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 الرابعة

ادارة المعاملات

Transactions Management

المعاملة Transaction

- عبارة عن عمل أو سلسلة أعمال تنفذ من قبل مستفيد واحد أو برنامج تطبيقي، والذي يقرأ أو يحدث محتويات قاعدة البيانات (وحدة منطقية من العمل على قاعدة البيانات).
- قد تكون برنامج كامل أو جزء من برنامج، أو أمر مفرد (مثل الأمر Select في SQL أو أمر Update)، وقد يتضمن أي عدد من العمليات على قاعدة البيانات.
- مثال: تحديث الراتب لموظف محدد ومعطى رقمه الوظيفي x

Read(staff No=x, salary)

Salary = salary*1.1

Write(staff No = x, salary)

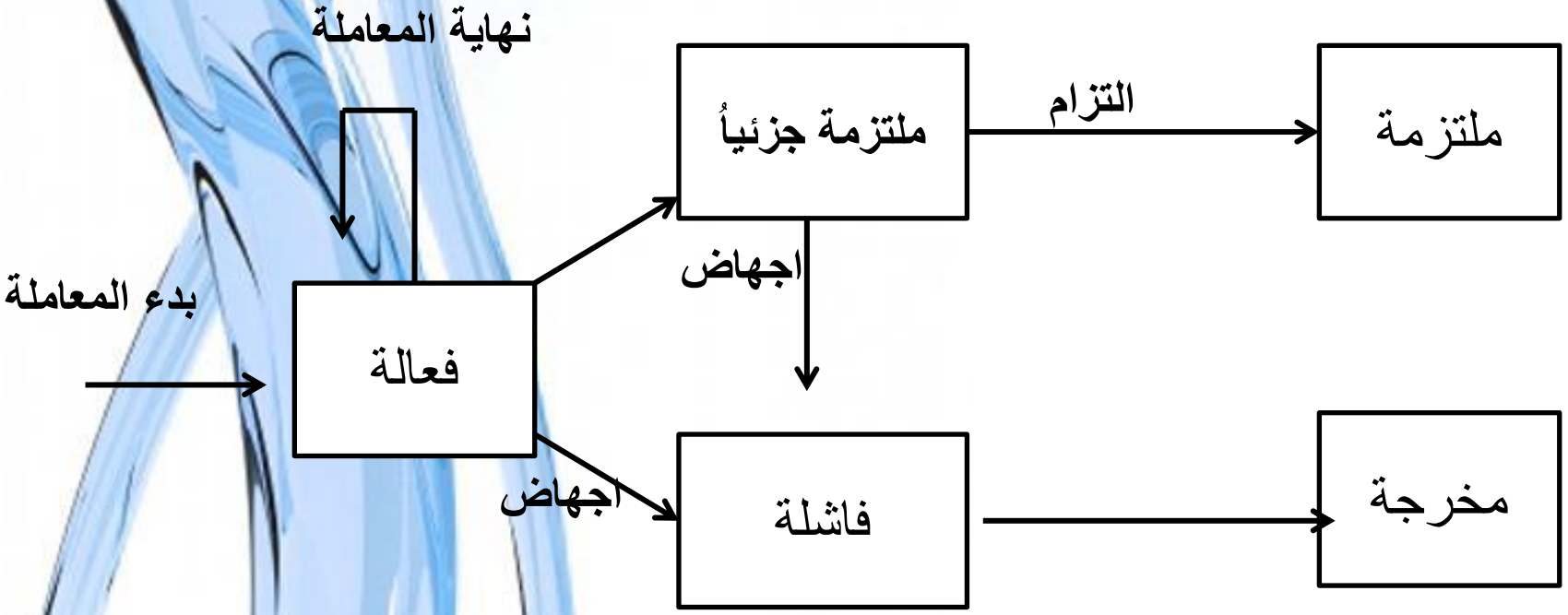
- يمكن أن يكون للمعاملة نتيجة واحدة من اثنين:

- اذا أكملت المعاملة بنجاح، يقال للمعاملة ملتزمة (Committed) وتوصل قاعدة البيانات الى حالة متناغمة جديدة.

