Tema 1

Definiciones

Sistema: Conjunto de elementos o partes coordinadas que responden a una ley o que contribuyen a un determinado objeto o función.
Datos: Representación simbólica de un atributo a variable cuantitativa/cualitativa.
Información: Conjunto organizado de datos procesados, que constituyen un mensaje que cambia el estado de conocimiento del sujeto o sistema que recibe dicho mensaje.

Sistemas de información

Sistema automatizado o manual, que engloba a personas, máquinas y/o métodos organizados para la recopilación, procesamiento, almacenamiento, transmisión, visualización, diseminación y organización de información. Está compuesto por hardware, software, datos y personas.

Sistema de información empresarial

Conjunto de elementos interrelacionados con fin de apoyar las actividades de una empresa (gestión de información, comunicación, resolución de problemas). Favorecidos por la complejidad de los sistemas actuales y la capacidad de los ordenadores.

Un gerente tiene múltiples funciones (planificar, toma de decisiones, organizar…). Los niveles gerenciales son:

- Planificación estratégica: Ejecutivos
- Control gerencial: Directores de producto, jefes de división…
- Control operativo: Jefes de departamento, de proyecto…

Hay que tener en cuenta el origen de la información (interior/entorno) y su presentación (detallada/resumida).

Según el nivel al que se dirigen los SI son:

- EIS (Executive Information Systems): Permiten a los ejecutivos conocer el estado actual del funcionamiento de la empresa, de forma visual y sencilla. Destinados a satisfacer necesidades de simulación de escenarios, y ofreciendo vista general global a modo de resumen (cuadro de mando)
- DSS (Decision Support Systems): Dan apoyo a los gestores en la toma de decisiones, ayudando a la resolución de problemas semiestructurados. Permiten analizar el impacto de las decisiones tomadas en el funcionamiento de la empresa. Según Alter sus funciones son:
    - Recuperación de información
    - Creación de informes a partir de múltiples fuentes
    - Estimación de las consecuencias de una decisión
    - Realización de propuestas
    - Ejecución de decisiones

  Los sistemas de ayuda a la decisión apoyan al ejecutivo/gerente/analista, pero no lo sustituyen. Ejemplo:
    - Sistemas expertos (KBS): Programas que codifican el conocimiento de un experto en forma de heurísticas, con potencial de ampliar la capacidad de resolución de problemas de una persona. Capaces de explicar cómo se obtuvo la solución, pero careciendo de intuición.
- Management Information Systems (MIS): Ayuda a desempeñar la tarea a los gestores de las organizaciones proporcionándoles información necesaria (evolución histórica, informes, simulaciones…):
    - Marketing: Apoyo a la resolución de problemas relacionados con el mercado y las ventas
    - Producción: Apoyo a la tarea principal de la empresa.
    - Finanzas: Todo lo relacionado con la situación económica de la empresa
    - Recursos humanos: Todo lo relacionado con el personal.

- Online Transaction/Analytical Processing (OLTP/OLAP): Procesan la información operative que se produce en la empresa (recopilación, manipulación y almacenamiento de datos). Ejemplos:
    - Gestión de pedidos
    - Control de inventario
    - Facturación
    - Contabilidad
- Planificador de recursos empresariales (ERP): Permite integrar distintos flujos de información de la empresa de forma modular y adaptada al cliente. Es un único Si integrado para toda la compañía, que coordina la información de los diferentes procesos de negocio, usando una DB centralizada y actualizada a tiempo real. Ejemplos:
    - SAP
    - Oracle E-Business Suite
    - ABW
    - Microsoft Dynamics
- Gestor de relaciones con el cliente (CRM): SI de Marketing para la gestión integral de relaciones con el cliente.

Tema 2

Las etapas del proceso de desarrollo del software son:

- Planificación: Ámbito del proyecto, estudio de viabilidad, análisis de riesgos, estimación, planificación temporal y asignación de recursos.
- Análisis: Elicitación de requerimientos (funcionales y no funcionales) y modelado de datos y procesos.
- Diseño: Estudio de alternativas y diseño arquitectónico (diseño de DB y aplicaciones).
- Implementación: Adquisición de componentes, creación e integración de recursos para el sistema.
- Pruebas: De unidad, integración, alfa/beta y test de aceptación.
- Instalación/despliegue
- Uso/mantenimiento: Adaptativo, correctivo, perfectivo y evolutivo

<u>Modelos de ciclo de vida</u>

- En cascada: Ciclo de vida clásico, se pasa de una etapa a otra sin vuelta a atrás. Problemas puesto que solo hay versión operativa al final. Útil cuando los requisitos se conocen y se sabe que no van a cambiar.
- De prototipos: Basado en el diseño y construcción de prototipos. El prototipo puede dar ideas falsas al cliente, pero orienta en las ideas buenas. Estos prototipos deberían desecharse.
- En espiral: Cíclicamente itera por las etapas de planificación, identificación de riesgos, desarrollo y revisión. Bueno para sistemas complejos que pueden cambiar.

