CODER HOUSE

SQL

Comisión 34945

Proyecto Final

Profe: Daniela Blanco

Tutor: Fernando Maldonado

Alumno: Sebastián Bellesi

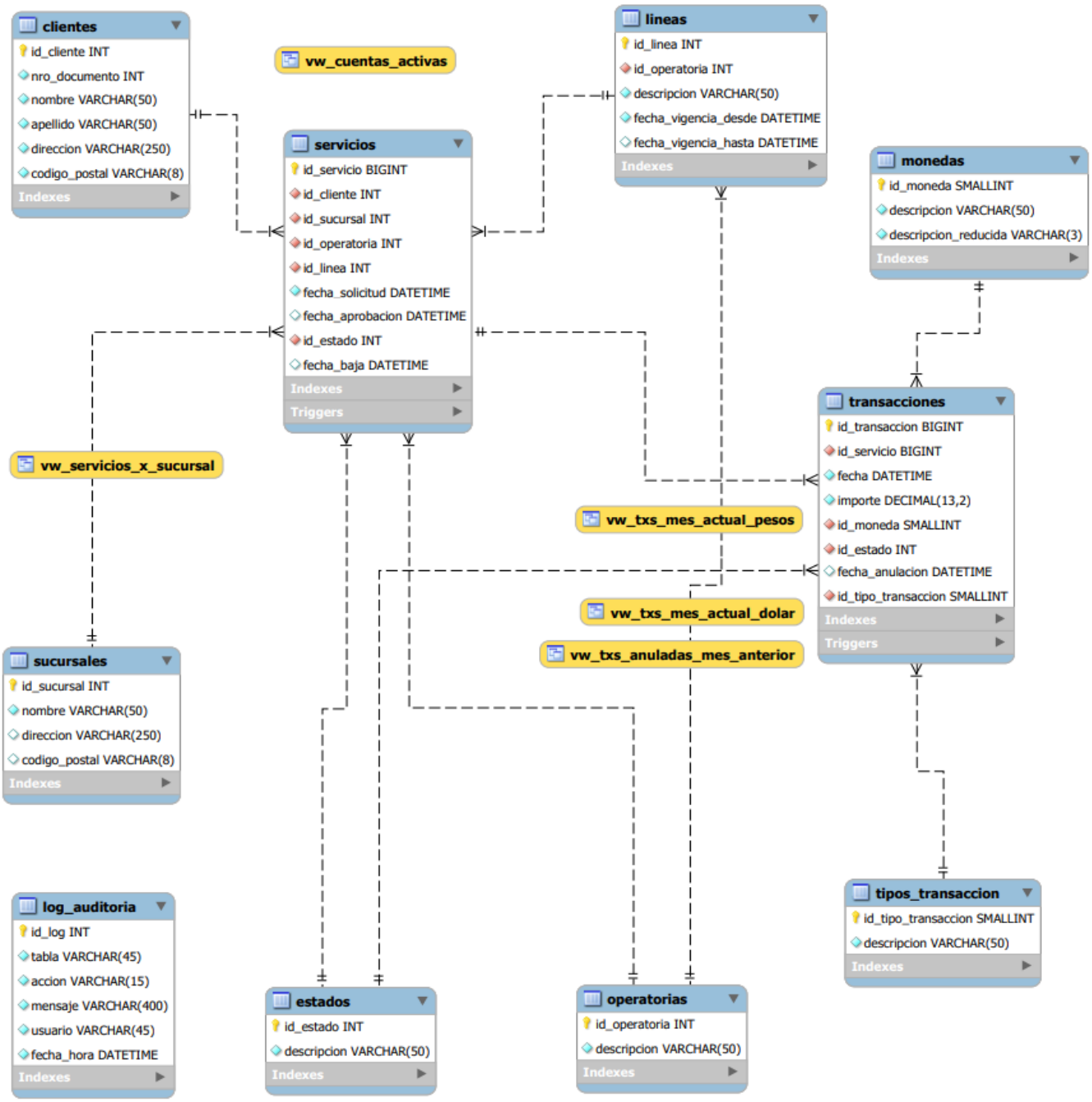
Tabla de contenido

Descripción de la temática de la Base de Datos:

Se propone la creación de una Base de Datos relacional basada en un modelo de negocio que permita gestionar los distintos productos contratados por los clientes en un Banco.

