الباب الرابع
مفاهيم متقدمة

١٤ . استخدام استعلام داخل استعلام آخر

١٥ . التحكم في العمليات Transaction

الفصل الرابع عشر

استخدام استعلام داخل استعلام آخر

Using Sub-Query

كلمة **Sub-query** *عبارة عن استعلام أو* **Query** *صغيرة تستخدم داخل* **Query** *حيث يتم تمرير نتيجة الاستعلام* **Query** *إلى استعلام أو* **Query** *أخرى .*

نتحدث عن استخدام **sub-query** *. هذا الفصل يفترض فهمك جيداً للفصول السابقة لأنه يشرح مفاهيم متقدمة في التعامل مع الاستعلام* **Query**.

بالانتهاء من هذا الفصل ستكتسب المعارف وتتدرب على المهارات التى تجعلك قادرا على:

- *تكوين* **SUB-QUERY**
- *استخدام الدوال الإحصائية مع* **Sub-Queries**
- *استخدام* **Exist** *مع* **sub-query**
- *استخدام* **ALL** *مع* **Sub-Query**
- *استخدام* **ANY** *مع* **Sub-Query**
- *استخدام* **Some** *مع* **Sub-Query**

تكوين SUB-QUERY

مفهوم **SUB-QUERY** عبارة عن استعلام أو **QUERY** صغيرة تستخدم داخل **QUERY** أخرى ويتم تمرير نتيجة الاستعلام إلى استعلام آخر . والأمثلة التالية سوف توضح ذلك . يتم تكوين **sub-query** بكتابة جملة داخل جملة بالشكل العام التالي :

```
Select * from table1
Where table1.field =
(select field from table2
where field = somevalue)
```

تأمل الصياغة السابقة ، ستجد أنها عبارة عن query داخل query وتعتمد نتيجة التنفيذ على مقارنة القيمة في الجدول الأول على القيمة في الجملة الثانية ، ولنأخذ أمثلة من الواقع وسنستخدم في هذه الأمثلة جدول stock وجدول orders وبياناتهما كالتالي :

```
Select * from stock
```

PART_NO	DESCRIPTION	QTY
MB01	Mother Board Giga	120
MB02	Mother Board Accorp	444
MO02	LG 17" Monitor	80
MO03	Samsong 15"	100
MO04	Samsong 17"	85
P001	1.7GHz	40
P003	2.4GHz	110
HD01	HDD 40GB	75
HD02	HDD 80GB	175
HD03	HDD 115GB	275

سنطبع الآن محتويات الجدول الثاني orders :

```
select * from orders
```

part_no	qty	price
MB01	20	350
MO01	10	600
MO04	5	540
P002	5	450
P004	10	980
HD01	4	320

كما قد تلاحظ يوجد حقل مشترك بين كل من جدول **stock** وجدول **orders** ولكننا نريد هنا طباعة أوامر التوريد **orders** لصنف معين ، لذلك سنقوم باستخدام وصف الصنف **description** من الجدول الأول **stock** فكيف ستقوم **sub-query** بتنفيذ ذلك :

Transact-SQL

```
Select *
From orders
Where part_no =
(select *
from stock
where description like 'hdd 4%')
```

Part_no	Qty	Price
HD01	75	320

(1 row(s) affected)

سنستخدم نفس **query** السابقة مع **MySQL** للتأكد من أن **sub-query** تعمل أيضا مع نظام **MySQL** حيث لم تكن تعمل في الاصدارات قبل **MySQL 4.0** :

MySQL

```
Select *
From orders
Where part_no =
(select part_no
from stock
where description like 'hdd 4%');
```

Part_no	qty	price
HD01	75	320

هل يمكن تنفيذ **sub-query** بالطريقة العادية ؟

طبعا المثال السابق بسيط ولا تستخدم **sub-query** في العمليات البسيطة ، ومع ذلك سوف نحاول كتابة نفس المثال بالطريقة العادية :

Transact-SQL

```
Select orders.part_no, qty
From orders, stock
```

```
where orders.part_no = stock.part_no
and description like 'hdd 4%'
```

Part_no	QTY	price
HD01	75	320

(1 row(s) affected)

MySQL

```
Select orders.part_no, stock.qty
From orders, stock
where orders.part_no = stock.part_no
and description like 'hdd 4%';
```

part_no	qty	price
HD01	75	320

استخدام الدوال الإحصائية مع Sub-Queries

فيما يلي سنقوم باستخدام الدوال الإحصائية مثل **SUM, COUNT, MIN, MAX, AVG** مع **Sub-Query** لتحقيق أكبر فائدة من استخدام استعلام داخل استعلام **(subquery)** . في المثال التالي سنقوم باستخراج متوسط حاصل ضرب الكمية في السعر وذلك بفرض أن السعر في جدول **prices** والكمية في جدول آخر هو **orders** ، بشرط أن يكون رقم الصنف **part_no** واحدا . أي يتطابق رقم الصنف في الجدول الأول مع رقم الصنف في الجدول الثاني:

MySQL

```
select avg(o.qty * p.price)
from orders o, prices p
where o.part_no = p.part_no;
```

avg(o.qty * p.price)
4225.7143

generated 6/7/2005 8:44:59 PM by MySQL-Front 1.22

في المثال السابق تم استخراج متوسط الكمية x السعر وهو الموجود في الجدول كنتيجة لتنفيذ جملة query السابقة .

فيما يلي سوف نقوم بطباعة الأصناف التي إجمالي قيمتها فوق المعدل ، لاحظ أننا سنستخدم الجملة السابقة ولكن Sub-Query للجملة الأساسية بحيث سنقوم بتنفيذ الجملة الأساسية في حالة زيادة القيمة عن المعدل avg :

MySQL

```
select p.part_no, o.qty * p.price total
from orders o, prices p
where o.part_no = p.part_no
and
o.qty * p.price >
 (select avg(o.qty * p.price)
from orders o, prices p
where o.part_no = p.part_no);
```

نتيجة لتنفيذ الجملة السابقة ، تم طباعة الأصناف فقط التي تزيد قيمتها عن المعدل أو المتوسط أو تزيد قيمة حاصل ضرب الكمية في السعر عن المتوسط وهو ٤٢٢٥,٧١٤٣ ، إذن نحن قارنا نتيجة الضرب مع المتوسط القادم لنا من sub-query ، تبدأ جملة الاستعلام الداخلي " sub-query " بجملة SELECT الثانية وتوضع بين قوسين ( ) . بعبارة أخرى كما قلنا في أول الفصل قمنا بتمرير نتيجة الاستعلام " sub-query " الثاني إلى الاستعلام الأول " QUERY " . وبالتالي حصلنا على النتيجة التالية :

part_no	total
MB01	6000
MO01	5000
P004	6000
HD03	5680

generated 6/7/2005 9:05:54 PM by MySQL-Front 1.22

تداخل Sub-Queries

ويسمى **Nested Sub-Queries** ومعنى ذلك أننا نقوم بتنفيذ **Sub-Query** ثم نمرر النتيجة إلى **Sub-Query** واستنادا إلى هذه النتيجة يتم تنفيذ **Sub-Query** التي تعتمد عليها ثم نصل إلى **Query** الأساسية ومعنا نتيجة تنفيذ **Sub-Query** الأخيرة وهو ما يمكن أن نكتبه هكذا

```
select * from xxxxx where (sub-query(sub-query(sub-query)));
```

طبعا المقصود ب **xxxxx** هو اسم جدول أو جداول يتم العمل عليها ، وطبعا كل **Sub_query** عبارة عن جملة **select** .

ولتوضيح ذلك فسوف نستخدم نفس المثال السابق باعتباره **Sub-Query** وسوف نكتب **Query** الأساسية لتستخدم المثال السابق ، ولكننا سوف نستخدم ٣ جداول من قاعدة البيانات **Procurement** ، سوف نطبع الآن محتويات جدول **Customer** :

Transact-SQL

```
Select * from customer
Go
```

Name	Address	City	State
COMPUSIENCE	gamaa street	giza	giza
Egypt Cables	orabi street	mohandseen	giza
United Group	saleem street	ziton	cairo
Sara Intl Group	makram obaid st	nasr city	cairo
Basic Solutions	akkad st	nasr city	cairo
Computer Technology	horia st	helipolis	cairo
Beta for Export	falaky st	babel look	cairo
Gama limited	mansour st	Helwan	cairo
Byte for IT	aly fadl	loran	Alexandria
Guide a abdelaziz	kafr	abdo	Alexandria

(10 row(s) affected)

Transact-SQL

```
select ALL c.name, c.address, c.city, c.state
from customer c
where c.name IN
(select o.name
from orders o, part p
where o.part_no = p.part_no
AND
o.quantity * p.price >
(select avg(o.quantity * p.price)
from orders o, part p
where o.part_no = p.part_no))
go
```

Name	Address	City	State
COMPUSIENCE	gamaa street	giza	giza
Sara Intl Group	makram obaid st	nasr city	cairo
Basic Solutions	akkad st	nasr city	cairo
Beta for Export	falaky st	babel look	cairo

(4 row(s) affected)

