Desarrollo de Sistemas de Información

Tema 1. Introducción

Marta Elena Zorrilla Pantaleón

Dpto. de Matemáticas, Estadística y Computación

Introducción

- ¿Qué es un Sistema de Información?
  - Término que surge en 1960
  - Difícil de definir de forma precisa

"Sistema que recoge, almacena, procesa y distribuye información"

+

"diseñado y construido por ingenieros"

+

"para un determina dominio"

+

"con objeto de facilitar la planificación, el control, la coordinación y la toma de decisiones en una organización"

Introducción

- ¿Qué es un Sistema de Información?
  - Funciones que debe realizar:
    - **Memoria**: mantiene la representación del estado del dominio

      (almacena la información que se le suministra con un determinado esquema)
    - **Informativo**: suministra información sobre el estado del dominio

      (responde a consultas sobre el estado del dominio)
    - **Activo**: realizar acciones que cambien el estado del dominio

      (nuevas inserciones, actualizaciones, borrados)
  - Y que requiere: hardware, software, infraestructura y personas

Definiciones

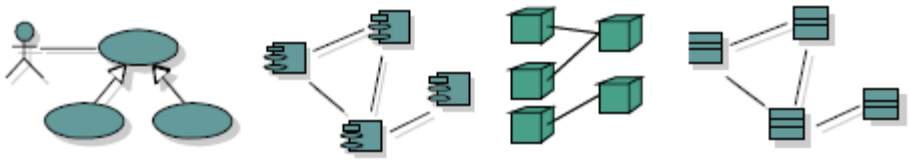
- **Información**: es un conjunto organizado de datos procesados, que constituyen un mensaje que cambia el estado de conocimiento del sujeto o sistema que recibe dicho mensaje (conjunto de datos, relaciones y restricciones de un dominio).
- **Dato**: atributo o característica de una entidad del dominio
- **BD**: colección organizada de datos, relativa a un problema concreto, que puede ser compartida por un conjunto de usuarios/ aplicaciones. Sirven para almacenar, actualizar, consultar y controlar la información.
- **SGBD**: programa o conjunto de programas que sirve para mantener bases de datos y responder consultas sobre ellas

Ciclo de vida de un SI

- Definición: **¿Qué quiero hacer?**
  - Estudio de oportunidades
  - Análisis de requisitos
- Diseño: **¿Cómo lo haré?**
  - Modelos software
- Construcción: **Implementación**
- Evaluación:
  - Pruebas
  - Puesta en marcha
- Mantenimiento:
  - Bugs
  - Incorporar nuevas funcionalidades

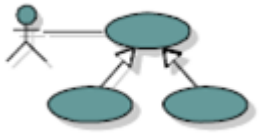
¿Qué es un modelo de software?

- A description of (part of) a system written in **a well-defined language**. (Equivalent to specification.) [Kleppe, 2003]
- A representation of a part of the **function, structure and/or behavior** of a system [MDA, 2001]
- A description or specification of the system and its **environment** for some certain **purpose**. A model is often presented as a combination of drawings and text. [MDA Guide, 2003]
- A set of **statements** about the system. [Seidewitz, 2003] (Statement: expression about the system that can be considered true or false.)



¿Qué es un modelo de software? (y 2)

- Entonces, se puede concretar en:



  - Un **modelo es una abstracción de un sistema o entidad del** mundo real.

  - Una abstracción es una **simplificación, que incluye sólo** aquellos detalles relevantes para algún determinado propósito

  - El modelado permiten **abordar la complejidad de los** sistemas

  - Un **modelo de datos** se puede definir como un conjunto de herramientas conceptuales para describir la representación de la información en términos de datos. Esto es un conjunto de conceptos, reglas y convenciones que permiten especificar *datos, las relaciones entre ellos, su semántica asociada y las restricciones de integridad*.

¿Para qué sirven?

