Triggers stored Procedurals

Triggers

المشغلات هي كائنات قاعدة بيانات مسماة يتم تشغيلها ضمنيًا عند حدوث حدث تشغيل. يمكن تشغيل إجراء التشغيل قبل أو بعد حدث التشغيل. تشبه المشغلات الإجراءات المخزنة ولكنها تختلف في طريقة استدعاءها.

يتم تضمين دعم المشغلات في MySQL فقط بدءًا من الإصدار 5.0.2. لا يمكن ربط المشغل إلا بجدول وتحديده للتشغيل عند تنفيذ عبارة INSERT أو DELETE أو UPDATE على الجدول. لا يسمح MySQL بمشغلين لهما نفس توقيت التشغيل (قبل أو بعد) وحدث أو بيان تشغيل (INSERT أو DELETE أو UPDATE) يتم تعريفهما في جدول. على سبيل المثال، لا يمكنك تحديد مشغلين قبل الإدراج أو مشغلين بعد التحديث للجدول. جميع المشغلات المحددة في MySQL هي مشغلات صفوف، مما يعني أن الإجراء المحدد للمشغلات يتم تنفيذه لكل صف يتأثر ببيان التشغيل.

تضمن معالجة الأخطاء أثناء تنفيذ المشغل لجداول المعاملات إكمال كل من بيان التشغيل وإجراء المشغل بنجاح أو عدم تنفيذ بيان المشغل أو إجراء المشغل، أي أن جميع التغييرات التي تم إجراؤها هي تراجع عند الفشل. بالنسبة للجداول غير المتعلقة بالمعاملات، تظل جميع التغييرات التي تم إجراؤها قبل نقطة الخطأ سارية المفعول.

ما يلي هو بناء الجملة لإنشاء مشغل في MySQL:

CREATE TRIGGER <trigger name>

{ BEFORE | AFTER }

{ INSERT | UPDATE | DELETE }

ON <table name>

FOR EACH ROW

<triggered action>

في Oracle، يمكن تشغيل المشغلات عند حدوث إحدى العمليات التالية:

عبارات DML (INSERT أو DELETE أو UPDATE) التي تقوم بتعديل البيانات في جدول أو طريقة عرض

تصريحات DDL

أحداث المستخدم مثل تسجيل الدخول وتسجيل الخروج

أحداث النظام مثل بدء التشغيل وإيقاف التشغيل ورسائل الخطأ

تسمح Oracle بتحديد مشغلات متعددة بنفس توقيت التشغيل وحدث التشغيل في الجدول؛ ومع ذلك، لا يمكن ضمان تنفيذ هذه المشغلات بأي ترتيب محدد. يمكن تعريف المشغلات على أنها مشغلات الصف أو مشغلات البيانات. يتم تشغيل مشغلات العبارة مرة واحدة لكل عبارة تشغيل بغض النظر عن عدد الصفوف في الجدول المتأثر بعبارة التشغيل. على سبيل المثال، إذا قامت عبارة DELETE بحذف عدة صفوف من جدول، فسيتم تشغيل مشغل العبارة مرة واحدة فقط.

نموذج التنفيذ لمشغلات Oracle هو المعاملات. يجب أن تنجح جميع الإجراءات التي يتم تنفيذها نتيجة لبيان التشغيل، بما في ذلك الإجراءات التي يتم تنفيذها بواسطة المشغلات التي تم تشغيلها؛ وإلا فسيتم التراجع.

## Stored Procedures

## توفر الإجراءات المخزنة طريقة قوية لترميز منطق التطبيق الذي يمكن تخزينه على الخادم. يستخدم كل من MySQL وOracle الإجراءات والوظائف المخزنة. تشبه الوظائف المخزنة الإجراءات، باستثناء أن الوظيفة ترجع قيمة إلى البيئة التي يتم استدعاؤها فيها. في MySQL، تسمى الإجراءات والوظائف المخزنة بشكل جماعي بالروتينات.