هل فهمت تداخل الجمل وقمت بتحليل النتيجة ؟

دعنا نقوم بتحليل الجملة السابقة وتقسيمها الى عدد من Queries فيما يلي

الجملة الأساسية

```
---------------------------------- الجملة الأساسية Query ----------------------------
select ALL c.name, c.address, c.city, c.state
from customer c
where c.name IN
---------------------------------- أول جملة Sub-Query ----------------------------
(select o.name
from orders o, part p
where o.part_no = p.part_no
AND
o.quantity * p.price >
```

```
---------- Sub-Query ---------- الجملة الأخيرة ----------
(select avg(o.quantity * p.price)
from orders o, part p
where o.part_no = p.part_no))
go
```

سنقوم الآن بتنفيذ الجملة الأخير بمفردها :

```
Transact-SQL
select avg(o.quantity * p.price)
from orders o, part p
where o.part_no = p.part_no
go
(no Column Name)
58102
(1 row(s) affected)
```

هذه الجملة بعد تنفيذها سوف تعطينا نتيجة واحدة أو قيمة واحدة ، هذه القيمة سوف يتم تمريرها إلى جملة Sub-Query الأولى

سنقوم الآن بتنفيذ الجملة الأولى والثانية معا ، وهذا معناه تمرير القيمة التي حصلنا عليها من الجملة الأخيرة إلى الجملة التي تسبقها :

```
Transact-SQL

select o.name
from orders o, part p
where o.part_no = p.part_no
AND
o.quantity * p.price >
(select avg(o.quantity * p.price)
from orders o, part p
where o.part_no = p.part_no)
go

Name
--------------------
```

COMPUSIENCE
COMPUSIENCE
Sara Intl Group
Basic Solutions
Beta for Export
(5 row(s) affected)

ما حدث أن الجملة الأخيرة مررت نتيجة واحدة هي 58102 الى الجملة الأولى وكنتيجة لتنفيذ الجملة الأولى ومعها نتيجة الجملة الأخيرة حصلنا على النتيجة الحالية ، هذه النتيجة سوف يتم تمريرها إلى الجملة الأساسية !

تعال نقوم بتنفيذ الجملة الأساسية دون استخدام Sub-Query :

```
select ALL c.name, c.address, c.city, c.state
from customer c
where c.name IN سنقوم بتعطيل هذه الجملة
```

Transact-SQL

Name	Address	City	State
COMPUSIENCE	gamaa street	giza	giza
Egypt Cables	orabi street	mohandseen	giza
United Group	saleem street	ziton	cairo
Sara Intl Group	makram obaid st	nasr city	cairo
Basic Solutions	akkad st	nasr city	cairo
Computer Technology	horia st	helipolis	cairo
Beta for Export	falaky st	babel look	cairo
Gama limited	mansour st	helwan	cairo
Byte for IT	aly fadl	loran	alexandria
Guide	a abdelaziz	kafrabdo	Alexandria

(10 row(s) affected)

أي أن الجملة الأساسية التقليدية سوف تطبع محتويات الجدول ، ولكن عند تفعيل جملة where c.name IN فسوف يتم تنفيذ الجملة الأساسية مع الأخذ في الاعتبار نتيجة تنفيذ Sub-Query الأولى والتي سوف يتم تنفيذه مع الأخذ في الاعتبار نتيجة تنفيذ جملة Sub-Query التي تسبقها وهكذا !!

هل نستخدم Where فقط للتعامل مع Sub-Query ؟

الجواب لا ، إلا أنك تستطيع استخدام كل من Having و Group By كما سنرى في المثال التالي :

```
select name, avg(quantity)
from orders
GROUP BY name
HAVING avg(quantity) >
(SELECT avg(quantity)
from orders)
go
```

Name	(No Column Name)
Basic Solutions	320
Beta for Export	310
Computer Technology	430
Egypt Cables	196
Sara Intl Group	220

(5 row(s) affected)

استخدام Exist مع sub-query

تقوم Exists باختبار وجود قيمة معينة في جدول فاذا وجدت هذه القيمة تكون النتيجة true وعدم وجود القيمة يعطينا نتيجة false وبعدها يستمر تنفيذ query مع أخذ هذه القيمة في الاعتبار :

Transact-SQL

```
select * from stock
where exists
(select * from stock
where description = '1.7GHz');
```

part_no	description	qty
MB01	Mother Board Giga	120

MB02	Mother Board Accorp	444
MO02	LG 17" Monitor	80
MO03	Samsong 15"	100
MO04	Samsong 17"	85
P001	1.7GHz	40
P003	2.4GHz	110
HD01	HDD 40GB	75
HD02	HDD 80GB	175
HD03	HDD 115GB	275

(10 row(s) affected)

