

Database 3

القواعد

Triggers

القواعد

- القواعد هي وحدات منطقية مقادة بالأحداث مزروعة في قاعدة البيانات.
- يكون القادح مرفقاً بجدول واحد في قاعدة البيانات و يتم تفعيله عند إعطاء أوامر إدراج أو تعديل أو حذف على سطور هذا الجدول، و نقول أن هذه الأحداث تطلق القادح.
- عملية تنفيذ القواعد شفافة بالنسبة للمستخدم (أي لا يرى عملية التنفيذ بل نتيجتها).
- تنفذ القواعد من قبل قاعدة البيانات أوتوماتيكياً عندما تنفذ أوامر معينة خاصة بالتعامل مع البيانات على جداول محددة.

Database

Applications

UPDATE t SET ...;

INSERT INTO t ...;

DELETE FROM t ...;

Table t

Update Trigger

BEGIN
... .

Insert Trigger

BEGIN
... .

Delete Trigger

BEGIN
... .

الغاية من القواعد

- التحقق من صلاحية البيانات المدخلة و صيانة قيود الكمال المعقدة:
- بعض قيود الكمال قد لا تكون مصرحة باستخدام آليات الكمال الأساسية لنموذج بيانات معطى، و يمكن أن تستخدم القواعد لبرمجة مثل هذه القيود.
- الاستخدام كمنبهات
- تحديد الأفعال التي قد تم إنجازها على قاعدة البيانات لمستخدمين محددين.
- صيانة معلومات التدقيق لمدير قاعدة البيانات
- دعم أفعال التوزيع
- مثل عمل نسخة مطابقة (replication) من البيانات أوتوماتيكياً عبر شبكة.

أجزاء القادح

- الحدث القادح أو العبارة القادحة
 - عملية INSERT أو DELETE أو UPDATE على جدول معين
 - عملية CREATE أو DROP أو ALTER على أي غرض في قاعدة البيانات
- شرط القدح
 - تعبير منطقي يجب أن يكون true حتى يتم إطلاق القادح
 - الفعل الذي سيتم تنفيذه
- هو إجرائية PL/SQL أو برنامج خارجي تحوي تعليمات SQL التي ستنفذ عندما:
 - يقع الحدث القادح
 - و عندما يكون شرط القدح true

بنية القادح

CREATE [OR REPLACE] TRIGGER <trigger_name>	اسم القادح
BEFORE AFTER	توقيت القادح
INSERT OR UPDATE [OF <column(s)>] OR DELETE ON <table>	الحدث القادح
[FOR EACH ROW]	نوع القادح
WHEN (<condition>)	شرط القادح
<PL/SQL block>	جسم القادح

أنواع القوادر

حسب مستوى التنفيذ

• على مستوى السطر Row-level

• على مستوى العبارة Statement-level

حسب الحدث القادر

• INSERT

• UPDATE

• DELETE

حسب توقيت التنفيذ

• قبل الحدث القادر BEFORE

• بعد الحدث القادر AFTER

Event	Trigger time point		Trigger Type	
	BEFORE	AFTER	Statement	Row
INSERT	✓	✓	✓	✓
UPDATE	✓	✓	✓	✓
DELETE	✓	✓	✓	✓

القوادح على مستوى السطر و على مستوى العبارة

- القوادح على مستوى السطر
 - تعرف باستخدام العبارة **FOR EACH ROW**
 - ينفذ القادح على مستوى السطر مرة واحدة من أجل كل سطر قبل/بعد الحدث
- القوادح على مستوى العبارة
 - تنفذ مرة واحدة قبل/بعد الحدث بشكل مستقل عن عدد الأسطر المتأثرة بالحدث
 - على سبيل المثال، قادح سطري من النوع **AFTER UPDATE** ينفذ من أجل كل سطر يتأثر بعملية التعديل
 - إذا كان التعديل يؤثر على ٢٠ سطر فالقادح سينفذ ٢٠ مرة، من أجل كل سطر مرة
 - بالمقابل لو كان القادح على مستوى العبارة فسينفذ مرة واحدة

القوادح على مستوى السطر و على مستوى العبارة

- فقط مع القوادح السطرية يمكن الوصول إلى قيم الواصفات لسطر قبل و بعد التغيير (لأن القادح ينفذ مرة من أجل كل سطر)
- من أجل قادح تعديل UPDATE
 - يمكن الوصول إلى القيمة القديمة للواصفة باستخدام :old.<column>
 - يمكن الوصول إلى القيمة الجديدة للواصفة باستخدام :new.<column>
- من أجل قادح إدراج INSERT
 - يمكن فقط استخدام :new.<column> القيمة الجديدة المدرجة
 - لأنه ليس هناك قيمة قديمة
- من أجل قادح حذف DELETE
 - يمكن فقط استخدام :old.<column> القيمة القديمة المحذوفة
 - لأنه ليس هناك قيمة جديدة

القوادح على مستوى السطر و على مستوى العبارة

- إذاً في القوادح السطرية يمكن عمل مقارنة بين قيم القديمة و الجديدة للواصفات في كتلة PL/SQL

```
IF :OLD.SAL<:NEW.SAL THEN ...
```

- في القادح السطري إذا كان التوقيت هو BEFORE من الممكن أيضاً تعديل القيم الجديدة للسطر

```
:NEW.SAL := :NEW.SAL * 1.05  
:NEW.SAL := :OLD.SAL
```

