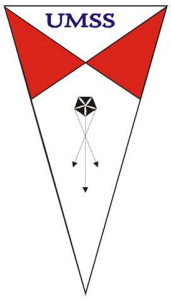


UNIVERSIDAD MAYOR DE SAN SIMÓN

CARRERA DE INGENIERÍA INFORMÁTICA - SISTEMAS

PROYECTO INMOBILIARIA

MOTORES, SEGURIDAD, TRIGGERS Y BITÁCORAS EN BASE DE DATOS

GRUPO: 13

NOMBRES:

- Vera Viscarra Emerson Denis
- Revollo Huanca Andres Juan Jose
- Rojas Becerra Guillermo
- Limachi Cuiza Daniel Boris

MATERIA:

Taller De Base De Datos

DOCENTE:

Lic. Boris Marcelo Calancha Navia

FECHA:

14-06-2023

Cochabamba–Bolivia

I. Introducción

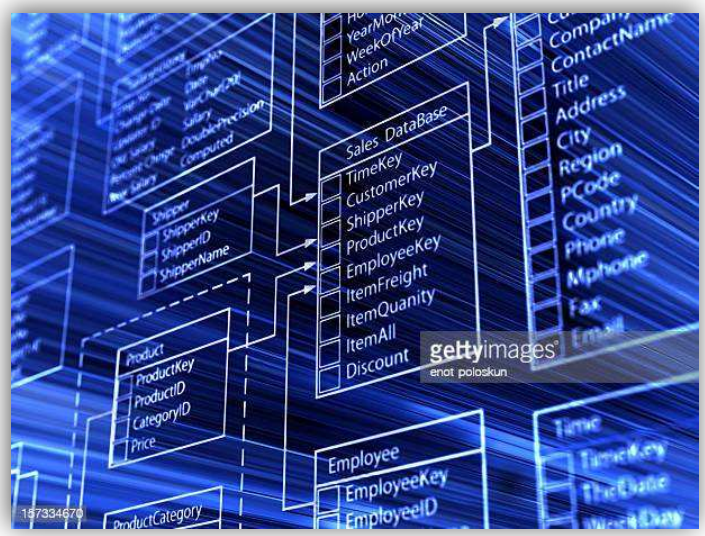
En el marco de este trabajo, se ha llevado a cabo una exhaustiva investigación a través de foros oficiales y externos, así como la revisión de páginas web y videos informativos, con el objetivo de profundizar en el conocimiento de los motores, la seguridad, los TRIGGERS y las bitácoras en una base de datos específicamente diseñada para una proveedora de inmuebles.

La seguridad de la base de datos es un aspecto crucial en la protección de los datos sensibles y confidenciales de los clientes y propiedades y todo lo relacionado a ellos. Se explorarán las mejores prácticas para asegurar la integridad, confidencialidad y disponibilidad de la información, así como la implementación de mecanismos de autenticación y autorización.

Los TRIGGERS, o disparadores, son procedimientos almacenados en la base de datos que se ejecutan automáticamente en respuesta a eventos o acciones específicas. Se examinará cómo utilizar los TRIGGERS de manera efectiva para mantener la consistencia de los datos y aplicar acciones automáticas en función de ciertos desencadenantes.

Las bitácoras, o registros de transacciones, son registros que registran todas las operaciones realizadas en la base de datos. Estos registros son esenciales para el seguimiento de cambios, la recuperación ante fallos y la auditoría de la base de datos. Se discutirán las estrategias adecuadas para implementar y administrar bitácoras en el contexto de una base de datos inmobiliaria.

En este trabajo, se abordarán conceptos clave, se discutirán las mejores prácticas y se propondrán recomendaciones para optimizar el rendimiento y garantizar la seguridad en el contexto de una base de datos inmobiliaria.



II. Tipos de tablas en SQL server

En SQL Server, existen diferentes tipos de tablas que desempeñan un papel fundamental en la organización y gestión de los datos. Estas tablas son objetos de base de datos que almacenan información en un formato de filas y columnas, similar al de una hoja de cálculo.

Cada fila en una tabla representa un registro único, mientras que cada columna representa un campo específico dentro de ese registro. Esta estructura tabular facilita el almacenamiento y acceso eficiente de los datos en la base de datos. A continuación, se describen los diferentes tipos de tablas utilizados:

Tablas con particiones

Las tablas con particiones son aquellas en las que los datos se dividen horizontalmente entre múltiples grupos de archivos de una base de datos. Esta técnica facilita la administración de grandes conjuntos de datos y permite un acceso y administración eficientes de subconjuntos de datos específicos. En SQL Server, se admiten hasta 15,000 particiones de forma predeterminada, lo que brinda flexibilidad y rendimiento en el manejo de grandes volúmenes de datos.

Tablas Temporales

Las tablas temporales en SQL Server se almacenan en la base de datos tempdb. Existen dos tipos de tablas temporales: locales y globales, que se diferencian por su nombre, visibilidad y disponibilidad.

- Las tablas temporales locales se identifican por un único signo de número (#) al inicio de su nombre. Estas tablas son visibles únicamente para el usuario de la conexión actual y se eliminan automáticamente cuando el usuario se desconecta de la instancia de SQL Server. Son útiles para almacenar datos temporales y trabajar con ellos durante la sesión de usuario.
- Por otro lado, las tablas temporales globales se identifican por dos signos de número (##) al inicio de su nombre. Estas tablas son visibles para cualquier usuario después de su creación y se eliminan automáticamente cuando todos los usuarios que hacen referencia a la tabla se desconectan de la instancia de SQL Server. Son útiles cuando se necesita compartir datos temporales entre múltiples usuarios o sesiones.

Las tablas temporales proporcionan una forma conveniente de almacenar y manipular datos temporales durante una sesión o transacción. Su almacenamiento en la base de datos tempdb permite un acceso rápido y eficiente a los datos temporales. Es importante tener en cuenta que las tablas temporales son adecuadas para datos temporales y su uso excesivo o innecesario puede afectar el rendimiento del sistema.

Tablas del sistema

En SQL Server, los datos que definen la configuración del servidor y de todas sus tablas se almacenan en un conjunto especial de tablas conocido como tablas del sistema.

Estas tablas del sistema no pueden ser consultadas o actualizadas directamente por los usuarios. En su lugar, la información contenida en estas tablas del sistema está disponible a través de vistas del sistema.

