SERVICIOS DE BACKEND

Sistemas informáticos I

EJERCICIOS

Niveles de aislamiento

TRANSACCIÓN

Dada la siguiente transacción:

```
READ X
READ Y
WRITE Y
WRITE X
WRITE Y
READ Y
```

Identificar su grado de aislamiento si la secuencia de acciones realizadas sobre X e Y viene dada por los siguientes modelos equivalentes simples.

SECUENCIA DE ACCIONES 1

```
SLOCK X
READ X
UNLOCK X
SLOCK Y
READ Y
UNLOCK Y
XLOCK Y
WRITE Y
XLOCK X
WRITE X
WRITE Y
READ Y
UNLOCK X
UNLOCK Y
```

Grado 2

SECUENCIA DE ACCIONES 2

```
READ X
READ Y
XLOCK Y
WRITE Y
XLOCK X
WRITE X
WRITE Y
READ Y
UNLOCK X
UNLOCK Y
```

Grado 1

SECUENCIA DE ACCIONES 3

```
SLOCK X
READ X
SLOCK Y
READ Y
XLOCK Y
WRITE Y
XLOCK X
WRITE X
WRITE Y
READ Y
UNLOCK X
UNLOCK Y
```

Grado 3

EJERCICIOS

Transacciones en SQL

BASE DE DATOS INICIAL

```
SELECT * FROM film
```

Data Output | Explain | Messages | History

	film_id smallint	name character varying(45)
1	1	LOTR: The Fellowship of the Ring
2	2	Star Wars: Episode IV
3	3	Indiana Jones and the Last Crusade

```
SELECT * FROM actor
```

Data Output | Explain | Message

	actor_id smallint	name character varying(45)
1	1	Harrison Ford
2	2	Viggo Mortensen
3	3	Robert De Niro

```
SELECT * FROM film_actor
```

Data Output | Explain

	actor_id smallint	film_id smallint
1	1	2
2	1	3
3	2	1

EPS

EJECUCIÓN EN SQL

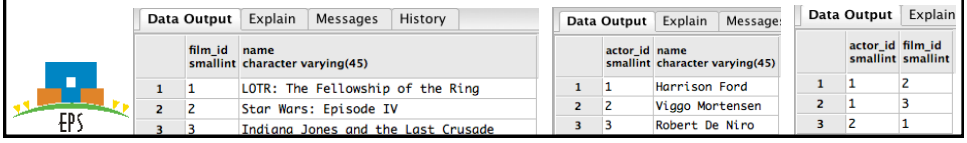
```
DELETE FROM actor WHERE actor_id=1
```

Data Output | Explain | **Messages** | History

```
ERROR:  update or delete on table "actor" violates foreign key constraint "fk_actor" on table "film_actor"
DETAIL:  Key (actor_id)=(1) is still referenced from table "film_actor".


********** Error **********

ERROR: update or delete on table "actor" violates foreign key constraint "fk_actor" on table "film_actor"
SQL state: 23503
Detail: Key (actor_id)=(1) is still referenced from table "film_actor".
```



Data Output | Explain | Messages | History

	film_id smallint	name character varying(45)
1	1	LOTR: The Fellowship of the Ring
2	2	Star Wars: Episode IV
3	3	Indiana Jones and the Last Crusade

Data Output | Explain | Messages

	actor_id smallint	name character varying(45)
1	1	Harrison Ford
2	2	Viggo Mortensen
3	3	Robert De Niro

Data Output | Explain

	actor_id smallint	film_id smallint
1	1	2
2	1	3
3	2	1

AHORA COMO TRANSACCIÓN

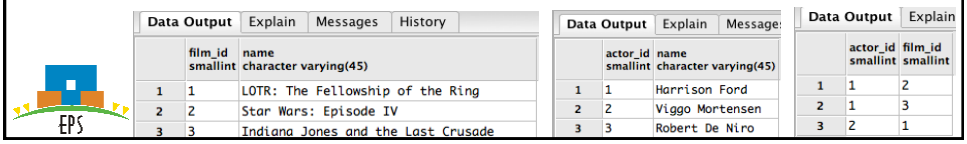
```
BEGIN;
DELETE FROM actor WHERE actor_id=3;
DELETE FROM actor WHERE actor_id=1;
COMMIT;
```