- مثل هذه التعديلات غير ممكنة مع القوادح السطرية ذات التوقيت AFTER
- بشكل عام ، ينصح باستخدام القوادح السطرية ذات التوقيت AFTER إذا كان السطر الجديد غير معدل في كتلة PL/SQL
- القوادح على مستوى العبارة تستخدم عادة مع القوادح ذات التوقيت AFTER فقط

تجميع أنواع القوادح

```
CREATE OR REPLACE TRIGGER emp_check
AFTER INSERT OR DELETE OR UPDATE ON EMP
FOR EACH ROW
BEGIN
    IF INSERTING THEN
        <PL/SQL block>
    END IF ;
    IF UPDATING THEN
        <PL/SQL block>
    END IF ;
    IF DELETING THEN
        <PL/SQL block>
    END IF ;
END;
```

مثال

AFTER UPDATE OF parts on hand ON inventory

WHEN (new.parts on hand < new.reorder point)

العبرة القادحة
شرط القدح
الفعل

FOR EACH ROW

```
DECLARE          /* a dummy variable for counting */
```

X NUMBER;

BEGIN

```
SELECT COUNT(*) INTO X /* query to find out if part has already been */
```

```
FROM pending orders      /* reordered-if yes, x=1, if no, x=0 */
```

WHERE part no = :new.part no;

```
IF x = 0 THEN      /* part has not been reordered yet, so reorder */
```

INSERT INTO pending orders

```
VALUES (new.part no, new.reorder quantity, sysdate);
```

[illegible]

END ;

RADS
RADIUS

AREAS	
RADIUS	AREA

```
CREATE TABLE RADS (
  RADIUS INTEGER
);
```

```
CREATE TABLE AREAS (
  RADIUS INTEGER,
  AREA NUMBER(14,2)
);
```

```
CREATE FUNCTION AREA(RAD IN INTEGER)
RETURN NUMBER IS
  PI CONSTANT NUMBER(7,2) := 3.14159;
BEGIN
  RETURN PI * POWER(RAD, 2);
END;
/
```

أمثلة

```
CREATE OR REPLACE TRIGGER INS_RADS
AFTER INSERT ON RADS
FOR EACH ROW
BEGIN
    INSERT INTO AREAS VALUES
        (:NEW.RADIUS, AREA(:NEW.RADIUS));
END;
/
```

```
INSERT INTO RADS VALUES (1);
INSERT INTO RADS VALUES (2);
```

- يطلق هذا القادح عند إدراج سطر جديد في الجدول RADS
- و يقوم بحساب المساحة و يدرجها (مع نصف القطر) في الجدول AREAS

RADS
RADIUS
1
2

AREAS	
RADIUS	AREA
1	3.14
2	12.56

أمثلة

```
CREATE OR REPLACE TRIGGER DEL_RADS
AFTER DELETE ON RADS
FOR EACH ROW
BEGIN
    DELETE FROM AREAS
    WHERE RADIUS = :OLD.RADIUS;
END;
/
```

```
DELETE FROM RADS WHERE RADIUS=1;
```

- يطلق هذا القادح عند حذف سطر من الجدول RADS
- و يقوم بحذف السطر المقابل من الجدول AREAS

RADS
RADIUS
1

AREAS	
RADIUS	AREA
1	3.14

أمثلة

```
CREATE OR REPLACE TRIGGER UPD_RADS
AFTER UPDATE ON RADS
FOR EACH ROW
BEGIN
    UPDATE AREAS
    SET RADIUS = :NEW.RADIUS,
        AREA = AREA(:NEW.RADIUS)
    WHERE RADIUS = :OLD.RADIUS;
END;
/
```

```
UPDATE RADS
    SET RADIUS = 10
    WHERE RADIUS = 1;
```

- يطلق هذا القادح عند تعديل سطر من الجدول RADS
- و يقوم بتعديل السطر المقابل من الجدول AREAS

RADS
RADIUS
10

AREAS	
RADIUS	AREA
10	314

- بالرغم من أن القواعد مفيدة من أجل تخصيص قاعدة البيانات، إلا أنه يجب استعمالها فقط عند الضرورة
- الاستخدام "المفرط" للقواعد قد يؤدي إلى ارتباطات معقدة فيما بينها و سيصبح من الصعب الحفاظ على هذه الارتباطات في التطبيقات الكبيرة.

SQL Statement

```
UPDATE t1 SET ...;
```

Fires the
UPDATE_T1
Trigger

UPDATE_T1 Trigger

```
BEFORE UPDATE ON t1
FOR EACH ROW
BEGIN
  .
  .
  INSERT INTO t2 VALUES (...);
  .
  .
END;
```

Fires the
INSERT_T2
Trigger

INSERT_T2 Trigger

```
BEFORE INSERT ON t2
FOR EACH ROW
BEGIN
  .
  .
  INSERT INTO ... VALUES (...);
  .
  .
END;
```

etc.

تعديل القوادر

تمكين القوادر و إبطال مفعولها

```
ALTER TRIGGER <Trigger_Name> DISABLE;  
ALTER TRIGGER <Trigger_Name> ENABLE;  
ALTER TABLE <Trigger_Name> ENABLE ALL TRIGGERS;  
ALTER TABLE <Trigger_Name> DISABLE ALL TRIGGERS;
```

حذف القوادر

```
DROP TRIGGER <Trigger_Name>;
```