## تقارن الأقسام التالية الإجراءات المخزنة في MySQL وOracle:

## بيانات SQL الفردية

## المتغيرات في الإجراءات المخزنة

## معالجة الأخطاء في الإجراءات المخزنة

## عبارات SQL الفردية

## يصف هذا القسم الاعتبارات المتعلقة بالعبارات أو البنيات التالية:

## استبدال البيان

## قم بالبيان

## بيان إعلان المجمع

## بيان المجموعة المركبة

## بيان الاستبدال

## عبارة REPLACE في MySQL عبارة عن عبارة ثنائية الغرض. وهي تعمل مثل عبارة INSERT عندما لا يكون هناك سجل في الجدول له نفس قيمة السجل الجديد لمفتاح أساسي أو فهرس فريد، وبخلاف ذلك فإنها تعمل مثل عبارة UPDATE.

## لا تحتوي Oracle على أي عبارات SQL مضمنة تدعم أغراض عبارة MySQL REPLACE. لتحويل هذا البيان إلى Oracle، يجب إنشاء دالة تمت مضاهاتها باستخدام كل من عبارات INSERT وUPDATE. يتم أولاً إجراء محاولة لوضع البيانات في الجدول باستخدام عبارة INSERT؛ وإذا فشل ذلك، يتم تحديث البيانات الموجودة في الجدول باستخدام عبارة UPDATE.

## بيان القيام

## كما يوحي اسمها، فإن عبارة DO في MySQL تفعل شيئًا ما ولكنها لا تُرجع أي شيء؛ على وجه التحديد، يقوم بتنفيذ قائمة التعبيرات المحددة بفواصل كمعلمات خاصة به. يتم تحويل عبارة DO إلى عبارة SELECT expr1 [, expr2,...] INTO ... FROM DUAL في Oracle.

## بيان الإعلان المركب

## يستخدم MySQL عبارة DECLARE للإعلان عن المتغيرات المحلية في الإجراءات المخزنة. لا يسمح PL/SQL بإعلانات متعددة؛ يجب أن يتم كل إعلان على حدة. لتحويل عبارات DECLARE المركبة إلى تعليمات PL/SQL البرمجية المكافئة وظيفيًا، يجب تحويل كل عبارة إعلان MySQL المتعددة إلى عبارات منفصلة مكافئة منطقيًا، واحدة لكل إعلان.

## على سبيل المثال، خذ بعين الاعتبار إعلان MySQL البسيط وبيانات الإعلان المتعددة التالية:

/\* Simple declaration \*/

DECLARE a INT;

/\* Compound declaration \*/

DECLARE a, b INT DEFAULT 5;

The PL/SQL functionally equivalent statements are:

/\* Simple declaration \*/

a INT;

/\* Multiple declarations \*/

a INT := 5;

b INT := 5;

## في هذا المثال، يتم تحويل عبارات MySQL DECLARE الأصلية إلى ثلاث عبارات تعريف PL/SQL مكافئة منطقيًا، مع عبارة تعريف PL/SQL واحدة لكل تعريف مستخدم ضمن عبارات MySQL DECLARE.

## بيان المجموعة المركبة

## يستخدم MySQL عبارة SET لتعيين قيم للمتغيرات (متغيرات المستخدم أو متغيرات النظام). يسمح MySQL بالعبارات المركبة التي تقوم بتعيين قيم لمتغيرين أو أكثر داخل نفس العبارة. يسمح PL/SQL فقط بالمهام البسيطة التي تقوم بتعيين قيمة واحدة لمتغير واحد. لتحويل عبارات SET المركبة إلى كود PL/SQL مكافئ وظيفيًا، قم بتقسيم كل عبارة مهمة متعددة من MySQL إلى عبارات مهمة بسيطة مكافئة منطقيًا.

