Base de datos - Teoría

# T2-Control de acceso.

1. Introducción.
   1. Ley Orgánica de Protección de Datos de Carácter Personal (LOPD)
      1. Garantiza las libertades y derechos de las personas en lo referente a datos personales.
      2. Raza, religión, datos médicos, condenas penales, morosidad… Son datos protegidos
      3. Agencia Española de Protección de Datos (AEPD) es la organización encargada.
   2. El administrador de base de datos es el encargado de la seguridad e integridad de los datos.
      1. Seguridad: Protección de los datos contra
         1. Revelación
         2. Alteración
         3. Destrucción
      2. Integridad: exactitud o validez de los datos.
      3. La cuenta de super-usuario debe estar bien protegida.
      4. Debe tener control de quien puede acceder y operar los datos de la base de datos.
2. Gestión de privilegios en SQL.
   1. Crear usuarios
      1. CREATE USER “user” IDENTIFIED BY “pass”;
   2. Borrar usuarios
      1. DROP USER “user”;
   3. Conceder privilegios:
      1. GRANT “privilegio” ON “base/tabla” TO “user”;
   4. Retirar privilegios
      1. REVOKE “privilegio” ON “base/tabla” FROM “user”;
   5. Ver permisos
      1. SHOW GRANTS;
      2. SHOW GRANTS FOR “user”;
   6. Propagación de privilegios.
      1. WITH GRANT OPTION
3. Roles.
   1. Muchos usuarios pueden necesitar exactamente los mismos permisos, para ellos usamos los roles.
   2. Crear un rol:
      1. CREATE ROLE “rol”;
   3. Asignar privilegios a un rol
      1. GRANT “privilegio” ON “base/tabla” TO “rol”
   4. Asignar rol a un usuario
      1. GRANT “rol” TO “user”;
   5. Activar rol
      1. SET ROLE “rol”;
   6. Mostrar rol
      1. SELECT current\_role();
   7. Rol por defecto
      1. SET DEFAULT ROLE “rol” TO “user”
   8. Mostrar privilegios de un usuario con un rol activo
      1. SHOW GRANTS FOR “user” USING “rol”;
   9. Retirar privilegios de un rol
      1. REVOKE “privilego” ON “base/tabla” FROM “rol”;
   10. Eliminar rol
       1. DROP “rol”;
4. Vistas y procedimientos almacenados.
   1. Vistas
      1. Consultar almacenadas que permiten limitar la visibilidad de los datos
      2. Crear vista
         1. CREATE VIEW “nombre” AS “select”;
      3. Borrar vista
         1. DROP VIEW “nombre”;
   2. Rutina – Complican el mantenimiento
      1. Porciones de código SQL reutilizables
      2. No se deben utilizar para implementar lógica de negocios
      3. Dos tipos:
         1. Procedimientos
            1. Cambiar delimitador

DELIMITER //

* + - * 1. Crear procedimiento

CREATE PROCEDURE nombre(parámetros)  
BEGIN  
<código>  
END //

* + - * 1. Cambiar delimitador

DELIMITER ;

* + - * 1. Ver proceso

SHOW PROCEDURE STATUS WHERE DB=”base”

* + - * 1. Llamar procedimiento

CALL “procedimiento”();

* + - 1. Funciones

1. Otras técnicas.
   1. DOS
      1. Crear muchas conexiones a la base de datos para bajar el rendimiento o bloquearla
      2. Limitar el uso de los usuarios MAX\_QUERIES\_PER\_HOUR
   2. SQL-Injection
      1. Inyectar código malicioso en una consulta SQL
      2. Sanear mediante parametrización el contenido de la consulta

# T3-Auditoria

1. Auditoria en SGBD
   1. Introducción
      1. Definición
         1. Es el proceso que consiste en recoger, agrupar y evaluar evidencias
            1. Salvaguarda el activo empresarial
            2. Mantiene la integridad de los datos
            3. Cumple con las leyes y regulaciones
            4. Utiliza eficientemente los recursos
      2. Tipos
         1. Rutinaria
            1. Objetivo

