

BASES DE DATOS AVANZADA

Actividad 5: Efectos de Concurrency

Autores:

Nicolás Correa
Joaquín Fernández
David Pazán

Profesor:

Juan Ricardo Giadach

mayo 16 del 2021

Índice

1. Introducción:	2
1.1. Herramientas	3
2. Desarrollo:	4
2.1. Conceptos importantes	4
2.2. Pasos de la actividad	5
2.3. Desarrollo de la actividad	9
2.3.1. Creación de entorno	9
3. Análisis de resultados	14
4. Conclusión	16
5. Bibliografía	17

1. Introducción:

Al trabajar con bases de datos es común la existencia de conexiones simultáneas, las cuales ingresan y solicitan información. Esto se puede observar en los distintos sistemas informáticos que utilizan internet de forma cotidiana, por ejemplo instagram, donde las personas suben y eliminan fotos a cada instante y sus seguidores tienen permitido acceder a estos datos para visualizarlos en su dispositivo.

En esta actividad mediante el uso múltiples terminales o consolas, se revisarán los problemas que pueden ocurrir al realizar accesos simultáneos a la base de datos, se busca conocer las soluciones que da una base de datos concurrente como lo es PostgreSQL a cada escenario donde necesite gestionar el acceso.

1.1. Herramientas

El ordenador que se utiliza para esta actividad es el mismo que se utilizó para las anteriores, siendo poseedor de las siguientes especificaciones.

- Acer Aspire E-15-575G-76P4
- Sistema Operativo: Linux, con distribución Elementary Os versión Hera
- Intel core i7-7500U 4 cpus.
- SSD KINGSTON SA400 con capacidad 480gb
- Velocidad para lectura de 550 MB/s y escritura 450 MB/s.

Dicho hardware se mantiene en el desarrollo de los trabajos con la intención de tener datos relacionados entre ellos.

2. Desarrollo:

2.1. Conceptos importantes

Concurrencia:

Este concepto se refiere al hecho de que los Sistemas Administradores de Base de Datos permiten que muchas transacciones accedan a un mismo dato a la vez. Cuando existen varios usuarios intentando modificar los datos al mismo tiempo, se necesita establecer algún tipo de control para que dichas modificaciones de un usuario no interfieran en las de los otros, a este sistema se le denomina control de concurrencia.

El control de concurrencia es el encargado de administrar las distintas transacciones que se ejecutan en paralelo donde existe un potencial conflicto. Podemos definir que dos operaciones entran en conflicto si fueron realizadas por transacciones distintas, siendo una de ellas de escritura y ambas accediendo a la misma data.

Es importante tener presente que ninguna transacción debe ver el resultado de otras transacciones inconclusas, si esto no fuera así estaríamos leyendo datos inconsistentes, lo cual no puede ocurrir en una base de datos relacional ya que estas cumplen firmemente con el criterio ACID.

- Consistency: la transacción solo termina si la data es consistente.
- Isolation: la transacción es independiente de otras transacciones.
- Atomicity: todas las acciones en la transacción se cumplen o no se cumple ninguna.
- Durability: cuando la transacción termina el resultado de esta es perdurable.

Para tener control de estos conflictos, la base de datos utiliza bloqueos, estos pueden bloquear tanto una fila como una tabla, y pueden ser proporcionados por el usuario como por la misma base de datos de forma automática. Su principal función es proveer exclusividad generando tiempos de espera a las demás transacciones, y de esta manera regular los accesos a los datos con tal de cuidar la concurrencia de estos.

Si existen problemas en el control, pueden existir problemas como Deadlock o Livelock.

- Deadlock: Un interbloqueo ocurre cuando dos o más tareas se bloquean permanentemente entre sí porque cada tarea tiene un bloqueo en un recurso que las otras tareas están tratando de bloquear. En otras palabras, Deadlock es una situación en la que dos procesos, cada uno con un bloqueo en una pieza de datos, intentan adquirir esta característica en la pieza del otro. Cada proceso esperaría indefinidamente a que el otro libere la sección a la que quiere ingresar, a menos que uno de los procesos del usuario finalice.
- Livelock: Esta es una situación en la que una solicitud de bloqueo exclusivo se niega repetidamente, ya que muchos bloqueos compartidos superpuestos siguen interfiriendo entre sí. Los procesos siguen cambiando su estado, lo que les impide completar la tarea.

2.2. Pasos de la actividad

La presente actividad consta de una gran cantidad de pasos, realizados mediante dos consolas de forma simultánea (C1 y C2).