Las tablas del sistema son fundamentales para el funcionamiento interno de SQL Server, ya que contienen datos críticos sobre la configuración del servidor, las bases de datos, las tablas, los índices, los usuarios y otros elementos del sistema. Sin embargo, el acceso directo a estas tablas está restringido para evitar modificaciones no autorizadas o errores en la configuración del sistema.

En su lugar, los usuarios pueden acceder a la información de las tablas del sistema mediante vistas del sistema. Estas vistas proporcionan una capa de abstracción que presenta los datos de manera más legible y comprensible. Las vistas del sistema permiten a los usuarios obtener información importante sobre la configuración del servidor, la estructura de la base de datos y otros metadatos relevantes.

Tablas Anchas

Las tablas anchas en SQL Server utilizan columnas dispersas para aumentar el número total de columnas permitidas hasta 30,000. Las columnas dispersas son columnas normales con un almacenamiento optimizado para valores NULL, lo que reduce los requisitos de espacio para dichos valores a costa de una mayor sobrecarga al recuperar valores no NULL. Una tabla ancha tiene un conjunto de columnas definido, que es una representación XML sin tipo que combina todas las columnas dispersas de una tabla en una salida estructurada. Además, el número máximo de índices y estadísticas se incrementa a 1,000 y 30,000 respectivamente. El tamaño máximo de una fila en una tabla ancha es de 8,019 bytes, lo que significa que la mayoría de los datos en una fila deben ser NULL. Sin embargo, el número máximo de columnas no dispersas más las columnas calculadas en una tabla ancha sigue siendo 1,024.

Las tablas anchas tienen implicaciones en el rendimiento que se deben tener en cuenta:

- Pueden aumentar el costo de mantenimiento de los índices de la tabla. Se recomienda limitar el número de índices en una tabla ancha a aquellos que sean necesarios para la lógica de negocio. Un aumento en el número de índices también puede resultar en un mayor tiempo de compilación de las operaciones de manipulación de datos (DML) y requerimientos de memoria. Se sugiere que los índices no clúster sean índices filtrados que se apliquen a subconjuntos de datos.
- Las aplicaciones pueden agregar y quitar columnas de las tablas anchas de forma dinámica, lo cual invalida los planes de consulta compilados. Por lo tanto, es importante tener en cuenta este aspecto al diseñar las aplicaciones.
- El rendimiento puede verse afectado cuando se agregan o eliminan datos de una tabla ancha. El diseño de las aplicaciones debe estar en consonancia con la carga de trabajo prevista para minimizar los cambios realizados en los datos de la tabla.

- Las operaciones de cambio de partición en las tablas anchas pueden ser lentas y pueden requerir grandes cantidades de memoria para su procesamiento.
- Al utilizar cursores de actualización que actualizan columnas específicas en una tabla ancha, es recomendable enumerar explícitamente las columnas en la cláusula FOR UPDATE para optimizar el rendimiento.

Tablas Persistentes

Las tablas persistentes en SQL Server son aquellas que permiten el almacenamiento y gestión de registros de manera permanente. Estas tablas pueden ser eliminadas o modificadas manualmente y se dividen en tres tipos principales:

Tablas base

Las tablas base son aquellas que almacenan toda la información de los registros sin aplicar ninguna validación adicional. En otras palabras, los datos se insertan directamente en la tabla base sin ningún procesamiento adicional. Esta es la forma más básica de tabla persistente en SQL Server.

Vistas

Las vistas son relaciones que se establecen en referencia a una fila o columna específica en una o varias tablas base. Estas vistas proporcionan una manera conveniente de acceder a datos seleccionados o procesados de una o varias tablas. Las vistas pueden ser utilizadas para simplificar consultas complejas, proporcionar una capa de seguridad adicional y ofrecer una presentación lógica y estructurada de los datos.

Tablas instantáneas (Instantáneas)

Las tablas instantáneas son registros que pueden ser vistos de manera inmediata con solo una referencia. En otras palabras, estas tablas permiten ver un estado específico de los datos en un momento dado. Esto es útil cuando se desea realizar una consulta o análisis basado en una instantánea específica de los datos sin afectar los registros actuales.

Tabla Montón

Una tabla de montón en SQL Server es una estructura de almacenamiento donde las filas se identifican por medio de un identificador de fila (RID, Row Identifier). El RID contiene información como el número de archivo, el número de página y el desplazamiento en la página de cada fila en la tabla.

En una tabla de montón, las filas no están organizadas en un orden específico, sino que se agregan a medida que se insertan nuevos registros. Esto significa que no hay una clave primaria o un índice que defina el orden de las filas. Las consultas en una tabla de montón pueden requerir escaneos completos de la tabla para encontrar los datos deseados, lo que puede resultar en un rendimiento subóptimo en comparación con una tabla indexada.

En el sistema de metadatos de SQL Server, se crea una entrada en el objeto de sistema "sys.partitions" para cada partición de una tabla de montón, con un valor de "id índice" igual a 0 para indicar que no hay un índice en la tabla.

Es importante tener en cuenta que las tablas de montón pueden ser útiles en ciertos escenarios, especialmente cuando no se requiere un acceso rápido a los datos o cuando la tabla es pequeña y las operaciones de búsqueda son infrecuentes. Sin embargo, en la mayoría de los casos, se recomienda utilizar índices para mejorar el rendimiento de las consultas y la eficiencia del almacenamiento de datos.

Tabla Agrupada

Una tabla agrupada en SQL Server es aquella que tiene un índice agrupado predefinido en una columna o múltiples columnas de la tabla. Este índice define el orden de almacenamiento de las filas dentro de las páginas de información y el orden de las páginas dentro de la tabla, basado en la clave de índice de agrupamiento.

Las tablas agrupadas tienen varias capacidades importantes, entre las cuales se incluyen:

Soporte de transacciones

Las tablas agrupadas permiten realizar operaciones transaccionales, lo que garantiza la integridad de los datos y la consistencia en caso de errores o fallos.

Soporte de procedimientos almacenados

Puedes utilizar procedimientos almacenados para realizar operaciones en las tablas agrupadas, lo que proporciona un mayor control y modularidad en el manejo de los datos.

Entorno gráfico de administración

SQL Server incluye una interfaz gráfica de administración que te permite realizar comandos de lenguaje de definición de datos (DDL) y lenguaje de manipulación de datos (DML) de manera visual.