Garantizar la información y verificar controles para prevenir riegos y posibles impactos

* + - * 1. Actividades

Revisar usuarios y permisos

Revisar configuración SGDB

* + - 1. Forense
         1. Objetivo

Encontrar problemas o fallos

* + - * 1. Actividad

Revisar logs/registros

* + 1. 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
    2. Acciones para auditar
       1. Usuarios
       2. Procesos
       3. Comandos
  1. Herramientas
     1. Logs de la BBDD
        1. Ventajas
           1. Nivel detallado y muy configurable
        2. Desventajas
           1. Puede provocar problemas de almacenamientos y es lento de analizar
     2. Aplicaciones de 3ºs
        1. Ventajas
           1. Facilita extracción de información
           2. Combinan información de diferentes sistemas.
        2. Desventajas
           1. Posible costo económico
           2. Posible impacto en rendimiento
     3. Triggers
        1. Saltan después de un evento concreto
        2. Dos tipos a nivel SQL
           1. Nivel fila: Por cada inserción modificación o eliminación en una tabla
           2. Nivel de sentencia: Por cada transacción independientemente del número de filas.
        3. MySQL solo soporta a nivel de fila.
        4. Ventajas
           1. Vigila la integridad de los datos
           2. Alternativa a tareas programadas
        5. Inconvenientes
           1. No permite hacer validaciones
           2. Difíciles de depurar
           3. No se debe usar lógica de aplicación en ellas
           4. Mayor carga en MySQL
        6. Crear Triggers
           1. CREATE TRIGGER “nombre” “cuando” “acción” on “tabla” FOR EACH ROW “acción”
        7. Ver Triggers
           1. SHOW TRIGGERS;
        8. Borrar Trigger
           1. DROP Trigger “nombre”

1. Auditoria y Blockchain
   1. Introducción a Blockchain
      1. Base de datos con un gran control de que y quien modifica sus datos.
      2. Se presento junto al Bitcoin como su libro de cuentas
      3. Basada en una base de datos distribuida
      4. Múltiples implementaciones más haya de bitcoin
      5. Esta formada por nodos que contienen una copia completa o parcial de la base de datos y sus transacciones
      6. 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
      7. 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.
      8. El nonce es un numero usado en el algoritmo Proof-of-work para validad y proteger junto con el hash de ciertos ataques.
      9. 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.
      10. Si dos nodos envían informaciones diferentes se reciben ambas y se espera a encontrar en nonce para validar cual era la verdadera.
      11. Este método requiere mucha energía
      12. Se empieza a migrar a proof-of-stake
          1. Un sistema de pujas
          2. Menor impacto ambiental.
   2. Redes Blockchain publicas vs privadas.
      1. Publicas
         1. Cualquier persona puede participa
         2. Descentralizadas
      2. Privadas
         1. No está abierta al publico
         2. Suelen estar gestionadas por empresas

# T4-Recuperación

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

# T5-Control de concurrencia

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

Fase de expansión

Reservas

Fase de contracción

Liberaciones

* + - * 1. Ventajas:

Es serializable

* + - * 1. Limitar la concurrencia
        2. Puede generar otros problemas

Deadlock

Starvation

* + - * 1. Variantes

Conservador

Se realizan todos los bloqueos arriba.

Si no es posible hacer la reserva se espera y se intenta más tarde

No tiene deadlock

Estricto

No se libera ninguna reserva de escritura hasta el final.

* + 1. Basados en marcas de tiempo
    2. De concurrencia multiversión
    3. Optimistas.