Data Output | Explain | **Messages** | History

```
ERROR:  update or delete on table "actor" violates foreign key constraint "fk_actor" on table "film_actor"
DETAIL:  Key (actor_id)=(1) is still referenced from table "film_actor".


********** Error **********

ERROR: update or delete on table "actor" violates foreign key constraint "fk_actor" on table "film_actor"
SQL state: 23503
Detail: Key (actor_id)=(1) is still referenced from table "film_actor".
```



Data Output | Explain | Messages | History

	film_id smallint	name character varying(45)
1	1	LOTR: The Fellowship of the Ring
2	2	Star Wars: Episode IV
3	3	Indiana Jones and the Last Crusade

Data Output | Explain | Messages

	actor_id smallint	name character varying(45)
1	1	Harrison Ford
2	2	Viggo Mortensen
3	3	Robert De Niro

Data Output | Explain

	actor_id smallint	film_id smallint
1	1	2
2	1	3
3	2	1

AHORA COMO TRANSACCIÓN

```
select * from actor
```

Data Output | Explain | Messages

	actor_id smallint	name character varying(45)
1	1	Harrison Ford
2	2	Viggo Mortensen
3	3	Robert De Niro

- No ha eliminado tampoco a Robert De Niro!!!
- Explicación: por el error del segundo DELETE nunca llega al COMMIT.

AUNQUE SI INTENTAMOS VER RESULTADOS INTERMEDIOS...

```
BEGIN;
DELETE FROM actor WHERE actor_id=3;
SELECT * FROM actor;
DELETE FROM actor WHERE actor_id=1;
COMMIT;
```

Data Output | Explain | **Messages** | History

```
ERROR:  update o delete en «actor» viola la llave foránea «fk_actor» en la tabla «film_actor»
DETAIL:  La llave (actor_id)=(1) todavía es referida desde la tabla «film_actor».
********** Error **********

ERROR: update o delete en «actor» viola la llave foránea «fk_actor» en la tabla «film_actor»
SQL state: 23503
Detail: La llave (actor_id)=(1) todavía es referida desde la tabla «film_actor».
```

Data Output | Explain | Messages | History

	film_id smallint	name character varying(45)
1	1	LOTR: The Fellowship of the Ring
2	2	Star Wars: Episode IV
3	3	Indiana Jones and the Last Crusade

Data Output | Explain | Messages

	actor_id smallint	name character varying(45)
1	1	Harrison Ford
2	2	Viggo Mortensen
3	3	Robert De Niro

Data Output | Explain

	actor_id smallint	film_id smallint
1	1	2
2	1	3
3	2	1

EJERCICIOS

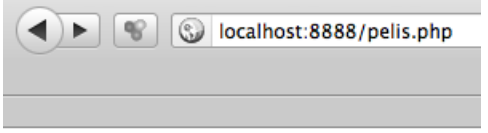
Transacciones en PHP

AHORA DESDE PHP

```
<?php
print "Ejemplo de transacciones en PHP-PDO.<p>";
try {
	$db=new PDO(...);
	$sql = "SELECT * FROM actor";
	foreach ($db->query($sql) as $row){
		print $row['actor_id'] .' - '. $row['name'] . '<br/>';
	}
}
catch (Exception $e){
	print "Un error<p>";
	echo $e->getMessage();
	print "<p>";
}

// Continuar la ejecución
echo 'Fin del script';
?>
```

AHORA EN PHP: SALIDA



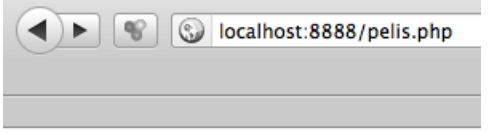
Ejemplo de transacciones en PHP-PDO.

1 - Harrison Ford
2 - Viggo Mortensen
3 - Robert De Niro
Fin del script

LO AGRUPAMOS EN TRANSACCIÓN

```
<?php
print "Ejemplo de transacciones en PHP-PDO.<p>";
try {
	$db=new PDO(...);
	$db->beginTransaction();
	$sql = "SELECT * FROM actor";
	foreach ($db->query($sql) as $row){
		print $row['actor_id'] .' - '. $row['name'] . '<br/>';
	}
	$db->commit();
}
catch (Exception $e){
	print "Un error<p>";
	echo $e->getMessage();
	print "<p>";
}
// Continuar la ejecución
echo 'Fin ';
?>
```

SALIDA



Ejemplo de transacciones en PHP-PDO.

