بسم الله الرحمن الرحيم

Database Programming

برمجة قواعد البيانات

عدد الساعات: ٢ نظري+٢عملي

الرمز: ٣١٤ حسب

المتطلبات: ٢٢٣ حسب (مبادئ قواعد البيانات)

أستاذات/المادة: م. لندا عمر البدري - م. نجلاء حسن

المحاضرة الخامسة والسادسة

تابع :ادارة المعاملات

Transactions Management

تمييز الجداول على أساس قابلية الاسترجاع
Characterizing Schedules Based on Recoverability

- عندما يتم تنفيذ المعاملات بشكل متداخل، فان ترتيب تنفيذ العمليات المكونه للمعاملات يُعرف باسم الجدول (Schedules).
- **جداول المعاملات:**

جدول المعاملات S والخاص بعدد n من المعاملات T1.T2,...,Tn هو ترتيب لعمليات المعاملات الموجودة في الجدول مع مراعاة أن عمليات كل معاملة Ti في S يجب أن تظهر بنفس ترتيبها في المعاملة Ti.بالرغم من أنه قد يحدث تداخل تأثير معاملة أخرى Tj مع تأثير المعاملة Ti الموجودة داخل الجدول.

- ولاغراض الاسترجاع والتحكم التزامني نركز بشكل كبير على العمليات write-item, Read-item, Commit,, Abort.
- مثال: الجدول Sa الذي يمثل المعاملتين T2,T1 يمكن التعبيرعنه كما يلي:

Time ↓

T1	T2
Read-item(x); x:=x-n; Write-item(x); Read-item(y); Y:=y+n' Write-item(y);	 Read-item(x); X:=x+m; Write-item(x);

Sa:r1(x);r2(x);w1(x);r1(y);w2(x);w1(y);

- وبنفس الطريقة يمكن التعبير عن الجدول Sb كما يلي مع افتراض أن المعاملة T1 اجهضت (Aborted) بعد اجراء العملية :read-item(y)
- مثال: الجدول Sb الذي يمثل المعاملتين T2,T1 يمكن التعبيرعنه كما يلي:

Time

T1	T2

Read-item(x);
x:=x-n;
Write-item(x);
 	
read-item(y);
	
	Read-item(x);
	X:=x+m;
	Write-item(x);

Sb:r1(x);w1(x);r2(x);w2(x);r1(y);a1;

- تكون عمليتين في حالة نزاع conflict اذا حققتا الشروط التالية:

١- اذا كانت كلتا العمليتين من معاملتين مختلفتين.

٢- اذا كانت كلتا العمليتين تتعاملان مع نفس العنصر (مثل x)

٣- أن تكون على الأقل أحد العمليتين هي عملية write-item ()

- مثال: في الجدول Sa العمليات في حالة نزاع هي:

r1(x) و w2(x) - r2(x) و w1(x) - w1(x) وw2(x)

أما العمليات التالية ليست في حالة نزاع:

r1(x) و r2(x) : لأن العمليتين هما قراءة ولاتوجد عملية write .

W2(x) و w1(y) :لأن كل عملية تتعامل مع عنصرين مختلفين.

W1(x) و r1(x) :لأن كلا العمليتين تنتميان الى نفس المعاملة.

الجدول الكامل

- يطلق على الجدول S الذي يشمل على n معاملة T1,T2,T3,…Tn جدول كامل اذا حقق الشروط التالية:

١- جميع العمليات الموجودة في الجدول هي كل العمليات الموجودة في المعاملات T1,T2,T3,…Tn ، وهذه العمليات تشمل العمليات commit أو abort كآخر عملية في أي معاملة.

٢- لكل عملية في أي معاملة Ti يجب أن تظهر في الجدول بنفس ترتيب حدوثها في المعاملة Ti .

٣-لأي عمليتين في حالة نزاع في الجدول احداهما يجب أن تحدث قبل الأخرى.

- قد يسمح في الجدول للعمليات الغير متنازعة بأن تحدث من غير تحديد أيهما حدثت أولاً، وفي هذه الحالة يُعرف الجدول على أنه ترتيب جزئ (Partially order) للعمليات في n معاملة.

