Base de datos - Teoría

T2-Control de acceso.

1. Introducción.
    a. Ley Orgánica de Protección de Datos de Carácter Personal (LOPD)
        i. Garantiza las libertades y derechos de las personas en lo referente a datos personales.
        ii. Raza, religión, datos médicos, condenas penales, morosidad… Son datos protegidos
        iii. Agencia Española de Protección de Datos (AEPD) es la organización encargada.
    b. El administrador de base de datos es el encargado de la seguridad e integridad de los datos.
        i. Seguridad: Protección de los datos contra
            1. Revelación
            2. Alteración
            3. Destrucción
        ii. Integridad: exactitud o validez de los datos.
        iii. La cuenta de super-usuario debe estar bien protegida.
        iv. Debe tener control de quien puede acceder y operar los datos de la base de datos.
2. Gestión de privilegios en SQL.
    a. Crear usuarios
        i. CREATE USER “user” IDENTIFIED BY “pass”;
    b. Borrar usuarios
        i. DROP USER “user”;
    c. Conceder privilegios:
        i. GRANT “privilegio” ON “base/tabla” TO “user”;
    d. Retirar privilegios
        i. REVOKE “privilegio” ON “base/tabla” FROM “user”;
    e. Ver permisos
        i. SHOW GRANTS;
        ii. SHOW GRANTS FOR “user”;
    f. Propagación de privilegios.
        i. WITH GRANT OPTION
3. Roles.
    a. Muchos usuarios pueden necesitar exactamente los mismos permisos, para ellos usamos los roles.
    b. Crear un rol:
        i. CREATE ROLE “rol”;
    c. Asignar privilegios a un rol
        i. GRANT “privilegio” ON “base/tabla” TO “rol”
    d. Asignar rol a un usuario
        i. GRANT “rol” TO “user”;
    e. Activar rol
        i. SET ROLE “rol”;
    f. Mostrar rol
        i. SELECT current_role();
    g. Rol por defecto

        i. SET DEFAULT ROLE “rol” TO “user”
    h. Mostrar privilegios de un usuario con un rol activo
        i. SHOW GRANTS FOR “user” USING “rol”;
    i. Retirar privilegios de un rol
        i. REVOKE “privilego” ON “base/tabla” FROM “rol”;
    j. Eliminar rol
        i. DROP “rol”;
4. Vistas y procedimientos almacenados.
    a. Vistas
        i. Consultar almacenadas que permiten limitar la visibilidad de los datos
        ii. Crear vista
            1. CREATE VIEW “nombre” AS “select”;
        iii. Borrar vista
            1. DROP VIEW “nombre”;
    b. Rutina – Complican el mantenimiento
        i. Porciones de código SQL reutilizables
        ii. No se deben utilizar para implementar lógica de negocios
        iii. Dos tipos:
            1. Procedimientos
                a. Cambiar delimitador
                    i. DELIMITER //
                b. Crear procedimiento
                    i. CREATE PROCEDURE nombre(parámetros)
BEGIN
<código>
END //
                c. Cambiar delimitador
                    i. DELIMITER ;
                d. Ver proceso
                    i. SHOW PROCEDURE STATUS WHERE DB=”base”
                e. Llamar procedimiento
                    i. CALL “procedimiento”();
            2. Funciones
5. Otras técnicas.
    a. DOS
        i. Crear muchas conexiones a la base de datos para bajar el rendimiento o bloquearla
        ii. Limitar el uso de los usuarios MAX_QUERIES_PER_HOUR
    b. SQL-Injection
        i. Inyectar código malicioso en una consulta SQL
        ii. Sanear mediante parametrización el contenido de la consulta