Arquitectura cliente-servidor

SQL Server funciona en un entorno cliente-servidor, donde los datos e información se almacenan en un servidor al que se puede acceder desde clientes remotos.

Administración de información de otros servidores

SQL Server tiene la capacidad de administrar información de otros servidores, lo que facilita la gestión de bases de datos distribuidas o con múltiples instancias.

Capacidades de bases de datos inteligentes

SQL Server ofrece características avanzadas como soporte para almacenamiento en memoria (in-memory), soporte de memoria persistente y optimización de tempdb para un mejor rendimiento.

Cifrado de datos y cumplimiento normativo

SQL Server cuenta con sistemas de protección, supervisión y clasificación de datos, lo que la convierte en una plataforma segura que cumple con las normativas de seguridad y privacidad.

Puedes utilizar procedimientos almacenados para realizar operaciones en las tablas agrupadas, lo que proporciona un mayor control y modularidad en el manejo de los datos.

Estas capacidades hacen de las tablas agrupadas en SQL Server una opción poderosa y versátil para el almacenamiento y gestión de datos en entornos empresariales.

Comprender los diferentes tipos de tablas en SQL Server es fundamental para diseñar y estructurar adecuadamente la base de datos de una proveedora de inmuebles. La elección correcta del tipo de tabla adecuado para cada situación contribuirá a la eficiencia y rendimiento general del sistema.

III. Mecanismos de seguridad propios de SQL server

La protección de la seguridad en SQL Server conlleva una serie de pasos que afectan a cuatro áreas: la plataforma, la autenticación, los objetos (incluidos los datos) y las aplicaciones que tienen acceso al sistema.

Seguridad de la plataforma y de la red

La plataforma de SQL Server incluye el hardware físico y los sistemas de redes que conectan los clientes con los servidores de bases de datos, así como los archivos binarios que se utilizan para procesar solicitudes de base de datos.

Seguridad física

Las recomendaciones de seguridad física limitan de forma estricta el acceso al servidor físico y a los componentes de hardware. Por ejemplo, use salas cerradas de acceso restringido para el hardware de servidor de base de datos y los dispositivos de red. Además, limite el acceso a los medios de copia de seguridad almacenándolos en una ubicación segura fuera de las instalaciones. La implementación de la seguridad de la red física comienza por mantener a los usuarios no autorizados fuera de la red. En la tabla siguiente se incluye más información sobre la seguridad de la red.

Seguridad del sistema operativo

SQL Server se integra con el Firewall de Windows para permitir la configuración de reglas de acceso a nivel de red. Puedes definir qué direcciones IP tienen permiso para acceder al servidor SQL Server y restringir el acceso desde ubicaciones no autorizadas.

Seguridad de los archivos del sistema operativo de SQL Server

SQL Server usa archivos del sistema operativo para el funcionamiento y el almacenamiento de datos. Las recomendaciones de seguridad de archivos indican que

se restrinja el acceso a estos archivos. En la tabla siguiente se ofrece información sobre estos archivos.

Entidades de seguridad y seguridad de objetos de base de datos

SQL Server permite otorgar permisos específicos a objetos individuales, como tablas, vistas, procedimientos almacenados, etc. Puedes controlar quién tiene acceso de lectura, escritura o ejecución a cada objeto.

Entidades de seguridad y seguridad de objetos de base de datos

Las entidades de seguridad son los individuos, grupos y procesos que tienen acceso a SQL Server. Los "elementos protegibles" son el servidor, la base de datos y los objetos incluidos en la base de datos. Cada uno de estos elementos dispone de un conjunto de permisos que pueden configurarse para reducir el área expuesta de SQL Server. se incluye información sobre las entidades de seguridad y los elementos protegibles:

- Usuarios, roles y procesos de servidor y base de datos.
- Seguridad de objetos de servidor y base de datos.
- La jerarquía de seguridad de SQL Server.

Cifrado

El cifrado no resuelve los problemas de control de acceso. Sin embargo, mejora la seguridad debido a que limita la pérdida de datos, incluso en el caso poco probable de que se superen los controles de acceso. Por ejemplo, si el equipo host de base de datos no está configurado correctamente y un usuario malintencionado obtiene datos confidenciales, como números de tarjetas de crédito, esa información robada podría resultar inservible si está cifrada, se muestran algunos escenarios donde se puede implementar el cifrado:

- La jerarquía de cifrado de SQL Server.
- Implementar conexiones seguras.
- Funciones de cifrado.

Certificados

Son "claves" de software que se comparten entre dos servidores que habilitan las comunicaciones seguras a través de una autenticación segura. Puede crear y usar certificados en SQL Server para mejorar la seguridad de objetos y conexiones. Como se pueden utilizar los certificados con SQL Server:

- Crear un certificado para utilizarlo con SQL Server.
- Usar un certificado para la creación de reflejo de la base de datos.

Estos mecanismos ayudan a garantizar la protección y seguridad de los datos almacenados en SQL Server, así como la autenticación y control de acceso a los recursos del sistema.

IV. Implementación de TRIGGERS

La implementación de TRIGGERS (disparadores) en una base de datos permite automatizar la ejecución de acciones o registros de datos después de que ocurran ciertos eventos los cuales se tienen planteados. Al implementar TRIGGERS, puedes lograr diferentes finalidades en la automatización del registro de datos. Algunas aplicaciones comunes incluyen:

- Auditoría de cambios: Puedes utilizar TRIGGERS para registrar automáticamente los cambios realizados en una tabla, almacenando información como quién realizó la modificación, cuándo y qué datos se modificaron. Esto es útil para fines de seguimiento, cumplimiento normativo o resolución de problemas.
- Mantenimiento de registros históricos: Los TRIGGERS pueden ayudar a mantener un registro histórico de los cambios en una tabla. Cada vez que se realiza una modificación en los datos, el TRIGGER puede copiar los valores antiguos en una tabla de historial, preservando así un seguimiento de las versiones anteriores.