<u>Análisis de requerimientos</u>

Recabar información sobre el uso que se piensa dar al sistema. Para ello se identifican los requisistos, de lo forma:

- Identificación de las principales áreas de a aplicación y de los distintos grupos de usuarios
- Estudio y análisis de la documentación existente relativa a las aplicaciones
- Estudio del entorno de operación actual
- Estudio del uso de la información

El resultado es un documento de especificación de requerimientos, en lenguaje natural y de forma jerárquica.

<u>Diseño conceptual</u>

Producir un esquema conceptual de la base de datos, independiente del SGBD. Objetivo principal de comprender la estructura, semántica, relaciones y restricciones de la BD, mediante una descripción estable del contenido. Es desable que tenga:

- Expresivida
- Sencillez
- Minimalidad
- Representación gráfica
- Formalidad

Se pueden realizar dos enfoques:

- Centralizado: Los requisitos de distintas aplicaciones y grupos de usuarios se combinan en un único conjunto de requisitos antes de comenzar el diseño del esquema.
- De integración de vistas: Diseño de un esquema para cada tipo de usuario o aplicación basado únicamente en sus requisitos. Durante la etapa de integración de vistas, dichos esquemas se combinan en uno global.

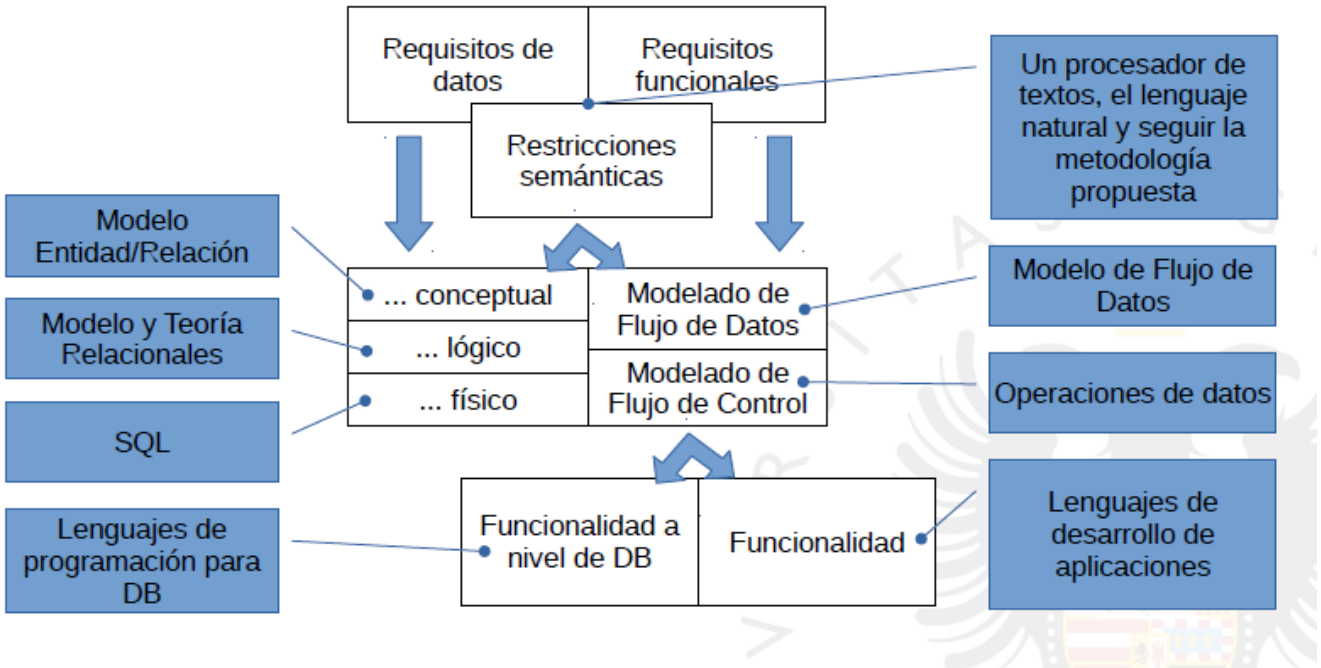
Como resultado se tiene un diagrama E/R, CASE*Method o de clases UML.