Cada producto contratado estará relacionado a un servicio en particular (por ejemplo: cuenta a la vista, tarjeta de crédito, plazo fijo, préstamo, etc). Adicionalmente, de cada servicio se van a gestionar también las distintas transacciones que ejecute el cliente a lo largo de la contratación del servicio (desde que se aprueba hasta que se da de baja).

Diagrama de Entidad-Relación de la Base de Datos:



Listado de las tablas que comprenden la Base de Datos:

Tabla **Sucursales**: almacenará la información de las distintas sucursales que posee el Banco.

Sucursales				
Campo	**Tipo de dato**	**PK**	**FK**	**Descripción**
id_sucursal	INT	SI	NO	Identificador único de la sucursal
nombre	VARCHAR(50)	NO	NO	Nombre de la sucursal
direccion	VARCHAR(250)	NO	NO	Dirección de la sucursal
codigo_postal	VARCHAR(8)	NO	NO	Código postal de la sucursal en formato CPA. Ej: C1426BMD

Tabla **Estados**: almacenará la información de los distintos estados permitidos.

Estados				
Campo	**Tipo de dato**	**PK**	**FK**	**Descripción**
id_estado	INT	SI	NO	Identificador único del estado
descripcion	VARCHAR(50)	NO	NO	Descripción del estado

Tabla **Clientes**: almacenará la información de los clientes que posee el Banco.

Clientes				
Campo	**Tipo de dato**	**PK**	**FK**	**Descripción**
id_cliente	INT	SI	NO	Identificador único del cliente
nro_documento	INT	NO	NO	Número de documento del cliente
nombre	VARCHAR(50)	NO	NO	Nombre del cliente
apellido	VARCHAR(50)	NO	NO	Apellido del cliente
direccion	VARCHAR(250)	NO	NO	Dirección del cliente
codigo_postal	VARCHAR(8)	NO	NO	Código postal del cliente en formato CPA. Ej: C1426BMD

Tabla **Operatorias**: almacenará la información de las operatorias que posee el Banco. Se define como “operatoria” a un grupo genérico de servicio brindado por el Banco, como por ejemplo: Cuentas a la Vista, Tarjetas, Préstamos, Plazos Fijo, Seguros, etc.

Operatorias				
Campo	**Tipo de dato**	**PK**	**FK**	**Descripción**
id_operatoria	INT	SI	NO	Identificador único de la operatoria
descripcion	VARCHAR(50)	NO	NO	Descripción de la operatoria

Tabla **Líneas**: almacenará la información de las líneas que posee el Banco por cada Operatoria. Se define como "línea" a un subgrupo de servicio brindado por el Banco dentro de cada Operatoria, como por ejemplo:

- Para Operatoria "Cuenta a la Vista": líneas Caja de Ahorro, Cuenta Corriente, Cuenta Sueldo, etc
- Para Operatoria "Tarjetas": líneas Tarjeta de Crédito, Tarjeta de Débito, Tarjeta Prepaga, etc
- Para Operatoria "Préstamos": líneas Prendario Comercial, Amortizable Comercial, Pago Integro Comercial, etc

Lineas				
Campo	**Tipo de dato**	**PK**	**FK**	**Descripción**
id_linea	INT	SI	NO	Identificador único de la línea
id_operatoria	INT	NO	SI (Tabla: Operatorias)	Identificador único de la operatoria
descripcion	VARCHAR(50)	NO	NO	Descripción de la línea
fecha_vigencia_desde	DATETIME	NO	NO	Fecha de vigencia "desde" de la línea
fecha_vigencia_hasta	DATETIME	NO	NO	Fecha de vigencia "hasta" de la línea

Tabla **Servicios**: almacenará la información de los servicios contratados por los distintos clientes que posee el Banco.

Servicios				
Campo	**Tipo de dato**	**PK**	**FK**	**Descripción**
id_servicio	BIGINT	SI	NO	Identificador único del servicio
id_cliente	INT	NO	SI (Tabla: Clientes)	Identificador único del cliente
id_sucursal	INT	NO	SI (Tabla: Sucursales)	Identificador único de la sucursal
id_operatoria	INT	NO	SI (Tabla: Operatorias)	Identificador único de la operatoria
id_linea	INT	NO	SI (Tabla: Lineas)	Identificador único de la línea
fecha_solicitud	DATETIME	NO	NO	Fecha de solicitud del servicio
fecha_aprobacion	DATETIME	NO	NO	Fecha de aprobación del servicio
id_estado	INT	NO	SI (Tabla: Estados)	Identificador único del estado
fecha_baja	DATETIME	NO	NO	Fecha de baja del servicio

Tabla **Transacciones**: almacenará la información de las transacciones que se ejecuten a través del Banco, asociadas a los distintos servicios.

