

Instituto Tecnológico de Villahermosa
Ing. Sistemas Computacionales
6to Semestre



Taller de Base de Datos

Reporte Unidad 5 - Transacciones

Reynaldo Bernard de Dios de la Cruz



Domingo, 08 de Mayo
de 2016.

Taller de Base de Datos

Contenido

Introducción	3
5.1 - Conceptos básicos	3
Marco teórico	3
Transacción.....	3
Autocommit	3
5.2 - Propiedades de las transacciones	4
Marco teórico	4
• Atomicidad	4
• Consistencia	4
• Aislamiento	4
• Durabilidad	4
5.3 Grados de consistencia.....	4
Marco teórico	4
• Active	4
• Uncommitted	4
• Rolled Back	4
• Committed	4
5.4 Niveles de aislamiento	5
Marco teórico	5
5.5 COMMIT y ROLLBACK.....	6
Marco teórico	6
COMMIT	6
ROLLBACK	6
SAVE POINT y ROLLBACK TO SAVEPOINT	7
Marco practico	8
Transacción 1: Eliminación de registros	9
Transacción 2: Actualización de registros	10
Transacción 3: Dos consultas en una transacción.	11
Transacción 4: Eliminación de tabla - DROP TABLE	13
Transacción 5: Selección de punto de guardado - SAVEPOINT.....	14
Anexo	17
Conclusión	18
Bibliografía.....	18

Taller de Base de Datos

Introducción

Las transacciones son un conjunto de instrucciones SQL que tienen la cualidad de ejecutarse como una unidad, es decir, o se ejecutan todas o no se ejecuta ninguna. Si una transacción tiene éxito, todas las modificaciones de los datos realizados durante la transacción se guardan en la base de datos. Si una transacción contiene errores los cambios no se guardarán en la base de datos.

En un sistema ideal, las transacciones deberían garantizar todas las propiedades ACID (en la práctica, a veces alguna de estas propiedades se simplifica o debilita con vistas para obtener un mejor rendimiento).

ACID son las siglas de Atomicity, Consistency, Isolation y Durability (Atomicidad, Consistencia, Aislamiento, Durabilidad)

El ejemplo clásico de transacción es una transferencia bancaria, en la que quitamos saldo a una cuenta y lo añadimos en otra. Si no es posible abonar el dinero en la cuenta de destino, no debemos quitarlo de la cuenta de origen. Para implementar transacciones en MySQL hay que utilizar la versión 5.0 o superior y el motor de almacenamiento InnoDB.

5.1 - Conceptos básicos

Marco teórico

Transacción

Una transacción en un Sistema de Gestión de Bases de Datos (SGBD), es un conjunto de órdenes que se ejecutan formando una unidad de trabajo, es decir, en forma indivisible o atómica.

Un SGBD se dice transaccional, si es capaz de mantener la integridad de los datos, haciendo que estas transacciones no puedan finalizar en un estado intermedio. Cuando por alguna causa el sistema debe cancelar la transacción, empieza a deshacer las órdenes ejecutadas hasta dejar la base de datos en su estado inicial (llamado punto de integridad), como si la orden de la transacción nunca se hubiese realizado.

Para esto, el lenguaje de consulta de datos SQL (Structured Query Language), provee los mecanismos para especificar que un conjunto de acciones deben constituir una transacción.

- **BEGIN:** Especifica que va a empezar una transacción.
- **COMMIT:** Le indica al motor que puede considerar la transacción completada con éxito.
- **ROLLBACK:** Indica que se ha alcanzado un fallo y que debe restablecer la base al punto de integridad.

Autocommit

MySQL tiene una variable de entorno llamada **AUTOCOMMIT**, que por defecto tiene el valor 1. Configurado de esta manera no se pueden usar transacciones, porque MySQL automáticamente hace un **COMMIT** luego de cada consulta.

Para usar transacciones entonces, hay que poner autocommit a 0 (desactivarlo).

Nota: Si autocommit se pone a cualquier número $N > 1$, MySQL hace un COMMIT automático luego de N consultas.

Taller de Base de Datos

Para cambiar el valor de autocommit, simplemente se usa **SET AUTOCOMMIT = 0;**

```
mysql> SET AUTOCOMMIT = 0;  
Query OK, 0 rows affected (0.05 sec)  
mysql>
```

5.2 - Propiedades de las transacciones

Marco teórico

Una unidad lógica de trabajo debe exhibir cuatro propiedades, conocidas como propiedades ACID (atomicidad, coherencia, aislamiento y durabilidad), para ser calificada como transacción.

- **Atomicidad:** es la propiedad que asegura que la operación se ha realizado o no, y por lo tanto ante un fallo del sistema no puede quedar a medias.
- **Consistencia:** Integridad. Es la propiedad que asegura que sólo se empieza aquello que se puede acabar. Por lo tanto se ejecutan aquellas operaciones que no van a romper las reglas y directrices de integridad de la base de datos. La propiedad de consistencia sostiene que cualquier transacción llevará a la base de datos desde un estado válido a otro también válido.
- **Aislamiento:** es la propiedad que asegura que una operación no puede afectar a otras. Esto asegura que la realización de dos transacciones sobre la misma información sean independientes y no generen ningún tipo de error.
- **Durabilidad:** es la propiedad que asegura que una vez realizada la operación, ésta persistirá y no se podrá deshacer aunque falle el sistema.