- **Los modelos sirven para:**
  - **Especificar el sistema**
    - Estructura, comportamiento,…
    - Comunicarse con los distintos *stakeholders*
  - **Comprender el sistema (si ya existe)**
  - **Razonar y validar el sistema**
    - Detectar errores y omisiones en el diseño
  - Prototipado (*ejecutar el modelo)*
  - Inferir y demostrar propiedades
  - **Guiar la implementación**

Características de los modelos [Selic, 2003]

- Abstractos
  - Enfatizan ciertos aspectos, mientras que ocultan otros
- Comprensibles
  - Expresados en un lenguaje comprensible por los usuarios y clientes
- Precisos
  - Fieles representaciones del objeto o sistema modelado
- Predictivos
  - Deben de poder ser usados para inferir conclusiones correctas
- Baratos
  - Más fáciles y baratos de construir y estudiar que el propio sistema

Limitaciones actuales de los modelos de software

- Sólo se usan como documentación
  - Que además no se actualiza!
- "Gap" entre el modelo y la implementación del sistema
  - Grandes diferencias semánticas en los lenguajes respectivos
  - No hay herramientas de propagación automática de cambios
    - Cambios en el modelo no se reflejan en el código
    - Cambios en el código no se reflejan en el modelo

    (el modelo no vuelve a usarse jamás tras la primera implementación)
- Los distintos modelos del sistema no se armonizan
  - Suponen vistas de un mismo sistema, pero no hay forma de relacionarlas
  - No hay herramientas de integración de modelos
  - Cada lenguaje de vista tiene una semántica distinta del resto
- No hay ni lenguajes ni herramientas para manejar modelos
  - Solo editores, pero no hay "compiladores", "optimizadores", "validadores", "transformadores de modelos", etc.

Lenguajes de modelado para BD

- UML (Unified Modeling Language)
  - Es un lenguaje de modelado visual de *propósito general* orientado a objetos.
  - Impulsado por el Object Management Group (www.omg.org)
- ORM (Object-Role Modeling) (www.orm.net)
  - Gráfico
  - Propuesto por Halpin (Halpin, 2009)
- ER (diversas notaciones)
  - Chen
  - IDEF1X
  - Merise
  - Etc.

Fases del diseño e implantación de BD

- Aunque el diseño de un SI debe incluir además del modelo de datos, los procesos, la interfaz de usuario y la seguridad. En este apartado se aborda exclusivamente las fases para el diseño e implantación de la BD.
- Análisis de requisitos. Descripción de la información a gestionar y sus procesos. Así como información de volumen de datos, volatilidad, normas de validación, gestor de implantación, etc..
  - Técnicas: Entrevistas con usuarios y expertos. Lectura de documentación. Observación del entorno. Cuestionarios. Prototipado. Mapas Conceptuales
  - Salida: documento con los requisitos
- Diseño conceptual: traducción del análisis de requisitos al esquema conceptual.
  - Parte **estática** Representación generalmente gráfica de las entidades con sus atributos y sus relaciones
    - Técnicas: ER, UML u ORM.
  - Parte **dinámica**: Detalle de procesos (aspectos que cambian con el tiempo)
    - Técnicas: casos de uso, modelos de comportamiento, diagramas de estado.

Fases del diseño e implantación de BD (y 2)

- Diseño lógico: traducción del modelo conceptual al LDD del gestor correspondiente. Modelo relacional, Orientado Objeto, Objeto-Relacional, schemas XML, etc.
- Diseño físico: Transformar el modelo lógico (generalmente en SQL) al físico adaptándolo a las características del gestor y al rendimiento que se espera de la BD (tiempo de respuesta, usuarios concurrentes, nº de transacciones, volumen de datos, procesos sobre tablas…)
- Carga de datos y pruebas: Carga inicial y pruebas de verificación y validación de los requisitos del sistema (tratar de violar las reglas de integridad, comportamiento ante valores límite de los tipos de dato, tiempos de respuesta en consultas frecuentes y en consultas complejas, etc.)
- Operación:
  - Puesta en marcha
  - Tareas de mantenimiento y monitorización