## المتغيرات في الإجراءات المخزنة

## يدعم MySQL ثلاثة أنواع من المتغيرات في الإجراءات المخزنة: المتغيرات المحلية، ومتغيرات المستخدم، ومتغيرات النظام.

## يتم الإعلان عن المتغيرات المحلية ضمن الإجراءات المخزنة وهي صالحة فقط ضمن كتلة BEGIN...END التي تم الإعلان عنها فيها. يجب الإعلان عن المتغيرات المحلية داخل كتلة BEGIN...END قبل أن يتم الرجوع إليها في عبارات أخرى في الكتلة، بما في ذلك أي كتل BEGIN...END متداخلة. إذا كان المتغير المحلي المعلن داخل كتلة BEGIN...END المتداخلة له نفس اسم المتغير المحلي المعلن في كتلة BEGIN...END المحيطة به، فإن المتغير المحلي في الكتلة المتداخلة يكون له الأسبقية حيثما تتم الإشارة إلى المتغير المحلي في BEGIN...END المتداخلة حاجز. يمكن أن تحتوي المتغيرات المحلية على أي نوع بيانات SQL. يوضح المثال التالي استخدام المتغيرات المحلية في إجراء مخزن.

CREATE PROCEDURE p1()

BEGIN

/\* declare local variables \*/

DECLARE x INT DEFAULT 0;

DECLARE y, z INT;

/\* using the local variables \*/

SET x := x + 100;

SET y := 2;

SET z := x + y;

BEGIN

/\* local variable in nested block \*/

DECLARE z INT;

SET z := 5;

/\* local variable z takes precedence over the one of the

same name declared in the enclosing block. \*/

SELECT x, y, z;

END;

SELECT x, y, z;

END;

mysql> call p1();

+-----+---+---+

| x | y | z |

+-----+---+---+

| 100 | 2 | 5 |

+-----+---+---+

1 row in set (0.00 sec)

+-----+---+-----+

| x | y | z |

+-----+---+-----+

| 100 | 2 | 102 |

+-----+---+-----+

1 row in set (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

## تعد متغيرات المستخدم خاصة بجلسة المستخدم ولا يمكن للمستخدمين الآخرين رؤيتها أو استخدامها. وهي صالحة فقط لمدة جلسة المستخدم ويتم تحريرها تلقائيًا عند انتهاء جلسة المستخدم. متغيرات المستخدم لها نطاق جلسة؛ وبالتالي، فإن كافة المراجع إلى متغير مستخدم يحمل نفس الاسم داخل الجلسة تشير إلى نفس المتغير. في إجراءات MySQL المخزنة، تتم الإشارة إلى متغيرات المستخدم بعلامة الضم (@) مسبوقة باسم متغير المستخدم (على سبيل المثال، @x و@y). يوضح المثال التالي استخدام متغيرات المستخدم في إجراءين مخزنين.

## معالجة الأخطاء في الإجراءات المخزنة

## يقوم كل من Oracle PL/SQL وMySQL بتنفيذ آلية معالجة الأخطاء لإجراءاتهم المخزنة. يتم التحقق من وجود أخطاء في كل عبارة SQL في الإجراء المخزن قبل معالجة العبارة التالية. في حالة حدوث خطأ، يتم تمرير التحكم على الفور إلى معالج الأخطاء. على سبيل المثال، إذا لم تعثر عبارة SELECT على أي صفوف في قاعدة البيانات، فسيظهر خطأ، ويتم تنفيذ التعليمات البرمجية للتعامل مع هذا الخطأ.