5.3 Grados de consistencia

Marco teórico

Consistencia es un término más amplio que el de integridad. Podría definirse como la coherencia entre todos los datos de la base de datos. Cuando se pierde la integridad también se pierde la consistencia. Pero la consistencia también puede perderse por razones de funcionamiento.

Una transacción finalizada (confirmada parcialmente) puede no confirmarse definitivamente (consistencia). Si se confirma definitivamente el sistema asegura la persistencia de los cambios que ha efectuado en la base de datos. Si se anula los cambios que ha efectuado son deshechos.

Una transacción que termina con éxito se dice que está comprometida (committed), una transacción que haya sido comprometida llevará a la base de datos a un nuevo estado consistente que debe permanecer incluso si hay un fallo en el sistema. En cualquier momento una transacción sólo puede estar en uno de los siguientes estados:

- **Active** (Activa): el estado inicial; la transacción permanece en este estado durante su ejecución.
- **Uncommitted** (Parcialmente comprometida): Después de ejecutarse la última transacción.
- **Failed** (Fallida): tras descubrir que no se puede continuar la ejecución normal.
- **Rolled Back** (Abortada): después de haber retrocedido la transacción y restablecido la base de datos a su estado anterior al comienzo de la transacción.
- **Committed** (Comprometida): tras completarse con éxito.

Taller de Base de Datos

5.4 Niveles de aislamiento

Marco teórico

Las transacciones especifican un nivel de aislamiento que define el grado en que se debe aislar una transacción de las modificaciones de recursos o datos realizadas por otras transacciones. Los niveles de aislamiento se describen en cuanto a los efectos secundarios de la simultaneidad que se permiten, como las lecturas desfasadas o ficticias.

Control de los niveles de aislamiento de transacción:

- Controla si se realizan bloqueos cuando se leen los datos y qué tipos de bloqueos se solicitan.
- Duración de los bloqueos de lectura.
- Si una operación de lectura que hace referencia a filas modificadas por otra transacción:
 1. Se bloquea hasta que se libera el bloqueo exclusivo de la fila.
 2. Recupera la versión confirmada de la fila que existía en el momento en el que empezó la instrucción o la transacción.
 3. Lee la modificación de los datos no confirmados.

El nivel de aislamiento para una sesión SQL establece el comportamiento de los bloqueos para las instrucciones SQL. El estándar ANSI/ISO SQL define cuatro niveles de aislamiento transaccional en función de tres eventos que son permitidos o no dependiendo del nivel de aislamiento. Estos eventos son:

- **Lectura sucia.** Las sentencias SELECT son ejecutadas sin realizar bloqueos, pero podría usarse una versión anterior de un registro. Por lo tanto, las lecturas no son consistentes al usar este nivel de aislamiento.
- **Lectura no repetible.** Una transacción vuelve a leer datos que previamente había leído y encuentra que han sido modificados o eliminados por una transacción cursada.
- **Lectura fantasma.** Una transacción vuelve a ejecutar una consulta, devolviendo un conjunto de registros que satisfacen una condición de búsqueda y encuentra que otros registro que satisfacen la condición han sido insertadas por otra transacción cursada.

Niveles de aislamiento: Comportamiento permitido			
Nivel de aislamiento	Lectura		
	Sucia	No repetible	Fantasma
Lectura no comprometida	Sí	Sí	Sí
Lectura comprometida	No	Sí	Sí
Lectura repetible	No	No	Sí
Secuenciable	No	No	No

Taller de Base de Datos

El estándar SQL trataba de establecer los niveles de aislamiento que permitirían a varios grados de consistencia para queries ejecutadas en cada nivel de aislamiento. Las lecturas repetibles "REPEATABLE READ" es el nivel de aislamiento que garantiza que un query un resultado consistente

En la definición SQL estándar, la lectura comprometida "READ COMMITTED" no regresa resultados consistentes, en la lectura no comprometida "READ UNCOMMITTED" las sentencias SELECT son ejecutadas sin realizar bloqueos, pero podría usarse una versión anterior de un registro. Por lo tanto, las lecturas no son consistentes al usar este nivel de aislamiento.

A mayor grado de aislamiento, mayor precisión, pero a costa de menor concurrencia. La sintaxis es la siguiente:

```
SET [SESSION | GLOBAL] TRANSACTION ISOLATION LEVEL  
  
{ READ UNCOMMITTED | READ COMMITTED | REPEATABLE READ | SERIALIZABLE}
```

Los niveles de aislamiento de transacciones globales y de sesión pueden consultarse con estas sentencias:

```
SELECT @@global.tx_isolation;  
SELECT @@tx_isolation;
```

5.5 COMMIT y ROLLBACK

Marco teórico

COMMIT

Esta instrucción de SQL se utiliza para confirmar como permanentes las modificaciones realizadas en una transacción. Por ejemplo:

```
BEGIN;  
    Insert into Mvtos Values ('0150', 'Dep', 1500, '08-10-2008');  
    Insert into Mvtos Values ('7120', 'Ret', 1500, '08-10-2008');  
    Update Cuentas Set Saldo = Saldo + 1 where Numero = '0150';  
    Update Cuentas Set Saldo = Saldo - 1 Where Numero = '7120';  
  
COMMIT;
```

Con la instruccion commit se pone una marca para saber hasta qué punto se hizo la transacción.

