Relaciones y operaciones transaccionales

Parte II

{desafío} latam_

- Objetivo de la sesión
- Conceptualización
- Ejercitación
- Reflexión

Inicio

15 Minutos

{desafío} latam_

Objetivo

/*Identificar qué son las transacciones y qué utilidad nos genera al momento de gestionar una base de datos.*/

/*Integrar las transacciones en sentencias SQL para la realización por lote de consultas*/

/*Cargar una base de datos utilizando dump.*/

Desarrollo

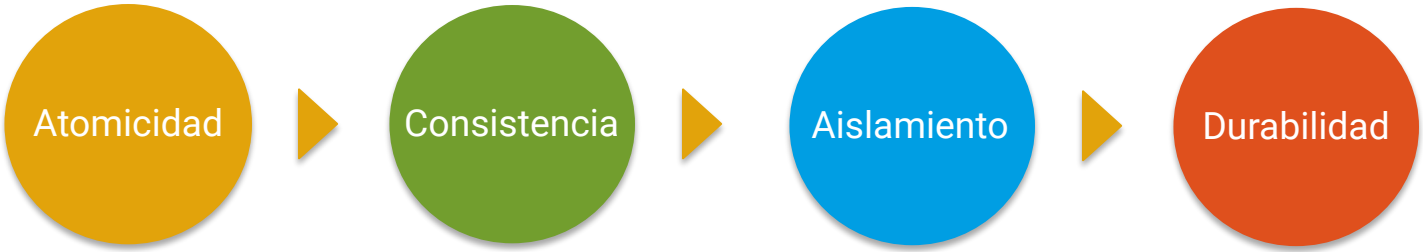
150 Minutos

{desafío} latam_

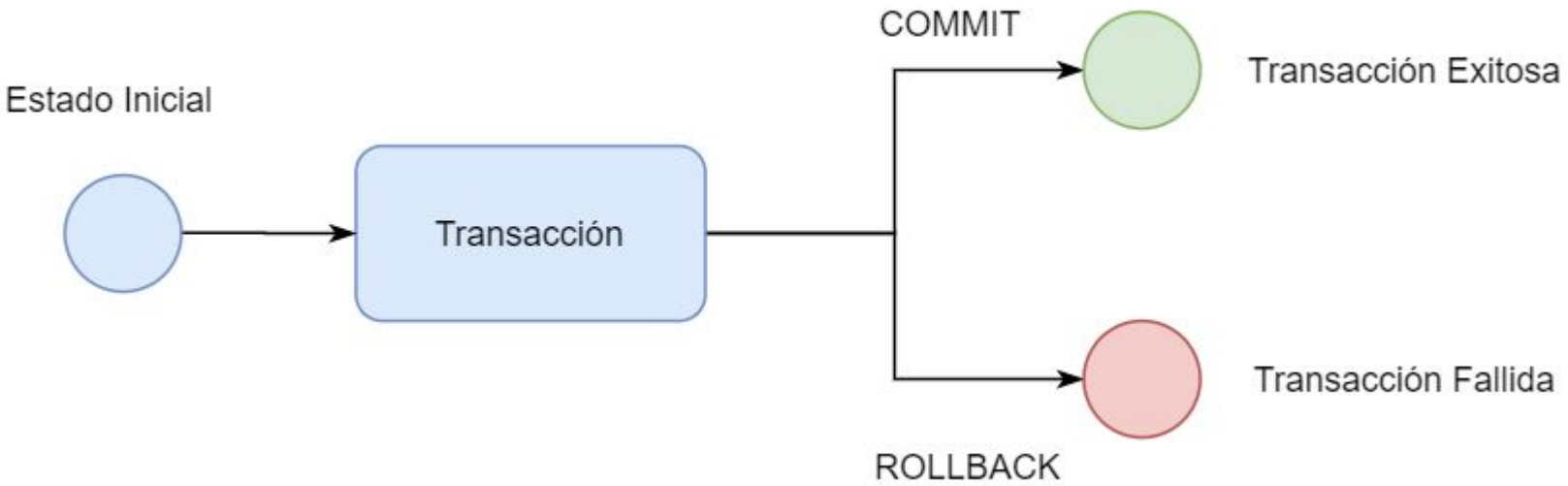
Transacciones

Las transacciones son secuencias de instrucciones ordenadas, las cuáles pueden ser indicadas de forma manual o pueden ser aplicadas automáticamente.

