Universidad Nacional del Nordeste

Facultad de Ciencias Exactas, Naturales y Agrimensura

Carrera: Licenciatura en Sistemas de Información
Materia: Base de datos I
Profesor: Walter Oscar Vallejos
Año: 2023

Conceptos del Manejo de transacciones y transacciones anidadas

Integrantes grupo N°2:
Roch, Ignacio Fernando – 41612299
Mieres, Julian Edgardo – 41442242
Sinatra Marisa Florencia Isabel – 42203243
Valenzuela Luciano Juan Ignacio – 43346145

ÍNDICE

- **INTRODUCCIÓN ……..….. 2**
- **MARCO CONCEPTUAL o REFERENCIAL …… 3**
- **METODOLOGÍA ……4**
- **DESARROLLO DEL TEMA ……....6**
- **RESULTADOS …… .11**
- **INTEGRACIÓN DE OTROS TEMAS ……17**
- **CONCLUSIÓN ……...19**
- **BIBLIOGRAFÍA ……20**

INTRODUCCIÓN

El siguiente trabajo de investigación busca comprender y familiarizarse con los conceptos referidos al manejo de **transacciones** simples y anidadas, que son parte fundamental de una buena gestión de bases de datos.

Para ello nos debemos hacer una serie de preguntas, las cuales buscaremos esclarecer a los largo del proceso investigativo en el actual informe. Las preguntas en cuestión deben indagar a fondo lo que hay que plantearse para unas mejor comprensión del tema, estas pueden ser:
¿Qué significa la palabra transacción?, ¿Que significado se le aplica a la palabra transacción en las bases de datos?, ¿Cuál es su importancia en base de datos?, ¿Cuáles son sus características?, ¿Qué ejemplos podemos obtener de una transacción? y ¿Que casos de usos podemos evidenciar?.

A los largo del siguiente material, se podrá ir entendiendo mucho mejor el tema a desarrollar, las cuestiones que surgieron durante el proceso de búsqueda de información, los planteamientos y aplicaciones con sus respectivos resultados, las cuales brindaran una apreciación más práctica de las transacciones.
Desde el comienzo ya planteamos un marco conceptual sobre el cual trabajar, la metodología de la cual haremos uso para alcanzar los objetivos. Para llegar al punto de ir desarrollando cada concepto que se necesita saber y poder tener una visión más amplia y concreta de las transacciones, y poder llegar por último a una conclusión grupal que exprese nuestras sensaciones al final del proceso investigativo.

MARCO CONCEPTUAL o REFERENCIAL

Fundamentamos esta investigación con los principales conceptos de una base de datos. Para ello analizaremos la fundamentación de la misma, las ventajas que estas conllevan y los conceptos que nos permitieron verificar el modelado final.
Base de datos, el concepto surge de la necesidad de organizar y almacenar grandes cantidades de información para después realizar consultas de forma eficaz. En la actualidad las industrias generan una gran cantidad de información la cual debe poderse analizar y modificar en su caso, de manera rápida y accesible. Sistemas de Bases de Datos, es una forma de organización que a través de una computadora pueda contener registros, es decir una colección de archivos de datos con los cuales se puedan realizar múltiples operaciones. El registro de los datos queda de forma permanente en varios archivos y se utilizan diversas aplicaciones para extraer los registros o consultas o añadir registros a los archivos adecuados.
Ventajas del uso de una Base de Datos:

- Se puede acceder a cualquier dato en todo momento.
- Es integra.
- Evita la duplicidad de registros, es no redundante.
- Integridad referencial, elimina todos los registros relacionados dependientes.
- Define unívocamente a todos los demás atributos de la tabla a través de una llave primaria.
- Establece la relación existente en dos tablas a través de una llave foránea.
- Favorece la normalización por ser más comprensible y aplicable.
- Factores de seguridad.
- Está organizada en tablas.
- Poseen una llave primaria.

Nuestro caso de estudio es “Manejo de transacciones y transacciones anidadas” aplicados, por lo que fundamentamos cada detalle con los conceptos teóricos de transacciones que vayamos encontrando(internet, bibliografía de la cátedra o recomendada) y buscando comprender a lo largo del desarrollo del presente informe, y poder aplicarlo de la mejor manera a un caso práctico, y al final tener un mejor conocimiento el tema.

METODOLOGÍA

Para este proyecto decidimos aplicar la metodología **SCRUM** porque es la metodología más utilizada en las empresas, pues es la que nos permite ser más rápidos, eficientes y productivos ya que está enfocada directamente en las tareas a realizar, en la interacción y colaboración de los equipos de trabajo (el trabajo colaborativo con esta metodología es fundamental).
En SCRUM se trabaja con Sprints, es decir, el proyecto lo dividimos en pequeñas partes para poder abordarlas de forma más rápida y eficiente.
Para cumplir con el objetivo del proyecto abordaremos el mismo dividiéndolo en 2 *sprints* siguiendo una serie de fases para abordar cada tarea.
Se dividió en cuatro fases:

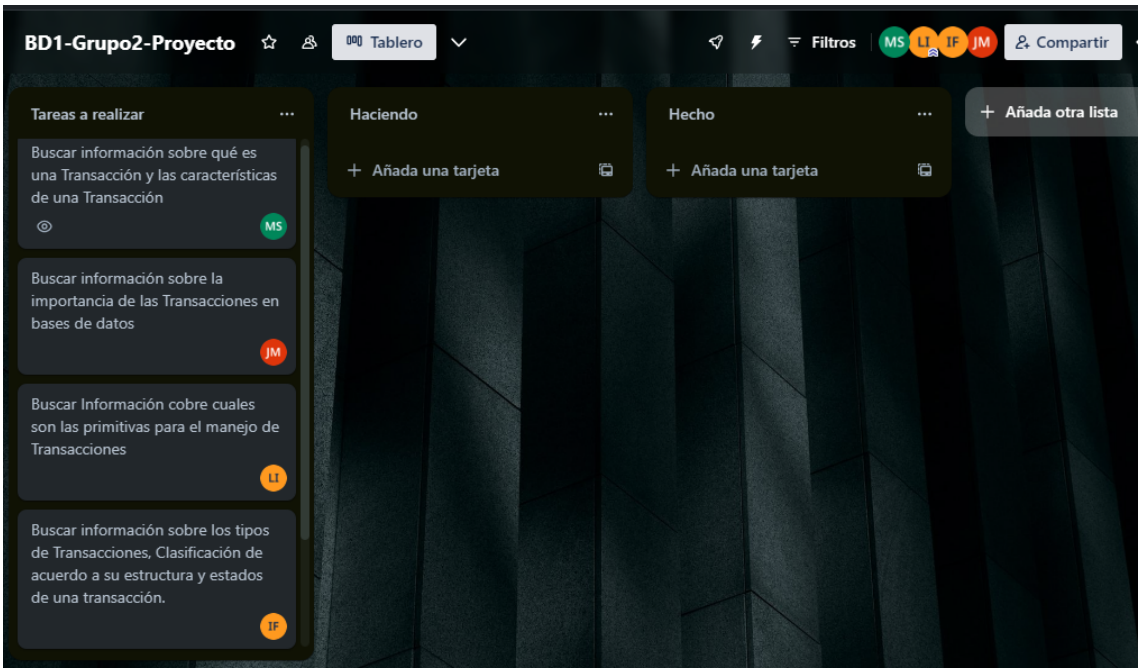
1. **Sprint planning**. La planificación del sprint es la primera fase de SCRUM donde se describe qué tareas se asignan a cada miembro del grupo de trabajo, así como el tiempo que necesita para concluirse.
2. **Scrum team meeting**. Suelen ser diarias y cortas, donde evaluamos el trabajo realizado, el que se va a abordar en el día y qué problemas se han presentado o se intuye que se van a presentar.
3. **Backlog refinement**. Es un repaso de las tareas con el fin de evaluar el tiempo y esfuerzo empleado en cada tarea y para resolver cualquier inconveniente encontrado en el camino.
4. **Sprint Review**. Son reuniones que tienen como objetivo mostrar los resultados obtenidos, y así poder conseguir un feedback más real.

Herramientas utilizadas:

- Creación de base de datos:
  - SQL Management Studio
- Administración de la base de datos:
  - SQL Management Studio
- Interacción del grupo y coordinación para el desarrollo del proyecto:
  - Whatsapp
  - Google Drive
  - Github
  - Google meet
  - Trello
- Bibliografía utilizada para el desarrollo:
  - Material teórico de la plataforma Moodle la materia de Base de Datos I
  - Transact – SQL

Sprint n°1

1. **Sprint planning - Desarrollo del tema**. Búsqueda de información detallada sobre los siguientes temas: transacción, características de una transacción, importancia de las transacciones en bases de datos, primitivas para el manejo de transacciones, tipos de Transacciones Clasificación de acuerdo a su estructura y estados de una transacción.
2. **Scrum team meeting**. Nos reunimos vía Google meet y evaluamos la información encontrada.
3. **Backlog refinement**. Utilizamos la herramienta Trello para poder organizar y asignarnos las tareas a realizar y para que cada integrante del equipo pueda ver en que estado se encuentra cada tarea.

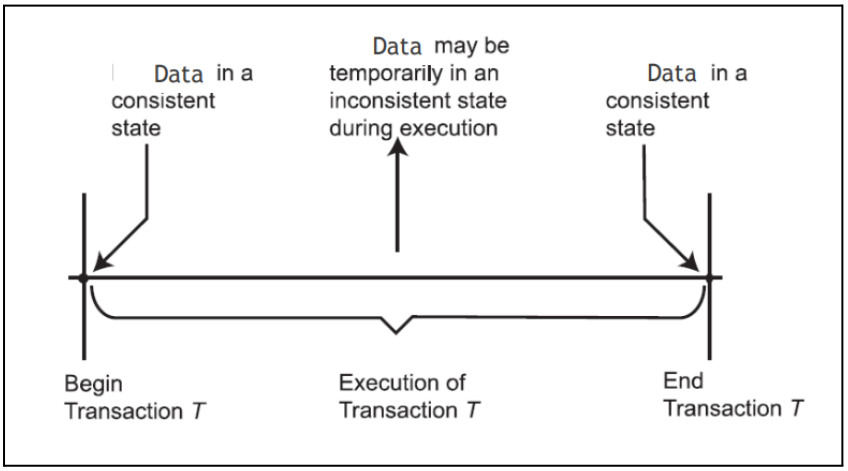


4. **Sprint Review**. Nos reunimos nuevamente vía Google meet para visualizar los resultados.

DESARROLLO DEL TEMA

¿Una pregunta que debemos hacernos es cómo se gestionan los datos en las bases de datos? Existen diferentes puntos que desembocan en una buena gestión de las mismas y las transacciones en base de datos son una parte fundamental de este proceso.
Una **transacción** en base de datos es una secuencia de operaciones que se realizan de manera indivisible, operaciones agrupadas como una unidad. Es como un paquete que contiene múltiples acciones que deben completarse en su totalidad o deshacerse por completo. Las operaciones que contiene una transacción se van almacenando temporalmente, no a nivel de disco. Es hasta que termina la transacción que se tienen efecto de manera permanente o no. Esto asegura que los datos se mantengan consistentes y evita problemas en caso de errores o fallos en el sistema.