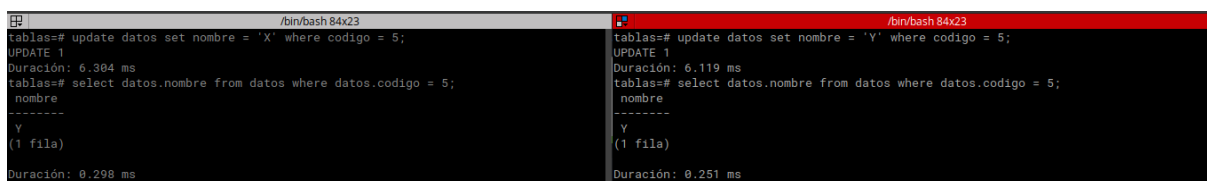
Paso 1:

En C1 se actualiza el NOMBRE de la fila de CÓDIGO 5.

En C2 se actualiza a un NOMBRE diferente del anterior el NOMBRE de la fila de CÓDIGO 5.

En C1 se consulta el NOMBRE de la fila de CÓDIGO 5.

En C2 se consulta el NOMBRE de la fila de CÓDIGO 5.



```

/bin/bash 84x23
tablas=# update datos set nombre = 'X' where codigo = 5;
UPDATE 1
Duración: 6.304 ms
tablas=# select datos.nombre from datos where datos.codigo = 5;
nombre
-----
X
(1 fila)
Duración: 0.298 ms

/bin/bash 84x23
tablas=# update datos set nombre = 'Y' where codigo = 5;
UPDATE 1
Duración: 6.119 ms
tablas=# select datos.nombre from datos where datos.codigo = 5;
nombre
-----
Y
(1 fila)
Duración: 0.251 ms
```

Figura 1: Segunda inserción, la tabla Personas1

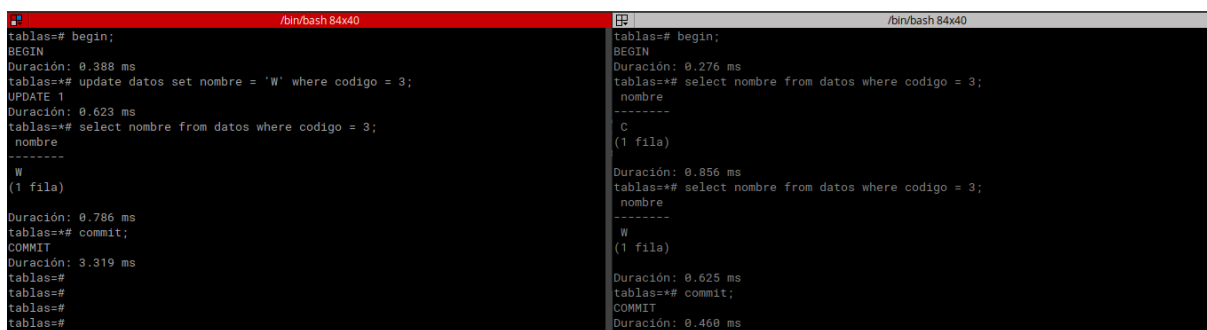
Paso 2:

En C1 se abre una transacción (BEGIN) y se actualiza el NOMBRE de la fila de CÓDIGO 3. Posterior se consulta el NOMBRE de la fila de CÓDIGO 3.

En C2 se abre una transacción (BEGIN) y se consulta el NOMBRE de la fila de CÓDIGO 3.

En C1 se cierra la transacción con COMMIT.

En C2 se vuelve a consultar el NOMBRE de la fila de CÓDIGO 3 y se cierra la transacción con COMMIT.



```

/bin/bash 84x40
tablas=# begin;
BEGIN
Duración: 0.388 ms
tablas== update datos set nombre = 'W' where codigo = 3;
UPDATE 1
Duración: 0.623 ms
tablas== select nombre from datos where codigo = 3;
nombre
-----
W
(1 fila)
Duración: 0.786 ms
tablas== commit;
COMMIT
Duración: 3.319 ms
tablas=#
tablas=#
tablas=#

/bin/bash 84x40
tablas=# begin;
BEGIN
Duración: 0.276 ms
tablas== select nombre from datos where codigo = 3;
nombre
-----
C
(1 fila)
Duración: 0.856 ms
tablas== select nombre from datos where codigo = 3;
nombre
-----
W
(1 fila)
Duración: 0.625 ms
tablas== commit;
COMMIT
Duración: 0.460 ms
```

Figura 2: Segunda inserción, la tabla Personas1

Paso 3:

En C1 se abre una transacción (BEGIN) y se actualiza el NOMBRE de la fila de CÓDIGO 7.