Transacciones				
Campo	**Tipo de dato**	**PK**	**FK**	**Descripción**
id_transaccion	BIGINT	SI	NO	Identificador único de la transacción
id_servicio	BIGINT	NO	SI (Tabla: Servicios)	Identificador único del servicio
fecha	DATETIME	NO	NO	Fecha de la transacción
importe	DECIMAL(13,2)	NO	NO	Importe de la transacción
id_moneda	SMALLINT	NO	SI (Tabla: Monedas)	Identificador único de la moneda
id_estado	INT	NO	SI (Tabla: Estados)	Identificador único del estado
fecha_anulacion	DATETIME	NO	NO	Fecha de anulación de la transacción
Id_tipo_transaccion	SMALLINT	NO	SI (Tabla: TiposTransaccion)	Identificador único del tipo de transacción

Tabla **Monedas**: almacenará la información de las monedas con las que trabaje el Banco.

Monedas				
Campo	**Tipo de dato**	**PK**	**FK**	**Descripción**
id_moneda	SMALLINT	SI	NO	Identificador único de la moneda
descripcion	VARCHAR(50)	NO	NO	Descripción de la moneda
descripcion_reducida	VARCHAR(3)	NO	NO	Descripción reducida de la moneda. Ej: ARS, USD, EUR, etc.

Tabla **TiposTransaccion**: almacenará la información de los tipos de transacción que se pueden llear a cabo (débito, crédito, reversa débito, reversa crédito).

Tipos_Transaccion				
Campo	**Tipo de dato**	**PK**	**FK**	**Descripción**
id_tipo_transaccion	SMALLINT	SI	NO	Identificador único del tipo de transacción
descripcion	VARCHAR(50)	NO	NO	Descripción del tipo de transacción

Tabla **LogAuditoria**: almacenará auditoria de otras tablas, como por ejemplo quien inserto/modifico/eliminó un registro de una tabla en particular.

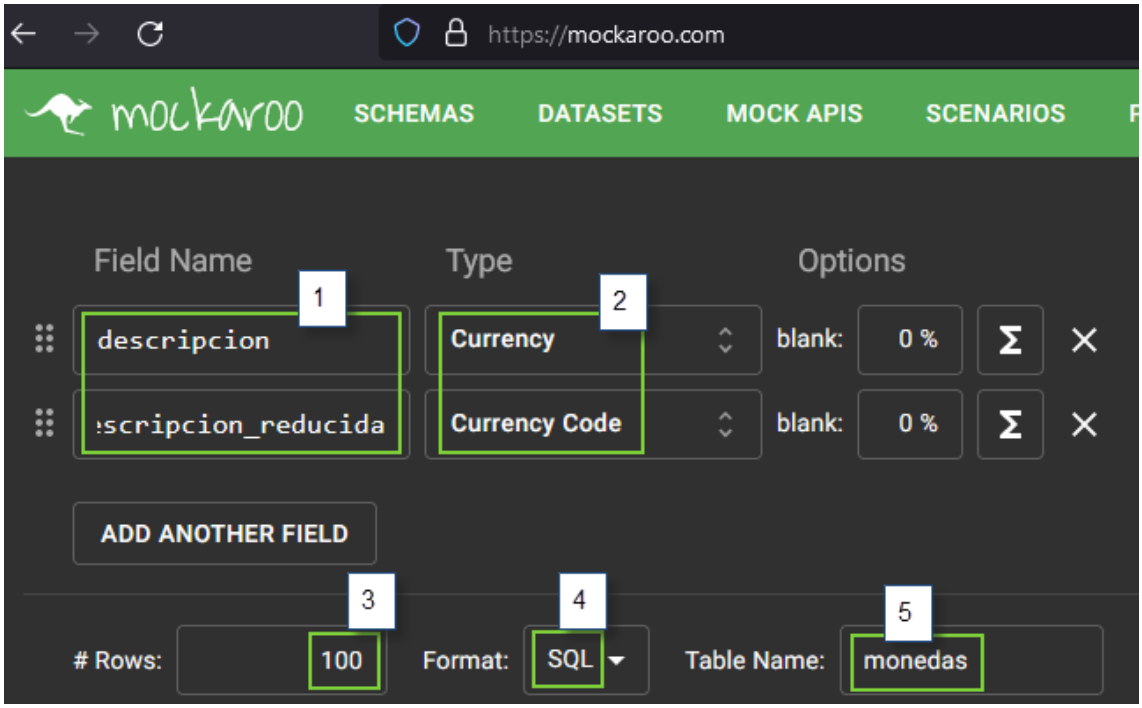
Log_Auditoria				
Campo	**Tipo de dato**	**PK**	**FK**	**Descripción**
id_log	BIGINT	SI	NO	Identificador único del log
tabla	VARCHAR(45)	NO	NO	Tabla relacionada a la acción que disparó la auditoria
accion	VARCHAR(15)	NO	NO	Acción que disparó la auditoria
mensaje	VARCHAR(400)	NO	NO	Mensaje descriptivo de lo que se logueó
usuario	VARCHAR(45)	NO	NO	Usuario que disparó la auditoria
fecha_hora	DATETIME	NO	NO	Fecha y hora en que se realizó la acción

