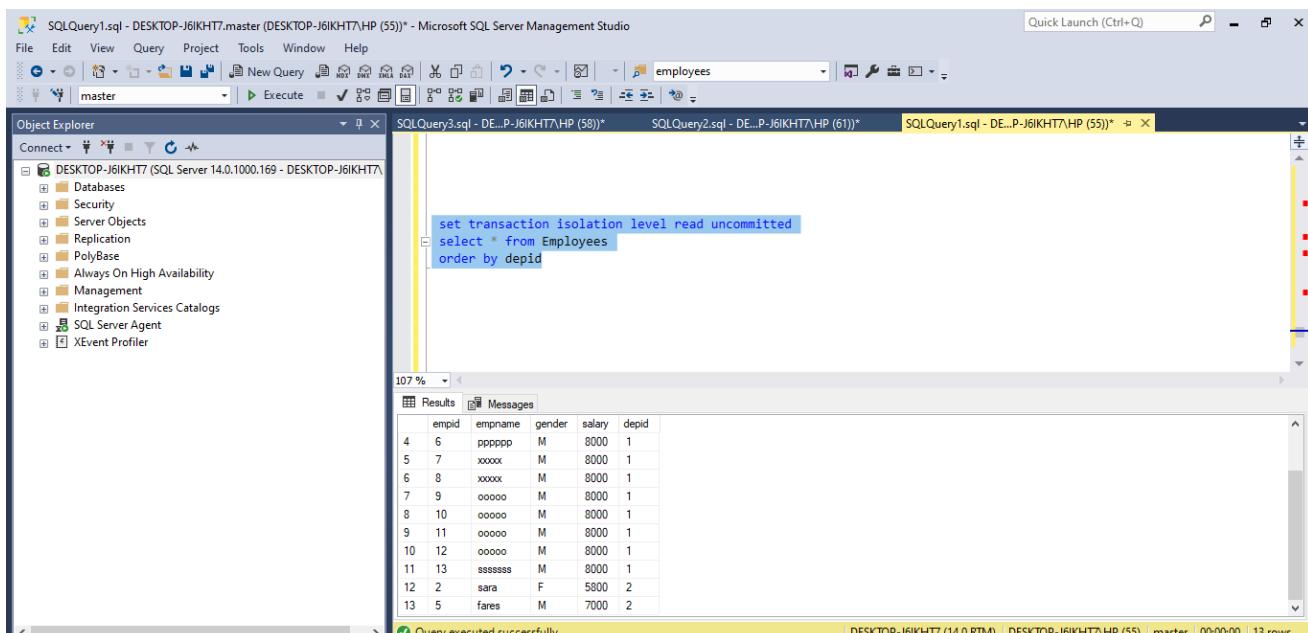
في ال transactions يوجد معينة قليلة الاستخدام وهي كثير من الأحيان يكون استخدامها خاطئ.



"If you get tired, learn to rest.
Not to quit"

set transaction isolation level read uncommitted

وفي هذه الحالة ستقرأ ال *data* ولو كانت *dirty data* أي *uncommitted*.
 أي لو قمنا بتشغيل ال *transaction* السابق مع هذه ال *select query* فلنفذنا ال *feature* فإنه سيقوم مباشرة بإحضار ال *data* وبدون أن يدخل في حالة *waiting* ولكن هذا بالتأكيد أمر خطأ جدًا فال *result* التي عادت وأصبحت موجودة لدينا وقمنا باتخاذ قرارات وتنفيذ تعليمات على أساسها يمكن أن يحدث *roll back* لها ضمن ال *transaction* في أي لحظة... لذلك فإن هذه ال *feature* لا يجب استخدامها أبداً.



```
set transaction isolation level read uncommitted
select * from Employees
order by depid
```