1. Técnicas de control de concurrencia
   1. Hasta ahora solo hemos considerado que las transacciones solo leen y modifican objetos pero en realidad también pueden crear y eliminar objetos.
   2. Esta creación o eliminación producen lecturas fantasmas:
      1. Solución:
         1. Re-ejecutar lecturas completas
            1. Se registran las clusulas where de todas las operaciones
            2. Al hacer commit se ejecutan esas porciones where y se verifica que coinciden con lo registrado.
            3. Esto degrada el rendimiento especialmente si las tablas están en disco.
         2. Reservas/locks de predicados
            1. Consiste en hacer reservas a las clausulas where sobre un espacion concreto, no sobre toda la fila
            2. Muy difícil de implementas.
         3. Reservas/locks sobre índices
            1. Los índices contienen metainformacion sobre el contenido de la tabla.
            2. Se pueden realizar reservas sobre índices para evitar que se modifiquen y sobre un rango de huecos
2. Niveles de aislamiento
   1. El estándar SQL define diferentes niveles de aislamiento.
   2. Esto permite ajustar si los planes son serializables.
      1. A mas estricto, menor es el rendimiento.
   3. Controla el grado al que se expone una transacción a posibles concurrencias.
   4. Mayor grado de libertad provoca
      1. Lecturas fantasmas
      2. Lecturas irrepetibles
      3. Lecturas sucias.
   5. Tabla

      Descripción generada automáticamente
   6. Serializable
      1. Equivale a 2PL estático y reservas sobre índices
   7. Repeatable Reads
      1. Equivalente a Serializable pero sin usar reservas sobre índices
   8. Read Committed
      1. Equivalente a Repeatable Reads pero los RLOCKs se liberan inmediatamente
   9. Read Uncommitted
      1. Igual que Read Committed pero sin usar RLOCKs

# T6 – Optimización

1. Índices
   1. Definición
      1. Estructura de datos que acelera la obtención de los datos
      2. Se utilizan en ciertas condiciones de búsquedas
      3. Se puede construir sobre 1 o varios campos de una tabla.
      4. Existen diferentes tipos
         1. B-Tree, Hash, GiST…
      5. 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.
   2. Desventajas
      1. 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
      2. Retrasan también las operaciones INSERT UPDATE y DELETE ya que no solo tienen que actualizar la tabla si no también el indice
      3. Dependiendo de la consulta puede que el indice no se aproveche o ni se use.
   3. Índices B-Tree
      1. Los B-Trees son una familia de estructuras de datos
      2. Tipos de B-Trees:
         1. B-Tree
         2. B+Tree
         3. B\*Tree
         4. Blink-Tree
   4. Índices B+Tree
      1. Son estructuras de datos ordenados y que permiten hacer operaciones con coste logarítmico
      2. Son básicamente una generalización del árbol de búsqueda binaria
      3. 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.
      4. 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?
            1. Si -> Fin
            2. No -> Dividir las claves de H entre H y un nuevo nodo hoja H2
      5. 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?
            1. Si -> Fin
            2. No: Re-distribuir elementos, cogiendo del nodo hoja cercano. Si no fuera posible fusionar H con el nodo hoja cercano.
2. Procesamiento de consultas
   1. Definición
      1. 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.
      2. Por ello, para hacer una consulta el SGBD para por las siguientes fases:
         1. Análisis gramáticos (“parse”)
            1. 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
            1. El optimizadle es el que usa el árbol sintáctico creado en el apartado anterior y toma las siguientes decisiones:

Determinar que operaciones Join se deben hacer.

Determinar si es mejor usar índices o “table scan”

Determinar el orden de las operaciones.

* + - * 1. El resultado de este optimizadle es un plan de ejecución.
        2. 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.
      1. Ejecución