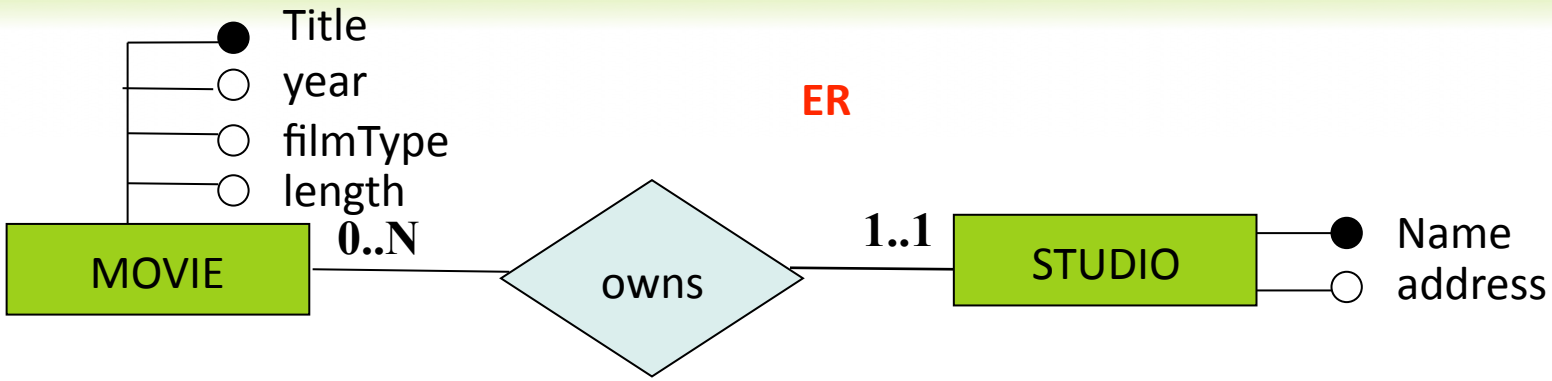
Modelo conceptual vs modelo lógico

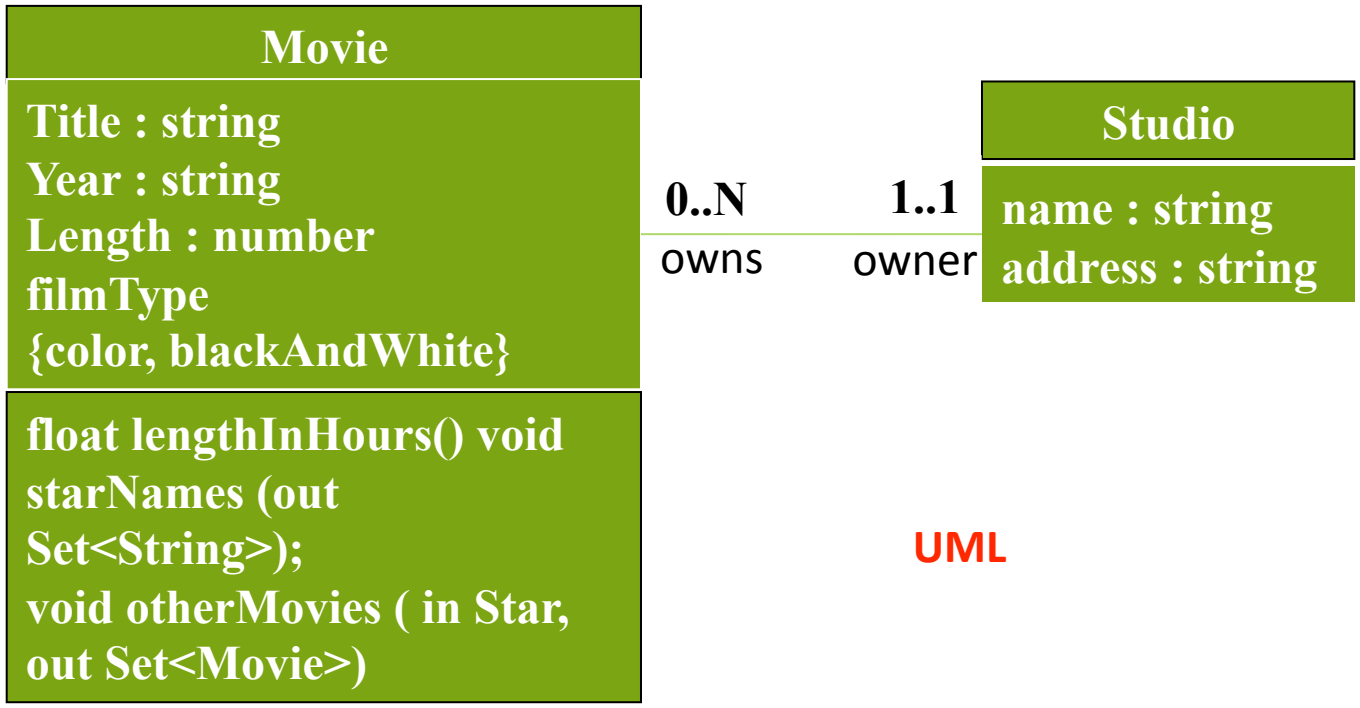
Modelo conceptual	Modelo lógico
Independientes del SGBD	Dependen de la tecnología
Mayor nivel de abstracción	Más próximos al ordenador
Más enfocados al diseño de alto nivel	Más enfocados a la implementación
Mayor capacidad semántica	Poca capacidad semántica
Interfaz usuario /informático	Interfaz informático/sistema
Ej.: ER, UML, ORM	Ej: relacional, jerárquico, Objeto-Relacional, Orientado a Objetos, Schemas XML, etc.