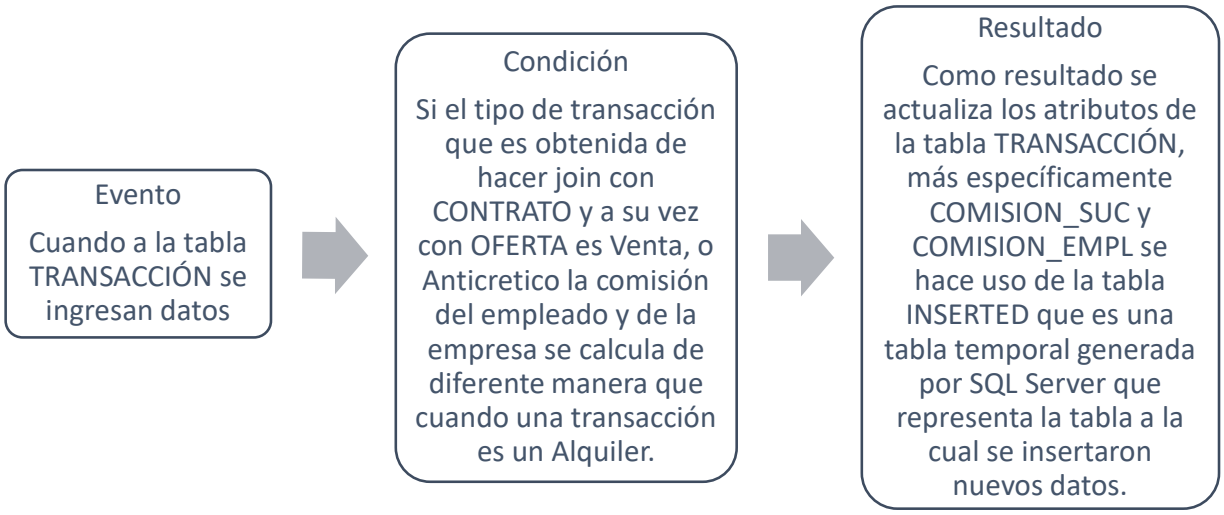
A continuación, se plantean diferentes eventos que requieren la implementación de disparadores.

Calculo de Comisiones

Cada que se culmine una transacción de venta, anticrético o alquiler calcular automáticamente los montos de comisión para la empresa y comisión para el agente inmobiliario, los porcentajes son:

- Para la venta o anticrético: es el 3% del monto de venta o de anticrético para la empresa y 0,06% del porcentaje correspondiente a la comisión de la empresa para el agente inmobiliario.
- Si es alquiler es el 50% del monto del alquiler para la empresa y el 20% del monto que gana la empresa para el agente inmobiliario.

Desarrollo



TRIGGER

```sql
CREATE TRIGGER CalculoComision
ON TRANSACCION
AFTER INSERT
AS
BEGIN
    -- Variables para almacenar los datos necesarios
    DECLARE @tipoTransaccion CHAR(20)
    DECLARE @monto money
    DECLARE @comisionEmpresa money
    DECLARE @comisionEmpleado money

    -- Obtener el tipo de transacción y el monto utilizando subconsultas
    SELECT @tipoTransaccion = O.TIPO_OFERTA,
           @monto = T.MONTO_TRANS
    FROM TRANSACCION AS T
    INNER JOIN inserted AS I ON T.COD_TRANSACCION = I.COD_TRANSACCION
    INNER JOIN CONTRATO AS C ON T.COD_CONTRATO = C.COD_CONTRATO
    INNER JOIN OFERTA AS O ON C.COD_OFERTA = O.COD_OFERTA

    -- Realizar el cálculo de las comisiones según el tipo de transacción
    IF @tipoTransaccion = 'Venta' OR @tipoTransaccion = 'Anticretico' -- Venta o anticrético
    BEGIN
        SET @comisionEmpresa = @monto * 0.03
        SET @comisionEmpleado = @comisionEmpresa * 0.0006
    END
    ELSE IF @tipoTransaccion = 'Alquiler' -- Alquiler
    BEGIN
        SET @comisionEmpresa = @monto * 0.5
        SET @comisionEmpleado = @comisionEmpresa * 0.2
    END

    -- Actualizar los montos de comisión en la tabla Transaccion
    UPDATE TRANSACCION
    SET COMISION_SUC = @comisionEmpresa, COMISION_EMPL = @comisionEmpleado
    WHERE COD_TRANSACCION IN (SELECT COD_TRANSACCION FROM inserted)
END
```

Estado actual de la tabla TRANSACCIÓN

COD_TRANSACCION	COD_CONTRATO	FECHA_TRANS	MONTO_TRANS	COMISION_SUC	COMISION_EMPL
1	1	2023-07-01 00:00:00.000	1000.0000	100.0000	10.0000
2	2	2023-02-22 00:00:00.000	2000.0000	200.0000	20.0000
3	3	2023-10-03 00:00:00.000	1500.0000	150.0000	15.0000
4	4	2023-05-03 00:00:00.000	3000.0000	300.0000	30.0000
5	5	2023-12-05 00:00:00.000	4000.0000	400.0000	40.0000
6	1	2023-05-03 00:00:00.000	5500.0000	550.0000	55.0000
7	2	2023-07-23 00:00:00.000	6800.0000	680.0000	68.0000
8	3	2024-01-28 00:00:00.000	2000.0000	200.0000	20.0000
9	4	2023-02-13 00:00:00.000	3000.0000	300.0000	30.0000
10	5	2023-01-31 00:00:00.000	3100.0000	310.0000	31.0000
11	6	2023-05-24 00:00:00.000	90000.0000	2700.0000	1.6200
13	7	2023-05-29 00:00:00.000	65000.0000	1950.0000	1.1700
12	5	2023-05-28 00:00:00.000	70000.0000	2100.0000	1.2600
14	8	2023-05-25 00:00:00.000	30000.0000	900.0000	0.5400
NULL	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*

Se insertará una nueva transacción:

```
INSERT INTO TRANSACCION(COD_TRANSACCION, COD_CONTRATO, FECHA_TRANS, MONTO_TRANS)
VALUES(15,12,'2023-07-08 00:00:00.000',1800)
```

Solo se ingresaron los primeros 4 datos las comisiones se calculan.