Condiciones de terminación
Una transacción aplica a datos recuperables, puede estar formada por operaciones simples o compuestas y su intención es que sea atómica.



Una transacción siempre termina, aun en la presencia de fallas. Si una transacción termina de manera exitosa se dice que la transacción hace un **commit** (consumación).
Si la transacción se detiene sin terminar su tarea, se dice que la transacción aborta. Cuando la transacción es abortada, su ejecución se detiene y todas las acciones ejecutadas hasta el momento se deshacen (undone) regresando a la base de datos al estado antes de su ejecución. A esta operación también se le conoce como **rollback**

Características de una transacción
Una transacción en base de datos debe cumplir con las siguientes características, conocidas como las propiedades ACID:

- **Atomicidad**: Una transacción se considera atómica, lo que significa que se ejecuta como una unidad completa o no se ejecuta en absoluto. Si alguna de las operaciones falla, se deshacen todas las operaciones anteriores, asegurando que los datos no queden en un estado inconsistente.

- **Consistencia**: Una transacción debe llevar la base de datos desde un estado válido a otro estado válido. Esto garantiza que se respeten las restricciones y reglas definidas para los datos.
- **Aislamiento**: Cada transacción se ejecuta de manera aislada y no se ve afectada por otras transacciones concurrentes. Esto evita problemas de concurrencia y garantiza la integridad de los datos.
- **Durabilidad**: Una vez que una transacción se ha completado correctamente, los cambios realizados en la base de datos se mantienen permanentemente, incluso en caso de fallos del sistema. Los datos actualizados son duraderos y no se perderán.

Un ejemplo que podemos analizar es, imaginar que estás realizando una compra en línea. Cuando seleccionas los productos y procedes al pago, se lleva a cabo una transacción en la base de datos. En este caso, la transacción incluye operaciones como reducir el inventario de los productos comprados, registrar la transacción en la cuenta del cliente y generar un recibo de compra. Si algo falla durante este proceso, todas las operaciones se deshacen para evitar inconsistencias.

Importancia de las transacciones en bases de datos

Las transacciones son vitales en entornos donde múltiples usuarios pueden acceder y modificar la misma información al mismo tiempo. Garantizan la integridad y consistencia de los datos, evitando conflictos y asegurando que los cambios se realicen de manera segura.

Casos de uso de las transacciones

Las transacciones son utilizadas en una amplia gama de aplicaciones y sistemas, como:

- **Sistemas bancarios:** Cuando realizas una transferencia de dinero en línea, se utiliza una transacción para asegurar que el dinero se deduzca de una cuenta y se acredite en otra de manera precisa.
- **Sistemas de reserva:** Al reservar un vuelo o una habitación de hotel, se utiliza una transacción para garantizar que la disponibilidad se actualice correctamente y que no se realicen reservas duplicadas.
- **Sistemas de gestión de inventarios:** Cuando se registra una venta o se actualiza el inventario de productos, una transacción asegura que los datos se actualicen de manera coherente.

Primitivas para el manejo de transacciones

- **BEGIN TRAN**: comienza una transacción y aumenta en 1 @@TRANCOUNT.
- **COMMIT TRAN**: reduce en 1 @@TRANCOUNT y si @@TRANCOUNT llega a 0 guarda la transacción.

- **ROLLBACK TRAN**: deshace la transacción actual, o si estamos en transacciones anidadas deshace la más externa y todas las internas. Además pone @@TRANCOUNT a 0.
- **SAVE TRAN**: guarda un punto (con nombre) al que podemos volver con un ROLLBACK TRAN si es que estamos en transacciones anidadas y no queremos deshacer todo hasta la más externa.
- **ABORT_TRANSACTION:** deshacer operación.

Tipos de Transacciones Clasificación de acuerdo a su estructura

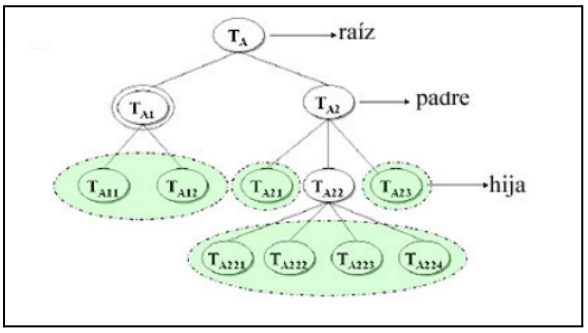
Transacciones planas: Estas transacciones tienen un punto de partida simple (Begin_transaction) y un punto simple de terminación (End_transaction).

```
Begin_transaction RESERVAR
begin
        EXEC SQL   UPDATE  cuentas  SET saldo=saldo-1000  WHERE  id_cuenta=1111
        EXEC SQL   UPDATE  cuentas  SET saldo=saldo+1000  WHERE  id_cuenta=222
end
```

Transacciones anidadas: las operaciones de una transacción anidada pueden incluir otras transacciones.

```
BeginTransaction Reservación
        BeginTransaction Vuelo
    ...
        EndTransaction {Vuelo}
        BeginTransaction Hotel
        ...
        endTransaction {Hotel}
      BeginTransaction Car
        ...
        endTransaction {Car}

EndTransaction {Reservación}
```

Una transacción anidada dentro de otra transacción conserva las mismas propiedades que la de sus padres, esto implica, que puede contener así mismo transacciones dentro de ella.