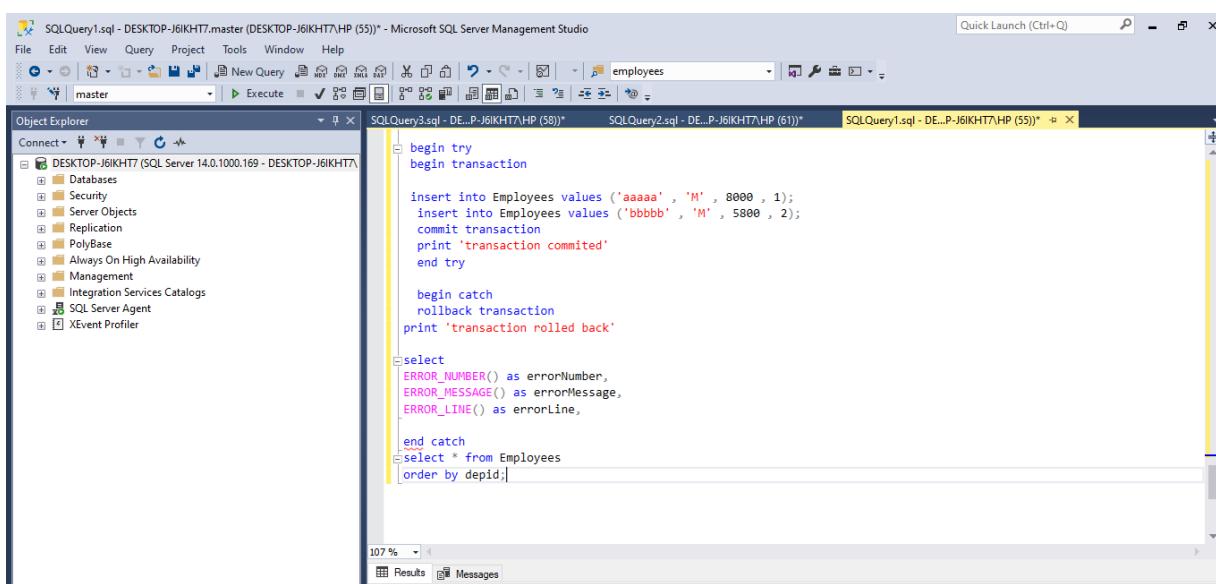
empid	emname	gender	salary	depid
4	pppppp	M	8000	1
5	xxxxxx	M	8000	1
6	xxxxxx	M	8000	1
7	oooooo	M	8000	1
8	oooooo	M	8000	1
9	oooooo	M	8000	1
10	oooooo	M	8000	1
11	oooooo	M	8000	1
12	sara	F	5800	2
13	fares	M	7000	2

Query executed successfully.

ملاحظة:

عادة يتم في ال *transaction* استخدام *try catch* لتحديد الخطأ والعملية التي تستنفذ على هذا الأساس.

مثال:



```
begin try
begin transaction
insert into Employees values ('aaaaa', 'M', 8000, 1);
insert into Employees values ('bbbb', 'M', 5800, 2);
commit transaction
print 'transaction committed'
end try

begin catch
rollback transaction
print 'transaction rolled back'

end catch
select
ERROR_NUMBER() as errorNumber,
ERROR_MESSAGE() as errorMessage,
ERROR_LINE() as errorLine,
;

end catch
select * from Employees
order by depid;
```

transaction committed




SQLQuery1.sql - DESKTOP-J6IKHT7.master (DESKTOP-J6IKHT7\HP (55)) - Microsoft SQL Server Management Studio

```

begin try
begin transaction

insert into Employees values ('aaaaa', 'M', 8000 , 1);
insert into Employees values ('bbbb', 'M', 5800 , 2);
commit transaction
print 'transaction committed'
end try

begin catch

```

Results Messages

```

(1 row affected)
(1 row affected)
transaction committed
(15 rows affected)

Completion time: 2023-12-02T00:41:54.1971915-06:00

```

كما نلاحظ أن هذا ال *transaction* تمت معالجته بشكل صحيح.