ROLLBACK

Esta funcion del SQL se utiliza para deshacer todas las modificaciones realizadas desde la última confirmación. Por ejemplo:

```
BEGIN;  
    Insert into Mvtos Values ('0150', 'Dep', 1500, '08-10-2008');  
    Insert into Mvtos Values ('7120', 'Ret', 1500, '08-10-2008');  
  
ROLLBACK;
```

Taller de Base de Datos

Esto cancelará las dos consultas de inserción y es como si no hubiera insertado nada.

SAVE POINT y ROLLBACK TO SAVEPOINT

En MySQL 5.0 y superiores, InnoDB soporta los comandos sql SAVEPOINT y ROLLBACK TO SAVEPOINT.

Para guardar un punto de guardado es necesario ejecutar en primer lugar BEGIN y posteriormente los puntos de guardado, la sintaxis es la siguiente, **SAVEPOINT nombre_del_punto_de_guardado**. Por ejemplo:

```
mysql> SAVEPOINT guardado1;  
Query OK, 0 rows affected (0.03 sec)  
  
mysql>
```

Si se desea regresar a un punto de guardado en específico la sintaxis a utilizar será la siguiente **ROLLBACK TO SAVEPOINT nombre_del_punto_de_guardado**. Por ejemplo:

```
mysql> ROLLBACK TO SAVEPOINT guardado1;  
Query OK, 0 rows affected (0.00 sec)  
mysql>
```

El comando **SAVEPOINT** crea un punto dentro de una transacción con un nombre. Si la transacción actual tiene un punto con el mismo nombre, el antiguo se borra y se crea el nuevo.

El comando **ROLLBACK TO SAVEPOINT** deshace una transacción hasta el punto nombrado. Las modificaciones que la transacción actual hace al registro tras el punto se deshacen en el ROLLBACK, pero InnoDB no libera los bloqueos de registro que se almacenaron en memoria tras el punto. (Tenga en cuenta que para un nuevo registro insertado, la información de bloqueo se realiza a partir del ID de transacción almacenado en el registro; el bloqueo no se almacena separadamente en memoria. En este caso, el bloqueo de registro se libera al deshacerse todo.) Los puntos creados tras el punto nombrado se borran.

Si un comando retorna el siguiente error, significa que no existe ningún punto con el nombre especificado:

ERROR 1305 (42000): SAVEPOINT guardado2 does not exist

Todos los puntos de la transacción actual se borran si ejecuta un COMMIT, o un ROLLBACK que no nombre ningún punto.

NOTA: Hay varias sentencias SQL en las que no se puede utilizar ROLLBACK ya que implícitamente MySQL realiza un COMMIT:
CREATE / ALTER / DROP DATABASE
CREATE / ALTER / DROP / RENAME / TRUNCATE TABLE
CREATE / DROP INDEX
CREATE / DROP EVENT
CREATE / DROP FUNCTION
CREATE / DROP PROCEDURE

Taller de Base de Datos

Para este tipo de declaraciones SQL no importa si está o no activado el AUTOCOMMIT, MYSQL ejecutará un COMMIT tan pronto ejecute la declaración.

Marco practico

Primero que nada hay que establecer una conexión a la base de datos, se puede usar la consola o utilizar el servicio de MySQL command line de MySQL Server.

```
Microsoft Windows [Versión 6.3.9600]
(c) 2013 Microsoft Corporation. Todos los derechos reservados.

C:\Users\Rodolfo>mysql -u root -p
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3
Server version: 5.7.11-log MySQL Community Server (GPL)

Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

Posteriormente hay que colocar la variable AUTOCOMMIT en falso, es decir, en cero como se muestra a continuación:

```
mysql> SET AUTOCOMMIT =0;
Query OK, 0 rows affected (0.00 sec)

mysql>
```

Ahora se pueden empezar las transacciones sin que se guarden en la base de datos, es decir, los cambios se harán pero no se guardarán hasta que hagamos un COMMIT.

Se puede consultar el valor de la variable usando la siguiente consulta, SELECT @@AUTOCOMMIT;

```
mysql> SELECT @@AUTOCOMMIT;
+-----+
| @@AUTOCOMMIT |
+-----+
|          0   |
+-----+
1 row in set (0.03 sec)
```


Taller de Base de Datos

Transacción 1: Eliminación de registros

Se consulta primeramente la tabla barcos para ver todos los registros.