- اذا لم تنفذ المعاملة بنجاح، فان المعاملة تخرج من النظام (Aborted)

- اذا اخرجت المعاملة فيجب اعادة تخزين قاعدة البيانات الى الحالة المتناغمة كما كانت قبل أن تبدأ المعاملة.مثل هذه المعاملة تعاد الى السابق Roll back أو تهمل العمل السابق undone.
- لا يمكن اخراج المعاملة الملتزمة،واذا كانت خطأ فيجب انجاز معاملة أخرى تعويضيه لعكس تأثيرها.

المعاملة المجهضة وتم اعادة العمل فيها يمكن اعادة تنفيذها بعد ذلك ، واعتماداً على سبب الفشل ، قد تنفذ بنجاح ويتم الالتزام بها ، الشكل يوضح ذلك :



نقل حالة معاملة

من الرسم السابق بالاضافة الي الحالات الطبيعية (active,committed,aborted) توجد حالتين :

- التزام جزئي partially Committed: يكون موجود بعد تنفيذ الجملة الأخيرة.قد يتم اكتشاف أن المعاملة قد انتهكت التسلسل العملي Serializability أو أنها انتهكت شرط سلامة البيانات ويجب اجهاضها. أو لم يتم تسجيلها بأمان في المخازن الثانوية وفي هذه الحالة تكون في حالة فشل ويجب اجهاضها.اما اذا كانت المعاملة ناجحة فان أي تحديث يمكن تسجيله بأمان وتكون المعاملة في حالة التزام committed
- فاشلة Failed :تكون موجودة في حالة عدم استطاعة المعاملة الالتزام أو أن المعاملة قد تم اجهاضها بينما هي في حالة العمل Active، ربما بسبب أن المستفيد أجهض المعاملة أو كنتيجة لسياق السيطرة التزامنية التي تجهض المعاملة لضمان التسلسل العملي Serializability.

خصائص المعاملات Transaction Properties

تمتلك المعاملات أربعة خصائص تسمى ACID هي:

- **الذرية Atomicity**: خاصية " الجميع أو لا شئ". المعاملة وحدة لا يمكن تجزئتها بحيث أنها اما تنفذ بصورة كاملة أو لا يتم تنفيذها.
- **التناغم Consistency**: تحفظ المعاملات تناغم قاعدة البيانات، المعاملة تنقل حالة التناغم لقاعدة البيانات الى حالة تناغم أخرى ، بدون ضرورة المحافظة على التناغم في جميع النقاط الوسيطة.

تابع:خصائص المعاملات Transaction Properties

- **العزل Isolation**: تعزل المعاملات الواحدة عن الأخرى، بالرغم من أن هنالك العديد من المعاملات تنفذ بصورة متزامنة فان أي معاملة تحديث موجودة تعزل عن البقية الى أن يتم الزام هذه المعاملة.
- **القدرة على المتانة Durability**: حالما يتم الزام المعاملة، فان تحديثاتها تبقى في قاعدة البيانات، حتى اذا كان الناتج تحطم النظام.

نظام قاعدة البيانات ذو المستخدم –المفرد:

- يضمن بصورة اوتوماتيكية التسلسل العملي Serializability وعزل Isolation لقاعدة البيانات، لأن معاملة واحدة تنفذ في وقت واحد. كما يجب ان يضمن خاصية الذرية والمتانة .

قواعد البيانات المتعدد المستخدمين:

- مثالياً مرتبطة مع معاملات التزامن المتعددة لذلك في هذه الحالة يجب تطبيق بعض القيود لضمان التسلسل العملي والعزل للمعاملات بالاضافة الى الذرية والمتانة من أجل حفظ سلامة وتناغم قاعدة البيانات ، وفي هذه الحالة يجب أن تدار المعاملات من خلال استخدام تقنيات سيطرة التزامن لتجنب مثل هذه الحالات غير المرغوبة .