T3-Auditoria

1. Auditoria en SGBD
    a. Introducción
        i. Definición
            1. Es el proceso que consiste en recoger, agrupar y evaluar evidencias

a. Salvaguarda el activo empresarial
b. Mantiene la integridad de los datos
c. Cumple con las leyes y regulaciones
d. Utiliza eficientemente los recursos

ii. Tipos
1. Rutinaria
a. Objetivo
i. Garantizar la información y verificar controles para prevenir riegos y posibles impactos
b. Actividades
i. Revisar usuarios y permisos
ii. Revisar configuración SGDB
2. Forense
a. Objetivo
i. Encontrar problemas o fallos
b. Actividad
i. Revisar logs/registros

iii. Requisitos
1. No puede impedir el funcionamiento normal del sistema
2. Debe ser útil relativo a seguridad, apoyo a las decisiones y al funcionamiento de la empresa
3. Que no sea quien las implemente el que lo revise

iv. Acciones para auditar
1. Usuarios
2. Procesos
3. Comandos

b. Herramientas
i. Logs de la BBDD
1. Ventajas
a. Nivel detallado y muy configurable
2. Desventajas
a. Puede provocar problemas de almacenamientos y es lento de analizar

ii. Aplicaciones de 3ºs
1. Ventajas
a. Facilita extracción de información
b. Combinan información de diferentes sistemas.
2. Desventajas
a. Posible costo económico
b. Posible impacto en rendimiento

iii. Triggers
1. Saltan después de un evento concreto
2. Dos tipos a nivel SQL
a. Nivel fila: Por cada inserción modificación o eliminación en una tabla
b. Nivel de sentencia: Por cada transacción independientemente del número de filas.
3. MySQL solo soporta a nivel de fila.
4. Ventajas
a. Vigila la integridad de los datos

        b. Alternativa a tareas programadas
    5. Inconvenientes
        a. No permite hacer validaciones
        b. Difíciles de depurar
        c. No se debe usar lógica de aplicación en ellas
        d. Mayor carga en MySQL
    6. Crear Triggers
        a. CREATE TRIGGER “nombre” “cuando” “acción” on “tabla” FOR EACH ROW “acción”
    7. Ver Triggers
        a. SHOW TRIGGERS;
    8. Borrar Trigger
        a. DROP Trigger “nombre”
2. Auditoria y Blockchain
    a. Introducción a Blockchain
        i. Base de datos con un gran control de que y quien modifica sus datos.
        ii. Se presento junto al Bitcoin como su libro de cuentas
        iii. Basada en una base de datos distribuida
        iv. Múltiples implementaciones más haya de bitcoin
        v. Esta formada por nodos que contienen una copia completa o parcial de la base de datos y sus transacciones
        vi. Las operaciones se validan mediante un hash el cual no es mas que un numero decimal puesto en forma hexadecimal que debe ser inferior a un cierto valor puesto por la Blockchain
        vii. El minado es el tiempo que cada bloque necesita para encontrar dicho hash inferior al número de umbral el cual hace que tarde mas o menos tiempo el cual varía según la Blockchain que usemos.
        viii. El nonce es un numero usado en el algoritmo Proof-of-work para validad y proteger junto con el hash de ciertos ataques.
        ix. La red funciona así:
            1. Cada transacción se envía a todos los nodos.
            2. Cada nodo recopila las nuevas transacciones en un nuevo bloque.
            3. Cada nodo busca el nonce para su bloque
            4. Cuando el nonce es encontrado envía su bloque al resto de nodos
            5. El resto de los nodos aceptan el bloque y validan las transacciones
            6. Una vez aceptado se empieza a trabajar en el siguiente siempre teniendo en cuenta el hash del nodo anterior.
        x. Si dos nodos envían informaciones diferentes se reciben ambas y se espera a encontrar en nonce para validar cual era la verdadera.
        xi. Este método requiere mucha energía
        xii. Se empieza a migrar a proof-of-stake
            1. Un sistema de pujas
            2. Menor impacto ambiental.
    b. Redes Blockchain publicas vs privadas.
        i. Publicas

            1. Cualquier persona puede participa
            2. Descentralizadas
        ii. Privadas
            1. No está abierta al publico
            2. Suelen estar gestionadas por empresas

