BASE DE DATOS – RESUMEN

Sumario

Clase 1:

Definiciones:

- **Dato:** Representación atómica de algo
- **Base de Datos:** Conjunto de información organizado
- **Sistema de Gestión de Base de Datos (SGBD):** Es el sistema que realiza las s de persistencia, consulta y actualización de los datos de una Base de Datos
- **Transacción:** Conjunto de operaciones sobre los datos almacenados en un SGBD que para garantizar la consistencia de los datos deben ejecutarse todas o ninguna de las operaciones que la forman

Funciones de un SGBD Relacional:

- Permitir la definición de los datos y su manipulación
- Permitir configurar la seguridad y monitoreo de los datos
- Asegurar la integridad de los datos
- Permitir la recuperación de los datos y administrar la concurrencia

Tipos de Usuarios:

- **Finales:** Usan la base de datos para alcanzar algún objetivo
- **Programadores de Aplicaciones:** Desarrollan software para proveer a los usuarios finales una interfaz con la base de datos para que puedan interactuar con la información
- **Administración de Base de Datos (DBA):** Hacer las tareas de instalación y administración del SGBD y las BBDD.
- **Programador de SGBD:** Desarrollan el software del SGBD

Opciones básicas Sobre los Datos:

- **Recuperación**
- **Creación ó Inserción**
- **Actualización**
- **Eliminación**

Estructuras Jerárquicas de la Información:

- JSON
- XML

Comparaciones entre JSON Y XML:

- XML y JSON tienen el mismo poder expresivo
- XML suele ser más largo
- XML permite diferenciar entre data y metadata
- JSON tiene una sintaxis específica para listas

Distintos Paradigmas para SGBD:

Modelo Relacional:

- Los datos se organizan en tablas que se relacionan entre sí
- Estructura fija pre-diseñada
- Aseguran un ACID

Modelo Key-Value:

- Los datos se guardan en largos diccionarios
- No hay estructura ni relaciones
- Las claves pueden ser “strings, hashes, lists, sets, sorted sets, bitmaps, etc”

Column:

- Los datos se almacenan en registros con una clave que vincula los datos del registro
- Los registros son conjuntos **Key-Value**
- No hay estructura, cada registro puede tener diferentes columnas
- Escalan bien

Document:

- Los datos se almacenan en documentos con una clave
- Los documentos son estructuras simples ó muy complejas, permitiendo anidación de documentos
- Se basan en estándares como JSON
- Son menos performantes, pero permiten estructuras más complejas

Graph:

- Usa nodos y arcos para organizar la información}

ACID:

- **Atomicity:** Asegura que las transacciones se ejecutan completamente, ó no se ejecutan

- **Consistency:** Asegura que se verifican las reglas de consistencia indicadas en el diseño de la BBDD, y que los usuiarios recuperan al mismo tiempo la misma información
- **Isolation:** Asegura que cada transacción se ejecuta de manera aislada e independiente de las otras transacciones
- **Durability:** Asegura que una vez que la transacción se completa, los datos se registran de manera permanente

El Modelo Relacional:

- Modelo formal que representa los datos mediante el uso de tablas y prescribe una forma de administrarlos

Aspectos del Modelo Relacional

- Estructura
- Integridad
- Manipulación

Elementos del Modelo Relacional

- Relaciones
- Dominios
- Tuplas
- Atributos
- Cardinalidad
- Grado
- Claves

Características de las Relaciones

- Los valores son atómicos
- Cada tupla es única
- Los valores de un atributo son todos del mismo dominio
- El orden de los atributos y/o tuplas no es significativo
- Cada atributo tiene un nombre único

Notación Relacional

- **Un esquema de relación R** es un conjunto de atributos. Cada atributo A € R tiene un dominio asociado **dom(A)** de valores posibles
    - dom(PILOTO) = string
    - dom(VUELO) = integer