الآن سنقوم بتنفيذ *transaction* آخر يحوي على عملية *insert* خاطئة أو مخالفة للشرط *constraint* وعندها : *roll back*



SQLQuery1.sql - DESKTOP-J6IKHT7.master (DESKTOP-J6IKHT7\HP (55)) - Microsoft SQL Server Management Studio

```

begin try
begin transaction

insert into Employees values ('AAAAA', 'M', 8000 , 1);
insert into Employees values ('BBBBB', 'M', 5800 , 2);
commit transaction
print 'transaction committed'
end try

begin catch

```

Results Messages

errorNumber	errorMessage	errorLine
547	The INSERT statement conflicted with the CHECK constraint "CK_Employee_Salary". The conflict occurred in database "master", table "Employees", column "salary". The statement has been terminated.	7

Query executed successfully.



SQLQuery1.sql - DESKTOP-J6IKHT7.master (DESKTOP-J6IKHT7\HP (55)) - Microsoft SQL Server Management Studio

```

begin try
begin transaction

insert into Employees values ('AAAAA', 'M', 8000 , 1);
insert into Employees values ('BBBBB', 'M', -5800 , 2);
commit transaction
print 'transaction committed'
end try

begin catch

```

Results Messages

```

(1 row affected)
(0 rows affected)
transaction rolled back
(1 row affected)
(15 rows affected)

Completion time: 2023-12-02T00:58:00.4062865-06:00

```

Query executed successfully.

نلاحظ طباعة *2 rows* وعدم إضافة ال *transaction roll back*.



Dead Lock

ليكن لدينا هذين الـ *tables* 2 على سبيل المثال t_1, t_2 :

	t_1	t_2	
<i>ID</i>	<i>Name</i>	<i>ID</i>	<i>Name</i>
1	t_1	1	t_2

لنفترض وجود شخص قام ببدء *transaction* على الـ t_1 الأولى *table* وقام خلالها بعملية *update* على الـ

ولتكن من الشكل:

```
Begin transaction
update t1
set Name = 't1t1'
where ID = 1
:
```



تحدثنا مسبقاً أن هذه التعليمات ستقوم بعمل *lock* على الـ t_1 *table*.

وسنفترض أيضاً وجود شخص ثانٍ في *session* ثانٍ قام أيضاً ببدء *transaction* على الـ t_2 الثانية وقام خلالها بعملية *update* على الـ *table* ولتكن من الشكل:

```
Begin transaction
update t2
set Name = 't2t2'
where ID = 1
:
```

وأيضاً هذه التعليمات ستقوم بعمل *lock* على الـ t_2 *table*.

ولحد هذه النقطة فإن التنفيذ سيكون سليم وليس هناك مشاكل ولكن المشكلة الحقيقة ستكون لو قمنا في تتمة

الأول بالقيام بالعملية التالية:

```
update t2
set Name = 'ttt2'
where ID = 1;
```

وبنفس الطريقة لو قمنا بالثاني بالعملية التالية:

```
update t1
set Name = 'ttt1'
where ID = 1;
```

وفي هذه الحالة نكون قد دخلنا بحالة *dead lock* لا نهائية فكلا الـ *transaction* ينتظر الآخر كي ينتهي.