T4-Recuperación

1. Introducción
    a. El SGBD debe asegurarse de que los datos están disponibles y dar mecanismos para que en caso de error estos sean recuperables
    b. Dos situaciones para considerar
        i. Recuperación a nivel de tabla/BD
            1. Copia de seguridad y log binario
        ii. Recuperación a nivel de transacción
            1. Log de recuperación
2. Transacciones
    a. Conjunto de operaciones que llevan la BD de un estado a otro.
    b. Se considera
        i. Unidad lógica de procesamiento
        ii. Unidad lógica de integridad
        iii. Unidad lógica de recuperacion
    c. El SGBD se encarga de la ejecución de las transacciones
    d. Deben cumplicas las propiedades ACID
        i. Atomicity
            1. Una transacción no puede ejecutarse a medias
        ii. Consistency
            1. Una transacción siempre parte de un estado consistente a otro
        iii. Isolation
            1. La ejecución de una transacción no puede interferir en la ejecución de otra transacción
        iv. Durability
            1. Los cambios realizados por una transacción confirmada deben persistir en la base de datos.
3. El diario de recuperación
4. Algoritmos de recuperación
5. Estructura de InnoDB
    a. Motor de almacenamiento
    b. Provee un buen balance entre fiabilidad y rendimiento
        i. Soporta las propiedades ACID
        ii. Control de concurrencia multi-usuario
    c. Estructuras en memoria
        i. Buffer Pool
            1. Cache de datos
            2. Gran velociada
            3. Organiza los valores por menos recientemente utilizado
            4. Ocupa 128MB
        ii. Log Buffer
            1. Contiene los datos de las transacciones que se deben de escribir en el Redo Log
            2. Por defecto 16MB

d. La configuración de InnoDB afecta a MySQL
e. Mayor bolcado del Buffer Pool significa mayor rendimiento y mayor tiempo de carga

T5-Control de concurrencia

1. Motivación
    a. Hoy en día varios usuarios suelen hacer uso de los mismos datos.
    b. Se debe asegurar que cada transacción se ejecuta de forma independiente del resto (Propiedad Isolation de ACID)
    c. Concurrencia
        i. Un procesador ejecuta varios procesos de forma simultánea.
    d. Paralelismo
        i. Varios procesadores ejecutan varios procesos de forma simultánea.
2. Problemas de concurrencia
    a. Ejecutar 2 o mas transacciones puede crear problemas de concurrencia.
    b. Perdida de actualizaciones
        i. La T2 no lee el valor que ha actualizado antes T1
    c. Actualizaciones temporales (Lectura de valores sucios)
        i. T1 actualiza un valor que luego T2 lee pero T1 hace rollback lo que provoca que T1 tenga su valor original pero T2 haya leído el erróneo.
    d. Totales incorrectos (Lectura de valores fantasma)
        i. T2 usa un valor que posteriormente es actualizado por T1 lo que hace que T2 no trabaje con el ultimo valor de la variable.
    e. Lectura inconsistente
        i. Cada vez que lees el valor en una transacción el valor cambia.
3. Planes de transacciones
    a. Un plan de transacciones es una serie de operaciones intercaladas que se ejecutan de forma concurrente.
    b. En un plan puede haber conflictos si:
        i. Las operaciones pertenecen a transacciones diferentes
        ii. Acceden a la misma variable/elemento.
        iii. Al menos una de las operaciones es write.
    c. Al permitir intercalar operaciones existen muchos órdenes.
        i. Alto riesgo de conflicto.
    d. Una opción para arreglarlo es planificaciones serie.
        i. Plan donde las operaciones de cada transacción se ejecutan consecutivamente.
        ii. Las operaciones de las transacciones no se intercalan.
        iii. Libres de conflictos.
        iv. Siempre correcta
        v. Poco eficiente
        vi. Esto es posible si para las mismas transacciones produce un resultado equivalente
4. Protocolos de serialización
    a. Existen 4 tipos de protocolos basados en las reglas de la teoría de la serialización.
        i. Basados en reservas
            1. Se basan en asegurar a través de reservas (locks)

2. Esto no garantiza la serialización de los planes
3. Es necesario seguir un protocolo que indique donde se deben colocar las reservas y sus liberaciones
4. Para eso se usa el 2PL
    a. Dos fases
        i. Fase de expansión
            1. Reservas
        ii. Fase de contracción
            1. Liberaciones
    b. Ventajas:
        i. Es serializable
    c. Limitar la concurrencia
    d. Puede generar otros problemas
        i. Deadlock
        ii. Starvation
    e. Variantes
        i. Conservador
            1. Se realizan todos los bloqueos arriba.
            2. Si no es posible hacer la reserva se espera y se intenta más tarde
            3. No tiene deadlock
        ii. Estricto
            1. No se libera ninguna reserva de escritura hasta el final.
ii. Basados en marcas de tiempo
iii. De concurrencia multiversión
iv. Optimistas.

