الفصل الخامس عشر

التحكم في العمليات

Controlling Translations

تعلمنا في الفصول السابقة أساسيات استخدام SQL Query وحصلنا على المعلومات التي تمكننا من التعامل مع قواعد البيانات ، يمكنك أن تتدرب وتتقن الفصول السابقة.

سوف نتحدث في هذا الفصل عن التحكم في العمليات Controlling Transactions

وقد سبق أن أشرنا أن لغة SQL تتكون من ثلاثة أقسام يطلق علي القسم الذي يخص بالتحكم في العمليات DCL (Translations) أو Data Control Language.

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

- ما هي العمليات Transaction
- بدء عملية Beginning transaction
- إنهاء العمليات Finishing Transaction
- إلغاء عملية Canceling Transaction
- تداخل العمليات
- استخدام Transaction Savepoints

ما هي العملية Transaction

فهمي لكلمة Transaction بعيدا عن العمل في SQL أنها مجموعة عمليات مثل تلك التي تتم بالسحب والإيداع من البنك فمثلا في كشف الحساب الذي يرسله إليك البنك تظهر فيه عمليات إيداع وسحب خلال فترة معينة . هذه العمليات خلال الفترة التي يغطيها

كشف الحساب تسمى Transactions أما هنا مع SQL فإنها تعني عملية (Unit of Work) أو الخطوات التي تتم منذ بداية العمل وحتى انتهائه . فمثلا إدخال تعديل على حقل معين يبدأ من لحظة التأكد من وجود الحقل ثم التعديل ثم انتهاء التعديل ، أو عدم وجود الحقل والعودة لنقطة البداية ، هذه تعتبر عملية لها نقطة بداية ونقطة انتهاء لذا تسمى Transaction أو عملية Unit Of Work . وما يحدث مثلا عند ذهابك الى آلة الصارف الآلي لسحب نقود ، أنك تقوم بوضع الكارت وكلمة السر في الآلة لتبدأ Transaction وبعد التأكد تنتهي Transaction ثم تبدأ أخرى لاتمام عملية السحب بسؤالك عن المبلغ ثم اعداد القيمة واستلامك لها . في هذه الأثناء يتم خصم القيمة من رصيدك بالبنك أو اخطارك بأن الرصيد لا يكفي وهنا تنتهي العمليات Transaction إما باستلام النقدية أو برسالة موجعة تقول لك "عفوا الرصيد لا يكفي" . أما موظف البنك فان هذه العملية تسمى Transaction سواء كان يعرف أن هذا مصطلح في قاعدة البيانات أو لا ، المهم أنها تسمى Transaction

لا تدعم MYSQL موضوع هذا الفصل وهو Transaction Control أو التحكم في العمليات ولذلك فإن الأمثلة الواردة في هذا الفصل تستخدم فقط ORACLE و SYBASE لذا لزم التنويه لتراجع نظام قاعدة البيانات الموجود على جهازك .

فهم التحكم في العمليات Basics of Transactions Control

تلقي محمود اتصالا هاتفيا من زوجته وجرى الحوار التالي :

- مالك فرحان ليه
- خلصت البرنامج والداتا بيز وتم تركيبهم
- وبعدين
- ارسلت زميلي حسن ليضع ألفين جنيه عن طريق الماكنة

- هه
- وارسلت علي ليسحب ٥٠٠ جنيه
- ايوه
- اما جمال فقد ذهب الى ماكينة أخرى ليستعلم عن الرصيد
- امتى
- من نصف ساعة
- على الحساب بتاعنا
- أيوه

سمع محمود صوت وضع السماعة دون أن يسمع كلمة مبروك وبعد قليل جاء جمال ليقول الاستعلام لم يعمل والماكينة توقفت عن العمل ذهبت الى ماكنة أخرى فظهرت رسالة أن هناك خطأ في النظام وبعدها قال علي لا يوجد رصيد فلم أستطع السحب وقبل أن يفق محمود من هذه الأخبار تلقى اتصالا من زوجته "مبروك ، النظام الجديد حلو ، مفيش خمس دقائق كنت سحبت كل المبلغ" ، وكان هذا أول اختبار للنظام !!!

كان محمود في حالة حنق شديد من زوجته ، وفي حالة احباط من المشاكل الحادثة في النظام خاصة وقد بدأ قسم العملاء يتلقى العيد من الاتصالات ، بينما كان قد تسرع وأخبر رئيسه في العمل بالانتهاء من النظام وتركيبه ، ذلك قبل أن يقوم بالعديد من الاختبارات اللازمة . أخذ يفكر ، قواعد البيانات مصممة لكي يتعامل العديد من المستخدمين مع جدول واحد أو أكثر في نفس الوقت وربما استخدام نفس الحقل field في نفس الوقت ولكن كيف يمكن التحكم في التعديلات التي قد تحدث على نفس الحقل بحيث تتم كل عملية وبعدها تبدأ عملية أخرى وهكذا حتى لا يحدث ارتباك أو تضارب .