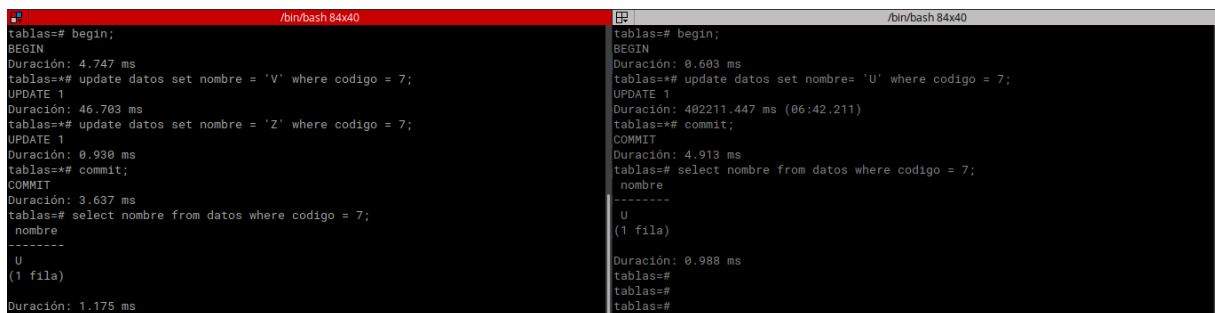
Actividad 5: Efectos de Concurrency

En C2 se abre una transacción (BEGIN) y se actualiza el NOMBRE de la fila de CÓDIGO 7 a un NOMBRE diferente.

En C1 se actualiza de nuevo el NOMBRE de la fila de CÓDIGO 7 con un NOMBRE diferente a los anteriores y se cierra la transacción con COMMIT.

En C2 se cierra la transacción con COMMIT.

Se consulta el NOMBRE de la fila de CÓDIGO 7 en ambas consolas.



```

+ /bin/bash 84x40
tablas=# begin;
BEGIN
Duración: 4.747 ms
tablas== update datos set nombre = 'V' where codigo = 7;
UPDATE 1
Duración: 46.703 ms
tablas== update datos set nombre = 'Z' where codigo = 7;
UPDATE 1
Duración: 0.930 ms
tablas== commit;
COMMIT
Duración: 3.637 ms
tablas=# select nombre from datos where codigo = 7;
nombre
-----
U
(1 fila)
Duración: 1.175 ms

+ /bin/bash 84x40
tablas=# begin;
BEGIN
Duración: 0.603 ms
tablas== update datos set nombre= 'U' where codigo = 7;
UPDATE 1
Duración: 402211.447 ms (06:42.211)
tablas== commit;
COMMIT
Duración: 4.913 ms
tablas=# select nombre from datos where codigo = 7;
nombre
-----
U
(1 fila)
Duración: 0.988 ms
tablas=#
tablas=#

```

Figura 3: Segunda inserción, la tabla Personas1

Paso 4:

En C1 se abre una transacción (BEGIN) y se actualiza el NOMBRE de la fila de CÓDIGO 4.

En C2 se abre una transacción (BEGIN) y se actualiza el NOMBRE de la fila de CÓDIGO 6.

En C1 se consulta por el NOMBRE de la fila de CÓDIGO 6.

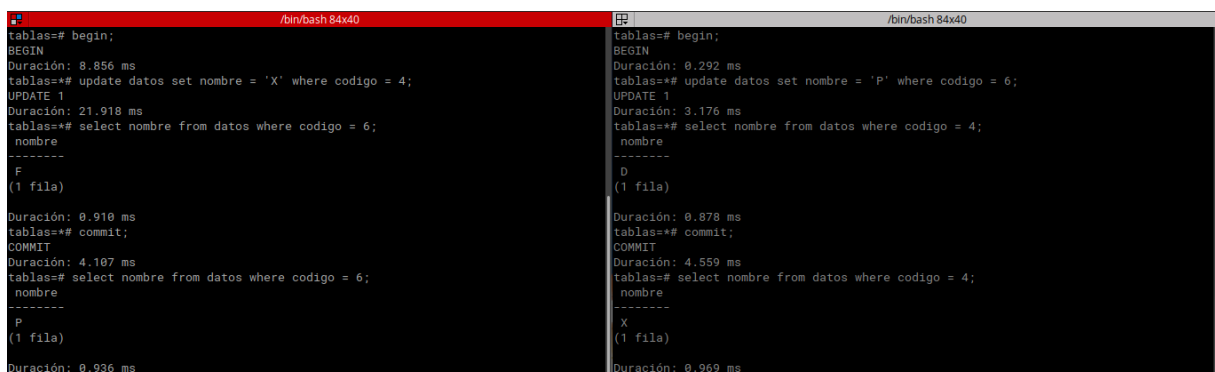
En C2 se consulta por el NOMBRE de la fila de CÓDIGO 4.

En C1 se cierra la transacción con COMMIT.

En C2 se cierra la transacción con COMMIT.

En C1 se consulta por el NOMBRE de la fila de CÓDIGO 6.