(قد تحدث هذه الحالة أيضاُ في حالة حدوث عمليتين في نفس الوقت بشكل متوازي)

تابع:الجدول الكامل

- بالرغم مما سبق ذكره فان ترتيب العمليات المتنازعة يجب أن يتم توضيحه في الجدول (شرط ٣)، وكذلك لأي عمليتين تنتميان الى نفس المعاملة (الشرط٢).
- حيث أن أي معاملة تنتهي بالعملية commit أو aborted ، فان الجدول الكامل لا يمكن أن يحتوي على معاملة فعالة (Active transactions).
- بشكل عام من الصعب وجود جدول كامل في نظام يعمل لأن كل المعاملات النشطة يتم تغذيتها الى النظام بشكل مستمر.

تمييز الجداول على أساس قابلية الاسترجاع

- بعض الجداول من السهل استردادها بعد فشل المعاملات بينما، نجد أن عملية الاسترداد هذه تكون أكثر تعقيداً في جداول أخرى.
- يمكن تمييز الجداول بناءً على قابلية استردادها الى:

- جداول قابلة للاسترداد recoverable schedules:

تحتوي على معاملات حدثت لها عملية التزام commited (ليس الضرورة أن يجرى لها عملية roll back)

- جداول غير قابلة للاسترداد non-recoverable schedules: لا تحتوى على معاملات ملتزمة.

تابع:تمييز الجداول على أساس قابلية الاسترجاع

- بفرض أن T تقرأ عناصر (مثل x) تم كتابتها بواسطة 'T، فان الجدول S قابل للاسترجاع :

- اذا كان لا يوجد به معاملة T ملتزمة الى أن تكون جميع المعاملات 'T التي تقوم بعملية كتابة العناصر التي تقرأ منها T قد تم الزامها.

- 'T لم يتم اجهاضها aborted قبل أن تقوم T بقراءة X

- لا توجد معاملة أخرى قامت بالكتابة مرة أخرى على x قبل أن تقوم T بقراءتها.

تابع:تمييز الجداول على أساس قابلية الاسترجاع

- الجداول القابلة للاسترجاع تتطلب عمليات معقدة لاسترجاعها ولكن اذا توفرت معلومات مناسبة (من سجل المعاملة) فانه من السهل ايجاد خوارزم الاسترجاع.
- الجدولين Sa، Sb (تم ذكرهما سابقاً) هما جداول قابلة للاسترجاع، حيث أنهما يحققان التعريف السابق.

Sa:r1(x);r2(x);w1(x);r1(y);w2(x);w1(y);

Sb:r1(x);w1(x);r2(x);w2(x);r1(y);a1;

والجدول التالي أيضاً قابل للاسترجاع وهو يشبه Sa ما عدا اضافة امري الالتزام C2,C1 .

S□a:r1(x);r2(x);w1(x);r1(y);w2(x);c2;w1(y);c1;

تابع:تمييز الجداول على أساس قابلية الاسترجاع

- مثال آخر:

Sc:r1(x);w1(x);r2(x);r1(y);w2(x);c2;a1;

Sd:r1(x);w1(x);r2(x); r1(y);w2(x);w1(y);c1;c2;

Se:r1(x);w1(x);r2(x); r1(y);w2(x);w1(y);a1;a2;

- الجدول Sc غير قابل للاسترجاع لأن T2 تقرأ العنصر x بعد أن تقوم T1 بكتابته وقد تم التزام T2 قبل التزام T1 . وذلك بسبب أنه في حالة اجهاض المعاملة T1 فان قيمة x التي قامت بكتابتها تكون غير متاحة لـ T2 وبالتالي فان T2 تكون معرضه للاجهاض بعد ان تم الزامها مما يجعل الجدول غير قابل للاسترجاع.

- وحتى يكون الجدول قابل للاسترجاع يجب أن يتم الزام T1 قبل T2 كما في الجدول Sd .

- في حالة أنه حدث اجهاض للمعاملة T1 فهذا يترتب عليه اجهاض للمعاملة T2 كما هو موضح في الجدول Se .

تابع:تمييز الجداول على أساس قابلية الاسترجاع

- في الجداول القابلة للاسترجاع لا توجد معاملات ملتزمة يحدث لها تراجع، وبالرغم من ذلك فانه من الممكن حدوث ما يعرف باسم الاجهاض المتعاقب أو التراجع المتعاقب

(cascading roll back or cascading abort)

حيث يتم تراجع للمعاملات الغير ملتزمة لأنها تقرأ عناصر من معاملات فاشلة، هذا ما نجده في الجدول Se حيث نجد ان المعاملة T2 حدث لها اجهاض (أو تراجع)لأنها تقرأ العنصر x من T1 و T1 قد تم اجهاضها.