رجع إلى الأسس التي تعلمها في قواعد البيانات وبرامج التطبيقات ، وجد أنه يجب أن تنتهي العمليات transaction دون أي أخطاء لتبدأ Transaction أخرى ، كما يجب

التأكد من عودة البيانات الى أصلها اذا حدث خطأ أثناء التنفيذ **Rollback** . أما سرعة تنفيذ العمليات فلن يشعر بها المستخدمين لأنها ثواني أو أجزاء من الثواني .

بدء عملية Beginning transaction

جميع نظم قواعد البيانات التي تستخدم وحدات عمل أو **Transactions** يجب أن تشتمل على طريقة لإعلام النظام أو إخباره عن بداية العملية (العملية هنا هي مجموعة عمليات لها بداية ونهاية) يجب أن تراجع الشكل العام المستخدم مع قاعدة البيانات التي تعمل عليها سواء كانت **Sybase SQL Server** أو **ORACLE** .

الشكل العام (**Syntax**) التالي هو المستخدم مع **ORACLE** .

SET TRANSACTION {READ ONLY | USE ROLLBACK SEGMENT segment}

وكما هو الحال مع عدد من نظم إدارة قواعد البيانات ، فان نظام **Oracle** يتيح لك تحديد متى تبدأ العمليات ومتى تنتهي ، كما يتأكد من عدم حدوث أي تعديلات أثناء تنفيذ عملية **Transaction** ، أي أن التعديلات لا تبدأ الا مع بداية عملية أخرى وهكذا .

وفيما يلي مثال بسيط لكيفية عمل استعلامات من خلال أوراكل مع استخدام تعليمات تمنع أي تعديلات في الجدول الذي نستخدمه :

Oracle SQL

```
SQL > SET TRANSACTION READ ONLY;
SQL > SELECT * FROM PERSONAL
        WHERE FNAME = 'MEDO';
    . . . OTHER COMMANDS . . . .
COMMIT;
```

سوف نناقش أمر **COMMIT** فيما بعد وللتوضيح نقول أن**SET TRANSACTION** تعني بداية العملية أما **READ ONLY** ومعناه قراءة فقط لتمنع أي عمليات أخري من التنفيذ قبل انتهاء العملية الحالية

تستخدم خاصية **Read Only** ضمن نظام أوراكل مع الأوامر التالية :

SELECT, LOCK TABLE, SET ROLE, ALTER SESSION, ALTER SYSTEM .

في حين يساند نظام **Oracle DBMS** استخدام خاصية **Read Only** فان **Sybase** لا تساند هذه الخاصية . أما الشكل العام لبداية عملية في نظام **SQL Server** فهو على النحو التالي :

BEGIN TRAN transaction_name

أو

BEGIN TRANSACTION transaction_name |

هناك اختلاف طفيف فيما بين **Oracle SQL** وبين **SQL-Server's Transact-SQL** ، بينما لا يسمح نظام **Sybase** باستخدام **Read Only** فانه يسمح لك بأن تضع اسما لالعمليات **transaction_name** هذا اذا كانت العمليات **transaction** مستخدمة

إنهاء عملية Finishing Transaction

العمليات **Transaction** تتضمن عدد من التعديلات أو الاضافات الى الجداول وعندما تنتهي بسلام نريد حفظ تلك التعديلات وهذا يشبه الى حد كبير العمل مع ملف **Word** أو **Excel** فعندما تقوم باغلاق الملف تظهر رسالة تأكيد "هل تريد حفظ التغييرات؟" يمكنك الاجابة بحفظ التعديلات أو الاجابة "لا" وتعني عدم الحفظ أو الغاء آخر تعديلات .
لانهاء عملية **Transaction** وحفظ الاجراءات التي تتضمنها من تعديل أو اضافة أو الغاء استخدم الأمر **COMMIT** ، وبعض المبرمجين يستخدمون هذا الأمر **COMMIT** في البداية للتأكد من انتهاء أي أعمال سابقة ، يكون نص هذا الأمر كالتالي :

ORACLE SQL
COMMIT [WORK]

```
[       COMMENT 'test'
|       FORCE 'text'  [, integer]  ];
```

تذكر القاعدة العامة التي تقول أن ما بين الأقواس [ ] اختياري في أوامر SQL ولذلك فلست في حاجة لإضافة COMMENT أو FORCE إلى أمر COMMIT .