En C2 se consulta por el NOMBRE de la fila de CÓDIGO 4.



```

+ /bin/bash 84x40
tablas=# begin;
BEGIN
Duración: 8.856 ms
tablas== update datos set nombre = 'X' where codigo = 4;
UPDATE 1
Duración: 21.918 ms
tablas== select nombre from datos where codigo = 6;
nombre
-----
F
(1 fila)
Duración: 0.910 ms
tablas== commit;
COMMIT
Duración: 4.107 ms
tablas=# select nombre from datos where codigo = 6;
nombre
-----
P
(1 fila)
Duración: 0.936 ms

+ /bin/bash 84x40
tablas=# begin;
BEGIN
Duración: 0.292 ms
tablas== update datos set nombre = 'P' where codigo = 6;
UPDATE 1
Duración: 3.176 ms
tablas== select nombre from datos where codigo = 4;
nombre
-----
D
(1 fila)
Duración: 0.878 ms
tablas== commit;
COMMIT
Duración: 4.559 ms
tablas=# select nombre from datos where codigo = 4;
nombre
-----
X
(1 fila)
Duración: 0.969 ms

```

Figura 4: Segunda inserción, la tabla Personas1

Paso 5:

En C1 se abre una transacción (BEGIN) y se actualiza el NOMBRE de la fila de CÓDIGO 2.

En C2 se abre una transacción (BEGIN) y se actualiza el NOMBRE de la fila de CÓDIGO 8.

En C1 se actualiza el NOMBRE de la fila de CÓDIGO 8 a un NOMBRE diferente a los anteriores.

En C2 se actualiza el NOMBRE de la fila de CÓDIGO 2 a un NOMBRE diferente a los anteriores.

En C1 se cierra la transacción con COMMIT.

En C2 se cierra la transacción con COMMIT.

En C1 se consulta por el NOMBRE de la fila de CÓDIGO 8.

En C2 se consulta por el NOMBRE de la fila de CÓDIGO 2.

```

# /bin/bash 84x40
tablas=# begin;
BEGIN
Duración: 0.442 ms
tablas=# update datos set nombre = 'R' where codigo = 2;
UPDATE 1
Duración: 25.350 ms
tablas=# update datos set nombre = 'T' where codigo = 8;
UPDATE 1
Duración: 64935.192 ms (01:04.935)
tablas=# commit;
COMMIT
Duración: 4.082 ms
tablas=# select nombre from datos where codigo = 8;
-----
T
(1 fila)
Duración: 0.973 ms
tablas=#
tablas=#
tablas=#
tablas=#
tablas=#

# /bin/bash 84x40
tablas=# begin;
BEGIN
Duración: 0.361 ms
tablas=# update datos set nombre = 'S' where codigo = 8;
UPDATE 1
Duración: 3.103 ms
tablas=# update datos set nombre = 'O' where codigo = 2;
ERROR: se ha detectado un deadlock
DETALLE: El proceso 24474 espera ShareLock en transacción 1094; bloqueado por proceso 25628.
El proceso 25628 espera ShareLock en transacción 1095; bloqueado por proceso 24474.
SUGERENCIA: Vea el registro del servidor para obtener detalles de las consultas.
CONTEXTO: mientras se actualizaba la tupla (0,2) en la relación «datos»
Duración: 1007.899 ms (00:01.008)
tablas=# commit;
ROLLBACK
Duración: 0.508 ms
tablas=# select nombre from datos where codigo = 2;
-----
R
(1 fila)
Duración: 0.217 ms

```

Figura 5: Segunda inserción, la tabla Personas1

Paso 6:

En C1 se abre una transacción (BEGIN) y se actualiza el NOMBRE de la fila de CÓDIGO 1.

En C2 se abre una transacción (BEGIN) y se actualiza el NOMBRE de la fila de CÓDIGO 1 a un NOMBRE diferente.

En C1 se actualiza el NOMBRE de la fila de CÓDIGO 9 a un NOMBRE diferente a los anteriores.

En C2 se actualiza el NOMBRE de la fila de CÓDIGO 9 a un NOMBRE diferente a los anteriores.

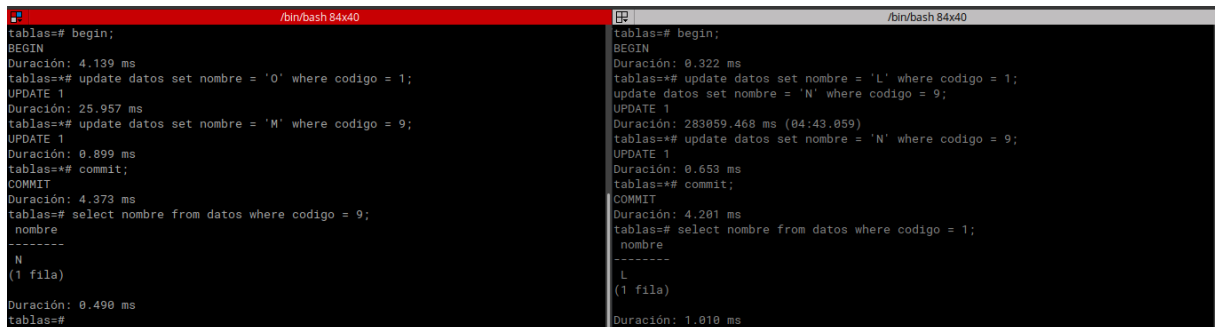
En C1 se cierra la transacción con COMMIT.