عند تنفيذ **Query** تم اختبار الجدول **stock** للتأكد من وجود **'description = '1.7GHz** باستخدام **exists** وبالتالي وجدت النتيجة **true** فتم تنفيذ الجملة **select** الأولى ، أي أننا نقول ببساطة تأكد من وجود وصف الصنف أولا فاذا وجدت قم بطباعة كامل الجدول ، واذا لم تجده فلا تطبع شيئا كما في المثال التالي :

```
select * from stock
where exists
(select * from stock
where description = 'VGA Card');
```

```
 part_no   description          qty
```

(0 row(s) affected)

في **Query** السابقة تم البحث عن **VGA Card** داخل الجدول فلم يعثر عليها . إذا نتيجة الشرط **Exists** سوف تكون هذه المرة **false** وبالتالي فلن تتم طباعة الجدول .

نلاحظ أن **EXISTS** تختبر فقط وجود قيمة من عدمه ولكنها لا تتدخل في طباعة الجدول فاذا وجدت القيمة في **SUB-QUERY** يتم ارسال النتيجة الى **QUERY** الأساسية وبناء على النتيجة تقوم **QUERY** الأساسية بالتنفيذ !

تأمل المثال التالي :

TRANSACT-SQL

```
select PART_NO, DESCRIPTION from stock S
where exists
(select PART_NO from ORDERS O
where S.PART_NO = O.PART_NO);
```

PART_NO	DESCRIPTION
MB01	Mother Board Giga
MO04	Samsong 17"
HD01	HDD 40GB
HD03	HDD 115GB

(4 row(s) affected)

في هذا المثال تم اختبار رقم الصنف **PART_NO** بين جدولين من خلال **SUB-QUERY** وكانت النتيجة وجود أربعة صفوف تشترك في رقم الصنف من هنا قامت **QUERY** الأساسية بطباعة كل سجل أو صف تحقق هذا الشرط ، من هنا نفهم أن **SUB-QUERY** تعمل على صف بعد الآخر !

نفس النتيجة ستحصل عليها باستخدام **MYSQL**

استخدام ALL مع Sub-Query :

نجرب الآن استخدام **ALL** بدلا من **EXISTS** بنفس **QUERY** السابقة لنرى النتيجة :

في المثال التالي نستخدم علامتي <> وتعني لا يساوي ، لذلك ترى النتيجة متحققة على كل السجلات فيما عدا السجل الذي تحقق فيه الشرط = ولكن داخل **SUB-QUERY** :

```
select PART_NO, DESCRIPTION from stock
where DESCRIPTION <> ALL
(select DESCRIPTION from STOCK
where DESCRIPTION = 'HDD 40GB');
```

PART_NO	DESCRIPTION
MB01	Mother Board Giga
MB02	Mother Board Accorp
MO02	LG 17" Monitor
MO03	Samsong 15"
MO04	Samsong 17"
P001	1.7GHz
P003	2.4GHz
HD02	HDD 80GB
HD03	HDD 115GB

نرى أن تحقق أي شرط في SUB-QUERY تكون من نتيجته تنفيذ جملة QUERY الأصلية طالما أن نتيجة تنفيذ SUB-QUERY هي TRUE !والشرط هنا هو ألا يرد وصف "HDD 40GB" في جدول stock وبالفعل هو لا يوجد نتيجة لتنفيذ الـ Sub-Query .

استخدام ANY مع Sub-Query :

أما المثال التالي فاستخدمنا فيه ANY ويعني أن النتيجة تتوقف على ما يتحقق في SUB-QURY كما نرى :

TRANSACT-SQL

```
select PART_NO, DESCRIPTION from stock
where DESCRIPTION = ANY
(select DESCRIPTION from STOCK
where DESCRIPTION = 'HDD 40GB');
```

PART_NO	DESCRIPTION
HD01	HDD 40GB

لذا فقد تمت طباعة السجل المتحقق فيه الشرط ! كل سجل يتحقق فيه الشرط المحدد في Sub-Query يتم طباعته

MySQL

```
select PART_NO, DESCRIPTION from stock
where DESCRIPTION = ANY
```

```
(select DESCRIPTION from STOCK
where DESCRIPTION = 'HDD 40GB');
```

PART_NO	DESCRIPTION
HD01	HDD 40GB

generated 3/9/2005 9:07:28 AM by MySQL-Front 1.22

استخدام Some مع Sub-Query

ومثل **ANY** نستخدم في المثال التالي **SOME** والنتيجة تعتمد على الشرط الذي تحقق في جملة **SUB-AUERY** !

```
select PART_NO, DESCRIPTION from stock
where DESCRIPTION = SOME
(select DESCRIPTION from STOCK
where DESCRIPTION = 'HDD 40GB');
```

```
PART_NO    DESCRIPTION
HD01       HDD 40GB
```