```
mysql> use nautica;
```

```
Database changed
```

```
mysql> SELECT * FROM barcos;
```

matricula	nombre	numero_amarre	cuota_pago
BA1054	Emilia	1478	2474.74
BA1084	Paola	987	7172.32
BA1102	Valeria	1298	3329.98
BA1109	America	165	6250.3
BA1117	Isabela	701	2224.57
BA1121	Maximiliano	772	4251.52
BA1168	Elena	1877	5172.4
BA1171	Linda	1440	3616.8
BA1200	Jaime	844	4676.2
BA1211	Alexia	1776	4519.07
BA1232	Mateo	1867	4074.95
BA1242	Lucian	439	4077.79
BA1258	Israel	313	7885.85
BA1277	Abraham	652	7781.45
BA1345	Lia	345	4467.25
BA1402	Alicia	1294	2438.25
BA1404	Sara	861	4112.8
BA1451	Nicolas	1941	2820.72
BA1487	Ada	885	5419.7
BA1566	Marco	1790	5693.04
BA1581	Marina	327	2840.57
BA1590	Fernanda	1858	2525.93
BA1607	Fernanda	1783	882.44
BA1610	Fabian	1333	3356.65
BA1656	Roberto	501	1901.07
BA1663	Virginia	72	6657.9
BA1739	Pedro	1782	915.24
BA1856	Noe	1273	2440.39
BA1858	Tatiana	218	731.96
BA1902	Guillermo	1884	2736.6

```
30 rows in set (0.07 sec)
```

Inicio de la transacción con el comando BEGIN y una eliminación completa de los registros en la tabla barcos.

```
mysql> BEGIN;
```

```
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> DELETE FROM barcos;
```

```
Query OK, 30 rows affected (0.16 sec)
```

```
mysql>
```

Si se consulta nuevamente la tabla, es evidente que no mostrará ni un registro, es más MySQL informará que la tabla está vacía (empty).

```
mysql> SELECT * FROM barcos;
```

```
Empty set (0.00 sec)
```

Taller de Base de Datos

Para confirmar la operación de borrado basta con poner COMMIT; y los cambios serán guardados, en este ejemplo se muestra cómo deshacer ese cambio(borrado de los registros de la tabla barcos), para ello se usará el comando ROLLBACK. Esto permitirá que se deshagan todas las consultas hechas después del comando BEGIN. Para este ejemplo, recuperar los registros de la tabla barcos.

```
mysql> ROLLBACK;  
Query OK, 0 rows affected (0.07 sec)
```

Transacción 2: Actualización de registros

Hay que recordar que para guardar los cambios se puede usar COMMIT o para deshacerlos completamente se usa ROLLBACK, de igual forma para decirle a MySQL que la transacción ha empezado se usa BEGIN.

Para este ejemplo se cambiará el apellido paterno de las personas que tengan su apellido con *De La Cruz* a *De La Torre*, para ello se muestra primeramente la tabla y ordenado por apellido paterno.

```
mysql> SELECT * FROM personas ORDER BY apellido_paterno;
```

rfc	nombre	apellido_paterno	apellido_materno	fecha_nac	tel	socio
AIAD552513	Elena	Alirio	Cordero	1998-02-23	9987653879	si
BBMG039362	Alfredo	Alvarez	Sixto	1998-04-13	9962697995	si
CKLJ610123	Jairo	Contreras	Jimenez	2000-07-13	9932773268	si
EGMA110456	Amaya	Contreras	Villegas	1999-05-17	9974604231	si
BGJG814539	Isabella	Cordero	Alvarez	1996-06-03	9960762854	no
MAIJ720187	Angelica	Cordero	Herrera	1997-02-07	9939966180	no
CCEM283256	Victor	De Dios	Suarez	2000-07-23	9943841431	si
EMBJ557200	Luis	De Dios	Jimenez	1995-04-24	9968384791	no
LFAI034723	Natalia	De Dios	Villegas	2000-01-25	9919482255	no
CEIE997246	Sandra	De La Cruz	De La Cruz	1995-11-01	9962286762	no
DFCH950877	Emiliano	De La Cruz	Cordero	1995-08-04	9981538338	no
EACJ483884	Elsa	De La Cruz	Jimenez	1997-02-16	9932517287	no
EDBF467297	Ruben	De La Cruz	Jimenez	2000-05-10	9997384647	si
FBLA376692	Noe	De La Cruz	Jimenez	1996-04-14	9971529299	si
LMLE981803	Uivian	De La Cruz	Contreras	1998-06-08	9968889570	no
MJLH264839	Eliana	De La Cruz	Villegas	2000-01-02	9982856587	no
MAIM669494	Rafael	Herrera	Alirio	2000-12-14	9993385142	no
AMKC324032	Gabriel	Jimenez	Contreras	1997-06-25	9949949432	no
LBJI891183	Rafael	Jimenez	De Dios	1996-11-22	9916794245	no
EGGF549945	Alina	Sanchez	Contreras	1998-12-28	9910665070	si
KCMN970213	Sebastian	Sanchez	Alirio	1995-02-11	9980622166	no
LFCM771604	Juliana	Sanchez	Alvarez	1997-11-01	9957446061	no
LGH8714834	Adrian	Sanchez	Villegas	2000-11-30	9992868540	no
DALH547074	Joaquin	Suarez	Herrera	1995-10-12	9953263950	si
DIIJ695472	Israel	Suarez	De Dios	1999-08-01	9981203938	no
IKCH898325	Mario	Suarez	Contreras	1997-12-18	9976367469	si
FLMH862656	Miranda	Villegas	Alirio	1995-12-07	9977106812	si
GJJB452660	Tobias	Villegas	De Dios	1999-02-28	9910319647	no
IEHC303916	Amanda	Villegas	Villegas	1999-12-15	9955807925	si
JMMH533503	Matias	Villegas	Cordero	1997-02-16	9915182693	si

A continuación se actualiza el valor de los apellidos.