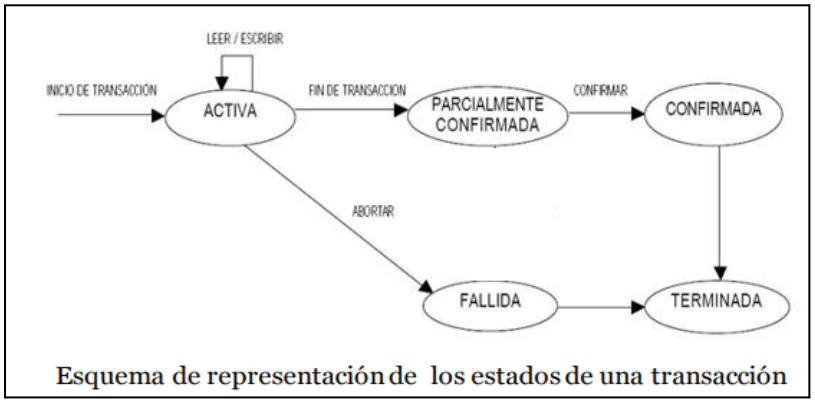


Existen **restricciones** para una transacción anidada:

- Debe empezar después que su padre y debe terminar antes que él.
- El commit de una transacción padre está condicionada al commit de sus transacciones hijas.
- Si alguna transacción hija aborta (rollback), la transacción padre también será abortada (rollback).

Estados de una transacción

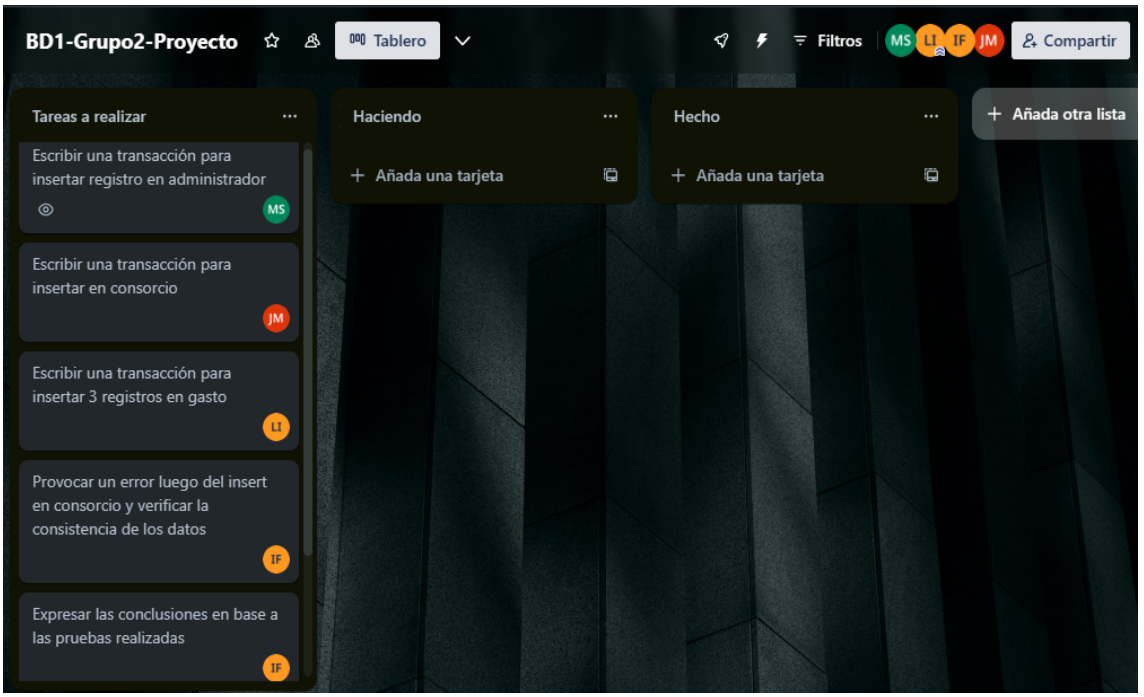
- **Transacción Activa**: se encuentra en este estado justo después de iniciar su ejecución.
- **Transacción Parcialmente Confirmada**: en este punto, se han realizado las operaciones de la transacción pero no han sido almacenados de manera permanente.
- **Transacción Confirmada**: Ha concluido su ejecución con éxito y se almacenan de manera permanente.
- **Transacción Fallida**: En este caso, es posible que la transacción deba ser cancelada.
- **Transacción Terminada**: indica que la transacción ha abandonado el sistema.



Esquema de representación de los estados de una transacción

Sprint n°2

1. **Sprint planning - Resultados**. Escribir una transacción para insertar registro en administrador, escribir una transacción para insertar en consorcio, escribir una transacción para insertar 3 registros en gasto, provocar un error luego del insert en consorcio y verificar la consistencia de los datos y expresar las conclusiones en base a las pruebas realizadas.
2. **Scrum team meeting**. Nos reunimos vía Google meet y evaluamos la creación de las transacciones y los casos de pruebas.
3. **Backlog refinement**. Utilizamos la herramienta Trello para poder organizar y asignarnos las tareas a realizar y para que cada integrante del equipo pueda ver en que estado se encuentra cada tarea.



