Las transacciones ofrecen cuatro características principales:

* Atomicidad.
* Consistencia.
* Aislamiento.
* Durabilidad (ACID por sus siglas en inglés).

Las transacciones son implementadas típicamente 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 usarlas correctamente para obtener ese beneficio).

Por defecto PDO trabaja en modo "autocommit", esto es, confirma de forma automática cada sentencia que ejecuta el servidor. Para trabajar con transacciones, PDO incorpora tres métodos:

* **beginTransaction**. Deshabilita el modo "autocommit" y comienza una nueva transacción, que finalizará cuando ejecutemos uno de los dos métodos siguientes.
* **commit**. Confirma la transacción actual.
* **rollback**. Revierte los cambios llevados a cabo en la transacción actual.

Una vez ejecutado un commit o un rollback, se volverá al modo de confirmación automática.

$ok = true;

$dwes->beginTransaction();

if($dwes->exec('DELETE …') == 0) $ok = false;

if($dwes->exec('UPDATE …') == 0) $ok = false;

…

if ($ok)

$dwes->commit(); // Si todo fue bien confirma los cambios

else

$dwes->rollback(); // y si no, los revierte

Ejemplo #1 Revertir una transacción

El siguiente ejemplo inicia una transacción y ejecuta dos sentencias que modifican la base de datos antes de revertir los cambios. En MySQL, sin embargo, la sentencia DROP TABLE automáticamente consigna la transacción, con lo cual, ninguno de los cambios en la transacción son revertidos.

<?php  
/\* Iniciar una transacción, desactivando 'autocommit' \*/  
$gbd->beginTransaction();  
  
/\* Cambiar el esquema y datos de la base de datos \*/  
$gsent = $gbd->exec("DROP TABLE fruit");  
$gsent = $gbd->exec("UPDATE dessert SET name = 'hamburger'");  
  
/\* Reconocer un error y revertir los cambios \*/  
$gbd->rollBack();  
  
/\* La conexión a la base de datos ahora vuelve al modo 'autocommit' \*/  
?>

**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 hacer dos actualizaciones independientes, pero encerrándolas en las llamadas [PDO::beginTransaction()](http://php.net/manual/es/pdo.begintransaction.php) y [PDO::commit()](http://php.net/manual/es/pdo.commit.php), 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 {  
  $gbd = 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 {    
  $gbd->setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);  
  
  $gbd->beginTransaction();  
  $gbd->exec("insert into staff (id, first, last)

values (23, 'Joe', 'Bloggs')");

$gbd->exec("insert into salarychange (id, amount, changedate)

values (23, 50000, NOW())");  
  $gbd->commit();  
    
} catch (Exception $e) {  
  $gbd->rollBack();  
  echo "Fallo: " . $e->getMessage();  
}  
?>

Otro formato

<?php

$ok = true;

$dwes->beginTransaction();

if($dwes->exec('DELETE …') == 0)

$ok = false; // error

if($dwes->exec('UPDATE …') == 0)

$ok = false; // error

…

if ($ok)

$dwes->commit(); // Si todo fue bien confirma los cambios

else

$dwes->rollback(); // y si no, los revierte

?>