Diseño lógico

Crear un esquema conceptual y los esquemas externos en el modelo de datos del SGBD elegido, es decir, transformar los esquemas del diseño funcional en un conjunto de estructuras propias del modelo abstracto de datos elegido.

Diseño físico

Estimar adecuadamente los diferentes parámetros físicos de nuestra BD mediante técnicas analíticas y/o experimentales, además de preparar las sentencias DDL correspondientes a las estructuras identificadas durante el diseño lógico. Como resultado se obtiene este conjunto de sentencias DDL en el lenguaje del SGBD.



Arquitecturas para el sistema de información

Pueden ser:

- Centralizadas: Con terminales “Dumb”, que únicamente sabían conectarse al servidor
- Cliente-servidor: Tienen problemas de portabilidad/mantenimiento, pues la lógica está en el cliente.
- Por niveles: La capa de aplicaciones contiene tanto la lógica como la funcionalidad.
- Web-service: Tecnología que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones mediante la adopción de estándares abiertos basados en la web.

Tema 3

Modelo de Datos

Representación, relativamente sencilla, orientada a la descripción de los datos del mundo real y que, usualmente describe la estructura, las condiciones que deben cumplir y cómo se manejan los datos.

Su representación tiene asociada una serie de conceptos (formados por construcciones lingüísticas y gráficas) que describen el conjunto de datos y las operaciones para manipularlos.

Existen dos clases:

- **Modelo Conceptual**, que representa la realidad en un alto nivel de abstracción.
  Genera el Esquema Conceptual
- **Modelo Lógico** (o de Base de Datos), que describe las relaciones lógicas entre los datos y la base de *datos.*
  Genera el Esquema Lógico.

Los elementos de un modelo representan Entidades genéricas.
Los valores concretos se denominan instancias u ocurrencias de una entidad.
Cada SGBD se suele asociar a un modelo de datos específico.

Mediante el Lenguaje de Definición de Esquemas se representan los datos, junto a sus propiedades, relaciones y restricciones.
Mediante el Lenguaje de Manipulación de Datos se representan las operaciones delos datos, las operaciones sobre relaciones de los datos y las relaciones entre operaciones.

Tipos:

- **Fichero plano**: El modelo más simple, formado por una matriz bidimensional de elementos sin relaciones con otras matrices. En él los miembros de una columna tienen valores del mismo tipo, los de una misma fila están relacionados entre ellos.
- **Jerárquico**: Datos organizados en una estructura arbórea, con relaciones 1:N. Las relaciones se establecen a nivel físico, y es muy eficiente en relaciones de datos con estructura jerárquica.
- **En red**: Generalización del modelo jerárquico que permite relaciones N:N, de modo que se reducen las redundancias y desaparece la herencia de los campos. La integridad de datos asociada a los arcos padre-hijo se mantiene.

Modelo de Datos Relacional

Basado en el concepto de Relación, que formalmente puede verso como un par de conjuntos (R,r) siendo **R** el esquema y **r** la instancia.

Esquema es un conjunto formado por pares de atributos junto a sus dominios.

Instancia es la aplicación de un esquema a un conjunto finito de datos (el contenido de una tabla o parte de ella). Visualmente es una estructura bidimensional formada por columnas (atributos) y filas (tuplas).

Por tanto una **base de datos relacional** es un conjunto finito de tablas. Y una **instancia de la BBDD** es una colección de instancias de las relaciones.

Para mantener la semántica se deben respetar las **restricciones de integridad**, es decir, reglas que mantienen correcta la información almacenada. Entonces el **esquema de una base de datos** es una colección de esquemas de relación junto a una serie de reglas de integridad.

