

1. 在teachers使用order by 改变游标中的执行顺序，使得教师查询按照工资高到低排序。

- 代码实现：

```
declare cursor_teachers cursor for
select * from teachers
order by salary desc;
--打开游标
open cursor_teachers
--读取游标中的数据
fetch next from cursor_teachers
while @@FETCH_STATUS = 0
begin
    fetch next from cursor_teachers
end;

close cursor_teachers

deallocate cursor_teachers;
```

- 实验结果：

	tid	tname	email	salary	
1	204711560	xarfindjtd	nsemd30@tsoi.net	4999	
	tid	tname	email	salary	
1	214445507	eghgugq	b7163n@zyfpi.com	4999	
	tid	tname	email	salary	
1	277877392	wbvmgkvi	o0pm@swhow.edu	4999	
	tid	tname	email	salary	
1	287866460	qzftezkyu	u2d5@nga.gov	4999	
	tid	tname	email	salary	
1	261399893	flwce	rxrz@tsb.edu	4998	
	tid	tname	email	salary	
1	287869210	abavr	alwbr@rsrjp.net	4997	
	tid	tname	email	salary	

通过实验结果可以看到，查询结果为按照工资进行降序排序

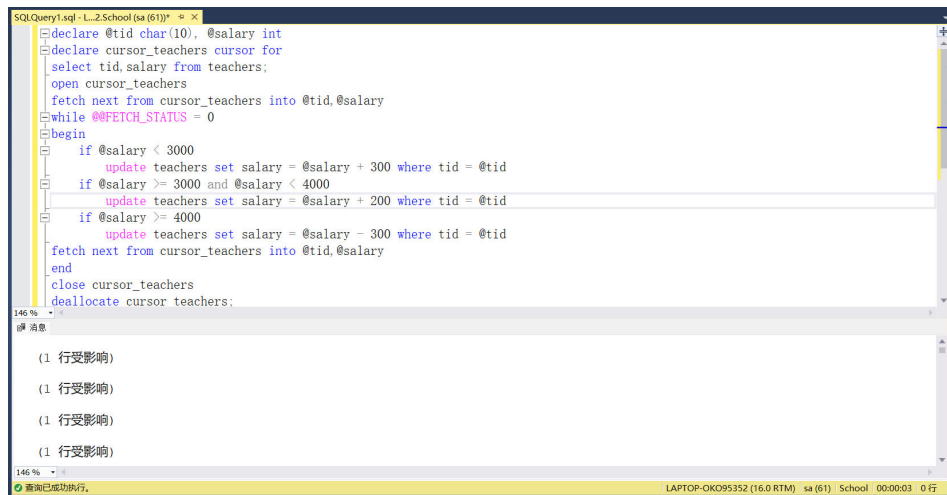
2. 实验并思考回答为什么实验示例4中更新语句不直接用类似于三条
update teachers set where语句替换。

- 代码实现:

```
-- 游标实现
declare @tid char(10), @salary int
declare cursor_teachers cursor for
select tid,salary from teachers;
open cursor_teachers
fetch next from cursor_teachers into @tid,@salary
while @@FETCH_STATUS = 0
begin
    if @salary < 3000
        update teachers set salary = @salary + 300
    where tid = @tid
    if @salary >= 3000 and @salary < 4000
        update teachers set salary = @salary + 200
    where tid = @tid
    if @salary >= 4000
        update teachers set salary = @salary - 300
    where tid = @tid
    fetch next from cursor_teachers into @tid,@salary
end
close cursor_teachers
deallocate cursor_teachers;

-- 三个update替换
if salary < 3000
    update teachers set salary = salary + 300
if salary >= 3000 and salary <4000
    update teachers set salary = salary + 200
if salary >= 4000
    update teachers set salary = salary - 300
```

- 实验结果:



```
SQLQuery1.sql - L_2.School (sa (61)) *
-- declare @tid char(10), @salary int
-- declare cursor_teachers cursor for
-- select tid,salary from teachers;
-- open cursor_teachers
-- fetch next from cursor_teachers into @tid,@salary
-- while @@FETCH_STATUS = 0
-- begin
--     if @salary < 3000
--         update teachers set salary = @salary + 300 where tid = @tid
--     if @salary >= 3000 and @salary < 4000
--         update teachers set salary = @salary + 200 where tid = @tid
--     if @salary >= 4000
--         update teachers set salary = @salary - 300 where tid = @tid
--     fetch next from cursor_teachers into @tid,@salary
-- end
-- close cursor_teachers
-- deallocate cursor_teachers;
```