- عملية التراجع المتعاقب cascading roll back تؤدي الى استهلاك الوقت بسبب انه قد يحدث تراجع لعدد كبير من المعاملات.

تابع:تمييز الجداول على أساس قابلية الاسترجاع

- يطلق على الجدول أنه بدون تراجع متعاقب (cascadeless) أو avoid cascading roll back اذا كان كل معاملة في الجدول تقرأ فقط العناصر المكتوبة من قِبَل معاملات ملتزمة.
- لتحقيق هذا الشرط فان العملية r2(x) في الجدولين Sd and Se يجب أن تؤجل الى أن يتم التزام المعاملة T1 (أو اجهاضها) وهذا يؤخر اجراء المعاملة T2 ولكن يضمن عدم وجود تراجع متعاقب.

تابع:تمييز الجداول على أساس قابلية الاسترجاع

- يوجد نوع آخر من الجداول يطلق عليه الجدول الصارم strict schedule وفيه لا يسمح لأي معاملة باجراء عمليه قراءة أو كتابة عنصر x الى أن يتم التزام آخر معاملة قامت بكتابة x .
- الجداول الصارمة من السهل استرجاعها.

تابع:تمييز الجداول على أساس قابلية الاسترجاع

- جميع الجداول الصارمة هي جداول بدون تراجع متعاقب (cascadeless) وجميع الجداول التي بدون تراجع متعاقب قابلة للاسترجاع (recoverable) .

- مما سبق يمكن تمييز الجداول بناءً على ثلاثة مصطلحات وهي:

١- القدرة على الاسترجاع recoverablity :

recoverable و un recoverable

٢- بدون التراجع المتعاقب cascading roll back:

cascadeless و cascading roll back

٣- الصرامة strictness:

strict schedule و un strict schedule

A

T1	T2
Read-item(x); x:=x-n; Write-item(x); Read-item(y); Y:=y+n' Write-item(y);	Read-item(x); X:=x+m; Write-item(x);

Time

B

T1	T2
Read-item(x); x:=x-n; Write-item(x); Read-item(y); Y:=y+n' Write-item(y);	Read-item(x); X:=x+m; Write-item(x);

C

T1	T2
Read-item(x); x:=x-n;	
	Read-item(x); X:=x+m;
Write-item(x); Read-item(y);	
	Write-item(x);
Y:=y+n; Write-item(y);	

Time

D

T1	T2
Read-item(x); x:=x-n; Write-item(x);	
	Read-item(x); X:=x+m; Write-item(x);
Read-item(y); Y:=y+n' Write-item(y);	

تمييز الجداول على أساس قابلية التسلسل
Characterizing Schedules Based on Serializability

- بفرض لدينا المعاملتين T1 وT2 ، في حالة عدم السماح بتداخل عمليتين من المعاملتين فان أحد الاحتمالات قد تحدث:

١- أن يتم تنفيذ جميع عمليات T1 بالترتيب ثم يتم تنفيذ جميع عمليات T2 (**schedule A**)

٢- أن يتم تنفيذ جميع عمليات T2 (بالترتيب) ثم تنفذ عمليات T1 (**schedule B**)

- اما في حالة السماح بوجود تداخل بين العمليات من المعاملتين فقد يكون لدينا عدد كبير من الاحتمالات لترتيب العمليات من المعاملتين.
- الجدولين C وD يوضحان أثنين من هذه الاحتمالات

- **مفهوم تسلسل الجداول:**

هو مفهوم يستخدم لتعريف أي الجداول سيكون صحيح عندما يتم تنفيذ عمليات المعاملات بشكل متداخل في الجدول.

- **التسلسل، عدم التسلسل (Serial, Non serial):**

الجدولين A و B يطلق عليهما جداول متسلسلة لان العمليات الخاصة بكل معاملة يتم تنفيذها بشكل متوالي وبدون تداخل للعمليات من المعاملات الاخرى.

الجدولين C وD تسمى جداول غير متسلسلةلأن في كل جدول نجد أن العمليات تنفذ بشكل متداخل بين المعاملتين.

وبشكل عام يطلق على الجدول S أنه جدول متسلسل اذا كان لكل معاملة T موجودة في الجدول فان جميع عمليات المعاملة T يتم تنفيذها بشكل متوالى في الجدول ، والا فان الجدول يكون جدول غير متسلسل (non serial achedules)

- في الجدول المتسلسل تكون معاملة واحدة فقط نشطة في كل مرة.
- Commit أو abort لأي معاملة يعني بداية المعاملة التالية.
- لا يسمح بتداخل العمليات في الجداول المتسلسلة.
- حيث ان كل معاملة سيتم تنفيذها بشكل متتال وبدون تداخل بين عمليات المعاملات فمن المتوقع ان كل معاملة ستتم بطريقة صحيحة، وقد يكون ليس من المهم اياُ من المعاملات سينفذ أولاُ.