Dentro de las restricciones de integridad, las **restricciones específicas** son aquellas que provienen de la semántica del atributo y son propias de cada base de datos concreta. Y las **restricciones genéricas** son meta-reglas, normas que se aplican a los atributos en función del papel que desempeñan (PK y FK)

Las **Transacciones ACID** garantizan la estabilidad de las operaciones, pero requieren una gestión sofisticada:
- **Atomicidad**: Propiedad que asegura si la operación se ha realizado o no, y, por lo tanto, no puede quedar a medias ante un fallo del sistema.
- **Consistencia** (Integridad): Sostiene que cualquier transacción llevará a la base de datos desde un estado válido a otro.
- **Aislamiento**: La ejecución concurrente de dos transacciones da como resultado un estado del sistema que sería el mismo que si se hubieran ejecutado secuencialmente (en cualquier orden).
- **Durabilidad** (Persistencia): Una vez establecida una transacción no se perderán los cambios aunque falle el sistema.

Un conjunto CC de atributos en una relación se dice que son **claves candidatas** si se verifica:
- **Unicidad**: Si dos instancias coinciden en los atributos de la clave candidata, entonces coinciden en el resto de atributos de la relación.
- **Minimalidad** (Irreductibilidad): La propiedad anterior no se verifica para cualquier subconjunto de CC

Se llama superclave si solo verifica la unicidad, y clave primaria a la clave candidata elegida por el diseñador.
Una **clave externa** respecto a una CP es el cuando el dominio activo de una CE está contenido en el de la CP.

Propiedades de las relaciones:
- No hay orden en las tuplas ni en los atributos
- No hay tuplas duplicadas
- Todo esquema de una relación tiene una clave primaria
- Los valores delos atributos son atómicos (no tienen estructura)

Para mantener la integridad del sistema, hay que respetar:
- **Integridad de entidad**: Los atributos de una CP no pueden tomar valores nulos. Para ello hay que comprobarlo en los procesos de Inserción y Actualización (y que no estén repetidos)
- **Integridad referencial**: El valor de una CE debe ser igual a un valor del dominio activo de la CP, o nulo. Se comprueba en la Inserción. En la Actualización también si se modifica la CE, si es la CP se actualizan en cadena las CE. En el Borrado de la CP se realiza un borrado en cadena o un cambio a valor nulo.

SQL es el lenguaje de consultas predominante basado en el Álgebra Relacional (de tipo procedimental) y en el Cálculo Relacional (de tipo declarativo, solo dice el resultado), este último orientado a tuplas y dominios.

SQL presenta dos sublenguajes:
- **DDL** (Data Description Lenguage) que permite definir y manejar esquemas de estructuras relacionales (relaciones, vistas, …)
- **DML** (Data Management Lenguage) que permite manipular intancias de estructuras relacionales (tuplas)

Desafíos:
- La BD no escala con el tráfico a un coste aceptable
- El tamaño del esquema de datos crece desproporcionalmente
- El sistema de información genera muchos datos temporales que no corresponden al almacén principal
- Tener que desnormalizar la BD por razones de rendimiento o conveniencia para utilizar los datos en una aplicación
- La BD contiene grandes cantidades de texto o imágenes
- Se ejecutan consultas sobre los datos que implican relaciones jerárquicas complejas
- Se usan transacciones locales que no suelen ser muy durables

<u>Modelo de Datos Orientado a Objetos</u>
Motivación:
- Los modelos de datos y las estructuras de datos de los LPOO están desacoplados
- Persistencia de Objetos más allá de los programas
- Almacenamiento más eficiente y gestión de datos en memoria secundaria
- Independencia de los datos respecto de los programas
- Lenguaje de consulta eficiente y de alto nivel independiente de la estructura física
- Gestión de transacciones que permita acceso concurrente, integridad, seguridad y recuperación ante fallos
- Limitación de las Bases de Datos Relacionales, no aconsejadas para estructuras de datos complejas:
  - Estructuras simples
  - Poca riqueza semántica
  - No soportan tipos definidos por el usuario ni recursividad
  - Falta de procedimientos/disparadores
  - No admiten herencia
- La traducción de objetos a tablas no conveniente porque implica:
  - Mayor tiempo de desarrollo
  - Errores en la traducción
  - Inconsistencias y mayores tiempos de ejecución debidos al ensamblaje/desemsamblaje

Manifiesto de los sistemas de base de datos orientados:
- Características obligatorias
  - Al se un SGBD: Persistencia, gestión de almacenamiento secundario, concurrencia, recuperación ante fallos y lenguajes ad-hoc para manipulación
  - Al ser OO: Objetos complejos, identidad del objeto, encapsulamiento, clases, herencia, polimorfismo, sobrecarga, vinculación dinámica, extensibilidad, completitud de cálculos
- Características opcionales
  - Herencia múltiple
  - Verificación e inferencia de tipo
  - Distribución
  - Transacciones de diseño
- Características abiertas
  - Paradigma de programación
  - Sistema de representación y de tipos