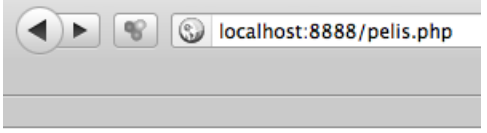
1 - Harrison Ford
2 - Viggo Mortensen
3 - Robert De Niro
Fin del script

¿Y SI LA TRANSACCIÓN NO TERMINA?

```
<?php
print "Ejemplo de transacciones en PHP-PDO.<p>";
try {
	$db=new PDO(...);
	$sql = "SELECT * FROM actor";
	$db->beginTransaction();
	foreach ($db->query($sql) as $row){
		print $row['actor_id'] .' - '. $row['name'] . '<br/>';
	}
	$db->rollBack();
}
catch (Exception $e){
	print "Un error<p>";
	echo $e->getMessage();
	print "<p>";
}

// Continuar la ejecución
echo 'Fin del script';
?>
```

SALIDA



Ejemplo de transacciones en PHP-PDO.

1 - Harrison Ford
2 - Viggo Mortensen
3 - Robert De Niro
Fin del script

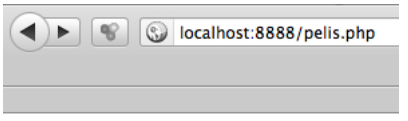
¿Y SI "TERMINA" DESPUÉS DEL ROLLBACK?

```
<?php
print "Ejemplo de transacciones en PHP-PDO.<p>";
try {
	$db=new PDO(...);
	$sql = "SELECT * FROM actor";
	$db->beginTransaction();
	foreach ($db->query($sql) as $row){
		print $row['actor_id'] .' - '. $row['name'] . '<br/>';
	}
	$db->rollBack();
	$db->commit();
}
catch (Exception $e){
	print "Un error<p>";
	echo $e->getMessage();
	print "<p>";
}

// Continuar la ejecución
echo 'Fin del script';
?>
```

SALIDA

- Ahora tenemos un error:
  - Intentó ejecutar un COMMIT después de ROLLBACK --> saltó una excepción



Ejemplo de transacciones en PHP-PDO.

1 - Harrison Ford
2 - Viggo Mortensen
3 - Robert De Niro
Un error

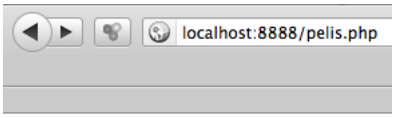
There is no active transaction

Fin del script

¿QUÉ PASA SI BORRAMOS?

```
<?php
function printTableActor($dbRef){
	$sql = "SELECT * FROM actor";
	foreach ($dbRef->query($sql) as $row){
		print $row['actor_id'] .' - '. $row['name'] . '<br/>';
	}
	print '<p>';
}
print "Ejemplo de transacciones en PHP-PDO.<p>";
try {
	$db=new PDO(...);
	printTableActor($db);
	$sql = "DELETE FROM actor WHERE actor_id=3";
	$db->exec($sql);
	printTableActor($db);
}
catch (Exception $e){
	print "Un error<p>";
	echo $e->getMessage() . "<p>";
}
// Continuar la ejecución
echo 'Fin del script';
?>
```

SALIDA



Ejemplo de transacciones en PHP-PDO.

1 - Harrison Ford
2 - Viggo Mortensen
3 - Robert De Niro

1 - Harrison Ford
2 - Viggo Mortensen

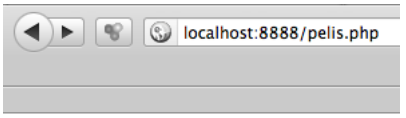
Fin del script

¿Y SI BORRAMOS Y LUEGO HACEMOS *ROLLBACK*?

```
<?php
print "Ejemplo de transacciones en PHP-PDO.<p>";
try {
	$db=new PDO("...");
	$db->beginTransaction();
	printTableActor($db);
	$sql = "DELETE FROM actor WHERE actor_id=3";
	$db->exec($sql);
	printTableActor($db);
	$db->rollBack();
	printTableActor($db);
}
catch (Exception $e){
	print "Un error<p>";
	echo $e->getMessage();
	print "<p>";
}

// Continuar la ejecución
echo 'Fin del script';
?>
```

SALIDA



Ejemplo de transacciones en PHP-PDO.