```
mysql> UPDATE personas SET apellido_paterno='De La Torre'  
-> WHERE apellido_paterno="De La Cruz";  
Query OK, 7 rows affected (0.00 sec)  
Rows matched: 7 Changed: 7 Warnings: 0
```

Se consulta nuevamente la tabla ordenada por apellido paterno.

Taller de Base de Datos

rfc	nombre	apellido_paterno	apellido_materno	fecha_nac	tel	socio
AIAD552513	Elena	Alirio	Cordero	1998-02-23	9987653879	si
BBMG039362	Alfredo	Alvarez	Sixto	1998-04-13	9962697995	si
CKLJ610123	Jairo	Contreras	Jimenez	2000-07-13	9932773268	si
ECMA110456	Amaya	Contreras	Villegas	1999-05-17	9974604231	si
BGJG814539	Isabella	Cordero	Alvarez	1996-06-03	9960762854	no
MAIJ720187	Angelica	Cordero	Herrera	1997-02-07	9939966180	no
CCEM203256	Victor	De Dios	Suarez	2000-07-23	9943841431	si
EMBJ557200	Luis	De Dios	Jimenez	1995-04-24	9968384791	no
LFAl034723	Natalia	De Dios	Villegas	2000-01-25	9919482255	no
CEIE997246	Sandra	De La Torre	De La Cruz	1995-11-01	9962286762	no
DFCH950877	Emiliano	De La Torre	Cordero	1995-08-04	9981538338	no
EACJ483884	Elsa	De La Torre	Jimenez	1997-02-16	9932517287	no
EDBF467297	Ruben	De La Torre	Jimenez	2000-05-10	9997384647	si
FBLA376692	Noe	De La Torre	Jimenez	1996-04-14	9971529299	si
LMLE981803	Uivian	De La Torre	Contreras	1998-06-08	9968889570	no
MJLH264839	Elia	De La Torre	Villegas	2000-01-02	9982856587	no
MAIM669494	Rafael	Herrera	Alirio	2000-12-14	9993385142	no
AMKC324032	Gabriel	Jimenez	Contreras	1997-06-25	9949949432	no
LBJI891183	Rafael	Jimenez	De Dios	1996-11-22	9916794245	no
EGGF549945	Alina	Sanchez	Contreras	1998-12-28	9910665070	si
KCMN770213	Sebastian	Sanchez	Alirio	1995-02-11	9980622166	no
LFGM771604	Juliana	Sanchez	Alvarez	1997-11-01	9957446061	no
LGBB714834	Adrian	Sanchez	Villegas	2000-11-30	9992868548	no
DALH547074	Joaquin	Suarez	Herrera	1995-10-12	9953263950	si
DIJ695472	Israel	Suarez	De Dios	1999-08-01	9981203938	no
IKCH898325	Mario	Suarez	Contreras	1997-12-18	9976367469	si
FLMH862656	Miranda	Villegas	Alirio	1995-12-07	9977106812	si
GJJB452660	Tobias	Villegas	De Dios	1999-02-28	9910319647	no
IEHC303916	Amanda	Villegas	Villegas	1999-12-15	9955807925	si
JMHM533503	Matias	Villegas	Cordero	1997-02-16	9915182693	si

Ahora en esta ocasión para guardar los cambios en la base de datos se utiliza COMMIT.

```
mysql> COMMIT;
Query OK, 0 rows affected (0.06 sec)

mysql>
```

Transacción 3: Dos consultas en una transacción.
A continuación se consulta la tabla barcos_salidas.

```
mysql> SELECT * FROM barcos_salidas;
```

rfc_persona	barcos_matricula	fecha	hora	destino
AIAD552513	BA1232	2004-02-08	13:21:05	Emiliano Zapata
AIAD552513	BA1242	2005-02-11	16:29:21	Nacajuca
CEIE997246	BA1902	2014-11-05	22:45:34	Macuspana
CKLJ610123	BA1902	2002-08-11	04:06:48	Huimanguillo
DFCH950877	BA1200	2006-02-07	03:04:48	Macuspana
DFCH950877	BA1232	2020-10-02	11:28:53	Cardenas
DIJ695472	BA1345	1995-06-16	16:20:43	Huimanguillo
EACJ483884	BA1663	2019-04-01	24:44:10	Macuspana
EDBF467297	BA1109	2013-12-27	07:03:57	Teapa
EDBF467297	BA1232	2010-06-27	16:39:41	Huimanguillo
EDBF467297	BA1607	2006-06-26	05:48:05	Huimanguillo
EMBJ557200	BA1242	2013-08-12	22:18:12	Nacajuca
EMBJ557200	BA1610	2014-11-19	14:14:29	Huimanguillo
EMBJ557200	BA1663	2004-09-01	22:14:57	Balancan
FLMH862656	BA1856	2008-10-26	14:44:49	Nacajuca
JMHM533503	BA1109	2012-10-18	02:15:32	Nacajuca
LFAl034723	BA1451	2004-11-22	23:30:27	Nacajuca
LMLE981803	BA1487	2015-10-05	14:11:30	Centro
MAIJ720187	BA1232	2014-08-27	15:08:50	Nacajuca
MAIM669494	BA1232	2016-06-05	14:01:47	Macuspana
MAIM669494	BA1581	2010-05-11	21:40:33	Emiliano Zapata
MAIM669494	BA1739	2013-12-09	17:23:58	Emiliano Zapata
MJLH264839	BA1054	2000-08-13	08:30:51	Centro
MJLH264839	BA1566	2018-10-13	10:40:56	Tacotalpa
MJLH264839	BA1902	2020-04-15	23:18:01	Centro

Taller de Base de Datos

En esta ocasión se eliminará y actualizarán registros en la tabla `barcos_salidas`. Para este ejemplo se eliminarán aquellos barcos que tengan como destino *Centro* y se actualizará el nombre de Balancán a Balankan. Primeramente se hace un `BEGIN` para indicar que la transacción iniciará.