- Uniformidad

Modelo de Datos Objeto Relacional

Soporta objetos, clases y herencia directamente en los esquemas de bases de datos y en el lenguaje de consulta.
Reglas:

- Cada clase persistente tiene una tabla de base de datos correspondiente
- Campos de objetos con tipos de datos primitivos se asignan a columnas en la tabla
- Cada fila corresponde a una instancia
- Cada relación de objeto de muchos a muchos requiere una tabla de join.
- La herencia es modelada a través de una relación uno-a-uno entre las tablas clase y subclase.

Manifiesto de los SGBD de 3ª Generación:

- Un SGBD-3G debe tener un sistema de tipos rico
- La herencia y las funciones son buenas ideas
- Los IDOs para los registros deberían asignarse por el SGBD sólo si no se dispone de una clave primaria
- Las reglas (disparadores, restricciones) se convertirán en una característica primordial de los sistemas

NoSQL

"Not only SQL", difieren del RDBMS en diferentes modos:

- No usan SQL como principal lenguaje de consultas
- Los datos almacenados no requieren estructuras fijas como tablas
- Normalmente no soportan operaciones JOIN
- No garantizan completamente ACID
- Habitualmente escalan bien horizontalmente

Usan el modelo BASE:

- **Basic Availability**: Siempre se obtiene una respuesta del sistema a una petición de datos, aunque esta sea un fallo o que los datos estén inconsistentes o en fase de cambio.
- **Soft-state**: El estado del sistema cambia constantemente a lo largo del tiempo, incluyo cuando no hay entradas de datos en ese periodo, debido a la consistencia eventual.
- **Eventual consistency**: Eventualmente el sistema se volverá consistente a partir del momento en el que deje de recibir datos. Los datos se propagarán, pero el sistema seguirá recibiendo datos sin evaluar la consistencia para cada transacción antes de avanzar a la siguiente.

Se han diseñado para potenciar aspectos como:

- **Flexibilidad**: Ofrecen esquemas flexibles que permiten un desarrollo más rápido y más iterativo, ideales para datos semiestructurados y no estructurados.
- **Escalabilidad**: Para escalar usando clústeres distribuidos en vez de con servidores caros.
- **Alto rendimiento**: Optimizadas para modelos de datos específicos y patrones de acceso.
- **Altamente funcional**: Proporcionan APIs altamente funcionales y tipos de datos específicamente diseñados para cada uno de sus respectivos modelos de datos.

El teorema de Brewer nos indica que es imposible para un sistema computacional distribuido ofrecer simultáneamente:

- **Consistencia**: Todos los nodos ven los datos al mismo tiempo
- **Disponibilidad**: Garantiza que cada petición recibe una respuesta acerca de si tuvo éxito o no. Cada cliente puede leer y escribir.
- **Tolerancia a particiones**: El sistema debe seguir funcionando aunque existan fallos o caídas parciales.

Taxonomía de soluciones NoSQL:

- Clave-Valor
- Tabular o columnar: Formado por familia de columnas en vez de por filas. Útil para gestión de tamaño, cargas de escritura masivas, disponibilidad, MapReduce
- Documentos: Colección de documentos con colecciones de clave-valor. Buenas en el modelado de datos natural, son amigables al programador, con desarrollo rápido y orientadas a la web
- Grafos: Formado por nodos y ralaciones con pares clave-valor en ambos. Buenas en modelar directamente un dominio en forma de grafo, con excelente rendimiento cuando los datos están interconectados y no tabulares, y útil para realizar operaciones transaccionales que exploten entidades.

A la hora de comparar con RDBMS hay que tener en cuenta:

- En NoSQL generalmente los datos son recuperados de manera más rápida, sin embargo las consultas pueden ser más limitadas y por tanto transmitir complejidad a la aplicación.
- NoSQL no usa locks y redos para garantizar ACID, por lo que para las aplicaciones que generan informes con consultas complejas no está recomendado.
- Aplicando MapReduce, NoSQL puede paralelizar operaciones complejas como filtros y agrupaciones.

Las RDBMS actuales evolucionan para incorporar capacidades NoSQL, pasando a llamarse NewSQL.