5. Técnicas de control de concurrencia
    a. Hasta ahora solo hemos considerado que las transacciones solo leen y modifican objetos pero en realidad también pueden crear y eliminar objetos.
    b. Esta creación o eliminación producen lecturas fantasmas:
        i. Solución:
            1. Re-ejecutar lecturas completas
                a. Se registran las clusulas where de todas las operaciones
                b. Al hacer commit se ejecutan esas porciones where y se verifica que coinciden con lo registrado.
                c. Esto degrada el rendimiento especialmente si las tablas están en disco.
            2. Reservas/locks de predicados
                a. Consiste en hacer reservas a las clausulas where sobre un espacion concreto, no sobre toda la fila
                b. Muy difícil de implementas.
            3. Reservas/locks sobre índices
                a. Los índices contienen metainformacion sobre el contenido de la tabla.

b. Se pueden realizar reservas sobre índices para evitar que se modifiquen y sobre un rango de huecos

6. Niveles de aislamiento
    a. El estándar SQL define diferentes niveles de aislamiento.
    b. Esto permite ajustar si los planes son serializables.
        i. A mas estricto, menor es el rendimiento.
    c. Controla el grado al que se expone una transacción a posibles concurrencias.
    d. Mayor grado de libertad provoca
        i. Lecturas fantasmas
        ii. Lecturas irrepetibles
        iii. Lecturas sucias.

	Lecturas sucias	Lecturas irrepetibles	Lecturas fantasma
Serializable	No	No	No
Repeatable Reads	No	No	Posible
Read Committed	No	Posible	Posible
Read Uncommitted	Posible	Posible	Posible

    e.
    f. Serializable
        i. Equivale a 2PL estático y reservas sobre índices
    g. Repeatable Reads
        i. Equivalente a Serializable pero sin usar reservas sobre índices
    h. Read Committed
        i. Equivalente a Repeatable Reads pero los RLOCKs se liberan inmediatamente
    i. Read Uncommitted
        i. Igual que Read Committed pero sin usar RLOCKs

T6 – Optimización

1. Índices
    a. Definición
        i. Estructura de datos que acelera la obtención de los datos
        ii. Se utilizan en ciertas condiciones de búsquedas
        iii. Se puede construir sobre 1 o varios campos de una tabla.
        iv. Existen diferentes tipos
            1. B-Tree, Hash, GiST…
        v. El tipo más común es el B-Tree que ordena todo como si fuera un árbol y evita tener que recorrer todos los valores como tendríamos que hacerlo si fuese una columna. Este tipo de estructuras funcionan para la mayoría de los tipos de datos.
    b. Desventajas
        i. Ocupan mucho espacio ya que se guardan como ficheros en disco y por tanto cuantos mas datos haya en la columna mayor será el indice