- **Una tupla t** sobre el esquema de relación R es una función de R donde: t(A) € dom(A) para A € R
    - t(PILOTO) = “Juan”
    - t(HORA) = 1230
- **Una instancia de relación r** es un conjunto de tuplas sobre el esquema de relación R. Es un subconjunto de todas las tuplas posibles: dom(A) x dom(B) x dom(c)…

```

asignado (PILOTO, VUELO, HORA, FECHA)
```

- Nombre de la relación: “asignado”
- Esquema de la relación: “(PILOTO, VUELO, HORA, FECHA)”
- Atributo: “FECHA”

Dependencias Funcionales:

- Es una **restricción** sobre el conjunto de tuplas, que puede aparecer en una instancia de relación
- Sea R un esquema de relación y X e Y dos subconjuntos de R, X → Y (X determina Y) se verifica en R sí en ninguna instancia aceptable de R pueden existir 2 tuplas **t1**, **t2** tales que
    - t1[X] = t2[X]
    - t1[Y] != t2[Y]

Axiomas de Inferencia

- Sean X Y W y Z conjuntos de atributos:
    - **Reflexión:** Sí Y está incluído en X, entonces X → Y
    - **Aumento:** Sí X → Y, entonces XW → YW
    - **Transitividad:** Sí X → Y y Y → Z, entonces X → Z

Reglas Adicionales de Inferencia

- **Unión:** Sí X → Y y X → Y, entonces X → YZ
- **Pseudotransitividad:** Sí X → Y y YW → Z, entonces XW → Z
- **Descomposición:** Sí X → YZ, entonces X → Y y X → Z

Claves:

- Dado un esquema R y un conjunto de Dfs F asoociado se dice que X incluido (ó igual) en R es una *clave* para el conjunto R sí

- $X \rightarrow R$
- No existe ningún Z incluido (ó igual) en X tal que $F \models Z \rightarrow R$ (condición de minimalidad)

Clase 2

Esquemas Indeseables

- **Redundancia**
  - Univ (ANUMERO, ANOMBRE, APELLIDO, MATERIA, IFECHA, EFECHA, NOTA)
  - Se unen datos independientes
- **Anomalías de Actualización**
  - Univ (ANUMERO, ANOMBRE, APELLIDO, MATERIA, IFECHA, EFECHA, NOTA)

    null null null math null 15/5/5 null
  - Valor nulo en clave primaria
- **Descomposición con pérdida de información**
  - Alumno (ANUMERO, ANOMBRE)
  - Resultado (ANUMERO, MATERIA, NOTA)
  - Examen (MATERIA, EFECHA, NOTA)
  - Inscripción (ANUMERO, MATERIA, IFECHA)

Modelo Entidad – Interrelación (DER)

- Herramienta de alto nivel para el modelado conceptual de los datos
- No formal
- Tiene por objetivo capturar el contenido semántico de un dominio de problema para convertirlo en un diseño relacional

Elementos del DER

- **Entidad**: Objeto del mundo real.
  - Una entidad se describe usando un conjunto de atributos.
- **Conjunto de entidades**: Una colección de entidades similares.
  - Ej. Empleados. Todas las entidades de un conjunto tienen el mismo conjunto de atributos.
- Cada entidad tiene una clave única.
- Cada atributo tiene un dominio.
- **Interrelaciones**: Son vínculos semánticos entre conjuntos de entidades
- Las interrelaciones también pueden tener atributos