Las transacciones tienen las siguientes propiedades:



Transacciones



Transacciones

Comando	Descripción
BEGIN	El sistema permite que se ejecuten todas las sentencias SQL que necesitemos.
COMMIT	Guarda los cambios de la transacción
ROLLBACK	Retrocede los cambios realizados
SAVEPOINT	Guarda el punto de partida al cual volver a la hora de aplicar ROLLBACK
SET TRANSACTION	Le asigna nombre a la transacción

Transacciones

- Los comandos sólo pueden ser usados con las operaciones INSERT, UPDATE y DELETE.
- Sintaxis:
  - COMMIT;
  - SAVEPOINT nombre_savepoint;
  - ROLLBACK [TO nombre_savepoint];
- Lo que está entre corchetes es de carácter opcional,

- Sintaxis para iniciar una transacción:

```
SET TRANSACTION [READ ONLY|WRITE][NAME
nombre_transaccion];
```

COMMIT

- COMMIT guarda los cambios de la transacción.
- **Ejemplo:** Pensar en el funcionamiento de las cuentas de los bancos. Al hacer una transferencia
  - *¿Cómo nos aseguramos que el dinero que se resta de una cuenta, se sume en la siguiente?*
  - *¿Cómo controlamos que efectivamente se mantenga la integridad de los datos en caso de fallas?*

Resolvamos en conjunto

Utilizando la siguiente tabla, se requiere registrar de 2 cuentas con un saldo de $1000 cada una. Luego realizar una transferencia de $1000 desde la cuenta 1 a la cuenta 2, una forma de asegurarnos que el monto de nuestro balance disminuya en $1000 y el de la segunda cuenta aumenta en la misma cifra.

```
CREATE TABLE cuentas (
numero_cuenta NUMBER NOT NULL UNIQUE PRIMARY KEY,
balance NUMBER(11,2) CHECK (balance >= 0.00)
-- check valida la condición que el monto sea mayor a cero
);
```

Resultado

```
COMMIT;
SET TRANSACTION NAME 'cuentas_update';
UPDATE cuentas SET balance = balance - 1000 WHERE numero_cuenta = 1;
-- esta consulta se ejecutará sin problemas.
UPDATE cuentas SET balance = balance + 1000 WHERE numero_cuenta = 2;
COMMIT;
```

numero_cuenta	balance
1	0
2	2000

Ahora te toca a ti

La empresa Mawashi Cars Spa ha detectado un problema con el sistema que permite registrar ventas de los autos que no tienen stock y ha provocado incomodidades con los clientes, por lo que necesita urgentemente que se arregle este error y evitar futuras ventas en donde no se tenga disponibilidad del auto que se quiera vender. En el apoyo lectura de esta sesión encontraras 2 archivos llamados autos.csv y ventas.csv para el desarrollo de este ejercicio propuesto.

Para empezar con las soluciones debes crear las tablas correspondientes a los archivos .csv deduciendo el tipo de dato de cada columna, definiendo la llave primaria y foránea correspondiente en las tablas, además de agregarle el comando CHECK a la columna stock en la tabla Autos.

ROLLBACK

- Con este comando podemos deshacer las transacciones que se hayan ejecutado, revirtiendo los cambios realizados por una transacción hasta el último COMMIT o ROLLBACK ejecutado.
- Permite controlar los flujos de ejecución en nuestras transacciones, de manera que volvamos a un estado anterior, sin alterar los datos almacenados.

Resolvamos en conjunto

En el ejemplo del banco, en el punto anterior realizamos una transacción en donde transferimos $1000 de la cuenta 1 a la cuenta 2, como verás no obtuvimos ningún error, no obstante podemos verificar que en las transacciones que terminan en error no se altera el estado de nuestros datos. ¿Y cómo lo hacemos? Ejecuta el siguiente código en tu terminal.

```
select * from cuentas;

COMMIT;
SET TRANSACTION NAME 'cuentas_update';
UPDATE cuentas SET balance = balance + 1000 WHERE numero_cuenta = 2;
UPDATE cuentas SET balance = balance - 1000 WHERE numero_cuenta = 1;
select * from cuentas;
ROLLBACK;
```

Ahora te toca a ti

Continuando con el ejercicio propuesto de Mawashi Cars, realiza otra transacción en donde insertes registros en la tabla ventas pero ahora con los autos de id 2, 3 y 5. Considera que alguno de estos autos podría no tener stock por lo que deberás realizar un rollback al recibir el error por consola y confirmar que los cambios no fueron realizados consultando ambas tablas.