        ii. Retrasan también las operaciones INSERT UPDATE y DELETE ya que no solo tienen que actualizar la tabla si no también el indice
        iii. Dependiendo de la consulta puede que el indice no se aproveche o ni se use.
    c. Índices B-Tree
        i. Los B-Trees son una familia de estructuras de datos
        ii. Tipos de B-Trees:
            1. B-Tree
            2. B+Tree
            3. B*Tree
            4. Blink-Tree
    d. Índices B+Tree
        i. Son estructuras de datos ordenados y que permiten hacer operaciones con coste logarítmico
        ii. Son básicamente una generalización del árbol de búsqueda binaria
        iii. Se considera una estructura de M caminos que cumple las siguientes propiedades:
            1. Balanceado, es decir, todos los nodos tienen la misma profundidad.
            2. Todos los nodos menos la raíz tienen M/2-1 a M-1 elementos
            3. Todos los nodos intermedios con k elementos tienen k+1 hijos no nulos.
        iv. Algoritmo de inserción de un elemento E
            1. Encontrar el nodo hoja H que le corresponde
            2. Inserta E en H
            3. ¿H tiene espacio para contener E?
                a. Si -> Fin
                b. No -> Dividir las claves de H entre H y un nuevo nodo hoja H2
        v. Algoritmo de eliminar un elemento E
            1. Encontrar el nodo hoja H que contiene E.
            2. Eliminar E.
            3. ¿H contiene más de M/2-1 elementos?
                a. Si -> Fin
                b. No: Re-distribuir elementos, cogiendo del nodo hoja cercano. Si no fuera posible fusionar H con el nodo hoja cercano.
2. Procesamiento de consultas
    a. Definición
        i. No es recomendable usar indices sin enderezar el impacto que estos pueden tener en el SGBD ya que SQL al ser un lenguaje declarativo no define como se recuperan los datos, solo cuales se recuperan o modifican.
        ii. Por ello, para hacer una consulta el SGBD para por las siguientes fases:
            1. Análisis gramáticos (“parse”)
                a. Divide la consulta en palabras y verifica que la sintaxis es correcta y que hace referencia a algo

que realmente existe en la base de datos o sistema de gestión de bases de datos. Y si esta correcta se crea un árbol sintáctico donde se representan los objetos y operaciones.

2. Optimización
    a. El optimizadle es el que usa el árbol sintáctico creado en el apartado anterior y toma las siguientes decisiones:
        i. Determinar que operaciones Join se deben hacer.
        ii. Determinar si es mejor usar índices o “table scan”
        iii. Determinar el orden de las operaciones.
    b. El resultado de este optimizadle es un plan de ejecución.
    c. El optimizador también se encargará de calcular el coste de cada alternativa el cual sirve para ver la complejidad de los diferentes plantes y elegir el de menor coste.
3. Ejecución

3. Estimación de costes
    a. Descripción:
        i. El coste estimado es una aproximación de la complejidad de una operación la cual se basa en estadísticas y datos de la consulta además de valores heurísticos del SGBD
        ii. Los datos estadísticos se obtienen del diccionario de datos que es una tabla del SGBD que actualiza las estadísticas en cada operación.
            1. En MySQL se llama Information_Schema

T7- NoSQL y MongoDB

1. Introducción a NoSQL
    a. Descripción
        i. En ciertas situaciones las bases de datos SQL resultan demasiado rígidas, por ejemplo, en aplicaciones de mensajería las longitudes de los mensajes pueden variar mucho y puede ser que los mensajes adjunten archivos.
        ii. Los sistemas NoSQL ofrecen una alternativa a los SQL con las siguientes características
            1. No requieren esquemas para definir los datos
            2. Es fácilmente escalable de forma horizontal, es decir se pueden añadir nodos fácilmente
            3. Y tiene particionado horizontal también conocido como sharding que favorece el acceso concurrente ya que los datos se distribuyen entre los nodos disponibles.
        iii. Existen diferentes categorías de sistemas NoSQL
            1. Basados en documentos
                a. Usan formatos JSON como puede ser MongoDB
            2. Clave-valor
                a. Cada dato tiene una clave como si fuera un HashMap, por ejemplo, Redis

3. En columnas
    a. Organizan los datos en filas por ejemplo Apache Druid
4. Basados en grafos
    a. Los datos se representan como grafos por ejemplo Neo4j

2. MongoDB
    a. Instalación
        i. Importar la clave pública del repositorio en apt.
        ii. Crear un fichero de listado para MongoDB.
        iii. Recargar el listado de paquetes disponibles en apt.
        iv. Instalar usando apt.
    b. Estructura de los datos
        i. Se organizan en formato JSON
        ii. Cada dato se almacena como clave: valor
        iii. La equivalencia con un sistema relacional seria:
            1. Tabla=Colección
            2. Fila=Documento
            3. Columna/Campo=Clave
        iv. Las principales diferencias son:
            1. No todos los documentos tienen las mismas claves
            2. No es necesario definir relaciones.
    c. Comandos CRUD
    d. Relaciones
        i. MongoDB no proporciona una técnica concreta para definir relaciones entre colecciones
        ii. Dos formas
            1. Documentos embebidos
                a. Adecuado para datos que no se solapan
                b. Tiene una agrupación lógica
                c. Pero pueden generar duplicidades
            2. Utilizar campos como referencias
                a. Adecuado para datos que se referencian entre si
                b. Elimina posibles duplicidades
                c. Complejo de gestionar
                d. Requieren agregaciones para obtener todos los datos.
        iii. Cuando usar cada uno
            1. Documentos embebidos
                a. Datos que estén fuertemente relaciones y sin duplicados
                b. Relaciones 1 a 1
                c. Relaciones 1 a N sin duplicados
            2. Referencias
                a. Datos de entidades independientes pero relacionadas
                b. Relaciones 1 a N
                c. Relaciones N a N
    e. Esquemas