En C2 se cierra la transacción con COMMIT.

En C1 se consulta por el NOMBRE de la fila de CÓDIGO 9.

En C2 se consulta por el NOMBRE de la fila de CÓDIGO 1.

Actividad 5: Efectos de Concurrency



```

# /bin/bash 84x40
tablas=# begin;
BEGIN
Duración: 4.139 ms
tablas=# update datos set nombre = 'O' where codigo = 1;
UPDATE 1
Duración: 25.957 ms
tablas=# update datos set nombre = 'M' where codigo = 9;
UPDATE 1
Duración: 0.899 ms
tablas=# commit;
COMMIT
Duración: 4.373 ms
tablas=# select nombre from datos where codigo = 9;
nombre
-----
M
(1 fila)
Duración: 0.490 ms
tablas=#

# /bin/bash 84x40
tablas=# begin;
BEGIN
Duración: 0.322 ms
tablas=# update datos set nombre = 'L' where codigo = 1;
update datos set nombre = 'N' where codigo = 9;
UPDATE 1
Duración: 283059.468 ms (04:43.059)
tablas=# update datos set nombre = 'N' where codigo = 9;
UPDATE 1
Duración: 0.653 ms
tablas=# commit;
COMMIT
Duración: 4.201 ms
tablas=# select nombre from datos where codigo = 1;
nombre
-----
L
(1 fila)
Duración: 1.010 ms

```

Figura 6: Segunda inserción, la tabla Personas1

Paso 7:

En C1 se abre una transacción (BEGIN) y se consulta el NOMBRE de la fila de CÓDIGO 10 usando la opción de actualización (SELECT ... FOR UPDATE).

En C2 se consulta normalmente por el NOMBRE de la fila de CÓDIGO 10 y se actualiza con un NOMBRE diferente.

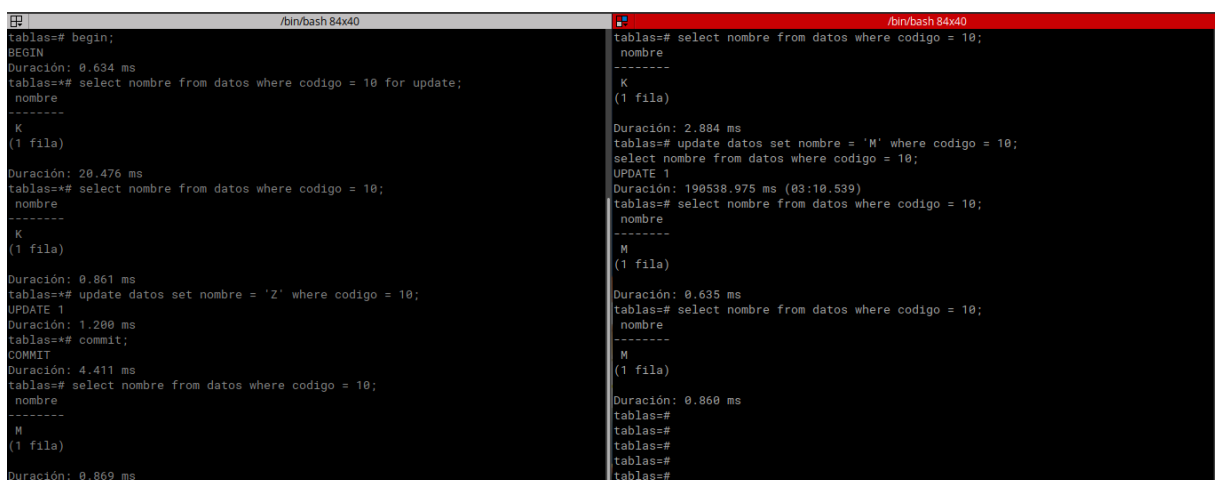
En C1 se consulta normalmente por el NOMBRE de la fila de CÓDIGO 10 y se actualiza con un NOMBRE diferente a los anteriores.

En C2 se consulta normalmente por el NOMBRE de la fila de CÓDIGO 10.

En C1 se cierra la transacción con COMMIT.

En C1 se consulta normalmente por el NOMBRE de la fila de CÓDIGO 10.