Internal Management الإدارة الداخلية

- أفرضي ان لدينا حسابين مصرفيين في قاعدة بيانات ونرغب في تحويل نقود من حساب الى حساب آخر، يتطلب هذا عمليتين منفصليتين في قاعدة البيانات المسندة على SQL :

```
UPDATE ACCOUNTS
  SET BALANCE=BALANCE-100
    WHERE ACCNO=1234;
UPDATE ACCOUNTS
SET BALANCE=BALANCE+100
        WHERE ACCNO=4567;
```

- قد تتحطم قاعدة البيانات بين تنفيذ الأمرين مما يترك البيانات في حالة غير متناغمة.

- تتطلب سلامة المعاملة أن تكون تأثيرات المعاملات اما كاملة أو لا تجرى أبداً.
- يوجد سياق **Roll back** و**Commit** من أجل سلامة المعاملة:

❖ **الالتزام Commit** هو عندما تحدث التغيرات على قاعدة البيانات بصورة دائمة.عندما تتحطم قاعدة البيانات فان التغيرات التي لم يتم الزامها (Committed) سوف تفقد.

- مع المعاملة السابقة يمكننا أن نضع أمر commit واضح عندما نضع أوامر التحديث هذه كما يلي:

```
UPDATE ACCOUNTS
  SET BALANCE=BALANCE-100
    WHERE ACCNO=1234;
UPDATE ACCOUNTS
SET BALANCE=BALANCE+100
       WHERE ACCNO=4567;
COMMIT;
```

- يعني هذا أن التحديث الأول عابر ولا يكون دائم الى أن ينفذ التحديث الثاني بمعنى اذا تحطمت قاعدة البيانات بين الأمرين فان تأثيرات التحديث الأول تكون مفقودة.

❖ **الرجوع ثانية Rollback :** هي آلية لإرجاع تأثيرات المعاملة عندما يوضع Rollback فان جميع تغيرات قاعدة البيانات منذ آخر commit ترجع كالسابق ولتوضيح تأثيراتها خذي تسلسل الأوامر التالية:

1- SELECT NAME FROM CUSTOMERS WHERE
REFNO=1;(RETURNES' P Abul')

2-UPDATE CUSTOMERS
Set NAME='J Jones'
where ref no=1;

3-COMMIT;

4-SELECT NAME FROM CUSTOERS WHERE
REFNO=1;
(RETURNS 'J JONES')

5-ROLLBACK;

6-SELECT NAME FROM CUSTOMERS WHERE
REFNO=1;
(RETURNS ' J Jones)

- يجعل الأمر commit في 3 التغيير دائم وأي تنفيذ للأمر rollback سوف يرجع التغييرات فقط التي عملت منذ آخر commit سابق.

- يمكن تغيير المثال السابق من خلال استخدام save point.
- عندما يستخدم الأمر roll back قد يكون اشارة الى save point .
- اذا لم يوضع أي commit منذ آخر save point فان قاعدة البيانات ترجع ثانية الى الوضع الذي يكون فيه save point بدلاً من آخر commit

1- SELECT NAME FROM CUSTOMERS WHERE REFNO=1;
(RETURNES' P Abul')

2-UPDATE CUSTOMERS

Set NAME='j Jones'

WHERE REFNO=1;

3-COMMIT;

4-SELECT NAME FROM CUSTOERS WHERE

REFNO=1;(RETURNS 'j jONES')

5-UPDATE CUSTOMERS

SET NAME='P smith'

WHERE REFNO=1;

6-SAVEPOINT 1;

7-UPDATE CUSTOMERS

SET NAME =' A SHARIF'

WHERE REFNO =1'

8-ROLLBACK TO SAVEPOINT 1;

9-SELECT NAME FROM CUSTOMERS WHERE

REFNO=1;(RETURNS ' P smith')

- يسترجع الامر الأخير البيانات التي تعكس حالة قاعدة البيانات في save point1 ، بسبب أننا نرجع الي هذه النقطة ، مرجعين تأثيرات الامر update في الامر 7 فقط.
- سوف يكون للرجوع rollback غير المشروط تأثير التراجع الى آخر أمر commit .