MODELOS DE DATOS

Modelos conceptuales	Modelo lógicos
Entity-Relationship (Chen, 1976)	Prerelacional: - BD jerárquicas y en red
Extended ER (Smith et al. 1977)	Relacional: (Codd, 1970) - SQL 92
RM/T (Codd, 1979)	Postrelacional: - OO (ODMG) - Object-relational (SQL3) - XML (SQL3) - NoSQL (no estandarizado)
UML (v1.0, 1977)	
ORM (Halpin, 1989)	**Modelos físicos:**
	Oracle, SQL Server, DB2, MySQL, Informix, Sybase, Posgresql, Cassandra,…

MODELO CONCEPTUAL-EJEMPLO





Modelo lógico - ejemplo

```
Create table studio (
  name char(10) not null primary key,
  address char(100) null
);

Create table movie (
  title char(20) not null primary key,
  year char(4) not null,
  length int not null,
  filmType char(2) not null check (filmtype in ('BW', 'C'),
  owns_to char(10) not null,
  foreign key owns_to references Studio (name) )
);
```

MODELO FÍSICO- EJEMPLO

ORACLE

```
Create table studio (
  name char2(10) not null primary key,
  address char2(100) null
);

Create table movie (
  title char2(20) not null primary key,
  year char2(4) not null,
  length number not null,
  filmType char2(2) not null check (filmtype in ('BW', 'C'),
  owns_to char2(10) not null,
  foreign key owns_to references Studio (name) )
);
```

Herramientas CASE (Computer Aided/Assisted Software/System Engineering)

- Herramientas para ayudar al analista/programador en la fase del diseño conceptual y su paso al lógico y físico

- Herramientas CASE para BD: ERWin, PowerDesigner, EasyCASE, Oracle designer (Discoverer), Visio (Microsoft), IBM Infosphere, etc.

- Estas herramientas permiten diseñar el modelo de datos conceptual siguiendo alguna de las técnicas mencionadas, generalmente ER o UML y obtener el esquema lógico, generalmente relacional escrito en lenguaje SQL estándar, o bien físico, esto es el mismo modelo pero llevado al lenguaje del gestor (Oracle, SQL Server, MySQL, etc.)

Herramientas CASE (y 2)

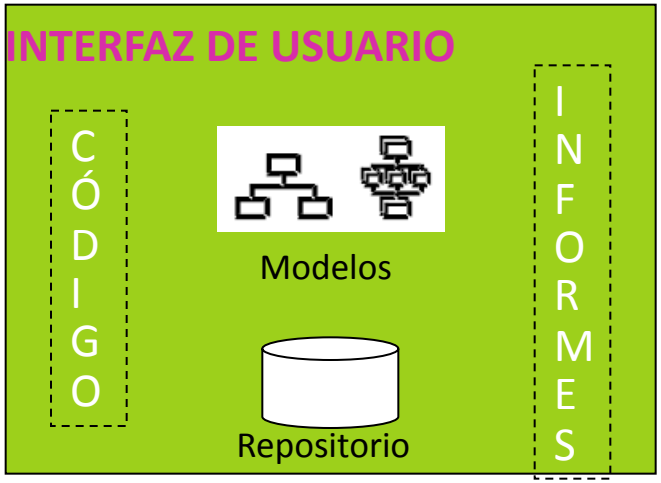
- Ventajas que aportan:
  - Ayudan al diseño (verificación de errores, validación respecto a la técnica, etc.)
  - Reducen el tiempo de desarrollo
  - Facilitan el mantenimiento del esquema de datos (ingeniería directa e inversa)
  - Reducen el tiempo de la documentación
  - Facilitan la portabilidad a otros gestores

Herramientas CASE (y 3)

- Deficiencias:
  - Generalmente no recogen toda la riqueza semántica del modelo de datos.
  - Falta de un modelo de restricciones que genere las reglas de negocio en automático.
  - No ayuda a especificar el modelo físico adecuado, lo indica el diseñador, pero no le da pautas o medidas de rendimiento.
  - No ofrecen la posibilidad de diseñar en entornos distribuidos, OO, activas, … no hay modelo que permita representarlo.
  - Los atributos derivados pueden estar en el conceptual por razones semánticas y en el físico por razones de eficiencia, el problema es que la regla por la que se genera no se puede modelizar.

