Transacciones y autoconsigna ("auto-commit") ¶

Una vez realizada una conexión a través de PDO, es necesario comprender cómo PDO gestiona las transacciones antes de comenzar a realizar consultas. Para aquellos que no han manejado anteriormente transacciones, estas ofrecen cuatro características principales: Atomicidad, Consistencia, Aislamiento y Durabilidad (ACID por sus siglas en inglés). En términos sencillos, se garantiza que cualquier trabajo llevado a cabo en una transacción, incluso si se hace por etapas, sea aplicado a la base de datos de forma segura, y sin interferencia de otras conexiones, cuando sea consignado. El trabajo transaccional puede también ser deshecho automáticamente bajo petición (siempre y cuando no se haya consignado ya), lo que hace más sencillo el manejo de errores en los scripts.
Una transacción se implementa normalmente para hacer que el lote de cambios se aplique a la vez. Esto tiene el buen efecto secundario de mejorar drásticamente la eficiencia de las actualizaciones. En otras palabras, las transacciones pueden hacer los scripts más rápidos y potencialmente más robustos (aún así es necesario utilizarlas correctamente para obtener tal beneficio).
Desafortunadamente, no todas las bases de datos admiten transacciones, por lo que PDO necesita ejecutarse en lo que es conocido como el modo "auto-commit" cuando se abra por primera vez la conexión. El modo auto-commit significa que toda consulta que se ejecute tiene su propia transacción implícita, si la base de datos la admite, o ninguna transacción si la base de datos no las admite. Para iniciar una transacción, si fuera necesario, se debe usar el método PDO::beginTransaction(). Si el controlador subyacente no admite transacciones, se lanzará una PDOException (independientemente de la configuración del manejo de errores: esto es siempre una condición de error serio). Una vez dentro de una transacción, se puede utilizar PDO::commit() o PDO::rollBack() para finalizarla, dependiendo del éxito del código que se ejecute durante la transacción.

Advertencia

PDO solamente comprueba la funcionalidad de transacciones a nivel del controlador. Si una cierta condición durante la ejecución implica que las transacciones no estén disponibles, PDO::beginTransaction() seguirá devolviendo **TRUE** sin ningún error si el servidor de bases de datos acepta la solicitud de iniciar una transacción.

Un ejemplo podría ser el intento de utilizar transacciones en tablas MyISAM en una base de datos MySQL.

Cuando el script finaliza o cuando una conexión está a punto de ser cerrada, si existe una transacción pendiente, PDO la revertirá automáticamente. Esto es una medida de seguridad que ayuda a evitar inconsistencia en los casos donde el script finaliza inesperadamente (si no se consignó la transacción, se asume que algo salió mal, con lo cual se realiza la reversión para la seguridad de los datos).

Advertencia

La reversión automática solamente ocurre si se inicia una transacción a través de PDO::beginTransaction(). Si se ejecuta manualmente una consulta que inicie una transacción,

PDO no tiene forma de saber nada sobre la misma y, por tanto, no puede revertirla si algo saliera mal.

Ejemplo #1 Ejecución de un lote en una transacción

En el siguiente ejemplo, se asume que se ha creado un conjunto de entradas para un nuevo empleado, al cual se le ha asignado el número de ID 23. Además de introducir los datos básicos de una persona, también es necesario registrar su sueldo. Es bastante simple realizar dos actualizaciones independientes, pero realizándolas entre las llamadas a PDO::beginTransaction() y PDO::commit() se garantiza que nadie más será capaz de ver los cambios hasta que se hayan completado. Si algo sale mal, el bloque catch revierte los cambios realizados desde que se creó la transacción y luego imprime un mensaje de error.

```
<?php
try {
  $mbd = new PDO('odbc:SAMPLE', 'db2inst1', 'ibmdb2',
     array(PDO::ATTR_PERSISTENT => true));
  echo "Conectado\n";
} catch (Exception $e) {
  die("No se pudo conectar: " . $e->getMessage());
}

try {
  $mbd->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

  $mbd->beginTransaction();
  $mbd->exec("insert into staff (id, first, last) values (23, 'Joe', 'Bloggs')");
  $mbd->exec("insert into salarychange (id, amount, changedate)
     values (23, 50000, NOW())");
  $mbd->commit();

} catch (Exception $e) {
  $mbd->rollBack();
  echo "Fallo: " . $e->getMessage();
}
?>
```

No exite límite al realizar actualizaciones en una transacción. También es posible ejecutar consultas complejas para extraer datos, con al posibilidad de utilizar esa información para construir más actualizaciones y consultas; mientras que la transacción esté activa, se garantiza que nadie más pueda realizar cambios mientras se esté en mitad del trabajo. Para más información sobre transacciones, consulte la documentación proporcionada por su servidor de base de datos.