```
mysql> BEGIN;
Query OK, 0 rows affected (0.00 sec)

mysql> DELETE FROM barcos_salidas WHERE destino="Centro";
Query OK, 3 rows affected (0.00 sec)

mysql> UPDATE barcos_salidas SET destino="Balankan" WHERE destino="Balancan";
Query OK, 1 row affected (0.00 sec)
Rows matched: 1 Changed: 1 Warnings: 0
```

Se vuelve a consultar la tabla para ver los cambios hechos.

```
mysql> SELECT * FROM barcos_salidas;
```

rfc_persona	barcos_matricula	fecha	hora	destino
AIAD552513	BA1232	2004-02-08	13:21:05	Emiliano Zapata
AIAD552513	BA1242	2005-02-11	16:29:21	Nacajuca
CEIE997246	BA1902	2014-11-05	22:45:34	Macuspana
CKLJ610123	BA1902	2002-08-11	04:06:48	Huimanguillo
DFCH950877	BA1200	2006-02-07	03:04:48	Macuspana
DFCH950877	BA1232	2020-10-02	11:28:53	Cardenas
DI IJ695472	BA1345	1995-06-16	16:20:43	Huimanguillo
EACJ483884	BA1663	2019-04-01	24:44:10	Macuspana
EDBF467297	BA1109	2013-12-27	07:03:57	Teapa
EDBF467297	BA1232	2010-06-27	16:39:41	Huimanguillo
EDBF467297	BA1607	2006-06-26	05:48:05	Huimanguillo
EMBJ557200	BA1242	2013-08-12	22:18:12	Nacajuca
EMBJ557200	BA1610	2014-11-19	14:14:29	Huimanguillo
EMBJ557200	BA1663	2004-09-01	22:14:57	Balankan
FLMH862656	BA1856	2008-10-26	14:44:49	Nacajuca
JMHM533503	BA1109	2012-10-18	02:15:32	Nacajuca
LFAI034723	BA1451	2004-11-22	23:30:27	Nacajuca
MAIJ720187	BA1232	2014-08-27	15:08:50	Nacajuca
MAIM669494	BA1232	2016-06-05	14:01:47	Macuspana
MAIM669494	BA1581	2010-05-11	21:40:33	Emiliano Zapata
MAIM669494	BA1739	2013-12-09	17:23:58	Emiliano Zapata
MJLH264839	BA1566	2018-10-13	10:40:56	Tacotalpa

22 rows in set (0.00 sec)

Efectivamente los cambios han sido realizado exitosamente, pero todavía no se ha ejecutado ni `COMMIT` ni `ROLLBACK`, para este ejemplo se desea regresar al estado original y para ello se ejecuta `ROLLBACK`.

```
mysql> ROLLBACK;
Query OK, 0 rows affected (0.08 sec)
```

Al ejecutar `ROLLBACK` las dos consultas (eliminación de los barcos con destino al centro y actualización de balancan a balankan) se han deshecho. Una vez más se verifica que `ROLLBACK` se ejecutó exitosamente, para ello se seleccionan los registros de la tabla `barcos_salidas`.

Taller de Base de Datos

```
mysql> SELECT * FROM barcos_salidas;
```

rfc_persona	barcos_matricula	fecha	hora	destino
AIAD552513	BA1232	2004-02-08	13:21:05	Emiliano Zapata
AIAD552513	BA1242	2005-02-11	16:29:21	Nacajuca
CEIE997246	BA1902	2014-11-05	22:45:34	Macuspana
CKLJ610123	BA1902	2002-08-11	04:06:48	Huimanguillo
DFCH950877	BA1200	2006-02-07	03:04:48	Macuspana
DFCH950877	BA1232	2020-10-02	11:28:53	Cardenas
DIJ695472	BA1345	1995-06-16	16:20:43	Huimanguillo
EACJ483884	BA1663	2019-04-01	24:44:10	Macuspana
EDBF467297	BA1109	2013-12-27	07:03:57	Teapa
EDBF467297	BA1232	2010-06-27	16:39:41	Huimanguillo
EDBF467297	BA1607	2006-06-26	05:48:05	Huimanguillo
EMBJ557200	BA1242	2013-08-12	22:18:12	Nacajuca
EMBJ557200	BA1610	2014-11-19	14:14:29	Huimanguillo
EMBJ557200	BA1663	2004-09-01	22:14:57	Balancan
FLMH862656	BA1856	2008-10-26	14:44:49	Nacajuca
JMHM533503	BA1109	2012-10-18	02:15:32	Nacajuca
LFQI034723	BA1451	2004-11-22	23:30:27	Nacajuca
LMLE981803	BA1487	2015-10-05	14:11:30	Centro
MAIJ720187	BA1232	2014-08-27	15:08:50	Nacajuca
MAIM669494	BA1232	2016-06-05	14:01:47	Macuspana
MAIM669494	BA1581	2010-05-11	21:40:33	Emiliano Zapata
MAIM669494	BA1739	2013-12-09	17:23:58	Emiliano Zapata
MJLH264839	BA1054	2000-08-13	08:30:51	Centro
MJLH264839	BA1566	2018-10-13	10:40:56	Tacotalpa
MJLH264839	BA1902	2020-04-15	23:18:01	Centro

25 rows in set (0.06 sec)

Transacción 4: Eliminación de tabla - DROP TABLE

Como ya se mencionó anteriormente, existen algunas sentencias que MySQL no puede revertir y por ende el ROLLBACK no tiene efecto sobre ellas. Para este ejemplo se dropeará (eliminará) una tabla completa, barcos_salidas.