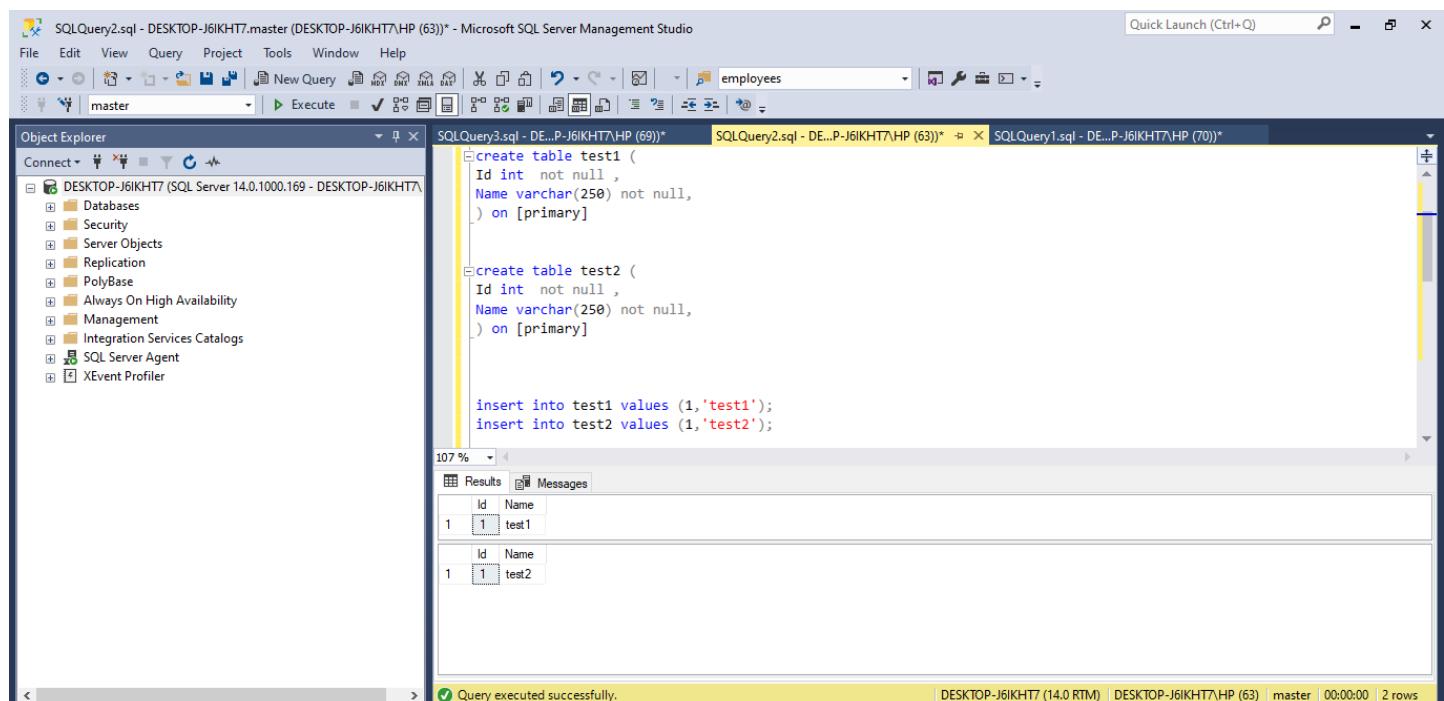
في هذه الحالة يقوم ال engine الخاص بال database بالانتظار لمدة 5 ثواني إما أن تتحقق إحدى ال sessions على حساب الأخرى أو تحصل على حالة dead lock وعندها فإن واحداً من هذين ال 2 sessions سيموت حتماً. ويتم تحديد ذلك بالنظر إلى ال transaction لكل منهما فال session ذو الأكبر ينتصر ويتم تنفيذه وعمل commit ويغلب ال session الآخر ويتم عمل roll back لل session الخاص به باعتباره الأضعف.

ملاحظة:

إن ال dead lock error هو من أكثر الأخطاء شيوعاً عند التعامل مع ال database وهو من أكثرها تأثيراً على ال performance وهو أيضاً من أكثرها تعقيداً في الحل.

ولتجنب مواجهته:

- من الخطوة الأولى لبناء ال database يجب أن يكون ال design بشكله الصحيح والمنتظم.
- تقليل زمن ال transaction قدر الإمكان أي تقسيم ال code أو ال logic transactions لأكثر من transaction.
- لا يكون DML Block كبير جداً.
- معالجة ال queries التي تستهلك زمناً طويلاً كبناء index يخدم هذه ال query على سبيل المثال:
لنشاهد التنفيذ الفعلي لما سبق.
سنقوم ببناء هذين ال 2 tables:



```

SQLQuery2.sql - DESKTOP-J6IKHT7.master (DESKTOP-J6IKHT7\HP (63)) - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
New Query Execute
master
Object Explorer
Connect
Databases Security Server Objects Replication PolyBase Always On High Availability Management Integration Services Catalogs SQL Server Agent XEvent Profiler
SQLQuery3.sql - DE...P-J6IKHT7\HP (69)* SQLQuery2.sql - DE...P-J6IKHT7\HP (63)* SQLQuery1.sql - DE...P-J6IKHT7\HP (70)*
create table test1 (
    Id int not null ,
    Name varchar(250) not null,
) on [primary]

create table test2 (
    Id int not null ,
    Name varchar(250) not null,
) on [primary]

insert into test1 values (1,'test1');
insert into test2 values (1,'test2');

Results Messages
Id Name
1 test1
Id Name
1 test2

```

Query executed successfully.

"All I want for Christmas is to
study Database"





الآن سنقوم بفتح 2 sessions وبدء transaction في كل واحدة منها ونقوم من خلالهما بتنفيذ تعليمات بنفس الشكل الذي ذكرناه في المثال السابق لرؤية حالة الdead lock.

The screenshot shows the Microsoft SQL Server Management Studio interface. The title bar indicates the connection is to DESKTOP-J6IKHT7\master (DESKTOP-J6IKHT7\HP (69)). The Object Explorer sidebar shows the database structure for the master database. The main pane displays a T-SQL script:

```
begin transaction

update test1
set name = 'test1 transaction '
where id =1
waitfor delay '00:00:05'

update test2
set name = 'test2 transaction '
where id =1
waitfor delay '00:00:05'

commit transaction
```

The execution results are shown in the Messages tab:

```
(1 row affected)
(1 row affected)

Completion time: 2023-12-02T09:14:41.0711557-06:00
```

A status bar at the bottom right shows "0 rows" affected.

The screenshot shows the Microsoft SQL Server Management Studio interface. The title bar indicates the connection is to DESKTOP-J6IKHT7 master (DESKTOP-J6IKHT7\HP (55)). The Object Explorer sidebar shows the database structure. The main pane displays a T-SQL script that attempts to update two tables simultaneously, leading to a deadlock:

```
begin transaction  
update test2  
set name = 'test2 transaction'  
where id = 1  
waitfor delay '00:00:05'  
  
update test1  
set name = 'test1 transaction'  
where id = 1  
waitfor delay '00:00:05'  
  
commit transaction
```

The status bar at the bottom shows the completion time as 2023-12-02T09:14:36.0303416-06:00.