عيوب الجداول المتسلسلة

١- تحد من التزامن وتداخل العمليات.

٢- اذا كان هنالك معاملة في انتظار اكتمال عملية I/O فانه لا يسمح للمعاملة الأخرى باستخدام الـ CPU وهذا يقلل من الاستفادة من المعالج.

٣- اذا كانت هناك معاملة T طويلة جداً فجميع المعاملات الأخرى تنتظر الى أن تكمل T جميع عملياتها .

* لذا تعتبر الجداول المتسلسلة غير عملية.

X=90

Y=90

N=3

M=2

X=90-3=87

Y=90+3=93

x=87+2=89

X=90+2=92

X=92-3=89

Y=90+3=93

- بفرض أنه تم اسناد القيم التالية للمتغيرات m, n, y, x:

X=90 ,Y=90, N=3 ,M=2

- عند اجراء المعاملتين T1 وT2 الموضحتين في الجدولين , a و b فان النتيجة تكون : y=93, x=89

وهذا يعني أن الجدولين يعطيان نتيجة صحيحة.

- بينما نجد أن الجدول c يعطي القيم y=93, x=92 بمعنى أنه يعطي قيمة خاطئة للمتغير x وذلك بسبب أن T2 تقرأ قيمة x قبل أن يتم تغييرها بواسطة T1.

- الجدول D يعطي قيم صحيحة للمتغيرين x وy بالرغم من أنه غير متسلسل.

* مما سبق يمكن أن نميز بين الجداول على أساس قابلية التسلسل الى :

جداول قابلة للتسلسل (Serializable schedule)

أو جداول غير قابلة للتسلسل (not serializable schedule)

- الجدول s المكون من n معاملة يكون قابل للتسلسل اذا كان يكافئ الجدول التسلسلى الذي يحتوي على نفس المعاملات n .
- يمكن أن نجد أكثر من جدول غير متسلسل يكافئ جدول متسلسل واحد (أو أكثر) وعليه تكون هذه الجداول قابلة للتسلسل.
- اما الجداول الغير متسلسلة التي لا تكافئ أي من الجداول المتسلسلة تكون جداول غير قابلة للتسلسل.
- السؤال؟؟ :كيف يمكن أن نطلق علي الجداول أنها متكافئة ؟
- توجد عدة طرق للمقارنة بين الجداول و تمييز ما إذا كانت متكافئة أم لا و ابسط طريقة هي مقارنة تأثير الجداول على قاعدة البيانات .
- يطلق على الجدولين أنهما متكافئي النتيجة (result equivalent) إذا كان تأثير الجدولين على قاعدة البيانات يوصلها الى نفس الحالة .

تطبيق PL/SQL

Iterative control statement

Loops

- Program structure that executes a series of program statements, and periodically evaluates an exit condition to determine if the loop should repeat or exit
- حلقات التكرار : هي تركيب برمجي يقوم بتنفيذ مجموعة (سلسلة) من الأوامر بشكل متكرر ، ويتم فيه تقييم شرط للخروج بشكل دوري وبناءً عليه يتم تحديد ما اذا كانت سيتم الاستمرار في الدوارة أم الخروج .
- Pretest loop: evaluates the exit condition before any program commands execute
- Pretest loop : في هذا النوع من الحلقات التكرارية يتم تقيم شرط الخروج من الدورة قبل تنفيذ أى أمر من الأوامر الموجودة داخل الحلقة التكرار .
- Posttest loop: executes one or more program commands before the loop evaluates the exit condition for the first time
- Posttest loop : في هذا النوع من الحلقات التكرارية يتم تنفيذ أمر أو أكثر قبل تقييم شرط الخروج من الدورة
- PL/SQL has 5 loop structures
- لغة PL/SQL توفر ٥ انواع من حلقات التكرار .