1 - Harrison Ford
2 - Viggo Mortensen
3 - Robert De Niro

1 - Harrison Ford
2 - Viggo Mortensen

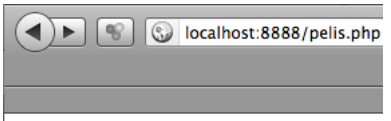
1 - Harrison Ford
2 - Viggo Mortensen
3 - Robert De Niro

Fin del script

RESULTADOS INTERMEDIOS DESDE FUERA DE LA TX

```
<?php
print "Ejemplo de transacciones en PHP-PDO.<p>";
function concurrente(){
   try {
      $db2 = new PDO("...");
      printTableActor($db2);
   }
   catch (PDOException $e){
       print "Un error en concurrente. <p>";
       echo $e->getMessage();
       print "<p>";
   }
}
try {
       $db=new PDO("...");
       $db->beginTransaction();
       printTableActor($db);
       $sql = "DELETE FROM actor WHERE actor_id=3";
       $db->exec($sql);
       concurrente();
       $db->rollBack();
       printTableActor($db);
}
...
```

RESULTADO



Ejemplo de transacciones en PHP-PDO.

1 - Harrison Ford
2 - Viggo Mortensen
3 - Robert De Niro

1 - Harrison Ford
2 - Viggo Mortensen
3 - Robert De Niro

desde concurrente()

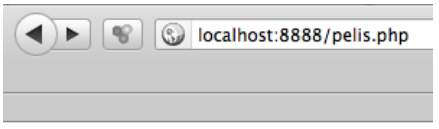
1 - Harrison Ford
2 - Viggo Mortensen
3 - Robert De Niro

Fin del script

O MEJOR AÚN

```
<?php
print "Ejemplo de transacciones en PHP-PDO.<p>";
...
try {
        $db=new PDO("...");
        $db->beginTransaction();
        printTableActor($db);
        $sql = "DELETE FROM actor WHERE actor_id=3";
        $db->exec($sql);
        concurrente();
        $db->commit();
        concurrente();
}
...
```

RESULTADO



Ejemplo de transacciones en PHP-PDO.

1 - Harrison Ford
2 - Viggo Mortensen
3 - Robert De Niro

1 - Harrison Ford
2 - Viggo Mortensen
3 - Robert De Niro

desde `concurrente()`

1 - Harrison Ford
2 - Viggo Mortensen

desde `concurrente()`

Fin del script

¿Y SI SE PRODUCE UN ERROR?

Valores devueltos Report a bug

PDO::exec() returns the number of rows that were modified or deleted by the SQL statement you issued. If no rows were affected, **PDO::exec()** returns *0*.

Advertencia

Esta función quizá devuelve Boolean **FALSE**, pero quizá también devuelve un valor non-Boolean que se evaluará como **FALSE**, como *0* o "". Por favor lea la sección en Booleans para más información. Use el operador === para testear el valor devuelto por esta función.

ROLLBACK EN CASO DE ERROR

```
<?php
      $db->beginTransaction();
      printTableActor($db);
      $sql = "DELETE FROM actor WHERE actor_id=3";
      $result = $db->exec($sql);
      if ($result === FALSE){
            print "Error detectado en DELETE 1<p>";
            $db->rollBack();
            die();
      } else {
            print "Filas afectadas por exec:" . $result . '<p>';
      }
      printTableActor($db);
      $sql = "DELETE FROM actor WHERE actor_id=1";
      $result = $db->exec($sql);
      if ($result === FALSE){
            print "Error detectado en DELETE 2<p>";
            $db->rollBack();
            die();
      } else {
            print "Filas afectadas por exec:" . $result . '<p>';
      }
      printTableActor($db);
      $db->commit();
...
```

SALIDA

Ejemplo de transacciones en PHP-PDO.

1 - Harrison Ford
2 - Viggo Mortensen
3 - Robert de Niro

Filas afectadas por exec:1

1 - Harrison Ford
2 - Viggo Mortensen

Error detectado en DELETE 2

¿Cuál será el contenido de la tabla después de la ejecución?