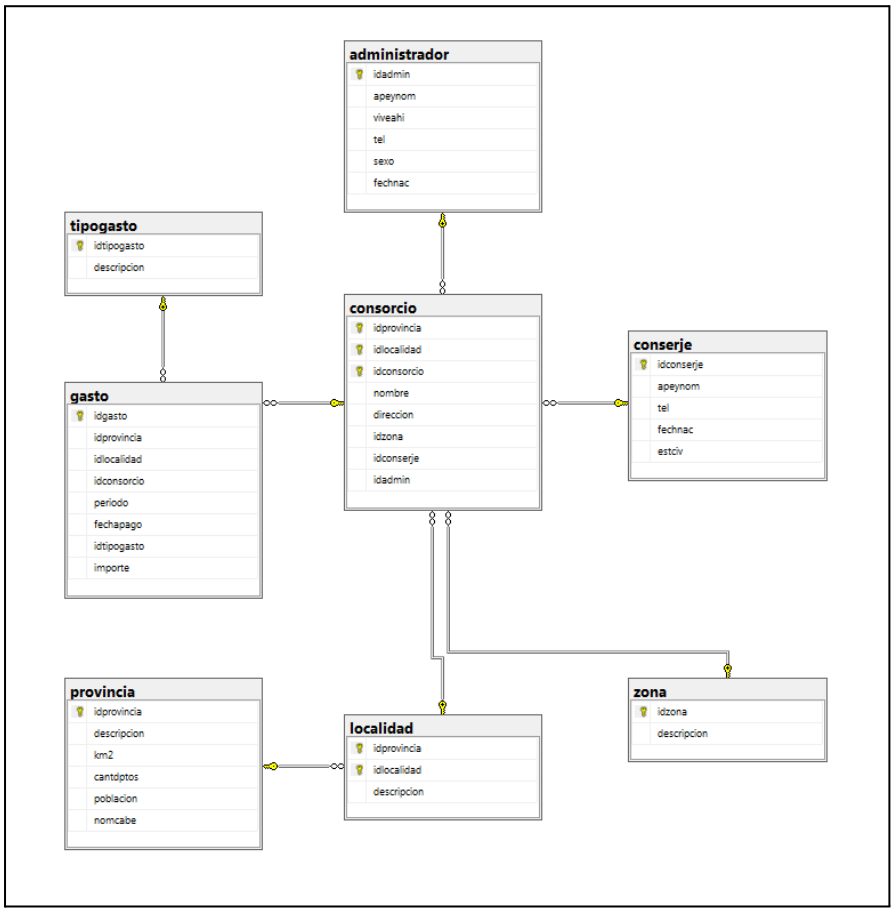
4. **Sprint Review**. Nos reunimos nuevamente vía Google meet para visualizar los resultados.

RESULTADOS

Desarrollo del tema en base al modelo asignado

Llegado a este punto ya tenemos más en claro cuál es la funcionalidad de las transacciones y por qué cumplen una función importante en la gestión de bases de datos. Al manejar de manera más clara los conceptos básicos de este tema, pudimos generar una serie de pruebas en base a las tareas asignadas específicamente.

Modelado físico de la base de datos Consorcio:



Tareas especificadas para ejemplificar de manera práctica una transacción:

- Escribir el código SQL que permita definir una transacción consistente en: Insertar un registro en Administrador, luego otro registro en consorcio y por último 3 registros en gasto, correspondiente a ese nuevo consorcio. Actualizar los datos solamente si toda la operación es completada con éxito.

- Sobre el código escrito anteriormente provocar intencionalmente un error luego del insert en consorcio y verificar que los datos queden consistentes (No se debería realizar ningún insert).

Prueba caso de error en transacción

Luego de seguir las instrucciones anteriores, se llegó al siguiente Script, donde se produce de manera intencional un fallo dentro de la transacción, lo que deshace toda modificación realizada, protegiendo así la integridad de los datos.

```
-- USE base_consorcio;

-- CREACION DE TRANSACCIONES
BEGIN TRY -- INICIAMOS EL BEGIN TRY PARA LUEGO COLOCAR LA LOGICA DENTRO Y ASEGURARNOS DE QUE SI ALGO FALLA IRA POR EL CATCH
    BEGIN TRAN -- COMENZAMOS LA TRANSACCION
    INSERT INTO administrador(apeynom, viveahi, tel, sexo, fechnac) -- UN INSERT A LA TABLA QUE QUEREMOS
    VALUES ('pablito clavounclavito', 'S', '37942222', 'M', '01/01/1996') -- LOS VALORES A INGRESAR A LA TABLA

    INSERT INTO consorcio(idprovincia, idlocalidad, idconsorcio, nombre, direccion, idzona, idconserje, idadmin)
    VALUES (999, 1, 1, 'EDIFICIO-111', 'PARAGUAY N 999', 5, 100, 1) -- GENERAMOS UN ERROR INGRESANDO EL ID PROVINCIA 999 QUE NO EXISTE.

    INSERT INTO gasto(idprovincia, idlocalidad, idconsorcio, periodo, fechapago, idtipogasto, importe)
    VALUES (1,1,1,6,'20130616',5,608.97)
    INSERT INTO gasto(idprovincia, idlocalidad, idconsorcio, periodo, fechapago, idtipogasto, importe)
    VALUES (1,1,1,6,'20130616',2,608.97)
    INSERT INTO gasto(idprovincia, idlocalidad, idconsorcio, periodo, fechapago, idtipogasto, importe)
    VALUES (1,1,1,6,'20130616',3,608.97)

    COMMIT TRAN -- SI TODO FUE EXITOSO FINALIZAMOS LA TRANSACCION
END TRY
BEGIN CATCH -- SI ALGO FALLA VENDRA AQUI
    SELECT ERROR_MESSAGE() -- MOSTRAMOS EL MENSAJE DE ERROR
    ROLLBACK TRAN -- VOLVEMOS HACIA ATRAS PARA MANTENER LA CONSISTENCIA DE LOS DATOS
END CATCH

```

Como se puede apreciar el resultado fue un error, ocasionando que se realice un rollback de la transacción completa y volver a un estado inicial. Al realizar una consulta para controlar que los registros específicos no habían sido cargados por la cancelación de la restricción, como se puede ver, la transacción hizo rollback y no se produjeron modificaciones.

Results | Messages

	(No column name)
1	The INSERT statement conflicted with the FOREIGN KEY constraint "FK_consorcio_pcia". The conflict occurred in database "base_consorcio", table "dbo.localidad".