The Transaction Log سجل المعاملات

- يستخدم نظام قاعدة البيانات سجل المعاملة لمتابعة جميع المعاملات التي تحدث لقاعدة البيانات.
- تُستخدم المعلومات المخزونة في هذا السجل من قبل نظام قاعدة البيانات من أجل متطلبات الاسترداد اما عند تنفيذ الأمر roll back أو فشل النظام(تناقض الشبكة، تحطم القرص) .
- تستخدم بعض نظم قاعدة البيانات سجل المعاملة لاسترداد تقدم قاعدة البيانات لحالة تناغم بعد فشل الSERVER ، وفي هذه الحالة ترجع ORACLE بصورة اوتوماتيكية المعاملات غير الملتزمة وتقدم المعاملات الملتزمة الى الامام.

تابع:سجل المعاملاتThe Transaction Log

يخزن سجل المعاملات ما يلي:

- قيد لبداية المعاملة.
- لكل معاملة (عبارة SQL)فانه يخزن:

- نوع العملية التي تم انجازها(تحديث،حذف، ادخال)
- اسماء المواضيع المتأثرة ومن قبل المعاملة (اسم الجدول)
- القيم «السابقة واللاحقة» للحقول التي تم تحديثها.
- مؤشرات الى مدخلات سجل المعاملة السابقة واللاحقة لنفس المعاملة.

- نهاية (التزام COMMIT) المعاملة

❖ بالرغم من أن استخدام سجل المعاملة سيزيد من جهد المعالجة في DBMS فان القدرة على اعادة خزن قاعدة البيانات المدمرة تعادل الثمن المبذول.

TRL ID	TRX-NUM PTR	PREV PTR	NEXT PTR	OPERATION	TABLE	ROW ID	ATTRIBUTE	BEFORE VALUE	AFTER VALUE
341	101	NULL	352	START	Xx start transac Tion				
352	101	341	363	UPDATE	PROD DUCT	15558 QWI	PROD QOH	25	23
363	101	352	365	UPDATE	CUSTO MER	10011	CUST-BALA NCE	525-75	615-73
365	101	363	NULL	COMMIT	Xx End Of Transac tIon				

TRL-ID : رقم سجل المعاملة ويتم تخصيصه بصورة اتوماتيكية من قبل DBMS

PTR : مؤشر الى رقم سجل المعاملة.

TRX-NUM :رقم المعاملة

تابع:سجل المعاملات
The Transaction Log

- سجل المعاملات نفسه عبارة عن قاعدة بيانات، يتم ادارته من قبل DBMS مثل أي قاعدة بيانات أخرى.
- يرتبط سجل المعاملات مع أخطار قاعدة البيانات الاعتيادية مثل شروط امتلاء القرص وتدمير القرص.
- لان سجل المعاملة يحتوى على بعض البيانات الهامة فان وضع نسخ منه على اقراص مختلفة يعمل على تقليص خطر فشل النظام.

استرداد النظام System Recovery

- يجب تهيئة النظام من أجل استرداده، ليس فقط من أعطال موقعية مثل وجود over flow ضمن معاملة مفردة لكن ايضاً من اعطال شاملة مثل انقطاع التيار الكهربائي .
- العطل الموقعي : يؤثر فقط على المعاملة التي يحصل فيها الفشل (مثل فقدان التحديث ، انقطاع التيار الكهربائي اثناء تنفيذ المعاملة) .
- العطل (الفشل) الشامل : يؤثر على جميع المعاملات أثناء التنفيذ في وقت الفشل ، وهذه لها تأثيرات كبيرة و مهمة على النظام .