Resultado

COD_TRAN...	COD_CONT...	FECHA_TR...	MONTO_TR...	COMISION...	COMISION...
1	1	2023-07-01 ...	1000.0000	100.0000	10.0000
2	2	2023-02-22 ...	2000.0000	200.0000	20.0000
3	3	2023-10-03 ...	1500.0000	150.0000	15.0000
4	4	2023-05-03 ...	3000.0000	300.0000	30.0000
5	5	2023-12-05 ...	4000.0000	400.0000	40.0000
6	1	2023-05-03 ...	5500.0000	550.0000	55.0000
7	2	2023-07-23 ...	6800.0000	680.0000	68.0000
8	3	2024-01-28 ...	2000.0000	200.0000	20.0000
9	4	2023-02-13 ...	3000.0000	300.0000	30.0000
10	5	2023-01-31 ...	3100.0000	310.0000	31.0000
11	6	2023-05-24 ...	90000.0000	2700.0000	1.6200
13	7	2023-05-29 ...	65000.0000	1950.0000	1.1700
15	12	2023-07-08 ...	1800.0000	900.0000	180.0000
12	5	2023-05-28 ...	70000.0000	2100.0000	1.2600
14	8	2023-05-25 ...	30000.0000	900.0000	0.5400
NULL	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*

Actualización del estado del Inmueble

Cada que se concluya una transacción de venta, de alquiler o de anticrético, se debe cambiar el estado del inmueble como “transacción culminada”, si no tiene este atributo se debe añadir.

Desarrollo

Evento

Cuando a la tabla TRANSACCIÓN se ingresan datos

Condición

Se considera que un inmueble tiene un estado "Transacción culminada" cuando este llega a formar parte de una transacción.

Resultado

Como resultado se actualiza los atributos de la tabla INMUEBLE, más específicamente DISPONIBLE este atributo ya existía en la base de datos original y se maneja como tipo de dato BIT o booleana.

TRIGGER

```
CREATE TRIGGER CambiarEstadoInmueble
ON TRANSACCION
AFTER INSERT
AS
BEGIN
    UPDATE INMUEBLE
    SET DISPONIBLE = 0
    WHERE COD_INMUEBLE IN(
        SELECT O.COD_INMUEBLE
        FROM TRANSACCION AS T
        INNER JOIN inserted AS I ON T.COD_TRANSACCION = I.COD_TRANSACCION
        INNER JOIN CONTRATO AS C ON T.COD_CONTRATO = C.COD_CONTRATO
        INNER JOIN OFERTA AS O ON C.COD_OFERTA = O.COD_OFERTA
        INNER JOIN INMUEBLE AS INM ON INM.COD_INMUEBLE = O.COD_INMUEBLE
    )
END
```

Estado actual de la tabla INMUEBLE.

COD_INMU...	COD_DOC...	COD_DIRE...	SUPERFICIE	DISPONIBLE	NUM_INM...
1	1	1	300	True	111
2	2	2	500	True	122
3	3	3	200	True	s/n
4	4	4	400	True	004
5	5	5	500	True	s/n
6	6	6	600	False	016
7	7	7	100	True	017

8	8	8	160	True	s/n
9	9	9	200	True	109
10	10	10	150	True	s/n
11	11	11	211	True	011
12	12	12	212	False	012
13	13	13	313	False	013
14	14	14	214	True	014
15	15	15	150	True	015
16	16	16	216	True	016
17	17	17	217	True	017
18	18	18	318	True	231
19	19	19	319	True	101
20	20	20	420	False	118
21	1	21	200	True	941
22	4	22	865	False	222
23	6	23	345	True	931
24	8	24	300	False	193
25	10	25	300	True	943
NULL	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*

Se modificarán los inmuebles con Código de inmueble ‘5’, ’4’ cada destacar que para que un inmueble llegue a una transacción debe formar parte de una oferta y a la vez parte de un contrato y contrato hace referencia al contrato hecho por el propietario y el cliente y transacción la última fase que consiste en el pago a la empresa y registro que ganancias.

Tabla OFERTA

COD_OFER...	COD_EMPL...	COD_INMU...	FECHA_INI_...	FECHA_FIN...	TIPO_OFER...
1	1	1	2023-06-01 ...	2023-07-01 ...	Alquiler ...
2	2	2	2023-02-01 ...	2023-03-28 ...	Anticretico ...
3	3	2	2023-08-20 ...	2023-09-20 ...	Venta ...
4	4	3	2023-10-20 ...	2023-11-20 ...	Alquiler ...
5	1	4	2023-11-01 ...	2023-11-21 ...	Alquiler ...
6	2	5	2023-12-05 ...	2024-01-28 ...	Alquiler ...
7	4	7	2023-12-08 ...	2024-01-31 ...	Venta ...

Tabla CONTRATO

COD_CONT...	COD_OFER...	PAGO_CTRT	FECHA_INI_...	FECHA_FIN...	DETALLE_C...
1	1	1500.0000	2023-02-01 ...	2023-03-31 ...	alquier de u...
2	2	20000.0000	2023-02-01 ...	2023-03-31 ...	venta de ca...
3	3	30000.0000	2023-03-01 ...	2024-05-31 ...	Venta ...
4	12	35000.0000	2023-02-28 ...	2023-04-01 ...	Venta ...
5	13	20000.0000	2023-01-15 ...	2023-03-15 ...	Anticretico ...
6	25	35000.0000	2023-05-24 ...	2024-05-24 ...	Venta ...
8	14	20000.0000	2023-05-25 ...	2023-12-28 ...	Anticretico ...
9	11	1800.0000	2023-05-26 ...	2023-08-26 ...	Alquiler ...
10	22	2000.0000	2023-07-02 ...	2023-10-10 ...	Alquiler ...
11	17	60000.0000	2023-06-25 ...	2023-06-27 ...	Venta ...
12	10	2000.0000	2023-07-06 ...	2023-11-06 ...	Alquiler ...
13	6	3000.0000	2023-06-08 ...	2023-12-01 ...	Alquiler ...
14	5	2500.0000	2023-07-01 ...	2023-09-01 ...	Alquiler ...
7	24	60000.0000	2023-07-26 ...	2023-10-01 ...	Venta ...
NULL	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*

Se ingresaron dos nuevas transacciones.

```
INSERT INTO TRANSACCION(COD_TRANSACCION, COD_CONTRATO, FECHA_TRANS, MONTO_TRANS)
VALUES
(16,13,'2023-06-07 00:00:00.000',3000),
(17,14,'2023-06-08 00:00:00.000',2500)
```

Tabla TRANSACCIÓN

COD_TRAN...	COD_CONT...	FECHA_TR...	MONTO_TR...	COMISION...	COMISION...
1	1	2023-07-01 ...	1000.0000	100.0000	10.0000
2	2	2023-02-22 ...	2000.0000	200.0000	20.0000
3	3	2023-10-03 ...	1500.0000	150.0000	15.0000
4	4	2023-05-03 ...	3000.0000	300.0000	30.0000
5	5	2023-12-05 ...	4000.0000	400.0000	40.0000
6	1	2023-05-03 ...	5500.0000	550.0000	55.0000
7	2	2023-07-23 ...	6800.0000	680.0000	68.0000