106 %

Results | Messages

	idadmin	apeynom	viveahi	tel	sexo	fechnac
1	174	ARAUJO GUILLERMO JOSE	S	3672235689	M	1985-10-13 00:00:00.000

idprovincia	idlocalidad	idconsorcio	nombre	direccion	idzona	idconserje	idadmin

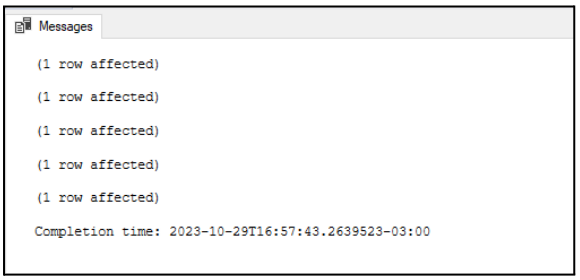
	idgasto	idprovincia	idlocalidad	idconsorcio	periodo	fechapago	idtipogasto	importe
1	8000	24	17	6	7	2017-07-02 00:00:00.000	5	869.17
2	7999	24	17	6	1	2017-01-01 00:00:00.000	2	1424.34
3	7998	24	17	6	8	2017-08-18 00:00:00.000	5	107.52

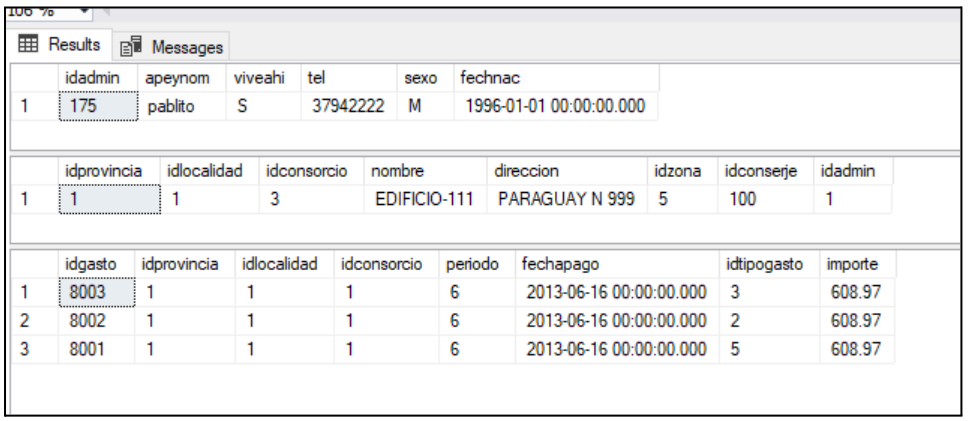
Prueba transacción terminada

Luego de verificar que en efecto al momento de una detección de error, la transacción queda cancelada, para el siguiente script lo único que hicimos fue corregir ese error en consorcio y el resultado al ejecutar fue el siguiente.

```
--- CASO Transacción Terminada
-- USE base_consorcio;

-- CREACION DE TRANSACCIONES
BEGIN TRY -- INICIAMOS EL BEGIN TRY PARA LUEGO COLOCAR LA LOGICA DENTRO Y ASEGURARNOS DE QUE SI ALGO FALLA IRA POR EL CATCH
    BEGIN TRAN -- COMENZAMOS LA TRANSACCION
    INSERT INTO administrador(apeynom, viveahi, tel, sexo, fechnac) -- UN INSERT A LA TABLA QUE QUEREMOS
    VALUES ('pablito clavounclavito', 'S', '37942222', 'M', '01/01/1996') -- LOS VALORES A INGRESAR A LA TABLA

    INSERT INTO consorcio(idprovincia, idlocalidad, idconsorcio, nombre, direccion, idzona, idconserje, idadmin)
    VALUES (1, 1, 3, 'EDIFICIO-111', 'PARAGUAY N 999', 5, 100, 1) -- AHORA INGRESAMOS EL CONSORCIO SIN INTENCION DE ERROR.
    INSERT INTO gasto(idprovincia, idlocalidad, idconsorcio, periodo, fechapago, idtipogasto, importe)
    VALUES (1,1,1,6,'20130616',5,608.97)
    INSERT INTO gasto(idprovincia, idlocalidad, idconsorcio, periodo, fechapago, idtipogasto, importe)
    VALUES (1,1,1,6,'20130616',2,608.97)
    INSERT INTO gasto(idprovincia, idlocalidad, idconsorcio, periodo, fechapago, idtipogasto, importe)
    VALUES (1,1,1,6,'20130616',3,608.97)

    COMMIT TRAN -- SI TODO FUE EXITOSO FINALIZAMOS LA TRANSACCION
END TRY
BEGIN CATCH -- SI ALGO FALLA VENDRA AQUI
    SELECT ERROR_MESSAGE() -- MOSTRAMOS EL MENSAJE DE ERROR
    ROLLBACK TRAN -- VOLVEMOS HACIA ATRAS PARA MANTENER LA CONSISTENCIA DE LOS DATOS
END CATCH
```

Cada proceso se realizó sin problemas y la transacción en conjunto fue realizada con éxito. Al verificar con una consulta que los registros fueron cargados, pudimos confirmar que los datos fueron cargados correctamente.