تطبيق PL/SQL

Conditional control statements

Conditional statements

- Conditional statements in PL/SQL have 3 types:

❑if-then

❑if-then-else

❑if-then-elsif

Conditional statements

Conditional Statements	syntax

if-then	if <condition> then
	<statement-list>
	end if;

if-then-else	if <condition> then
	<statement-list-1>
	else
	<statement-list-2>
	end if;

if-then-elsif	if <condition-1> then
	<statement-list-1>
	elsif <condition-2>
	<statement-list-2>
	.....
	elsif <condition-N>
	<statement-list-N>
	else
	<statement-list-N+1>
	end if;

PL/SQL Decision Control Structures

Use IF/ELSIF to evaluate many conditions:

– IF condition1 THEN

commands that execute if condition1 is TRUE;

ELSIF condition2 THEN

commands that execute if condition2 is TRUE;

ELSIF condition3 THEN

commands that execute if condition3 is TRUE;

...

ELSE

commands that execute if none of the conditions are TRUE;

END IF;

Conditional logic –IF statement

Examples

```
IF hourly_wage > 10  THEN
      hourly_wage := hourly_wage * 1.5;
ELSE
      hourly_wage := hourly_wage * 1.1;
END IF;


IF salary BETWEEN 1000AND 4000
     THEN
      bonus := 1500;
ELSIF salary > 4000 AND salary <=
     10000 THEN bonus := 1000;
ELSE bonus := 0;
END IF;
```

Comments

- You can put parenthesis around Boolean expression after the IF and ELSIF .
- You don't need to put {, } or BEGIN, END to surround several statements between IF and ELSIF/ELSE, or between ELSIF/ELSE and END IF;

Example

1)

```
if (cnum > 1000) and (cnum < 9000) then
dbms_output.put_line('Customer no ' || cnum);
end if;
```

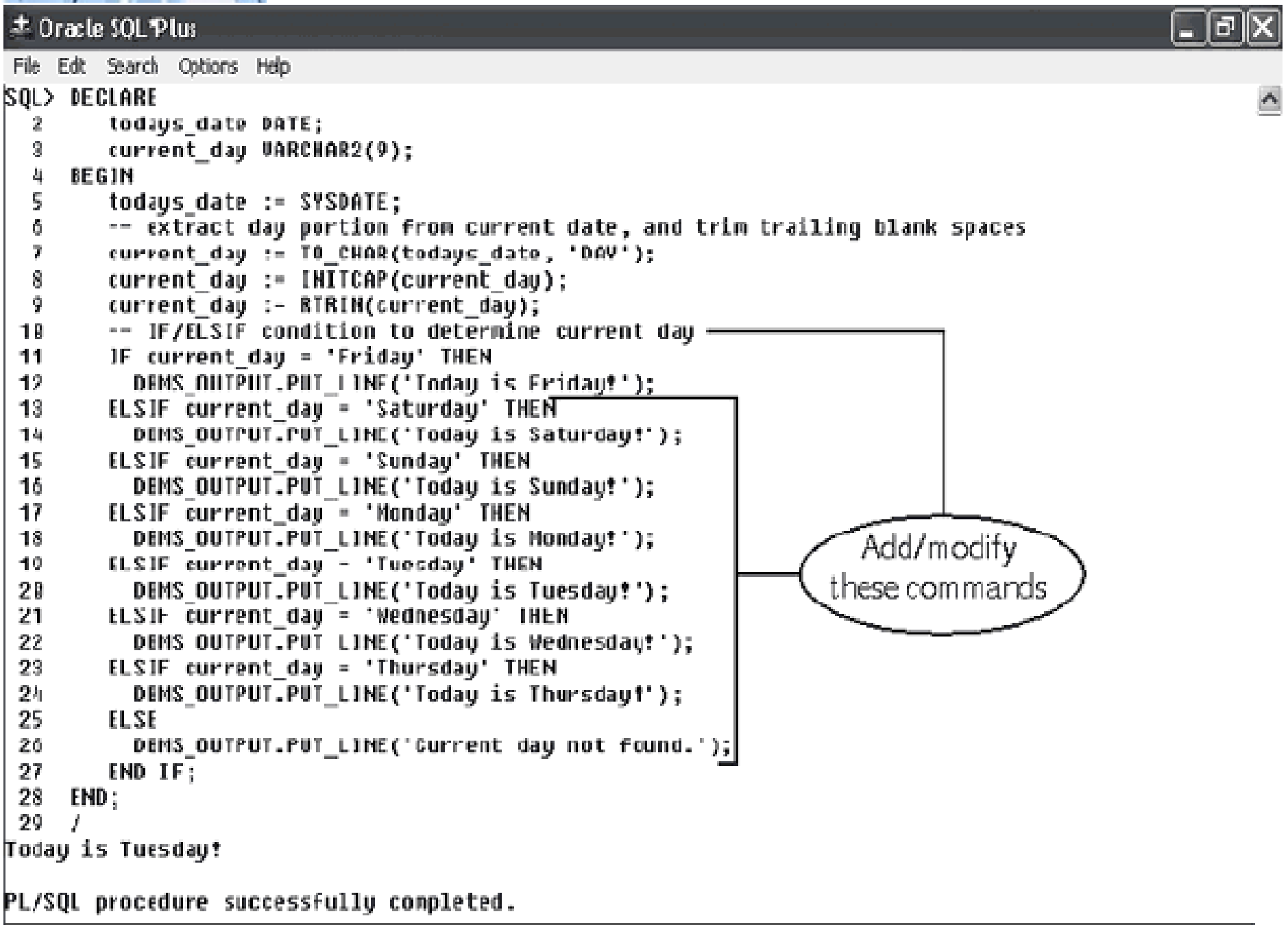
2)

```
if (cnum > 1000) and (cnum < 9000) then
i := i+1;
dbms_output.put_line(' Valid Customer ' || cnum);
else
j := j+1;
dbms_output.put_line('Invalid Customer ' || cnum);
end if;
```