i. Es posible usan MongoDB sin definir ninguna estructura, pero estas son utilizadas para controlar los datos de forma automática.
ii. Se puede definir la estructura que debe tener un documento y el esquema verifica que esa estructura se cumple y que los datos no son incorrectos.
iii. Dos opciones para controlar:
1. validationLevel
a. Controla como de estrica es la validación
i. Strict: Se comprueba toda inserción y modificación
ii. Moderate: Las modificaciones a documentos ya existentes no se comprueban
2. validationAction
a. Indica que hacer cuando un documento no cumple el esquema
i. Error: Se emite un error y se impide la inserción
ii. Warn: Se escribe un aviso en el log y se permite la inserción.

f. Índices
i. Se define para uno o varios campos de una colección
ii. Tienen las mismas desventajas que en los sistemas relaciones

g. Explain
i. La función Explain proporciona información sobre tiempos y plan de ejecución de una consulta.

h. Colecciones limitadas
i. Son colecciones con un límite máximo de documentos
ii. Llegar al límite de un documento borra el más antiguo
iii. Útiles cuando no interesa conservar los datos mas antiguos y ahorrar espacio en disco

i. Usuarios y roles
i. Necesario gestionar la actividad mediante usuarios y permisos
ii. En mongo se hace a través de roles
iii. Un rol es un conjunto de permisos
iv. Los roles predefinidos

Categoría	Rol	Permisos
Usuario de 1 BD	read	Leer todas las colecciones
	readWrite	Leer y modificar todas las colecciones
Administrador de 1 BD	userAdmin	Gestionar usuarios
	dbOwner	Equivale a userAdmin + readWrite
Administrador de todas las BBDD	readAnyDatabase	Leer cualquier colección
	readWriteAnyDatabase	Leer y modificar cualquier colección
	root	Control total

v. Por defecto Mongo no tiene ningún usuario ni control de usuario y solo permite conexiones locales.
vi. Un usuario se crea vinculado a una BD concreto y el usuario se autentica contra es BD lo cual no impide al usuario acceder a otras siempre que tenga los permisos

j. Copias de seguridad
    i. Dos formas de hacerlo
        1. Utilizando la nube
            a. Requiere contratar
        2. Manual
            a. Mongodump realiza una copia de una DB en ficheros BSON
            b. Mongorestore restaura una copia hecha con mongodump
            c. Puedes copiar la carpeta que contienen los datos con otras herramientas, pero ocupan más que con mongodump

k. Sharding
    i. Hasta ahora hemos trabajado con una única instancia de MongoDB pero este esta pensado para tener múltiples nodos distribuidos
    ii. Cada shard contiene una porción de los datos
    iii. El punto de acceso a una instalación distribuida de MongoDB es mongos
    iv. Los documentos de las colecciones se distribuyen entre los Shards en base a una Shard key
        1. Es uno de los campos del documento elegida por el administrador
        2. Deben elegirse con cuidado ya que afecta al rendimiento
    v. 2 escenarios posibles en unas instalaciones
        1. Consulta sin uso de Shard Key
            a. Mongos manda la consulta a todos los Shards lo que satura la red
        2. Consulta usando Shard Key
            a. Mongos envía la consulta al Shard que contiene la respuesta
    vi. La instalación distribuida de MongoDB se considera para atender consultas con muy baja latencia y conjuntos de datos muy grandes
    vii. Esta es muy rápida pero compleja de gestionar.