En C2 se consulta normalmente por el NOMBRE de la fila de CÓDIGO 10.



```

# /bin/bash 84x40
tablas=# begin;
BEGIN
Duración: 0.634 ms
tablas=# select nombre from datos where codigo = 10 for update;
nombre
-----
K
(1 fila)
Duración: 20.476 ms
tablas=# select nombre from datos where codigo = 10;
nombre
-----
K
(1 fila)
Duración: 0.861 ms
tablas=# update datos set nombre = 'Z' where codigo = 10;
UPDATE 1
Duración: 1.200 ms
tablas=# commit;
COMMIT
Duración: 4.411 ms
tablas=# select nombre from datos where codigo = 10;
nombre
-----
M
(1 fila)
Duración: 0.869 ms

# /bin/bash 84x40
tablas=# select nombre from datos where codigo = 10;
nombre
-----
K
(1 fila)
Duración: 2.884 ms
tablas=# update datos set nombre = 'M' where codigo = 10;
select nombre from datos where codigo = 10;
UPDATE 1
Duración: 190538.975 ms (03:10.539)
tablas=# select nombre from datos where codigo = 10;
nombre
-----
M
(1 fila)
Duración: 0.635 ms
tablas=# select nombre from datos where codigo = 10;
nombre
-----
M
(1 fila)
Duración: 0.860 ms
tablas=#
tablas=#
tablas=#
tablas=#

```

Figura 7: Segunda inserción, la tabla Personas1

2.3. Desarrollo de la actividad

2.3.1. Creación de entorno

```
1 create table datos(  
2     codigo numeric(2),  
3     nombre char(1)  
4 );  
5 insert into datos (codigo, nombre) values (1, 'A');  
6 insert into datos (codigo, nombre) values (2, 'B');  
7 insert into datos (codigo, nombre) values (3, 'C');  
8 insert into datos (codigo, nombre) values (4, 'D');  
9 insert into datos (codigo, nombre) values (5, 'E');  
10 insert into datos (codigo, nombre) values (6, 'F');  
11 insert into datos (codigo, nombre) values (7, 'G');  
12 insert into datos (codigo, nombre) values (8, 'H');  
13 insert into datos (codigo, nombre) values (9, 'J');  
14 insert into datos (codigo, nombre) values (10, 'K');
```

El primer paso esta actividad o taller fue crear el escenario, para ello fue creada la tabla “Datos”. Luego, fueron insertadas con “codigo” del 1 al 10 y “nombre” de la A a la J respectivamente.

Para finalizar con el espacio de trabajo, fueron inicializadas 2 consolas; en este caso C1 y C2, las cuales se conectaron a la base de datos.

Paso 1:

Para comenzar la actividad, en C1 se actualiza el NOMBRE, exactamente en la fila del CÓDIGO 5.

```
1 update datos set nombre = 'X' where codigo = 5;
```

A continuación, se realiza el mismo procedimiento para la segunda consola, con un NOMBRE distinto al anterior.

```
1 update datos set nombre = 'Y' where codigo = 5;
```

Por último se realiza la consulta de dicho NOMBRE en la fila de CÓDIGO 5.

```
1 select datos.nombre from datos where datos.codigo = 5;  
2 #Esta consulta se aplica en ambas consolas
```

Paso 2:

Siguiendo la linea, se tiene que en C1 se abre la transacción BEGIN y se vuelve actualizar el NOMBRE, ahora en la fila del CÓDIGO 3. Posteriormente se realiza la consulta del NOMBRE en dicha fila.

```
1 begin;  
2 update datos set nombre = 'W' where codigo = 3;  
3 select nombre from datos where codigo = 3;
```

Actividad 5: Efectos de Concurrency

A continuación, se abre la transacción BEGIN para la segunda consola y se consulta el NOMBRE en la fila del CÓDIGO 3.

```
1 begin;
2 select nombre from datos where codigo = 3;
```

Ya realizadas, se procede a cerrar la transacción de la C1.

```
1 commit;
```

En la segunda consola se vuelve a consultar el NOMBRE y posterior se cierra la transacción con COMMIT

```
1 select nombre from datos where codigo = 3;
2 commit;
```

Paso 3:

Para iniciar, en C1 fue abierta una transacción y dentro de esta fue actualizado el NOMBRE de la fila con “codigo” igual a 7.

```
1 begin;
2 update datos set nombre = 'V' where codigo = 7;
```

Luego, en C2 fue abierta una transacción y fue actualizado el nombre de la fila de CODIGO 7.

```
1 begin;
2 update datos set nombre = 'U' where codigo = 7;
```

Seguido de esto, en C1, por segunda vez fue actualizado el NOMBRE de la fila de CODIGO 7 a un nombre diferente a los anteriores y, fue cerrada la transacción:

```
1 update datos set nombre = 'Z' where codigo = 7;
2 commit;
```

En C2 fue cerrada la transacción:

```
1 commit;
```

Luego, en C1 y C2, se consulta el NOMBRE de la fila de CODIGO 7:

```
1 select nombre from datos where codigo = 7;
```

Paso 4:

En C1 se abre la transacción BEGIN y se actualiza el NOMBRE de la fila de CÓDIGO 4.

```
1 begin;
2 update datos set nombre= 'X' where codigo = 4;
```

A continuación en C2 se realiza el mismo procedimiento, pero ahora se actualiza el NOMBRE de la fila de CÓDIGO 6.