Sybase SQL Syntax
```
COMMIT [TRANSACTION  | TRAN |  WORK]  (transaction_name)
```

Microsoft Transact-SQL
```
COMMIT [TRANSACTION  | TRAN |  WORK]  (transaction_name)
```

سنستخدم هنا مثال يوضح كيفية انهاء العمليات (COMMIT Transaction) باستخدام نظم قواعد البيانات المختلفة :

ORACLE SQL
```
SET TRANSACTION;
INSERT INTO CUSTOMERS VALUES
        ('Arabian Gulf Co', '22 Aswan Street', 'Heliopolis', 'Cairo',
11133, 7);
COMMIT;
```

SYBASE SQL
```
1. BEGIN TRANSACTION
2. INSERT INTO CUSTOMERS VALUES
   ('Arabian Gulf Co', '22 Aswan Street', 'Heliopolis', 'Cairo',
   11133, 7)
3. COMMIT TRANSACTION
4. GO
```

وعند استعراض بيانات الجدول في الحالتين بواسطة الأمر SELECT * customers سيظهر السجل الجديد الذي قمت بإضافته في نهاية الجدول

MICRSOFT TRANSACT- SQL
```
BEGIN TRANSACTION
INSERT INTO CUSTOMERS VALUES
```

```
        ('Arabian Gulf Co', '22 Aswan Street', 'Heliopolis', 'Cairo',
        11133, 7)
COMMIT TRAN
GO
```

إلغاء عملية Canceling Transaction

ربما يحتاج المستخدم لتنبيهه هل تريد اتمام العمليات Transactions بما حدث فيها من تعديلات ؟ أم لا . يمكنك استخدام أمر Rollback لكي تقوم باعادة كل البيانات كما كانت قبل تنفيذ كل العمليات Transactions وكأنه لم يتم تنفيذ أي عمل فمثلا قم بهذه التجربة

```
MICRSOFT TRANSACT- SQL
BEGIN TRAN
INSERT INTO CUSTOMERS VALUES
        ('Egypt Import', '33 Hassan Street', 'Maadi', 'Cairo', 11443, 8)
Rollback TRAN
GO
SELECT * FROM CUSTOMERS
GO
```

في هذه الحالة ستحصل علي صفوف جدول customers كما كانت قبل إدخال بيانات السجل . بمعنى يتم إلغاء أمر INSERT وكأنه لم يحصل

تداخل العمليات

أحياناً تتداخل العمليات Nested Transactions كما يظهر في المثال التالي والمستخدم مع نظام Sybase DBMS :

```
1>      begin transaction new_account
2>      insert Customers values
3>      ('Arab Fruits' , 'Mokatam Deserts 25' , 'White House',
        'AL' , 32854, 6)
4>      if exists(select * from CUSTOMERS where name =
        'Arab Fruits')
5>      begin
6>      begin transaction
```

```
7>      insert BALANCES values(1250.76, 1431.26, 8)
8>      end
9>      else
10>     rollback transaction
11>     if exists(select * from BALANCES where Account_ID =
        8)
12>     begin
13>     begin transaction
14>     insert ACCOUNTS values(8, 6)
15>     end
16>     else
17>     rollback transaction
18>     if exists (select * from ACCOUNTS where Account_ID
        = 8 and Customer_ID = 6)
19>     commit transaction
20>     else
21>     rollback transaction
22>     go
(1 row(s) affected)
(1 row(s) affected)
(1 row(s) affected)
```

لاحظ أننا استخدمنا في السطر الأول من هذا المثال **Begin** للدلالة على بداية العمليات **Transaction** ثم بعدها قمنا بتنفيذ **Insert** بعدها قمنا باختبار الجملة "هل تم تنفيذها بصورة صحيحة " **if exists** " إذا كان الاختبار صحيح يتم تنفيذ العمليات التالية " سطر ٥" باستخدام الأمر **Begin** ، تنفيذ عملية داخل عملية أخرى يسمى **Nested Transactions** .

سنعود لشرح الأوامر التي لم نتناولها بعد مثل أمر **ROLLBACK** والموجودة في سطر ٢١ . تم استخدام هذا المثال نفسه دون أي تغيير مع **Microsoft SQL Server** وقد أعطى نفس النتائج التي حصلنا عليها عند استخدامه مع **Sybase** .

بعض نظم قواعد البيانات مثل **SQL Server** تستخدم **Autocommit** وفي هذه الحالة يتم التأكد من كل عملية **Transaction** . إذا تم تنفيذها دون أخطاء يستمر العمل

ويستمر في الوحدة التي تليها . أما إذا حدث خطأ يتم تنفيذ Rollback وإعادة كل شيء إلى ما كان عليه .