Importación de registros a la Base de Datos:

Para la importación de los datos se utilizó la web https://mockaroo.com/

Desde la misma se fueron generando datos aleatorios para las distintas tablas y luego se descargó la información con sentencia SQL para realizar el INSERT en la DB. Por ejemplo, para la tabla "monedas":

1. Se definieron los campos que contiene la tabla.
2. Se definió el tipo de dato aleatorio a generar.
3. Se seteó la cantidad de registros a generar.
4. Se eligió el formato de salida para descargar la información
5. Se declaró el nombre de la tabla en donde se almacenarán los registros.



Preview

```
insert into monedas (descripcion, descripcion_reducida) values ('Rupiah', 'IDR');
insert into monedas (descripcion, descripcion_reducida) values ('Rupiah', 'IDR');
insert into monedas (descripcion, descripcion_reducida) values ('Hryvnia', 'UAH');
insert into monedas (descripcion, descripcion_reducida) values ('Yuan Renminbi', 'CNY');
insert into monedas (descripcion, descripcion_reducida) values ('Rupiah', 'IDR');
insert into monedas (descripcion, descripcion_reducida) values ('Lek', 'ALL');
insert into monedas (descripcion, descripcion_reducida) values ('Peso', 'PHP');
insert into monedas (descripcion, descripcion_reducida) values ('Dollar', 'USD');
insert into monedas (descripcion, descripcion_reducida) values ('Real', 'BRL');
insert into monedas (descripcion, descripcion_reducida) values ('Peso', 'PHP');
insert into monedas (descripcion, descripcion_reducida) values ('Peso', 'MXN');
insert into monedas (descripcion, descripcion_reducida) values ('Sol', 'PEN');
insert into monedas (descripcion, descripcion_reducida) values ('Sol', 'PEN');
insert into monedas (descripcion, descripcion_reducida) values ('Won', 'KPW');
insert into monedas (descripcion, descripcion_reducida) values ('Peso', 'PHP');
insert into monedas (descripcion, descripcion_reducida) values ('Real', 'BRL');
insert into monedas (descripcion, descripcion_reducida) values ('Rupiah', 'IDR');
insert into monedas (descripcion, descripcion_reducida) values ('Ruble', 'RUB');
insert into monedas (descripcion, descripcion_reducida) values ('Peso', 'PHP');
insert into monedas (descripcion, descripcion_reducida) values ('Peso', 'PHP');
insert into monedas (descripcion, descripcion_reducida) values ('Dollar', 'USD');
insert into monedas (descripcion, descripcion_reducida) values ('Rupiah', 'IDR');
```

Rows: 100 DOWNLOAD DATA

Vistas:

Se realizó la creación de 5 vistas, cada una con las siguientes funcionalidades:

1. Vista que muestra los servicios del Banco asociados a Cuentas a la Vista que se encuentren en estado activo recuperando además información de los clientes y descripción del tipo de cuenta.
2. Vista que muestra la cantidad de servicios que posee el Banco agrupados por sucursal, mostrando los datos de cada sucursal.
3. Vista que muestra las transacciones en pesos realizadas en el mes actual, es decir, en estado "aplicado"; mostrando información del servicio y del cliente.
4. Vista que muestra las transacciones en dólares realizadas en el mes actual, es decir, en estado "aplicado"; mostrando información del servicio y del cliente.
5. Vista que muestra las transacciones anuladas en el mes anterior al actual, mostrando la moneda y la información del servicio y del cliente.