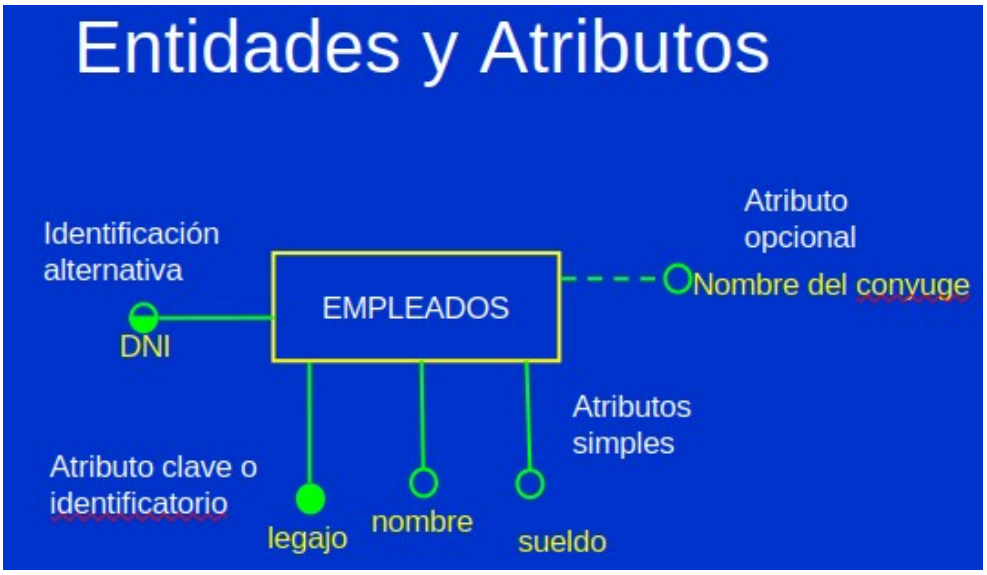
- Para modelar identificamos los conjuntos de entidades (clases), los atributos comunes de interés a los fines del problema y las interrelaciones entre los conjuntos de entidades.

Entidades

- Entidades fuertes: Poseen atributos propios que definen una clave.
- Entidades débiles: Necesitan de atributos de una entidad fuerte para formar su propia clave

Entidades Y Atributos

- Los atributos no tienen existencia por sí sólos, únicamente tienen sentido en la medida que pertenecen a una entidad.



Interrelaciones

- Interrelación: Asociación entre dos o más entidades.
  - Ej: Andrés trabaja en el Departamento Ventas.
- Conjunto de Interrelaciones: Colección de interrelaciones similares.

Tipos de Interrelaciones

- 1 a 1 (un empleado ocupa una oficina, una oficina es ocupada por un empleado)
- 1 a muchos (un empleado tiene muchos recibos de sueldo, un recibo es de un solo empleado)
- Muchos a muchos (un empleado tiene muchas habilidades, una habilidad la poseen muchos empleados)

Interrelación con Dependencia

- Interrelación con dependencia de existencia.
  - Por ejemplo, Empresa -> Empleado

- Interrelación con dependencia de identificación (incluye identificación).
    - Por ejemplo, Hotel -> Habitación

Proceso

- Analizar el enunciado detectando los sustantivos y sus interrelaciones, los sustantivos son candidatos a entidades
- Formar la matriz de entidades
- Construir un primer DER
- Análisis de cardinalidades
- Detectar e eliminar redundancias
- Convertir el DER en entidades relacionales (o tablas)
- Implementar en el motor usando DDL

Claves Foráneas

- Cuando un atributo o conjunto de atributos x de una relación R contiene el valor de los atributos clave (primaria o candidata) k de una relación S se dice que x en R es clave foránea de S.
- Constituye una restricción al dominio de x donde no puede tomar valores que no existan en alguna tupla de S en k.

Acciones Referenciales

- Cascade
- Restrict / No Action
- Set Null

Como Pasar del DER a Relaciones

- Las entidades se convierten en tablas
- Las relaciones uno a uno (y las “es un”) se pueden convertir en una tabla (si no hay muchos nulos) o en dos tablas donde los registros de la menor cantidad apuntan a la de mayor cantidad con una clave foránea.
- Las interrelaciones uno a muchos generan atributos de clave foránea del lado de la entidad de muchos que apuntan a la clave de la entidad del lado uno
- Las interrelaciones muchos a muchos generan una nueva entidad que tiene claves foráneas a ambas entidades

Clase 3

Algunas Definiciones

- **Dependencias funcionales no triviales**: aquellas que el lado izquierdo no es un superconjunto del derecho
- **Clave**: Un conjunto de atributos que determina de manera única y mínima a las tuplas de la relación
- **Clave Candidata**: Si hay mas de una clave en una relación las claves son llamadas claves candidatas.
- **Clave primaria**: Una clave candidata elegida para ser clave principal.
- **Atributos primarios**: Atributos que son parte de las claves candidatas.