8	3	2024-01-28 ...	2000.0000	200.0000	20.0000
9	4	2023-02-13 ...	3000.0000	300.0000	30.0000
10	5	2023-01-31 ...	3100.0000	310.0000	31.0000
11	6	2023-05-24 ...	90000.0000	2700.0000	1.6200
13	7	2023-05-29 ...	65000.0000	1950.0000	1.1700
15	12	2023-07-08 ...	1800.0000	900.0000	180.0000
16	13	2023-06-07 ...	3000.0000	1500.0000	300.0000
17	14	2023-06-08 ...	2500.0000	1500.0000	300.0000
12	5	2023-05-28 ...	70000.0000	2100.0000	1.2600
14	8	2023-05-25 ...	30000.0000	900.0000	0.5400
NULL	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*

Resultado

COD_INMU...	COD_DOC...	COD_DIRE...	SUPERFICIE	DISPONIBLE	NUM_INM...
1	1	1	300	True	111
2	2	2	500	True	122
3	3	3	200	True	s/n
4	4	4	400	False	004
5	5	5	500	False	s/n
6	6	6	600	False	016
7	7	7	100	True	017
8	8	8	160	True	s/n
9	9	9	200	True	109
10	10	10	150	True	s/n
11	11	11	211	True	011

Registro de la hora de entrada de los empleados

Cada que un empleado registre su entrada, evaluar considerando su horario de ingreso al turno si llego con retraso, si marco con retraso, guardar los datos que se requiera (puede ser un campo o una tabla) donde se guarde el tiempo de retraso y el dato retraso; se aclara que el ingreso debe ser igual o mayor a 10 minutos para ser considerado retraso. Se debe añadir lo que necesite ya sea tablas o atributos.

Desarrollo

Para la implementación de este trigger se añadieron dos tablas:

```
CREATE TABLE REGISTRO_ENTRADAS (
    COD_REGISTRO      INT IDENTITY(1,1)   not null,
    COD_EMPLEADO      INT                 not null,
    ENTRADA           TIME                not null,
    constraint PK_REGISTRO_ENTRADAS primary key nonclustered (COD_REGISTRO)
)
go
CREATE TABLE RETRASO (
    COD_RETRASO       INT IDENTITY(1,1)   not null,
    COD_EMPLEADO      INT                 not null,
    ENTRADA           TIME                not null,
    TIEMPO_RETRASO    INT                 not null,
    constraint PK_COD_RETRASO primary key nonclustered (COD_RETRASO)
)
go

alter table REGISTRO_ENTRADAS
    add constraint FK_REGISTRO_ENTRADAS_EMPLEADOS foreign key (COD_EMPLEADO)
       references EMPLEADO (COD_EMPLEADO)
go

alter table RETRASO
    add constraint FK_RETRASO_EMPLEADOS foreign key (COD_EMPLEADO)
       references EMPLEADO (COD_EMPLEADO)
go
```

Evento

Cuando a la tabla REGISTRO_ENTRADAS se ingresan datos

Condición

Se considera que un empleado tiene un retraso cuando llega 10 minutos después de su hora de entrada en este caso las 8:00 am.

Resultado

Como resultado se inserta un retraso en la tabla RETRASO incluyendo datos como el código del empleado, la hora de ingreso y el tiempo de retraso.

TRIGGER

```
CREATE TRIGGER EVALUAR_RETRASO
ON REGISTRO_ENTRADAS
AFTER INSERT
AS
BEGIN
    -- Variables necesarias
    DECLARE @IDEmpleado         INT
    DECLARE @HoraEntrada        TIME
    DECLARE @TiempoRetraso      INT
```

```
    -- Obtencion de datos de entrada
    SELECT @IDEmpleado = COD_EMPLEADO, @HoraEntrada = ENTRADA
    FROM inserted
    -- Calculo del tiempo
    SET @TiempoRetraso = DATEDIFF(MINUTE, CAST('08:00:00' AS TIME), @HoraEntrada) - 10

    IF @TiempoRetraso >= 0
        BEGIN
            INSERT INTO RETRASO (COD_EMPLEADO, ENTRADA, TIEMPO_RETRASO)
            VALUES(@IDEmpleado, @HoraEntrada, @TiempoRetraso)
        END
END
```

Se realizaron las siguientes inserciones

```
INSERT INTO REGISTRO_ENTRADAS(COD_EMPLEADO, ENTRADA)
VALUES (3,'08:00:00')
INSERT INTO REGISTRO_ENTRADAS(COD_EMPLEADO, ENTRADA)
VALUES (5,'08:30:00')
INSERT INTO REGISTRO_ENTRADAS(COD_EMPLEADO, ENTRADA)
VALUES (10,'09:00:00')
INSERT INTO REGISTRO_ENTRADAS(COD_EMPLEADO, ENTRADA)
VALUES (1,'07:00:00')
```

Resultado

Tabla REGISTRO_ENTRADAS

COD_REGIS...	COD_EMPL...	ENTRADA
1	10	08:30:00
2	5	08:00:00
3	3	08:00:00
4	4	08:30:00
NULL	*NULL*	*NULL*

Tabla RETRASO

COD_RETR...	COD_EMPL...	ENTRADA	TIEMPO_RE...
4	5	08:30:00	20
5	10	09:00:00	50
NULL	*NULL*	*NULL*	*NULL*

De las cuatro inserciones que se hicieron 2 empleados entraron tarde y estas fueron registradas en la tabla RETRASO.

V. Implementación de bitácoras

Las bitácoras, también conocidas como logs, son registros secuenciales que registran las transacciones y cambios realizados en la base de datos. Estas bitácoras se utilizan para mantener un registro detallado de las actividades y eventos que ocurren en el sistema de gestión de bases de datos (DBMS, por sus siglas en inglés).

Las bitácoras desempeñan un papel fundamental en la recuperación de la base de datos en caso de fallos o errores. Al registrar todas las transacciones y cambios, permiten realizar un seguimiento de las modificaciones realizadas en la base de datos, lo que facilita la recuperación de la base de datos a un estado consistente antes del fallo.