```
mysql> SHOW TABLES;
```

Tables_in_nautica
barcos
barcos_salidas
personas

3 rows in set (0.00 sec)

Ahora se eliminará barcos_salidas mediante DROP TABLE, recordando que BEGIN debe ser escrito para indicar que la transacción empezará.

```
mysql> BEGIN;  
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> DROP TABLE barcos_salidas;  
Query OK, 0 rows affected (0.32 sec)
```

Ahora si se consultan las tablas, barcos_salidas ya no existe.

```
mysql> SHOW TABLES;
```

Taller de Base de Datos

```
+-----+
| Tables_in_nautica |
+-----+
| barcos             |
| personas           |
+-----+
2 rows in set (0.00 sec)
```

Falta intentar deshacer el DROP TABLE con ROLLBACK, para ello se escribe ROLLBACK.

```
mysql> ROLLBACK;
Query OK, 0 rows affected (0.00 sec)
```

Se muestran las tablas.

```
mysql> SHOW TABLES;
+-----+
| Tables_in_nautica |
+-----+
| barcos             |
| personas           |
+-----+
2 rows in set (0.00 sec)
```

Y precisamente ese es el objetivo de esta transacción, identificar las sentencias o declaraciones de MySQL que no pueden ser afectadas por ROLLBACK, recordándolas son:

CREATE / ALTER / DROP DATABASE
CREATE / ALTER / DROP / RENAME / TRUNCATE TABLE
CREATE / DROP INDEX
CREATE / DROP EVENT
CREATE / DROP FUNCTION
CREATE / DROP PROCEDURE

Transacción 5: Selección de punto de guardado - SAVEPOINT

MySQL permite crear puntos de guardado y también regresarse hasta ellos, para este ejemplo la tabla a utilizar será barcos y se ejecutarán tres consultas diferentes, cada una con su punto de guardado.

```
mysql> SELECT * FROM barcos;
```

Taller de Base de Datos

matricula	nombre	numero_amarre	cuota_pago
BA1054	Emilio	1478	2474.74
BA1084	Paola	987	7172.32
BA1102	Valeria	1298	3329.98
BA1109	America	165	6250.3
BA1117	Isabela	701	2224.57
BA1121	Maximiliano	772	4251.52
BA1168	Elena	1877	5172.4
BA1171	Linda	1440	3616.8
BA1200	Jaime	844	4676.2
BA1211	Alexia	1776	4519.07
BA1232	Mateo	1867	4074.95
BA1242	Lucian	439	4077.79
BA1258	Israel	313	7885.85
BA1277	Abraham	652	7781.45
BA1345	Lia	345	4467.25
BA1402	Alicia	1294	2438.25
BA1404	Sara	861	4112.8
BA1451	Nicolas	1941	2820.72
BA1487	Ada	885	5419.7
BA1566	Marco	1790	5693.04
BA1581	Marina	327	2840.57
BA1590	Fernanda	1858	2525.93
BA1607	Fernanda	1783	882.44
BA1610	Fabian	1333	3356.65
BA1656	Roberto	501	1901.07
BA1663	Virginia	72	6657.9
BA1739	Pedro	1782	915.24
BA1856	Noe	1273	2440.39
BA1858	Tatiana	218	731.96
BA1902	Guillermo	1884	2736.6

30 rows in set (0.00 sec)

Indicar con BEGIN que la transacción se iniciará y se consulta la suma total de todos los registros en la columna cuota_pago.

```
mysql> BEGIN;
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT sum(cuota_pago) as total_pago FROM barcos;
```

```
+-----+
| total_pago |
+-----+
| 117448.45000000001 |
+-----+
1 row in set (0.00 sec)
```

Ahora se pasará a eliminar todos los barcos que tengan una cuota de pago superior a tres mil unidades monetarias.

```
mysql> DELETE FROM barcos WHERE cuota_pago > 3000;
Query OK, 18 rows affected (0.21 sec)
```

Posteriormente se guarda un punto de guardado llamado tres_mil.