The LOOP...EXIT Loop

```
LOOP
  [program statements]
  IF condition THEN
    EXIT;
  END IF;
  [additional program statements]
END LOOP
```

The LOOP...EXIT Loop

```
create table number_table(
      num NUMBER(10)
);
```

```
DECLARE
i       number_table.num%TYPE:=1;
BEGIN
LOOP
   IF i > 10 THEN EXIT ;
     END IF;
      INSERT INTO number_table
      VALUES(i);
      i := i + 1;
END LOOP;
END;
```

The LOOP...EXIT WHEN Loop

```
LOOP
    program statements
    EXIT WHEN condition;
END LOOP;
```

The LOOP...EXIT WHEN Loop

```
DECLARE
i       number_table.num%TYPE:=1;
BEGIN
LOOP
INSERT INTO number_table
      VALUES(i);
      i := i + 1;
EXIT WHEN I >10 ;
 END LOOP;
END;
```

The WHILE...LOOP

WHILE *condition* LOOP

program statements

END LOOP;

The WHILE...LOOP

```
DECLARE
i       number_table.num%TYPE:=1;
BEGIN
WHILE i<=10 LOOP
INSERT INTO number_table
      VALUES(i);
      i := i + 1;

 END LOOP;
END;
```

The Numeric FOR Loop

FOR *counter_variable* IN *start_value* .. *end_value*
LOOP
    program statements
END LOOP;

Loops: FOR Loop

```
DECLARE
   i        number_table.num%TYPE;
BEGIN
   FOR i IN 1..10 LOOP
      INSERT INTO number_table VALUES(i);
   END LOOP;
END;
```

Notice that i is incremented automatically

- Lastly the Cursor FOR Loop , which we will discuss later.

- النوع الأخير من حلقات التكرار هو Cursor FOR Loop، والذي سيتم نتاوله بالشرح لاحقاً .

GOTO statement

- When executed , the GOTO statement changes the flow of control in a PL/SQL block
- عندما يتم تنفيذ جملة GOTO فان ذلك سيؤدي الى تغير تسلسل تنفيذ الأوامر في ال PL/SQL block .
- The two parts needed to code a GOTO statement are :
- لكتابة جملة GOTO فان ذلك يتم على مرحلتين :

1. Defining the label name : a label name is optionally used to name the PL/SQL block .(defined by using angular brackets <<label name >>).

١. تعريف اسم عنوان (Lable) : ويتم ذلك عن طريق كتابة اسم العنوان داخل الأقواس من النوع << >> .

1. Using the GOTO statement to pass the control to the label .

٢. في المرحلة الثانية يستخدم الأمر GOTO لتغيير تسلسل الاوامر ليتم تنفيذ الأوامر من بداية العنوان المذكور في جملة GOTO .

```
DECLARE
.....
BEGIN
.....
<<label_name>>
.....
.....
GOTO label_name;
.....
END;
```

- Transfer of control with the GOTO statement is allowed in the following places :
- يُسمح بتغير تسلسل تنفيذ البرنامج باستخدام جملة GOTO ، في الأجزاء التالية من الوحدة البرمجية (Block):
- From a block or to an executable statement
- من داخل ال BLOCK ، الى أي امر أو اوامر قابلة للتنفيذ Executable statement
- Branch from an exception handler into an enclosing block .
- من جزء ال Exception handler الى داخل ال BLOCK.

```
DECLARE
OLD fare.fisrt%TYPE;
NEW OLD%TYPE;
Mroute_code fare.route_code%TYPE;
BEGIN
Mroute_code:='Ms_as';
Select first INTO OLD
FROM fare WHERE route_code= mroute_code;
For I in 1..3 LOOP
NEW := OLD-(OLD*0.20);
IF NEW<200 THEN
GOTO less200;
ELSE
INSERT INTO NEW_TABLE VALUES (mroute_code, OLD, NEW);
OLD:=NEW;
END IF;
END LOOP;
GO TO successful;
<<less200>>
DBMS_OUTPUT.PUT_LINE('Terminateed – fare less than 200');
<<successful>>
DBMS_OUTPUT.PUT_LINE('PROCESSING OVER');
END;
/
```

fare

First	Route_ code	…….	…….
300	Ms_as		
200	Ms_ds		

New_table

Mroute_ code_t	T_old	T_new
Ms_as	300	240

NULL Statement

- The NULL statement is used to explicitly specify in action. i.e it does nothing other than pass control to the next statement .
- جملة Null لا تقوم بعمل مهمة معينة غير انها تستخدم لتمرير تحكم تنفيذ الأومر ليتم تنفيذ الأمر التالي من أوامر ال Block
- For example :

```
IF newfare>90 THEN
Statements;
ELSE
NULL;
END IF ;
```

THE END