قد يشرح لنا الجزء التالي وفي الأسطر ١٨ الى ٢١ وجه من وجوه استخدام rollback ، نحاول معا استرجاع هذه الأسطر ونفهمها :

```
18>    if exists (select * from ACCOUNTS where Account_ID
   = 8 and Customer_ID = 6)
19>    commit transaction
20>    else
21>    rollback transaction
```

في السطر ١٨نقول إذا if exists فإذا كان الجواب نعم قمنا بحفظ العمل " commit transaction " كما يظهر في السطر ١٩ ، وفي السطر ٢٠ نقول else بمعنا اذا لم يكن موجودا فإننا نلغي العمليات rollback transaction كما يظهر في السطر ٢١ وسنعرف استخدامات أخرى لاحقا في هذا الكتاب .

لاحظ عندما تستخدم Rollback فانك تلغي كل ما تم تنفيذه من أوامر ، خاصة عندما تستخدم عدد من وحدات عمل متداخلة Nested Transactions فان أمر rollback يتم تنفيذه على العمليات الأساسية وجميع الوحدات بداخلها .

يجب أن يسبق كل أمر Commit الأمر Begin Transaction والا سوف تفاجأ برسالة خطأ

استخدام Transaction Savepoints

كنت أقص موضوعا على أحد الأصدقاء وفجأة اضطررت للتوقف فقلت له "خليك فاكر احنا وقفنا لحد فين" فرأيته يضحك ولا أدري لماذا ضحك الصديق ؟

أعتقد أننا سوف نستخدم نفس الطريقة مع قواعد البيانات ، حيث يمكنك كتابة query مهما كانت تحتوي من أوامر ومهما كانت طويلة ، ولكنك سوف تخبر قواعد البيانات كل فترة "خليك فاكر احنا واقفين فين" يطلق على هذا المفهوم في قواعد البيانات SavePoint وقد نسميها نقطة العودة نستخدمها حين نرغب في الغاء بعض الأوامر وآثارها تماما مثلما يحدث مع UNDO فانها تعطيك قائمة منسدلة تختار منها UNDO للعودة الى أي مرحلة .
والشكل العام لإنشاء SAVEPOINT كما يلي :

Oracle SQL
SAVPOINT savepoint_name

Sybase
SAVE TRANSACTION savepoint_name

Microsoft Transact-SQL
SAVE TRANSACTION savepoint_name

مثال

Oracle SQL
```
SQL > SET TRANSACTION;
SQL > UPDATE BALANCES SET  CURRENT_BAL = 25000 WHERE
SQL > ACCOUNT_ID=5;
SQL > SAVEPOINT save_it ;
SQL > DELETE FROM BALANCES WHERE ACCOUNT_ID = 5;
SQL > ROLLBACK TO SAVEPOINT save_it;
SQL > COMMIT;
SQL > SELECT * FROM BALANCES ;
```

Sybase / *Microsoft Transact-SQL*

```
1> begin transaction;
2> update BALANCES set  current_bal = 25000 WHERE
   account_id=5;
3> save transaction save_it
```

```
4> delete from BALANCES where account_id = 5;
5> rollback transaction save_it;
6> commit transaction;
7> go
8> select * from BALANCES;
9> go
```

عند تنفيذ الـ Query السابقة ستحصل على نفس النتيجة التالية :

Average	Current_Bal	Account_ID
1299	854	1
5427	6543	2
220	5433	3
38	432	4
1371	25000	5
1431	7865	6
346	6655	7

ومنها تلاحظ أننا بدأنا العمليات في السطر رقم ١ ثم عدلنا رصيد الحساب رقم ٥ ليصبح 2500 في السطر رقم ٢ . في السطر رقم ٣ وضعنا savepoint يعني نقطة يمكن أن نعود إليها . في السطر رقم ٤ حذفنا الحساب رقم ٥ الذي عدلناه وفي السطر رقم ٦ تم إلغاء العمليات ليس من أولها ولكن من عند النقطة SAVEPOINT الموجودة في السطر رقم ٣ بمعنى أن السجل الذي ألغي سيعود كما كان .

ثم أنهينا العمليات بأمر COMMIT TRANSACTION في السطر رقم ٦ . دقق في النتيجة التي حصلت عليها تلاحظ أن الحساب رقم ٥ موجود بالتعديلات التي تمت ولم يتأثر بأمر الإلغاء لأن SAVEPOINT كانت بعد أمر UPDATE وقبل أمر DELETE .