```
mysql> SAVEPOINT tres_mil;
Query OK, 0 rows affected (0.00 sec)
```

Taller de Base de Datos

Si se consulta el total, podrá verse que la cantidad ha cambiado.

```
mysql> SELECT sum(cuota_pago) as total_pago FROM barcos;
```

```
+-----+
| sum(cuota_pago) |
+-----+
|      24932.48   |
+-----+
1 row in set (0.00 sec)
```

A continuación se eliminan los registros con cuota de pago mayor a dos mil unidades monetarias.

```
mysql> DELETE FROM barcos WHERE cuota_pago > 2000;
```

Query OK, 8 rows affected (0.14 sec)

Se realiza un punto de guardado llamado dos_mil y se consulta el total.

```
mysql> SAVEPOINT dos_mil;
```

Query OK, 0 rows affected (0.00 sec)

```
mysql> SELECT sum(cuota_pago) FROM barcos;
```

Por último se eliminan los registros que tengan una cuota de pago superior a mil unidades monetarias.

```
mysql> DELETE FROM barcos WHERE cuota_pago > 1000;
```

Query OK, 1 rows affected (0.00 sec)

Se consulta el total.

```
mysql> SELECT sum(cuota_pago) as total_pago FROM barcos;
```

```
+-----+
| sum(cuota_pago) |
+-----+
| 2529.6400000000003 |
+-----+
1 row in set (0.00 sec)
```

Ahora, aquí viene la utilidad de **ROLLBACK TO SAVEPOINT** si se deseara regresar hasta la consulta que tenía solamente los registros superiores a 3000 unidades monetarias, entonces se usa, el total para ese entonces era **24932.48** .

```
mysql> ROLLBACK TO SAVEPOINT tres_mil;
```

Query OK, 0 rows affected (0.00 sec)

Para comprobar si se regresó hasta ese punto se ejecuta la siguiente consulta.

```
mysql> SELECT sum(cuota_pago) as total_pago FROM barcos;
```


Taller de Base de Datos

```
+-----+
| sum(cuota_pago) |
+-----+
|          24932.48 |
+-----+
1 row in set (0.00 sec)
```

Y efectivamente se ha devuelto hasta ese punto de guardado, pero ¿Qué pasa si quieres avanzar hasta un punto de guardado de más adelante? por ejemplo, hasta el punto de guardado *dos_mil*.

Este será el resultado:

```
mysql> ROLLBACK TO SAVEPOINT dos_mil;
ERROR 1305 (42000): SAVEPOINT dos_mil does not exist
```

Es sencillo de explicar, cuando vas creando puntos de guardado y avanzar a uno muy superior ya no puedes bajar más de ese punto, pero sí seguir subiendo a cada punto de guardado.

Supongamos que ahora no quieres ni un solo cambio, es decir llegar hasta donde la tabla estaba intacta (Esto quiere decir que los registros de tres mil unidades monetarias aún existe), para ello ese ejecuta ROLLBACK.

```
mysql> ROLLBACK;
Query OK, 0 rows affected (0.15 sec)
```

Y con esto se subió hasta el punto original de la transacción.

Anexo

Link de repositorio el GIT del script escrito en python para generar los datos, la base de datos en un archivo txt y además el reporte en formato PDF.

https://github.com/Reynald0/taller_bd

Sencilla página web que da un resumen del trabajo realizado y su seguimiento.

http://reynald0.github.io/taller_bd/

Taller de Base de Datos

Conclusión

Usar transacciones es muy simple: antes de ejecutar la primer consulta, se ejecuta una que solamente contiene BEGIN. Luego se ejecutan las consultas que deban ejecutarse. Si éstas resultan exitosas, se termina la transacción con COMMIT, lo cual provoca que los cambios hechos por las consultas anteriores sean permanentes. Si las consultas fallan en algún paso, se puede volver al estado anterior al comienzo de la transacción ejecutando ROLLBACK

Una transacción tiene dos finales posibles, COMMIT (se ejecutan todas las instrucciones y guardamos los datos) y ROLLBACK (se produce un error y no se guardan los cambios). Por defecto, MySQL trae activado el modo autocommit, por lo que cuando se realiza una transacción (INSERT, UPDATE o DELETE) esta se confirma automáticamente. Para desactivar esta opción se debe ejecutar SET AUTOCOMMIT = 0;

Hay que tener en cuenta que al realizar una transacción SQL que cuando se realice un INSERT, UPDATE o DELETE se generará un bloqueo sobre la tabla y que otros clientes no pueden acceder esta para escribir. Pero si podrán realizar lecturas, en las que no podrán ver los datos del primer cliente hasta que los mismos sean confirmados.

Bibliografía

Luciano. (13 de octubre de 2008). COMMIT de transacciones en MySQL. 05/5/2016, de LUAF Sitio web: <http://luauf.com/2008/10/13/commit-de-transacciones-en-mysql/>

Chonchoi. (12 de Diciembre de 2003). Transacciones en MySQL. 05/05/2016, de programacion.net Sitio web: http://programacion.net/articulo/transacciones_en_mysql_242

Desconocido. (2011). Transacciones en MySQL. 05/05/2016, de Blogger Sitio web: <http://apuntes-para-no-olvidar.blogspot.mx/2011/09/transacciones-en-mysql.html>