Algunos de los usos y beneficios de las bitácoras en una base de datos incluyen:

- Recuperación: Las bitácoras permiten recuperar la base de datos a un estado consistente en caso de fallos, como caídas del sistema, errores de hardware o interrupciones inesperadas.
- Auditoría y cumplimiento normativo: Las bitácoras proporcionan un registro detallado de todas las transacciones y cambios realizados en la base de datos, lo que facilita la auditoría y el cumplimiento de las normativas de seguridad y privacidad.
- Investigación de problemas: En caso de problemas o discrepancias en los datos, las bitácoras pueden ayudar a identificar y rastrear la causa raíz del problema.

Identificar las tablas críticas

En el contexto de la base de datos, las tablas críticas son aquellas que almacenan y manejan datos sensibles o de alta importancia para la empresa. Estos datos pueden incluir información financiera, datos personales, secretos comerciales u otra información confidencial que, si se modifica o se accede de manera no autorizada, puede ocasionar serios problemas o impactar negativamente en la empresa.

Tabla SALARIO asociada a EMPLEADO

Esta tabla almacena información sobre los salarios de los empleados de la empresa. Los datos relacionados con la remuneración y beneficios económicos de los empleados son de suma importancia y su modificación no autorizada puede afectar la estabilidad financiera y la confianza de los empleados en la empresa.

Tabla PRECIO asociada a OFERTA

En esta tabla se almacenan los precios asociados a las ofertas de productos o servicios de la empresa. La modificación no autorizada de los precios puede llevar a pérdidas económicas significativas, afectar la competitividad de la empresa y generar desconfianza en los clientes.

Tabla CONTRATO

Esta tabla contiene información sobre los contratos celebrados por la empresa con terceros, como proveedores, clientes o socios comerciales. Los datos relacionados con los contratos pueden incluir información confidencial, cláusulas legales, términos

financieros y acuerdos comerciales estratégicos. Su modificación no autorizada puede generar problemas legales, pérdida de relaciones comerciales y daños a la reputación de la empresa.

Tabla TRANSACCIÓN

En esta tabla se registran las transacciones financieras realizadas por la empresa, como pagos, compras, ventas y movimientos de capital. Estos datos son críticos para el seguimiento y control financiero de la empresa, y su modificación no autorizada puede generar desequilibrios contables, pérdidas económicas e incluso problemas legales.

La identificación de estas tablas críticas permite enfocar los esfuerzos de seguridad y protección en aquellas áreas de la base de datos que contienen información altamente sensible y valiosa. Se implementarán medidas adicionales de seguridad para garantizar la integridad y confidencialidad de estas tablas críticas.

Implementación de Bitácoras

Con el fin de tener un registro detallado de las tareas de actualización de datos en la tabla crítica "CONTRATO", se ha implementado una bitácora que registra todas las modificaciones realizadas en dicha tabla. Para ello, se creó una nueva tabla denominada "BITACORA_CONTRATO" que almacenará la información relevante sobre las actualizaciones realizadas en la tabla "CONTRATO".

La tabla "BITACORA_CONTRATO" tiene una estructura similar a la tabla "CONTRATO" e incluye los siguientes campos:

```
CREATE TABLE BITACORA_CONTRATO (
    ID              INT IDENTITY(1,1)   not null,
    COD_CONTRATO    INT                 not null,
    COD_OFERTA      INT                 not null,
    PAGO_CONTRATO   MONEY               not null,
    FECHA_INICIO    DATETIME            not null,
    FECHA_FIN       DATETIME            not null,
    FECHA_EVENTO    DATETIME            not null,
    EVENTO          VARCHAR(100)        not null,
    constraint PK_BITACORA_CONTRATO primary key nonclustered (ID)
)
go
```

El TRIGGER asociado

```
CREATE TRIGGER REGISTRAR_BITACORA_CONTRATO
ON CONTRATO
AFTER INSERT, UPDATE, DELETE
AS
BEGIN
    DECLARE @Cod_Contrato   INT
    DECLARE @Cod_Oferta     INT
```

```
    DECLARE @Pago_Contrato  MONEY
    DECLARE @Fecha_Inicio   DATETIME
    DECLARE @Fecha_Fin      DATETIME

    IF EXISTS (SELECT * FROM inserted)
    BEGIN
        SELECT
            @Cod_Contrato = COD_CONTRATO,
            @Cod_Oferta = COD_OFERTA,
            @Pago_Contrato = PAGO_CTRT,
            @Fecha_Inicio = FECHA_INI_CNTRT,
            @Fecha_Fin = FECHA_FIN_CNTRT
        FROM inserted
        -- Inserción de registros en BitacoraTransaccion para nuevas transacciones
        INSERT INTO BITACORA_CONTRATO
        VALUES(@Cod_Contrato, @Cod_Oferta, @Pago_Contrato, @Fecha_Inicio, @Fecha_Fin,
        GETDATE(), 'Actualizacion en Contrato')
    END
```

```
    ELSE IF EXISTS (SELECT * FROM deleted)
    BEGIN
        SELECT
            @Cod_Contrato = COD_CONTRATO,
            @Cod_Oferta = COD_OFERTA,
            @Pago_Contrato = PAGO_CTRT,
            @Fecha_Inicio = FECHA_INI_CNTRT,
            @Fecha_Fin = FECHA_FIN_CNTRT
        FROM deleted
        -- Inserción de registros en BitacoraTransaccion para transacciones modificadas o eliminadas
        INSERT INTO BITACORA_CONTRATO
        VALUES(@Cod_Contrato, @Cod_Oferta, @Pago_Contrato, @Fecha_Inicio, @Fecha_Fin,
        GETDATE(), 'Eliminacion en Contrato')
    END
END
```

Se realizarán las siguientes operaciones:

```
INSERT INTO CONTRATO(COD_CONTRATO, COD_OFERTA, PAGO_CTRT, FECHA_INI_CNTRT, FECHA_FIN_CNTRT, DETALLE_CNTRT)
VALUES
(16, 21,3000,'2023-06-12 00:00:00.000','2023-09-12 00:00:00.000','Alquiler de Casa amueblada')

UPDATE CONTRATO
SET PAGO_CTRT = 3500
WHERE COD_CONTRATO = 16

DELETE FROM CONTRATO
WHERE COD_CONTRATO = 16
```