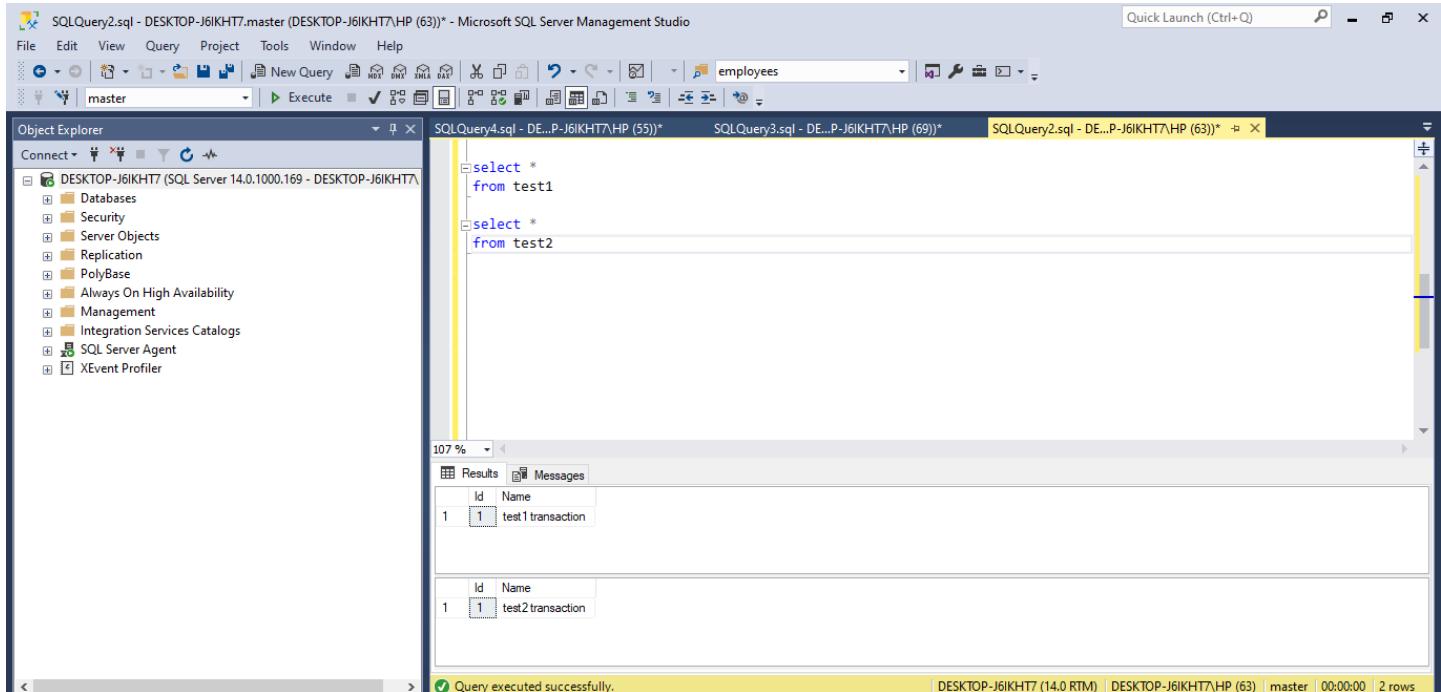
ملاحظة:

تَعْلِيمَةُ التَّأْخِيرِ *wait for* تم وضعها لنفترض وجود العديد من التعليمات بدلًا منها.



“Coffee in one hand, confidence
in another”

وكما لاحظنا أن ال *session* الأولى تم تنفيذها ونجحت على حساب ال *session* الثانية التي فشلت واعادت عمل *transaction* لل *roll back* وتم عمل *dead lock error*.