Clausura

- Clausura de F, F+, es F mas todas la DFs que pueden ser derivadas de F
- Por lo que si F |= X → Y entonces X → Y esta en F+
- Clausura de X, X+, es el mayor Z tal que X → Z esta en F+

Formas Normales

- Se utilizan para evitar los esquemas indeseables.
- FN1, FN2, FN3, FNBC, FN4, FN5 (se sólo hasta la FN4)

FN1

- 1NF: La relación no debe tener grupos repetitivos. Definición de Codd La relación no debe tener atributos no atómicos.

FN2

- Un esquema de relación está en FN2 si ningún atributo no clave es funcionalmente dependiente de solo una parte de la clave

FN3

- Un esquema de relación está en FN3 si ningún atributo no clave es funcionalmente dependiente de solo una parte de la clave (FN2) y ningún atributo no clave es funcionalmente dependiente de otro atributo no clave

FNBC

- Un esquema de relación está en FNBC si y solo sí toda dependencia funcional no trivial tiene como lado izquierdo una clave candidata.

FN2, FN3 Y FNBC

- Concluciones:
  - Todo campo no clave debe tener información referida a la clave
  - Se puede dividir para eliminar las anomalías de actualización

FN4

- Un esquema de relación esta en FN4 con respecto a una Dependencia Jerárquica si para toda Dependencia Multivaluada de la DJ no trivial de la forma $X \rightarrow\rightarrow Y$, X es una clave candidata de R.

Dependencias Multivaluadas

- Si R es un esquema de relación y XY esta incluido en R podemos definir entonces una dependencia multivaluada $X \rightarrow\rightarrow Y$ sobre R (X multidetermina a Y), como una restricción aplicable a cualquier instancia r de R, según la cual para cada valor de X existe un conjunto bien definido de valores de Y y ese conjunto es independiente del resto de los atributos R-XY

Dependencias Jerárquicas

- Si R es un esquema de relación y XYZ esta incluido en R podemos definir entonces una dependencia jerárquica $X \rightarrow\rightarrow Y|Z$ sobre R (X multidetermina jerárquicamente a Y y a Z), como una restricción aplicable a cualquier instancia r de R, si $X \rightarrow\rightarrow Y$ y $X \rightarrow\rightarrow Z$ y Y es independiente de Z.

Normalización

- Proceso de lograr que los esquemas de relación cumplan con las Formas Normales.
- Objetivos:
  - Eliminar ciertos tipos de redundancia
  - Eliminar ciertos anomalías de actualización
  - Producir un “buen” diseño del mundo real
  - Simplificar imponer reglas de integridad
- Como? Mediante la descomposición sin pérdidas

Clase 4

Base de Datos Orientada a Documentos

- Los datos se almacenan en colecciones de documentos que guardan información sobre objetos de una misma entidad/clase
- Los documentos se graban en JSON ó BSON
- Los documentos se pueden embeber de forma unos dentro de otros, formando una estructura jerárquica
- En los documentos los valores de los datos se almacenan dentro de conjuntos clave-valor
- Los valores pueden ser de diferentes tipos: numeros, strings, arreglos de valores, fechas, otros documentos
- Los documentos de una colección deberán almacenar documentos de datos de objetos de un mismo tipo
- Los documentos de una misma colección no necesariamente deben tener los mismos esquemas JSON
- Por diseño, se deberían identificar qué datos son obligatorios, y cuáles opcionales.