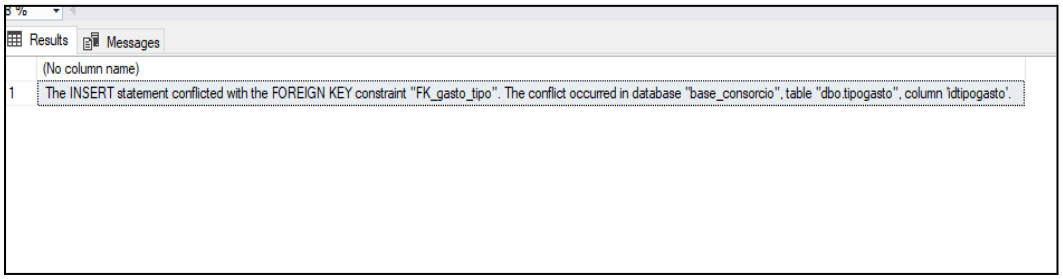
Prueba transacción anidada

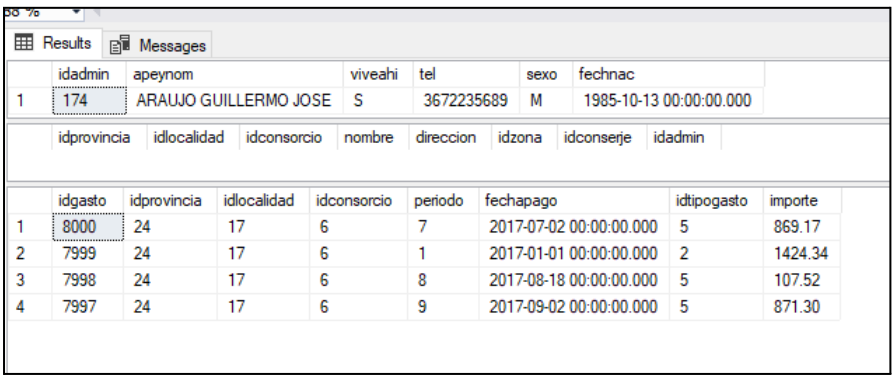
Prueba de error

Por último quisimos realizar una prueba de una transacción anidada, para ello utilizamos el script anterior, incorporando una pequeña transacción interna, que contenía un error y por ello se debería generar un rollback de toda la transacción en conjunto. Devolvimos la base de datos y el lote como al principio, y realizamos la prueba.

```sql
-- CREACION DE TRANSACCIONES
BEGIN TRY -- INICIAMOS EL BEGIN TRY PARA LUEGO COLOCAR LA LOGICA DENTRO Y ASEGURARNOS DE QUE SI ALGO FALLA IRA POR EL CATCH
    BEGIN TRAN -- COMENZAMOS LA TRANSACCION
    INSERT INTO administrador(apeynom, viveahi, tel, sexo, fechnac) -- UN INSERT A LA TABLA QUE QUEREMOS
    VALUES ('pablito clavounclavito', 'S', '37942222', 'M', '01/01/1996') -- LOS VALORES A INGRESAR A LA TABLA

    INSERT INTO consorcio(idprovincia, idlocalidad, idconsorcio, nombre, direccion, idzona, idconserje, idadmin)
    VALUES (1, 1, 3, 'EDIFICIO-111', 'PARAGUAY N 999', 5, 100, 1) -- GENERAMOS UN ERROR INGRESANDO EL ID PROVINCIA 999 QUE NO EXISTE.

    -------------------------
    BEGIN TRAN -- COMENZAMOS UNA TRANSACCION ANIDADA
        UPDATE consorcio SET nombre = 'EDIFICIO-222'; -- ACTUALIZAMOS EL REGISTRO DE CONSORCIO QUE CARGAMOS ANTES
        INSERT INTO gasto(idprovincia, idlocalidad, idconsorcio, periodo, fechapago, idtipogasto, importe)
            VALUES (1,1,1,6,'20130616',20,608.97) -- INSERT A LA TABLA GASTO, DEBE DAR UN ERROR POR TIPO DE GASTO
    COMMIT TRAN -- FINALIZAMOS UNA TRANSACCION ANIDADA
    ------------------------

    INSERT INTO gasto(idprovincia, idlocalidad, idconsorcio, periodo, fechapago, idtipogasto, importe)
    VALUES (1,1,1,6,'20130616',5,608.97)
    INSERT INTO gasto(idprovincia, idlocalidad, idconsorcio, periodo, fechapago, idtipogasto, importe)
    VALUES (1,1,1,6,'20130616',2,608.97)
    INSERT INTO gasto(idprovincia, idlocalidad, idconsorcio, periodo, fechapago, idtipogasto, importe)
    VALUES (1,1,1,6,'20130616',3,608.97)

    COMMIT TRAN -- SI TODO FUE EXITOSO FINALIZAMOS LA TRANSACCION
END TRY
BEGIN CATCH -- SI ALGO FALLA VENDRA AQUI
    SELECT ERROR_MESSAGE() -- MOSTRAMOS EL MENSAJE DE ERROR
    ROLLBACK TRAN -- VOLVEMOS HACIA ATRAS PARA MANTENER LA CONSISTENCIA DE LOS DATOS
END CATCH
```

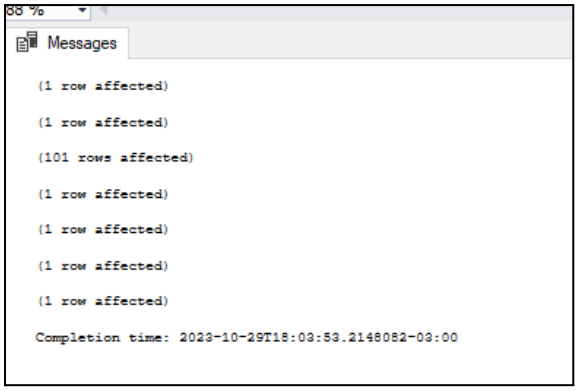
El resultado fue como se esperaba, generando un error por la transacción interna, y lo que debía suceder es que para nuestro caso, la transacción padre hiciera un rollback completo. Y así fue, al realizar las consultas y verificar que los procesos de la transacción en conjunto fueron cancelados, pudimos ver que tanto los procesos de la transacción padre, como los de la transacción hija, se deshicieron, ya que no hubo cambio en las tablas en cuestión.