Funciones:

Se realizó la creación de 2 funciones, cada una con las siguientes funcionalidades:

1. Función para obtener la fecha de la última transacción que realizó un cliente determinado para un producto (operatoria y línea) en particular y relacionado a un tipo de transacción seleccionada.

   Parámetros de entrada: id_cliente, id_linea, id_tipo_transaccion
   Parámetro de salida: DATETIME

2. Función para obtener la cantidad de transacciones que realizó un cliente determinado en un mes/año en particular, para un producto (operatoria y línea) específico y relacionado a un tipo de transacción seleccionada.

   Parámetros de entrada: id_cliente, id_linea, año-mes, id_tipo_transaccion
   Parámetro de salida: INT

Stored Procedures:

Se realizó la creación de 2 stored procedures, cada uno con las siguientes funcionalidades:

1. Procedimiento almacenado para dar de alta un cliente y una cuenta a la vista asociada. La cuenta quedará en estado “Solicitud pendiente” para su posterior aprobación. En caso de que ya exista dado de alta el cliente (verificación por índice único “nro_documento”) se dará de alta sólo la cuenta, mostrando un mensaje informativo con lo sucedido. En caso de que ya exista la cuenta a la vista para el cliente en cuestión (verificación por línea, sucursal y estado activo o pendiente aprobación), no se realizará ninguna acción, mostrando un mensaje informativo con lo sucedido. En caso de que haya errores en los parámetros de entrada (obligatoriedad, error de dato, etc), no se realizará ninguna acción, mostrando un mensaje informativo con lo sucedido.

   Parámetros de entrada: nro_documento, nombre, apellido, direccion, codigo_postal, id_sucursal, id_linea
   Parámetros de salida: cod_ejecucion, estado_ejecucion (*)

   Ejemplo de ejecución y recuperación de parámetros de salida:

   Ejecución con error por sucursal inexistente:

```
CALL sp_alta_cliente_cuenta(
      33514817,
      'Seba',
      'Bellesi',
      'Entre rios 988',
      'C2000ABC',
      99999,
      1,
      @cod_ejec,
      @estado_ejec);
SELECT @cod_ejec, @estado_ejec;
```

Ejecución OK:

```
CALL sp_alta_cliente_cuenta(
      33514817,
      'Seba',
      'Bellesi',
      'Entre rios 988',
      'C2000ABC',
      1,
      1,
      @cod_ejec,
      @estado_ejec);
SELECT @cod_ejec, @estado_ejec;
```

2. Procedimiento almacenado para anular una transacción pasando la misma a estado "Anulado". En caso de que la transacción no exista o no se encuentre en estado "Aplicado", no se realizará ninguna acción, mostrando un mensaje informativo con lo sucedido. En caso de que haya errores en los parámetros de entrada (obligatoriedad, error de dato, etc), no se realizará ninguna acción, mostrando un mensaje informativo con lo sucedido.

   Parámetros de entrada: id_transaccion
   Parámetro de salida: cod_ejecucion, estado_ejecucion (*)

   Ejemplo de ejecución y recuperación de parámetros de salida:

   Ejecución con error por transacción inexistente:

```
CALL sp_anula_transaccion(
      99999,
      @cod_ejec,
      @estado_ejec);
SELECT @cod_ejec, @estado_ejec;
```

   Ejecución OK:

```
CALL sp_anula_transaccion(
      125,
      @cod_ejec,
      @estado_ejec);
SELECT @cod_ejec, @estado_ejec;
```

   (*) El parámetro de salida "cod_ejecucion" permite conocer si el SP se ejecutó ok (devuelve 0) o si hubo algún problema (devuelve -1). En caso de que haya finalizado con código -1, se puede visualizar el error en el parámetro de salida "estado_ejecucion".

Triggers:

Se realizó la creación de 2 triggers, cada uno con las siguientes funcionalidades:

1. Trigger asociado a la tabla “servicios”, que se dispara con la acción de actualización y verifica si lo que se está modificando es el estado. En ese caso, si el estado al que se actualiza es “Dado de baja” setea la “fecha de baja” con la fecha/hora actual.

   Tabla asociada: servicios
   Acción asociada: UPDATE

2. Trigger asociado a la tabla “transacciones”, que se dispara con la acción de eliminación y guarda en una tabla de auditoria la transacción eliminada y datos del usuario que realizó la eliminación.