NoSQL

Ventajas - Desventajas

- Ventajas
  - Flexibilidad, sin esquemas
  - Mayor velocidad de desarrollo.
  - Mínima fragmentación. Son más rápidas leyendo y escribiendo.
- Desventajas
  - No hay un estándar. **Se debe documentar bien!**
  - Puede haber redundancia. Derrocha recursos. No hay normalización.
  - El cliente debe mantener la consistencia.

Objetivos de Diseño

- Almacenar datos
- Proveer la mejor performance
- Requerir un monto razonable de hardware
- (No se incluye) controla la integridad referencial de los datos. ← Objetivo secundario.

Diferencia Diseño Relacional VS NoSQL

- Relacional:
  - Diseño orientado a modelar el dominio del problema, y con foco en el control de la integridad
- NoSQL:
  - Diseño orientado a la aplicación,y con el foco en la performance.

Embeber VS Referenciar

Embeber

- Ventajas
  - Se puede recuperar toda la información relevante en una sola operación
  - Evita los joins y accesos múltiples a la BD
  - Insertar nueva información en una sola operación, evitando el uso de transacciones.
- Desventajas
  - Repetición de los datos.
  - Recuperar información que no es relevante.
- Cuándo Embeber?
  - Cuando la relación es 1:poco

Referenciar

- Ventajas
  - Documentos más pequeños
  - Menor duplicación de los datos
- Desventajas
  - Se necesita más de una query para recuperar los datos ó bien usar $lookup, con impacto en la performance
- Cuándo referenciar?
  - Cuando la relación es 1:muchos
  - Mucho a Muchos
  - Cuando necesito saber información sobre partes independientes de los productos.}

Reglas para Embeber ó Referenciar:

1. Embeber es la mejor opción siempre que no haya razones ineludibles para no embeber.

2. Tener que acceder objetos independientemente es una razón ineludible para NO embeber. (Ó sea, hay que referenciar.)
3. Evitar en lo posible, usar joins/lookups.
4. Los arrays enormes no son bien manejados, pueden provocar un overflow de documentos
5. El diseño de los esquemas estará supeditado al uso que se prevé por parte de la aplicación para hacerla más performante.

Bases de Datos Distribuidas

Definición:

- Una base de datos distribuida es un sistema de gestión de bases de datos donde los datos no están almacenados en un único servidor, sino que se distribuyen en múltiples nodos ó servidores conectados por una red. Esto permite mayor escalabilidad, disponibilidad y rendimiento, ya que la carga de trabajo se distribuye entre diferentes sistemas.

Teorema CAP (Consistency, Availability and Partition Tolerance)

- Establece que en un sistema distribuido, no es posible garantizar simultáneamente las 3 propiedades (consistencia, disponibilidad y tolerancia a particiones)

Consistencia (C)