## في إجراءات MySQL المخزنة، يمكن تعريف المعالجات للتعامل مع الأخطاء أو التحذيرات التي تحدث من تنفيذ عبارة SQL ضمن إجراء مخزن. يسمح MySQL بنوعين من المعالجات: معالجات CONTINUE ومعالجات EXIT. يختلف نوعا المعالجات عن نقطة التنفيذ التالية في الإجراء المخزن بعد تشغيل المعالج. بالنسبة للمعالج CONTINUE، يستمر التنفيذ في العبارة التالية بعد العبارة التي أدت إلى ظهور الخطأ. بالنسبة لمعالج EXIT، يتم إنهاء تنفيذ العبارة المركبة الحالية، المحاطة بزوج من عبارات BEGIN وEND، ويستمر التنفيذ في العبارة التالية (إن وجدت) بعد العبارة المركبة.

## يتم تعريف المعالجات للتعامل مع شرط واحد أو أكثر. قد يكون الشرط عبارة عن قيمة SQLSTATE، أو رمز خطأ MySQL، أو شرط محدد مسبقًا. هناك ثلاثة شروط محددة مسبقًا: SQLWARNING (تحذير أو ملاحظة)، لم يتم العثور عليه (لا يوجد المزيد من الصفوف) وSQLEXCEPTION (خطأ). قد يتم تعريف الشرط بشكل منفصل باسم ثم يتم الرجوع إليه لاحقًا في بيان المعالج. يتم إنشاء كافة تعريفات المعالج في بداية كتلة البيان المركبة.

## في إجراءات Oracle PL/SQL المخزنة، تسمى حالة الخطأ استثناءً. قد يتم تعريف الاستثناءات داخليًا (بواسطة نظام وقت التشغيل) أو معرفة من قبل المستخدم. بعض الاستثناءات الداخلية لها اسم محدد مسبقًا، مثل ZERO\_DIVIDE أو NO\_DATA\_FOUND. يتم رفع الاستثناءات الداخلية ضمنيًا (تلقائيًا) بواسطة نظام وقت التشغيل. يجب إعطاء أسماء للاستثناءات المعرفة ويجب رفعها بشكل صريح بواسطة عبارات RAISE في الإجراءات المخزنة. تعالج معالجات الاستثناءات الاستثناءات التي يتم رفعها.

## يمكن الإعلان عن معالجات الاستثناء لكتلة PL/SQL. يتم تضمين معالجات الاستثناءات هذه بين عبارات BEGIN وEND، وتتعامل مع الاستثناءات التي قد تثار بواسطة العبارات الموجودة في كتلة PL/SQL، بما في ذلك الكتل الفرعية. كتلة PL/SQL تشبه كتلة بيان MySQL المركبة. يمكن الإعلان عن الاستثناءات فقط في الجزء التصريحي من كتلة PL/SQL، وهي محلية لتلك الكتلة وعالمية لجميع كتلها الفرعية. وبالتالي، لا تستطيع الكتلة المتضمنة التعامل مع الاستثناءات المرفوعة في كتلة فرعية إذا كانت استثناءات محلية في الكتلة الفرعية. لا يتم نشر الاستثناءات المرفوعة في كتلة فرعية إلى الكتلة المتضمنة إذا تم تحديد معالجات الاستثناء للكتلة الفرعية ومعالجتها إذا لم يتم رفعها مرة أخرى في معالجات الاستثناء. بعد تشغيل معالج الاستثناء، تتوقف الكتلة الحالية عن التنفيذ ويستأنف التنفيذ في العبارة التالية في الكتلة المرفقة.

CREATE PROCEDURE adjust\_emp\_salary ()

BEGIN

DECLARE job\_id INT;

DECLARE employee\_id INT DEFAULT 115;

DECLARE sal\_raise DECIMAL(3,2);

DECLARE EXIT HANDLER FOR 1339;

SELECT job\_id INTO jobid FROM employees WHERE employee\_id = empid;

CASE

WHEN jobid = 'PU\_CLERK' THEN

SET sal\_raise := .09;

WHEN jobid = 'SH\_CLERK' THEN

SET sal\_raise := .08;

WHEN jobid = 'ST\_CLERK' THEN

SET sal\_raise := .07;

END CASE;

END;