Actividad 5: Efectos de Concurrency

```
1 begin;
2 update datos set nombre= 'P' where codigo = 6;
```

Se realiza una consulta del NOMBRE en C1 de la fila CÓDIGO 6

```
1 select nombre from datos where codigo = 6;
```

Se realiza una consulta del NOMBRE en C2 de la fila CÓDIGO 4

```
1 select nombre from datos where codigo = 4;
```

Se cierran ambas transacciones en orden, primero C1 y posteriormente C2, mediante un COMMIT.

```
1 commit;
```

Por último se consulta por el NOMBRE en la fila, primero C1 de la fila CÓDIGO 6.

```
1 select nombre from datos where codigo = 6;
```

y segundo C2 de la fila CÓDIGO 4.

```
1 select nombre from datos where codigo = 4;
```

Paso 5:

Ahora en C1 fue abierta una transacción y se actualiza el NOMBRE de la fila de CODIGO 2.

```
1 begin;
2 update datos set nombre = 'R' where codigo = 2;
```

Y, en C2 se abre otra transacción, donde se actualiza el NOMBRE de la fila de CODIGO 8:

```
1 begin;
2 update datos set nombre = 'S' where codigo = 8;
```

Seguido de esto, en C1 se actualiza el NOMBRE de la fila de CODIGO 8:

```
1 update datos set nombre = 'T' where codigo = 8;
```

En C2 se ejecuta una sentencia similar a la anterior, pero en la fila de CODIGO 2:

```
1 update datos set nombre = 'O' where codigo = 2;
```

Se cierran las transacciones tanto en C1 como en C2 con COMMIT:

```
1 commit;
```

Posterior a esto, en C1 fue consultado el “nombre” de la fila de “codigo” 8:

```
1 select nombre from datos where codigo = 8;
```

Y, en C2 fue consultado el “nombre” de la fila de “codigo” 2:

```
1 select nombre from datos where codigo = 2;
```

Para finalizar, fue registrado lo ocurrido en ambas consolas.

Paso 6: C1 parte abriendo la transacción BEGIN y se le actualiza el NOMBRE de la fila CÓDIGO 1

```
1 begin;
2 update datos set nombre = 'O' where codigo = 1;
```

C2 Realiza el mismo procedimiento anteriormente mencionado

```
1 begin;
2 update datos set nombre = 'L' where codigo = 1;
```

C1 se le actualiza el NOMBRE ahora de la fila CÓDIGO 9, con un NOMBRE diferente de los anteriores

```
1 update datos set nombre = 'M' where codigo = 9;
```

C2 se hace el mismo procedimiento

```
1 update datos set nombre = 'N' where codigo = 9;
```

Se cierra la transacción en ambas consolas mediante COMMIT, primero C1 después C2.

```
1 commit;
```

Se consulta por el nombre en cada una de las consolas, para C1 se tiene

```
1 select nombre from datos where codigo = 9;
```

Ejecutar la misma sentencia para C2 pero con CODIGO 1

```
1 select nombre from datos where codigo = 1;
```

Paso 7: En C1 se abre la transacción BEGIN, posterior se realiza la consulta del NOMBRE en la fila de CÓDIGO 10, esta vez mediante la sentencia de actualización (SELECT 'consulta' FOR UPDATE)

```
1 begin;
2 select nombre from datos where codigo = 10 for update;
```

Ahora en C2 se realiza una consulta por el NOMBRE de la fila CÓDIGO 10 y se actualiza el NOMBRE por uno diferente.

```
1 select nombre from datos where codigo = 10;
2 update datos set nombre = 'M' where codigo = 10;
```

Ahora se ejecuta el mismo paso anterior pero en C1

```
1 select nombre from datos where codigo = 10;
2 update datos set nombre = 'Z' where codigo = 10;
```

Se realiza nuevamente una consulta simple en C2 por NOMBRE en la fila CÓDIGO 10

Actividad 5: Efectos de Concurrency

```
1 select nombre from datos where codigo = 10;
```

Se cierra la transacción de C1 mediante un COMMIT, y se realiza una consulta SELECT por el NOMBRE en la fila de CÓDIGO 10

```
1 commit;  
2 select nombre from datos where codigo = 10;
```

Finalmente, en C2 se realiza la última consulta por el nombre con CODIGO 10

```
1 select nombre from datos where codigo = 10;
```

3. Análisis de resultados

Caso 1:

Como se puede apreciar en la imagen (ver Desarrollo, Pasos de la actividad, Figura 1, página 5), se realiza una actualización en C1 del nombre según el código proporcionado; Luego ocurre que al momento de ejecutar el mismo proceso en C2, el segundo UPDATE se sobrepone al de la terminal 1 de forma instantánea.