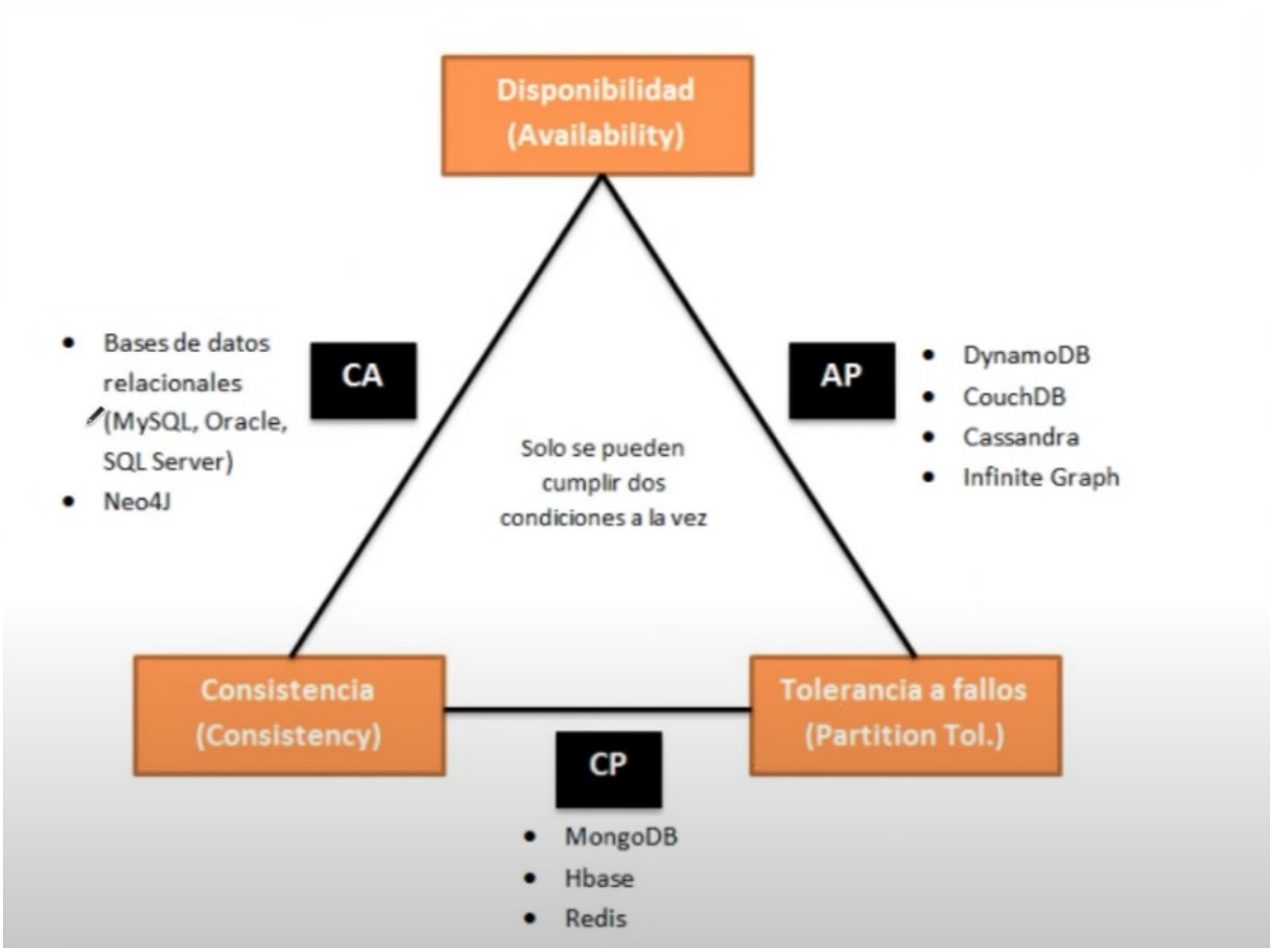
- Significa que todos los nodos del sistema ven los mismo datos en cualquier momento. Esto implica que cualquier lectura debe devolver la información más reciente, no importa qué nodo se consulte.

Disponibilidad (A)

- Significa que cada solicitud al sistema debe recibir una respuesta, ya sea exitosa ó no, aunque no sea la versión más reciente de los datos

Tolerancia a Particiones (P)

- Significa que el sistema debe continuar funcionando de manera correcta incluso sí algunos nodos se quedan aislados debido a fallos de red.



Arquitecturas

- **Peer to Peer (P2P)**: Los nodos se comunican directamente entre sí para la replicación y la consistencia. Todos los nodos son iguales.
- **Maestro-Esclavo**: Una instancia de MySQL se configura como maestro (acepta escrituras), y una ó más instancias esclavas replican los datos del maestro. Las aplicaciones pueden dirigir las lecturas a los esclavos para distribuir la carga.
    - **Multi-Maestro:** Caso particular, dónde cualquier nodo puede aceptar escrituras
- **Basada en Particionamiento (Sharding):** Los datos de una colección se dividen en “shards”, basados en una clave de particionamiento. Cada “shard” reside en un conjunto de servidores (replica set para alta disponibilidad). Un servidor de “mongos” actúa como enrutador, dirigiendo las consultas al “shard” apropiado