Tabla CONTRATO después del INSERT

COD_CONT...	COD_OFER...	PAGO_CTRT	FECHA_INI_...	FECHA_FIN...	DETALLE_C...
1	1	1500.0000	2023-02-01 ...	2023-03-31 ...	alquier de u...
2	2	20000.0000	2023-02-01 ...	2023-03-31 ...	venta de ca...
3	3	30000.0000	2023-03-01 ...	2024-05-31 ...	Venta ...
4	12	35000.0000	2023-02-28 ...	2023-04-01 ...	Venta ...
5	13	20000.0000	2023-01-15 ...	2023-03-15 ...	Anticretico ...
6	25	35000.0000	2023-05-24 ...	2024-05-24 ...	Venta ...
8	14	20000.0000	2023-05-25 ...	2023-12-28 ...	Anticretico ...
9	11	1800.0000	2023-05-26 ...	2023-08-26 ...	Alquiler ...
10	22	2000.0000	2023-07-02 ...	2023-10-10 ...	Alquiler ...
11	17	60000.0000	2023-06-25 ...	2023-06-27 ...	Venta ...
12	10	2000.0000	2023-07-06 ...	2023-11-06 ...	Alquiler ...
13	6	3000.0000	2023-06-08 ...	2023-12-01 ...	Alquiler ...
14	5	2500.0000	2023-07-01 ...	2023-09-01 ...	Alquiler ...
15	18	40000.0000	2023-08-06 ...	2024-08-06 ...	Anticretico ...
16	21	3000.0000	2023-06-12 ...	2023-09-12 ...	Alquiler de ...
7	24	60000.0000	2023-07-26 ...	2023-10-01 ...	Venta ...
NULL	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*

Tabla BITACORA_CONTRATO

ID	COD_CONT...	COD_OFER...	PAGO_CON...	FECHA_INI...	FECHA_FIN	FECHA_EVE...	EVENTO
1	16	21	3000.0000	2023-06-12 ...	2023-09-12 ...	2023-06-13 ...	Actualizacio...
NULL	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*

Tabla CONTRATO después del UPDATE

COD_CONT...	COD_OFER...	PAGO_CTRT	FECHA_INI_...	FECHA_FIN...	DETALLE_C...
1	1	1500.0000	2023-02-01 ...	2023-03-31 ...	alquier de u...
2	2	20000.0000	2023-02-01 ...	2023-03-31 ...	venta de ca...
3	3	30000.0000	2023-03-01 ...	2024-05-31 ...	Venta ...
4	12	35000.0000	2023-02-28 ...	2023-04-01 ...	Venta ...
5	13	20000.0000	2023-01-15 ...	2023-03-15 ...	Anticretico ...
6	25	35000.0000	2023-05-24 ...	2024-05-24 ...	Venta ...
8	14	20000.0000	2023-05-25 ...	2023-12-28 ...	Anticretico

8	14	20000.0000	2023-05-25 ...	2023-12-28 ...	Anticretico ...
9	11	1800.0000	2023-05-26 ...	2023-08-26 ...	Alquiler ...
10	22	2000.0000	2023-07-02 ...	2023-10-10 ...	Alquiler ...
11	17	60000.0000	2023-06-25 ...	2023-06-27 ...	Venta ...
12	10	2000.0000	2023-07-06 ...	2023-11-06 ...	Alquiler ...
13	6	3000.0000	2023-06-08 ...	2023-12-01 ...	Alquiler ...
14	5	2500.0000	2023-07-01 ...	2023-09-01 ...	Alquiler ...
15	18	40000.0000	2023-08-06 ...	2024-08-06 ...	Anticretico ...
16	21	3500.0000	2023-06-12 ...	2023-09-12 ...	Alquiler de ...
7	24	60000.0000	2023-07-26 ...	2023-10-01 ...	Venta ...
NULL	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*

Tabla BITACORA_CONTRATO

ID	COD_CONT...	COD_OFER...	PAGO_CON...	FECHA_INI...	FECHA_FIN	FECHA_EVE...	EVENTO
1	16	21	3000.0000	2023-06-12 ...	2023-09-12 ...	2023-06-13 ...	Actualizacio...
2	16	21	3500.0000	2023-06-12 ...	2023-09-12 ...	2023-06-13 ...	Actualizacio...
NULL	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*

Tabla CONTRATO después del DELETE

COD_CONT...	COD_OFER...	PAGO_CTRT	FECHA_INI_...	FECHA_FIN...	DETALLE_C...
1	1	1500.0000	2023-02-01 ...	2023-03-31 ...	alquier de u...
2	2	20000.0000	2023-02-01 ...	2023-03-31 ...	venta de ca...
3	3	30000.0000	2023-03-01 ...	2024-05-31 ...	Venta ...
4	12	35000.0000	2023-02-28 ...	2023-04-01 ...	Venta ...
5	13	20000.0000	2023-01-15 ...	2023-03-15 ...	Anticretico ...
6	25	35000.0000	2023-05-24 ...	2024-05-24 ...	Venta ...
8	14	20000.0000	2023-05-25 ...	2023-12-28 ...	Anticretico ...
9	11	1800.0000	2023-05-26 ...	2023-08-26 ...	Alquiler ...
10	22	2000.0000	2023-07-02 ...	2023-10-10 ...	Alquiler ...
11	17	60000.0000	2023-06-25 ...	2023-06-27 ...	Venta ...
12	10	2000.0000	2023-07-06 ...	2023-11-06 ...	Alquiler ...
13	6	3000.0000	2023-06-08 ...	2023-12-01 ...	Alquiler ...
14	5	2500.0000	2023-07-01 ...	2023-09-01 ...	Alquiler ...

14	5	2500.0000	2023-07-01 ...	2023-09-01 ...	Alquiler ...
15	18	40000.0000	2023-08-06 ...	2024-08-06 ...	Anticretico ...
7	24	60000.0000	2023-07-26 ...	2023-10-01 ...	Venta ...
NULL	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*

Tabla BITACORA_CONTRATO

ID	COD_CONT...	COD_OFER...	PAGO_CON...	FECHA_INI...	FECHA_FIN	FECHA_EVE...	EVENTO
1	16	21	3000.0000	2023-06-12 ...	2023-09-12 ...	2023-06-13 ...	Actualizacio...
2	16	21	3500.0000	2023-06-12 ...	2023-09-12 ...	2023-06-13 ...	Actualizacio...
3	16	21	3500.0000	2023-06-12 ...	2023-09-12 ...	2023-06-13 ...	Eliminacion ...
NULL	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*	*NULL*