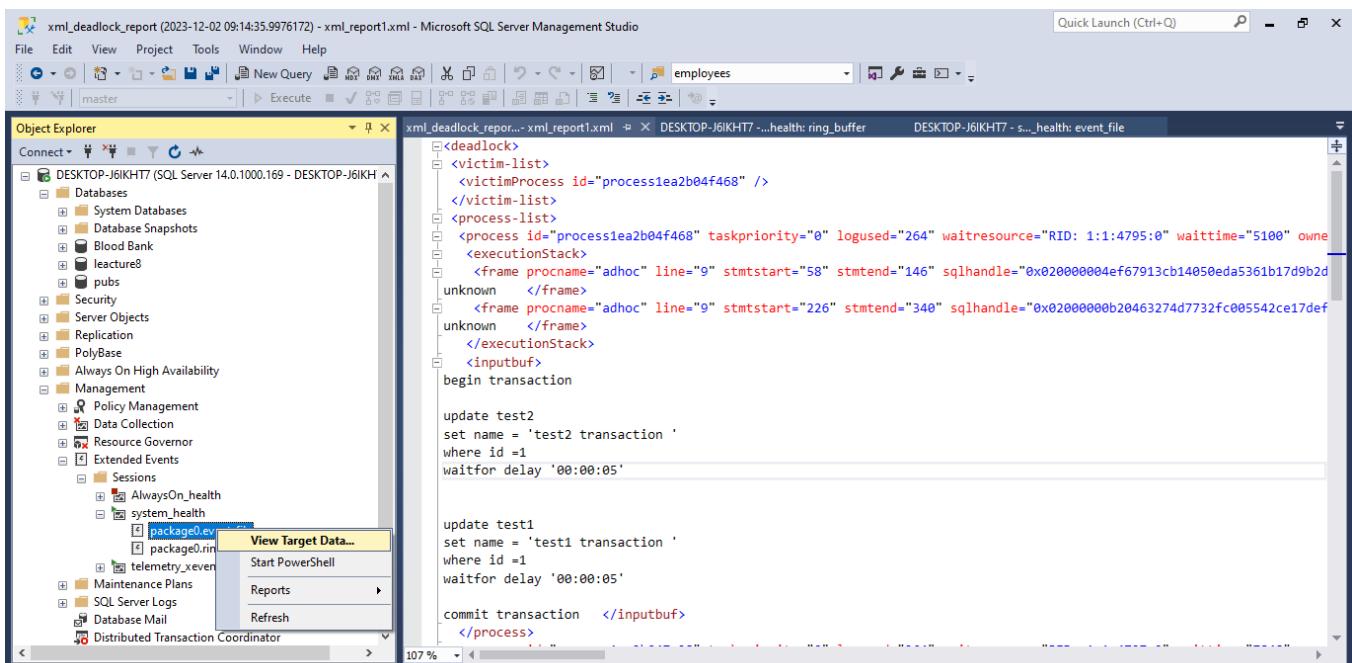
```

SQLQuery2.sql - DESKTOP-J6IKHT7.master (DESKTOP-J6IKHT7\HP (63)) - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
New Query Execute
master
Object Explorer
Connect
DESKTOP-J6IKHT7 (SQL Server 14.0.1000.169 - DESKTOP-J6IKHT7)
Databases Security Server Objects Replication PolyBase Always On High Availability Management Integration Services Catalogs SQL Server Agent XEvent Profiler
SQLQuery4.sql - DE...P-J6IKHT7\HP (55)* SQLQuery3.sql - DE...P-J6IKHT7\HP (69)* SQLQuery2.sql - DE...P-J6IKHT7\HP (63)*
select *
from test1
select *
from test2
Results Messages
Id Name
1 test1transaction
Id Name
1 test2transaction
Query executed successfully.
DESKTOP-J6IKHT7 (14.0 RTM) DESKTOP-J6IKHT7\HP (63) master 00:00:00 2 rows
    
```

ملاحظة:

هناك عدة طرق لمشاهدة ال *dead lock* ضمن ال *database* ومشاهدتها تأثيرها ومن هو ال *transaction* الذي فشل او كما يسمى *victim transaction* (الضحية) ومن هو ال *transaction* الذي قام بقتله.