Caso 2:

Se puede observar en la figura (ver Desarrollo, Pasos de la actividad, Figura 2, página 5), que al hacer la consulta por el nombre asociado al código 3. Luego de haber hecho un UPDATE en C1, y ambas terminales estando en una transacción; El SELECT en C2 no toma el valor actualizado al momento de ejecutar la consulta. Pero al momento de cerrar la transacción en C1 con COMMIT y realizar nuevamente un SELECT en C2, el valor se reescribe. Se puede decir entonces, que solo hasta cerrar la transacción se efectúan los cambios en ambas consolas.

Caso 3:

Como se puede apreciar en las transacciones de la imagen (ver Desarrollo, Pasos de la actividad, Figura 3, página 6), las consolas realizan transacciones y actualizaciones correspondientes en la misma fila (CÓDIGO 7), teniendo en cuenta que C1 lleva a cabo dos reescrituras de datos, y luego cierra su transacción. Al realizar un COMMIT en C2, el dato implicado mediante el UPDATE de dicha consola (C2), es el que prevalece en la fila (CÓDIGO 7) y esto se refleja en la consulta. De esta forma se concluye que la segunda transacción empleada es quien tiene la prioridad al efectuar último el COMMIT.

Caso 4:

Para el caso de la figura (ver Desarrollo, Pasos de la actividad, Figura 4, página 6) se aprecia un orden cronológico al momento de actualizar los nombres procedentes de las filas con códigos 4 y 6 (en C1 y en C2), en transacciones distintas. Posterior a ello se consulta a través de SELECT y se observa que la generación de transacciones produce operaciones independientes entre ellas. Se puede ver los datos actualizados en forma paralela por parte de los usuarios (C1 y C2). Tomando la última sentencia producida previo al cierre del COMMIT, en ambas terminales (C1 y C2).

Caso 5:

Para el caso de la figura (ver Desarrollo, Pasos de la actividad, Figura 5, página 7), una vez realizados los pasos cuidadosamente, fue posible apreciar un mensaje de error en C2, ya que, se está modificando datos en forma cruzada tanto en C1 como en C2 a los nombres con CODIGOS 2 y 8; generando así un punto muerto o *DEADLOCK*. Puesto que ambas terminales compiten por los mismos recursos llegando así a un interbloqueo en ambas consolas.

Caso 6:

Tras ver el comportamiento del sistema cuando un usuario (C1), ingresa una petición (ver Desarrollo, Pasos de la actividad, Figura 6, página 8), mientras que el otro cliente (C2), espera para modificar algún valor. Ya que, surge un deadlock que restringe el UPDATE o modificación de la fila con CODIGO 1. Como resultado de esto, al ingresar queries en C2 estas quedan en espera hasta que C1 termine su transacción y el punto muerto sea levantado.

Caso 7:

Para el caso final (ver Desarrollo, Pasos de la actividad, Figura 7, página 8), el uso de la sentencia `SELECT ... FOR UPDATE` bloquea el registro, con el fin de que la transacción se completa, luego se decide si se confirma la actualización o si también la revierte. Esto ocurre al momento de iniciar las transacciones mediante `BEGIN`, en caso contrario el bloqueo se levanta de inmediato. Por otro lado, en C2 ocurre algo similar al caso anterior, donde espera que el proceso en C1 finalice, y así se ejecuta. Finalmente al consultar el dato, se observa que resulta el ingresado en C2, debido a que esta es la actualización post `COMMIT` de C1.

4. Conclusión

El correcto control de concurrencias permite mantener información consistente en las bases de datos, así como también evitar la aparición de errores en las recuperaciones y respaldos que se realicen en esta.

Los problemas se presentan en el instante que la concurrencia no se controla, o las distintas transacciones se imponen ante un control insuficiente, se puede concluir que las bases de datos tradicionales como PostgreSQL mantienen un modelo de bloqueo para evitar la existencia de conflictos. En caso de que el conflicto supere el control y se genere un deadlock u otro tipo de bloqueo excepcional, la base de datos seguirá con su propiedad de concurrencia.

Gracias a lo indicado anteriormente se concluye que los objetivos propuestos para la presente actividad, fueron cumplidos.

5. Bibliografía

- Información relacionada a Livelock and Deadlock.
<https://www.c-sharpcorner.com/blogs/sql-server-deadlock-vs-livelock1>
Página consultada el 18 de marzo del 2021.
- Información relacionada a bloqueos Bloqueos.
<https://www.guru99.com/what-is-livelock-example.html>
Página consultada el 18 de marzo del 2021.