1. Estimación de costes
   1. Descripción:
      1. 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
      2. 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
   1. Descripción
      1. 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.
      2. 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.
      3. Existen diferentes categorías de sistemas NoSQL
         1. Basados en documentos
            1. Usan formatos JSON como puede ser MongoDB
         2. Clave-valor
            1. Cada dato tiene una clave como si fuera un HashMap, por ejemplo, Redis
         3. En columnas
            1. Organizan los datos en filas por ejemplo Apache Druid
         4. Basados en grafos
            1. Los datos se representan como grafos por ejemplo Neo4j
2. MongoDB
   1. Instalación
      1. Importar la clave pública del repositorio en apt.
      2. Crear un fichero de listado para MongoDB.
      3. Recargar el listado de paquetes disponibles en apt.
      4. Instalar usando apt.
   2. Estructura de los datos
      1. Se organizan en formato JSON
      2. Cada dato se almacena como clave: valor
      3. La equivalencia con un sistema relacional seria:
         1. Tabla=Colección
         2. Fila=Documento
         3. Columna/Campo=Clave
      4. Las principales diferencias son:
         1. No todos los documentos tienen las mismas claves
         2. No es necesario definir relaciones.
   3. Comandos CRUD
   4. Relaciones
      1. MongoDB no proporciona una técnica concreta para definir relaciones entre colecciones
      2. Dos formas
         1. Documentos embebidos
            1. Adecuado para datos que no se solapan
            2. Tiene una agrupación lógica
            3. Pero pueden generar duplicidades
         2. Utilizar campos como referencias
            1. Adecuado para datos que se referencian entre si
            2. Elimina posibles duplicidades
            3. Complejo de gestionar
            4. Requieren agregaciones para obtener todos los datos.
      3. Cuando usar cada uno
         1. Documentos embebidos
            1. Datos que estén fuertemente relaciones y sin duplicados
            2. Relaciones 1 a 1
            3. Relaciones 1 a N sin duplicados
         2. Referencias
            1. Datos de entidades independientes pero relacionadas
            2. Relaciones 1 a N
            3. Relaciones N a N
   5. Esquemas
      1. Es posible usan MongoDB sin definir ninguna estructura, pero estas son utilizadas para controlar los datos de forma automática.
      2. 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.
      3. Dos opciones para controlar:
         1. validationLevel
            1. Controla como de estrica es la validación

Strict: Se comprueba toda inserción y modificación

Moderate: Las modificaciones a documentos ya existentes no se comprueban

* + - 1. validationAction
         1. Indica que hacer cuando un documento no cumple el esquema

Error: Se emite un error y se impide la inserción

Warn: Se escribe un aviso en el log y se permite la inserción.

* 1. Índices
     1. Se define para uno o varios campos de una colección
     2. Tienen las mismas desventajas que en los sistemas relaciones
  2. Explain
     1. La función Explain proporciona información sobre tiempos y plan de ejecución de una consulta.
  3. Colecciones limitadas
     1. Son colecciones con un límite máximo de documentos
     2. Llegar al límite de un documento borra el más antiguo
     3. Útiles cuando no interesa conservar los datos mas antiguos y ahorrar espacio en disco
  4. Usuarios y roles
     1. Necesario gestionar la actividad mediante usuarios y permisos
     2. En mongo se hace a través de roles
     3. Un rol es un conjunto de permisos
     4. Los roles predefinidos

Tabla

Descripción generada automáticamente

* + 1. Por defecto Mongo no tiene ningún usuario ni control de usuario y solo permite conexiones locales.
    2. 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
  1. Copias de seguridad
     1. Dos formas de hacerlo
        1. Utilizando la nube
           1. Requiere contratar
        2. Manual
           1. Mongodump realiza una copia de una DB en ficheros BSON
           2. Mongorestore restaura una copia hecha con mongodump
           3. Puedes copiar la carpeta que contienen los datos con otras herramientas, pero ocupan más que con mongodump
  2. Sharding
     1. Hasta ahora hemos trabajado con una única instancia de MongoDB pero este esta pensado para tener múltiples nodos distribuidos
     2. Cada shard contiene una porción de los datos
     3. El punto de acceso a una instalación distribuida de MongoDB es mongos
     4. 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
     5. 2 escenarios posibles en unas instalaciones
        1. Consulta sin uso de Shard Key
           1. Mongos manda la consulta a todos los Shards lo que satura la red
        2. Consulta usando Shard Key
           1. Mongos envía la consulta al Shard que contiene la respuesta
     6. La instalación distribuida de MongoDB se considera para atender consultas con muy baja latencia y conjuntos de datos muy grandes
     7. Esta es muy rápida pero compleja de gestionar.