146 % 消息

(1 行受影响)
(1 行受影响)
(1 行受影响)
(1 行受影响)

146 % LAPTOP-OKO95352 (16.0 RTM) sa (61) School 00:00:03 0 行

不使用三个update替换的原因是因为一致性的问题，因为在游标中定义了@salary变量，因此如果有一个salary为2800，第一次进入了第一个if，然后加了300，变成了3100，但是这个更改是对于salary的而不是@salary，因此不会进入第二个判断。如果利用三个update进行操作则会导致直接更改salary，导致进入第二个if语句，导致数据的更改不一致。

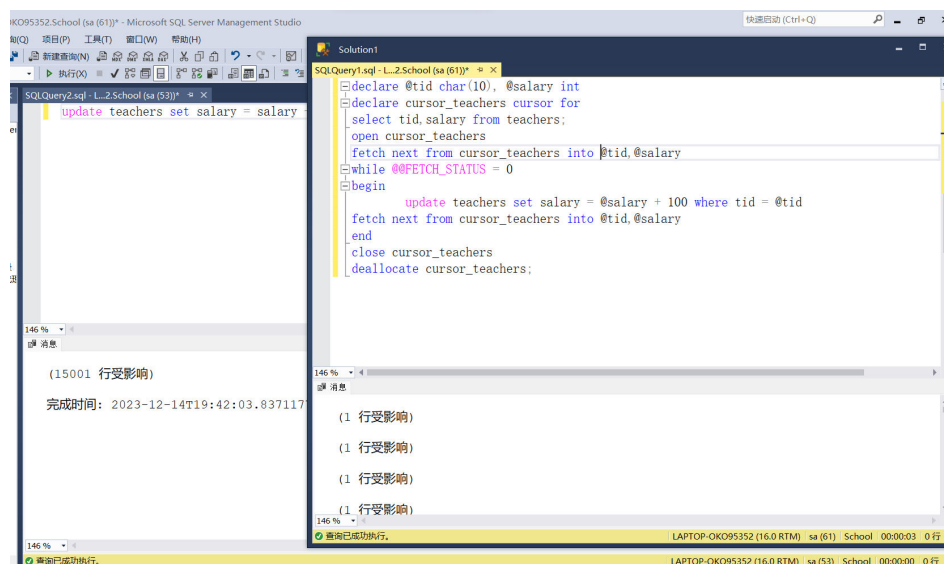
3. 执行两个连接，分别利用游标和正常的更新方式对教师工资进行更新(全部增加100)，记录两者运行的所用时间(见查询结果右下角),比较两者的执行效率。

- 代码实现：

```
-- 游标
declare @tid char(10), @salary int
declare cursor_teachers cursor for
select tid,salary from teachers;
open cursor_teachers
fetch next from cursor_teachers into @tid,@salary
while @@FETCH_STATUS = 0
begin
    update teachers set salary = @salary + 100
    where tid = @tid
    fetch next from cursor_teachers into @tid,@salary
end
close cursor_teachers
deallocate cursor_teachers;

-- update
update teachers set salary = salary + 100;
```

○ 实验结果：



通过结果发现，使用update更快，因此使用游标的执行效率更慢，原因如下：

1. 网络开销：

- 游标通常需要在客户端和数据库服务器之间进行多次交互，每次都要发送和接收数据。这增加了网络开销，特别是在大量行的情况下。

2. 锁定和事务开销：

- 使用游标时，可能需要使用显式锁定（如 `FOR UPDATE`）以确保事务的隔离性。这可能导致在事务中的某些操作需要等待其他事务释放锁。正常的更新方式可能更容易进行并发处理而不需要显式锁定。

3. 游标操作的额外开销：

- 游标通常需要额外的资源来维护，例如服务器端和客户端的内存。在每次 `FETCH` 和 `UPDATE` 之间，需要在数据库中进行额外的定位和处理。

4. 单行处理开销：

- 游标通常是基于单行的处理，每次 `FETCH` 和 `UPDATE` 都只处理一行。这种单行处理方式可能在大量数据的情况下引入额外的开销，相对于批量更新的方式而言更慢。

5. 编程语言和数据库驱动的开销：

- 游标操作通常需要更多的编程逻辑，而直接的更新语句可能更简单，这取决于所使用的编程语言和数据库驱动。

使用游标不一定就慢，而是相对于一次性执行批量更新的直接语句而言可能更慢。性能差异可能因数据库引擎、查询优化、索引、数据量和硬件等因素而异。