SAVEPOINT

- Permiten seleccionar qué partes de la transacción serán descartadas bajo ciertas condiciones, mientras el resto de las operaciones sí se ejecutan.
- Después de definir un punto de recuperación seguido de su nombre representativo, se puede volver a él por medio de ROLLBACK TO. Todos los cambios realizados por la transacción, entre el punto de recuperación y el rollback se descartan.

Resolvamos en conjunto

Con el siguiente código Intentemos registrar una nueva cuenta de número 3 en la tabla “cuentas” con un saldo de $5000 y justo luego guardar ese punto de la transacción con un SAVEPOINT de nombre “nueva_cuenta”. Luego intentar transferir a esta nueva cuenta $3000 desde la cuenta 2

```
BEGIN TRANSACTION
INSERT INTO cuentas(numero_cuenta, balance) VALUES (3,
5000);
SAVEPOINT nueva_cuenta;
```

Ahora te toca a ti

Realizar una nueva transacción intentando volver a registrar las ventas de los autos de id 2, 3 y 5, pero para evitar que no se realice ningún cambio crea un punto de guardado por cada auto que no arroje un error en su actualización de stock y si recibes algún error realiza un ROLLBACK a último SAVEPOINT. Posteriormente revisa ambas tablas para verificar que se realizaron los cambios correspondientes.

AUTOCOMMIT

- Oracle posee por defecto **autocommit OFF**; pero esto se puede modificar para casos en que quisiéramos que las transacciones se confirmen de inmediato.

```
-- para mostrar el estado del autocommit
show autocommit;
-- para activarlo
autocommit off;
-- para desactivarlo
autocommit on;
```

Cargar una base de datos utilizando Data Pump Export

Podemos realizar tareas de importar o exportar datos con nuestros clientes gráficos, mediante comandos por medio de nuestra consola. De esta forma podríamos realizar respaldos fuera de horario, programando tareas.

Podría parecer una tarea engorrosa si la comparamos con otros motores y su dump, pero esto se debe a motivos de seguridad, por ejemplo el asignar un directorio para los respaldos, nos asegura al menos un orden. Adicionalmente la exportación se realiza en archivos especiales para oracle.

Manos al código - Pongamos a prueba los conocimientos

La pizzeria nacional Hot Cheese, ofrece en su sitio web para el servicio de pizzas a domicilio y apesar de estar funcionando bien los primeros meses ha bajado su clientela por incomodidades en sus usuarios, que realizan pagos electrónicos en la aplicación y posteriormente reciben un correo de disculpas por la empresa diciendo que la pizza que compró ya no está disponible. El dueño de la pizzería ha tomado cartas sobre el asunto y ha solicitado contratar a un programador de bases de datos, para que cree una nueva base de datos aplicando las restricciones para evitar que siga sucediendo esta situación.

Para la solución de este ejercicio deberás realizar lo siguiente:

Manos al código - Pongamos a prueba los conocimientos

1. Crear una base de datos llamada “pizzeria”.
2. Conectarse a la base de datos pizzeria.
3. Crear 2 tablas llamadas “ventas” y “pizzas” con la siguiente estructura. La columna stock debe tener una restricción que diga que su valor debe ser mayor a 0.

cliente	fecha	monto	pizza(FK)

Modelo de la tabla ventas

id(PK)	stock	costo	nombre

Modelo de la tabla pizzas

Manos al código - Pongamos a prueba los conocimientos

4. Agregar 1 registro a la tabla “pizzas” seteando como stock inicial 0.
5. Realizar una transacción que registre una nueva pizza con un stock positivo mayor a 1.
6. Realizar una transacción que registre una venta con la pizza con stock 0 e intentar actualizar su stock restándole 1.
7. Realizar una transacción que intente registrar 1 venta por cada pizza, guardando un SAVEPOINT luego de la primera venta y volviendo a este punto si surge un error.
8. Exportar la base de datos “pizzeria” como un archivo pizzeria.sql.
9. Eliminar la base de datos “pizzeria”.
10. Importar el archivo pizzeria.sql.

Cierre

15 Minutos

{desafío} latam_

¿Existe algún concepto que no hayas comprendido?

Volvamos a revisar los conceptos que más te hayan costado antes de seguir adelante

Reflexionemos

{desafío} latam_

{desafío} latam_

Academia de talentos digitales