Example (cont.)

3)

```
if (score > 90) then
na := na+1;
elsif (score > 80) then
nb := nb+1;
elsif (score > 70) then
nc := nc+1;
elsif (score > 60) then
nd := nd+1;
else
nf := nf+1;
end if;
```


Oracle SQL*Plus
File Edit Search Options Help
SQL> DECLARE
2 todays_date DATE;
3 current_day VARCHAR2(9);
4 BEGIN
5 todays_date := SYSDATE;
6 -- extract day portion from current date, and trim trailing blank spaces
7 current_day := TO_CHAR(todays_date, 'DAY');
8 current_day := INITCAP(current_day);
9 current_day := RTRIM(current_day);
10 -- IF/ELSIF condition to determine current day
11 IF current_day = 'Friday' THEN
12 DBMS_OUTPUT.PUT_LINE('Today is Friday!');
13 ELSIF current_day = 'Saturday' THEN
14 DBMS_OUTPUT.PUT_LINE('Today is Saturday!');
15 ELSIF current_day = 'Sunday' THEN
16 DBMS_OUTPUT.PUT_LINE('Today is Sunday!');
17 ELSIF current_day = 'Monday' THEN
18 DBMS_OUTPUT.PUT_LINE('Today is Monday!');
19 ELSIF current_day = 'Tuesday' THEN
20 DBMS_OUTPUT.PUT_LINE('Today is Tuesday!');
21 ELSIF current_day = 'Wednesday' THEN
22 DBMS_OUTPUT.PUT_LINE('Today is Wednesday!');
23 ELSIF current_day = 'Thursday' THEN
24 DBMS_OUTPUT.PUT_LINE('Today is Thursday!');
25 ELSE
26 DBMS_OUTPUT.PUT_LINE('Current day not found.');
27 END IF;
28 END;
29 /
Today is Tuesday!
PL/SQL procedure successfully completed.
Add/modify these commands

Complex Conditions

- Created with logical operators AND, OR and NOT
- AND is evaluated before OR
- Use () to set precedence