   Tabla asociada: transacciones
   Acción asociada: DELETE

Herramientas y Tecnologías utilizadas:

- Motor DB MySQL 8.0.30
- MySQL Workbench 8.0.28

Apéndice:

Consultas sublenguaje DCL:

```
-- Nos posicionamos sobre la DB del sistema MySQL
USE mysql;

-- Creación de usuario de sólo lectura
CREATE USER 'userConsulta'@'localhost' IDENTIFIED BY 'pass12345';

-- Agregamos el permiso correspondiente para que el usuario sea de sólo lectura
sobre todos los esquemas y tablas
GRANT SELECT ON *.* TO 'userConsulta'@'localhost';

-- Creación de usuario de lectura, inserción y modificación
CREATE USER 'userAMC'@'localhost' IDENTIFIED BY 'pass123456789';

-- Agregamos los permisos correspondiente para que el usuario pueda consultar,
insertar y actualizar sobre todos los esquemas y tablas
GRANT SELECT, INSERT, UPDATE ON *.* TO 'userAMC'@'localhost';

-- Visualizamos permisos por usuario
SHOW GRANTS FOR 'userConsulta'@'localhost';
SHOW GRANTS FOR 'userAMC'@'localhost';

-- Verificación en tabla user de MySQL
SELECT * FROM `user`;

-- Eliminación de usuarios
-- DROP USER 'userConsulta'@'localhost';
-- DROP USER 'userAMC'@'localhost';
```

Consultas sublenguaje TCL:

```
-- Nos posicionamos sobre la DB a utilizar
USE coder_banco_34945;

/* ----------------------------------------------------------------------
      Ejemplo 1
 ---------------------------------------------------------------------- */
-- Inicializamos la transacción (internamente se pasa el @@AUTOCOMMIT a 0)
START TRANSACTION;

-- Verificamos registro a eliminar
SELECT * FROM transacciones AS t
WHERE t.id_transaccion = 123;

-- Eliminamos registro de la tabla "transacciones" con id_transaccion = 123
DELETE FROM transacciones
WHERE id_transaccion = 123;

-- Verificamos registro eliminado
SELECT * FROM transacciones AS t
WHERE t.id_transaccion = 123;

-- Hasta acá, la eliminación está pendiente hasta invocar al COMMIT (confirma
eliminación) o ROLLBACK (deshace eliminación)

-- Se deshacen los cambios
ROLLBACK;

-- Se confirman los cambios
-- COMMIT;

-- Verificamos como quedó luego de la transacción
SELECT * FROM transacciones AS t
WHERE t.id_transaccion = 123;

/* ----------------------------------------------------------------------
      Ejemplo 2
 ---------------------------------------------------------------------- */
```

```
-- Inicializamos la transacción (internamente se pasa el @@AUTOCOMMIT a 0)
START TRANSACTION;

-- Se insertan 3 registros dentro de la transacción
INSERT INTO clientes (nro_documento, nombre, apellido, direccion, codigo_postal)
VALUES (12345671, 'Cliente 1', 'Apellido 1', 'Direccion 1', 'L2000HGA'),
       (12345672, 'Cliente 2', 'Apellido 2', 'Direccion 2', 'L2000HGB'),
       (12345673, 'Cliente 3', 'Apellido 3', 'Direccion 3', 'L2000HGC');

-- Se agrega SAVEPOINT hasta el registro 3
SAVEPOINT registro_3;

-- Se insertan 2 registros más dentro de la transacción
INSERT INTO clientes (nro_documento, nombre, apellido, direccion, codigo_postal)
VALUES (12345674, 'Cliente 4', 'Apellido 4', 'Direccion 4', 'L2000HGD');
INSERT INTO clientes (nro_documento, nombre, apellido, direccion, codigo_postal)
VALUES (12345675, 'Cliente 5', 'Apellido 5', 'Direccion 5', 'L2000HGE');

-- Se agrega SAVEPOINT hasta el registro 5
SAVEPOINT registro_5;

-- Para quitar último SAVEPOINT
-- RELEASE SAVEPOINT registro_5;

-- Se deshacen los cambios hasta el savepoint registro_3
-- ROLLBACK TO SAVEPOINT registro_3;
-- Se deshacen los cambios hasta el savepoint registro_5
-- ROLLBACK TO SAVEPOINT registro_5;
-- Se deshacen todos los cambios
ROLLBACK;

-- Se confirman los cambios
-- COMMIT;
-- Verificamos como quedó luego de la transacción por nro_documento
SELECT * FROM clientes AS c
WHERE c.nro_documento IN (12345671, 12345672, 12345673, 12345674, 12345675);
```