Prueba sin error

Por último, lo que hicimos fue corregir el error en el script anterior y permitir que la transacción anidada se realizará correctamente. Y como se ve en los resultados, todos los procesos fueron realizados con éxitos y la transacción anidada fue terminada sin problemas, generando los cambios solicitados.



88 %

Results | Messages

	idadmin	apeynom	viveahi	tel	sexo	fechnac
1	178	pablito clavounclavito	S	37942222	M	1996-01-01 00:00:00.000

	idprovincia	idlocalidad	idconsorcio	nombre	direccion	idzona	idconserje	idadmin
1	1	1	3	EDIFICIO-222	PARAGUAY N 999	5	100	1

	idgasto	idprovincia	idlocalidad	idconsorcio	periodo	fechapago	idtipogasto	importe
1	8006	1	1	1	6	2013-06-16 00:00:00.000	3	608.97
2	8005	1	1	1	6	2013-06-16 00:00:00.000	2	608.97
3	8004	1	1	1	6	2013-06-16 00:00:00.000	5	608.97
4	8003	1	1	1	6	2013-06-16 00:00:00.000	5	608.97

INTEGRACIÓN DE OTROS TEMAS

En esta parte del informe se incorporan resumidas compresiones sobre los demás temas cuya investigación estuvo a cargo de otros grupos en la comisión.

Backup en línea y restore

El informe habla sobre la importancia de realizar copias de seguridad (backup) y restauraciones (restore) en SQL Server para proteger los datos críticos almacenados en las bases de datos. Se menciona que una estrategia bien diseñada de copia de seguridad y restauración ayuda a proteger las bases de datos de posibles pérdidas de datos causadas por errores.

Se destaca que la estrategia de copia de seguridad y restauración debe personalizarse según el entorno y los recursos disponibles. Se deben tener en cuenta factores como los objetivos de la organización, la naturaleza de las bases de datos y las restricciones de los recursos.

Además, se enfatiza la importancia de probar las copias de seguridad para garantizar su eficacia. Se recomienda realizar pruebas de restauración en un sistema de prueba y verificar la coherencia de la base de datos restaurada.

Por último, se menciona la restauración en línea, que permite restaurar datos mientras la base de datos está en línea. Se describen los pasos básicos de la restauración en línea y se concluye resaltando la importancia de realizar backups regularmente para proteger los datos importantes.

Índices Columnares en SQL Server

Si bien el informe no realiza una explicación teórica profunda del tema, y va más orientado hacia la aplicación práctica y argumentación durante el desarrollo.
Pudimos comprender que los índices columnares son estructuras de datos que mejoran la velocidad de recuperación de información en bases de datos al almacenar y organizar de manera eficiente los valores de una o más columnas específicas.
La clave de índice se construye a partir de una o más columnas definidas al crear el índice. Esta estructura facilita la búsqueda rápida y eficiente de datos.

Ventajas de los Índices Columnares:
Mejora del rendimiento de consultas, la presencia de índices columnares acelera la búsqueda y recuperación de datos, especialmente en operaciones de búsqueda y filtrado.

Escenarios de Uso: los índices columnares son particularmente beneficiosos en tablas con grandes conjuntos de datos, donde la optimización del rendimiento es crítica.

En resumen, los índices columnares en SQL Server son herramientas esenciales para mejorar el rendimiento de consultas, proporcionando una forma eficiente de acceder y recuperar datos en grandes bases de datos. Su diseño y mantenimiento adecuados son fundamentales para garantizar un rendimiento óptimo del sistema.

Optimización de consultas a través de índices

Este informe aborda la optimización de consultas mediante el uso de índices, centrándose en el contexto de auditoría de bases de datos. Se explora la diferencia entre consultas con y sin índices, destacando que los índices aceleran la velocidad de recuperación de datos.

El documento abarca tipo de índices, ventajas y desventajas, elección de columnas a indexar, y ofrece un ejemplo práctico de aplicación de índices en consultas SQL.

Se describe detalladamente el funcionamiento del motor de base de datos con índices agrupados y no agrupados. El caso de estudio presenta un análisis de consultas antes y después de la creación de índices, demostrando su impacto en el rendimiento.

En conclusión, se resalta la importancia de equilibrar el uso de índices para mejorar el rendimiento sin generar complejidades innecesarias en la base de datos.

CONCLUSIÓN

En resumen, una transacción en base de datos es una secuencia de operaciones que se realizan de manera indivisible para mantener la consistencia y la integridad de los datos. Las transacciones siguen las propiedades ACID, lo que garantiza que los cambios se realicen de manera segura y que los datos se mantengan en un estado coherente. Hemos podido corroborar cada detalle de manera práctica y es así como damos por finalizado este proceso investigativo, llevándonos el conocimiento teórico y práctico sobre las transacciones y dándoles el valor e importancia que se merecen para la gestión de bases de datos.

CAPÍTULO VI: BIBLIOGRAFÍA

- Silberchatz, korth, sudarshan. Fundamentos de base de datos (4ta Edición). (pág 367).
- Mercedes Marqués. (2011). Base de datos. https://bdigital.uvhm.edu.mx/wp-content/uploads/2020/05/Bases-de-Datos.pdf
- Wikipedia. Transacciones (informática). https://es.wikipedia.org/wiki/Transacci%C3%B3n_(inform%C3%A1tica)#:~:text=Una%20transacci%C3%B3n%20en%20un%20sistema,en%20forma%20indivisible%20o%20at%C3%B3mica.
- Transact-SQL Reference. https://learn.microsoft.com/en-us/sql/t-sql/language-reference?view=sql-server-ver16