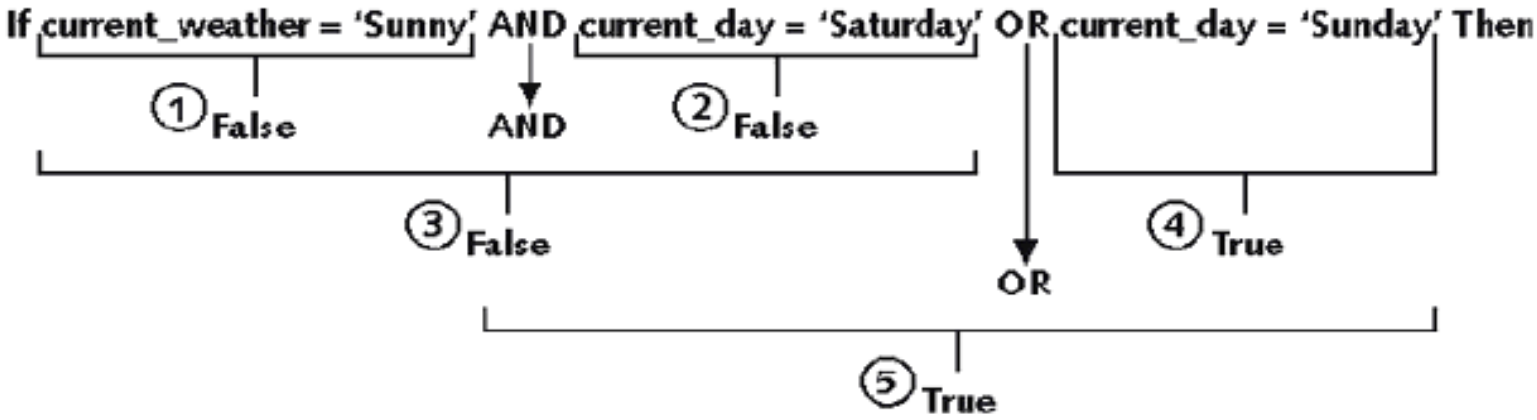


Figure 4-19 Evaluating AND and OR in an expression

Conditional logic

Condition:

```
If <cond>
   then <command>
elsif <cond2>
   then <command2>
else
      <command3>
end if;
```

Nested conditions:

```
If <cond>
   then
      if <cond2>
         then
            <command1>
      end if;
else <command2>
end if;
```

٣١

IF-THEN-ELSIF Statements

```
. . .
IF rating > 7 THEN
   v_message := 'You are great';
ELSIF rating >= 5 THEN
   v_message := 'Not bad';
ELSE
   v_message := 'Pretty bad';
END IF;
. . .
```

Suppose we have the following table:

```
create table mylog(
      who varchar2(30),
      logon_num number
);
```

mylog

who	logon_num
Hala	3
Amal	4
Mona	2

- Want to keep track of how many times someone logged on to the DB
- When running, if user is already in table, increment logon_num. Otherwise, insert user into table

Solution

```
DECLARE
  cnt  NUMBER;
BEGIN
  select count(*)
   into cnt
   from mylog
  where who = 'user';

  if cnt > 0 then
    update mylog
      set logon_num = logon_num + 1
    where who = 'user';
  else
    insert into mylog values('user',
1);
  end if;
  commit;
end;
/
```

Conditional logic –Simple CASE statement

CASE *selector*

WHEN *expression_1* THEN *statements*

[WHEN *expression_2* THEN *statements*]

[ELSE *statements*]

END CASE;

- ***selector* can be an expression of any datatype, and it provides the value we are comparing.**
- ***Expression_n* is the expression to test for equality with the selector.**

- If no WHEN matches the selector value, then the ELSE clause is executed.
- If there is no ELSE clause PL/SQL will implicitly supply:

ELSE RAISE CASE_NOT_FOUND;

which will terminate the program with an error (if the program ends up in the ELSE clause).

```
CASE grade
   WHEN 'A' THEN
    dbms_output.put_line('Excellent');
   WHEN 'B' THEN
    dbms_output.put_line('Very Good');
   WHEN 'C' THEN
    dbms_output.put_line('Good');
   WHEN 'D' THEN       dbms_output.put_line('Fair');
   WHEN 'F' THEN       dbms_output.put_line('Poor');
   ELSE dbms_output.put_line('No such grade');
END CASE;
```

THE END