وإحدى الطرق لرؤية ال *dead lock* هي من خلال ال *side bar* بالضغط على *system health* ← *sessions* ← *Events* ومن ثم *view target data*



```

xml_deadlock_report (2023-12-02 09:14:35.9976172) - xml_report1.xml - Microsoft SQL Server Management Studio
File Edit View Project Tools Window Help
New Query Execute
master
Object Explorer
xml_deadlock_report...xml_report1.xml DESKTOP-J6IKHT7 -...health: ring_buffer DESKTOP-J6IKHT7 - ... health: event_file
DESKTOP-J6IKHT7 (SQL Server 14.0.1000.169 - DESKTOP-J6IKHT7)
Databases System Databases Database Snapshots Blood Bank lecture8 pubs Security Server Objects Replication PolyBase Always On High Availability Management Policy Management Data Collection Resource Governor Extended Events Sessions AlwaysOn_health system_health package0.event_file telemetry.event Maintenance Plans SQL Server Logs Database Mail Distributed Transaction Coordinator
xml_report1.xml
<deadlock>
  <victim-list>
    <victimProcess id="process1ea2b04f468" />
  </victim-list>
  <process-list>
    <process id="process1ea2b04f468" taskpriority="0" logused="264" waitresource="RID: 1:1:4795:0" waittime="5100" owner=""
      <executionStack>
        <frame procname="adhoc" line="9" stmtstart="58" stmtend="146" sqlhandle="0x020000004ef67913cb14050eda5361b17d9b2d"
          unknown </frame>
        <frame procname="adhoc" line="9" stmtstart="226" stmtend="340" sqlhandle="0x02000000b20463274d7732fc005542ce17def"
          unknown </frame>
      </executionStack>
      <inputbuf>
        begin transaction
      </inputbuf>
    <process id="process1ea2b04f468" taskpriority="0" logused="264" waitresource="RID: 1:1:4795:0" waittime="5100" owner=""
      <executionStack>
        <frame procname="adhoc" line="9" stmtstart="58" stmtend="146" sqlhandle="0x020000004ef67913cb14050eda5361b17d9b2d"
          unknown </frame>
        <frame procname="adhoc" line="9" stmtstart="226" stmtend="340" sqlhandle="0x02000000b20463274d7732fc005542ce17def"
          unknown </frame>
      </executionStack>
      <inputbuf>
        update test2
        set name = 'test2 transaction'
        where id =1
        waitfor delay '00:00:05'
      </inputbuf>
    <process id="process1ea2b04f468" taskpriority="0" logused="264" waitresource="RID: 1:1:4795:0" waittime="5100" owner=""
      <executionStack>
        <frame procname="adhoc" line="9" stmtstart="58" stmtend="146" sqlhandle="0x020000004ef67913cb14050eda5361b17d9b2d"
          unknown </frame>
        <frame procname="adhoc" line="9" stmtstart="226" stmtend="340" sqlhandle="0x02000000b20463274d7732fc005542ce17def"
          unknown </frame>
      </executionStack>
      <inputbuf>
        update test1
        set name = 'test1 transaction'
        where id =1
        waitfor delay '00:00:05'
      </inputbuf>
    </process>
  </process-list>
</deadlock>
    
```



كما نلاحظ أنه هنا قام بإعطائنا كل التغيرات على الـ *database* ويمكننا ضمـن هذا الملف عبر الاختصار *ctrl + f* وايضاً يمكننا تحديد الـ *table* الذي نريد البحث فيه:

The screenshot shows the Microsoft SQL Server Management Studio interface. In the Object Explorer, the 'Extended Events' node is expanded, showing various session types like 'AlwaysOn_health', 'system_health', and 'package0.event_file'. A search dialog is open in the center, with 'Find what:' set to 'deadlock' and 'Look in:' set to 'Table Columns'. Below the search dialog, a table titled 'Event.xml_deadlock_report (2023-12-02 09:14:35.9976172)' displays a single row under the 'Details' tab, specifically the 'Deadlock' section.

وفي حالتنا ظهر لدينا هذا الـ *dead lock* الذي حصل نتيجة التنفيذ الذي قمنا به في المثال السابق.
نضغط عليه لنشاهد الـ *block of code* يشرح عملية الـ *dead lock* التي حصلت:

This screenshot shows the XML content of the deadlock report. It details two processes involved in the deadlock. Process 1 (victim) has a task priority of 0, log usage of 264, and is waiting for RID 1:1:4795:0 with a wait time of 5100ms. Process 2 (victim) has a task priority of 146, log usage of 264, and is waiting for RID 1:1:4795:0 with a wait time of 5100ms. Both processes are associated with the 'adhoc' stored procedure. The report also includes the 'begin transaction' and 'commit transaction' statements for both processes.

This screenshot shows the timeline of events with 59836 events displayed. The 'xml_deadlock_report' event is highlighted. Below it, the 'Details' pane shows the deadlock diagram. The diagram illustrates two transactions (test1 and test2) involving RID locks on pages 1:1 and 1:2. Transaction test1 (Server process 55) holds an 'X' lock on page 1:1 and requests a 'U' lock on page 1:2. Transaction test2 (Server process 69) holds a 'U' lock on page 1:2 and requests an 'X' lock on page 1:1. The 'Owner Mode' for both locks is 'X' (exclusive), which leads to a deadlock.

The End