Herramientas CASE: Componentes

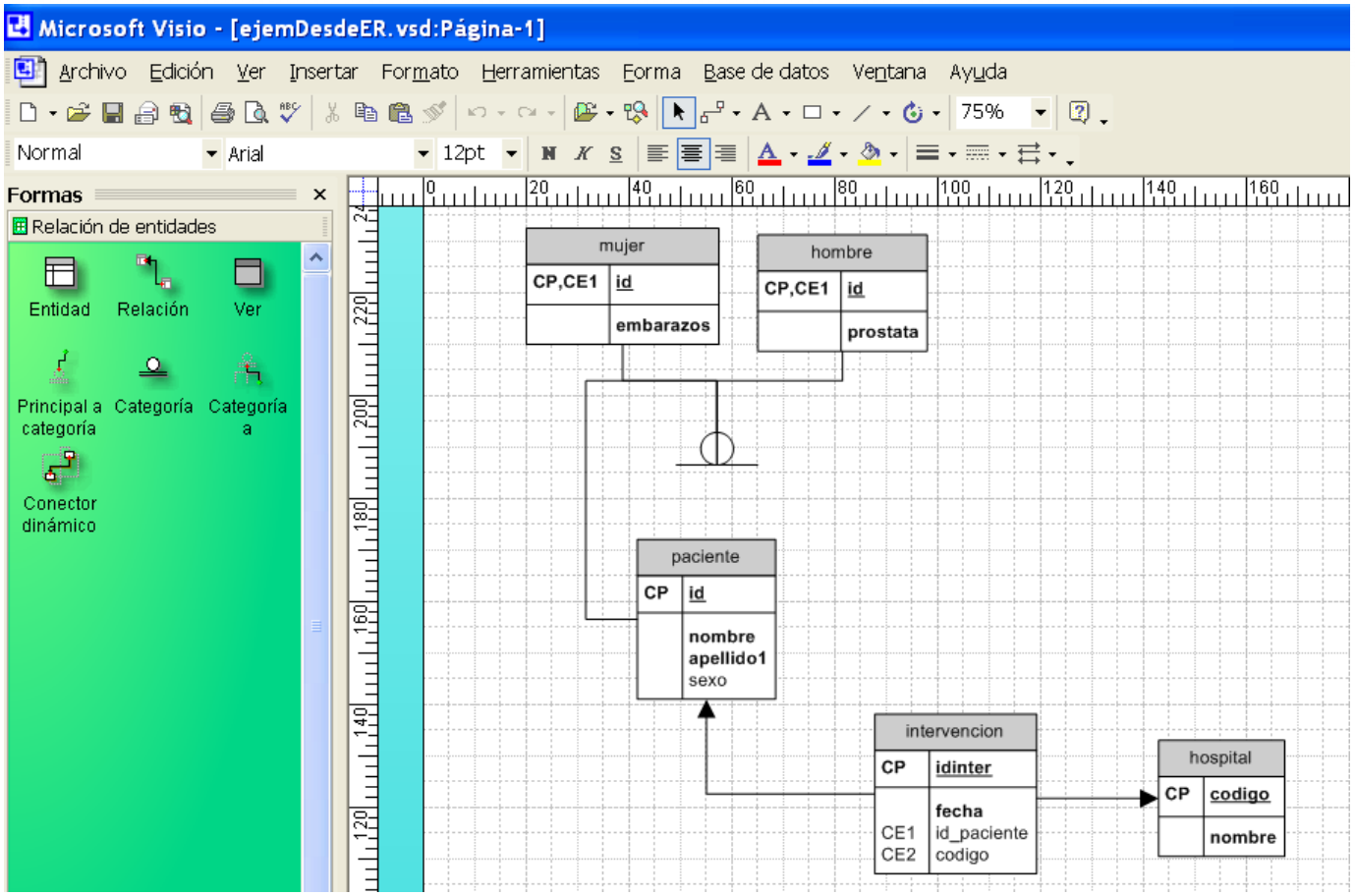
- Repositorio o diccionario de datos
  - Almacén de los elementos definidos
- Módulo diagramático
  - Editores que recogen las distintas técnicas
- Generador de código. Ingeniería inversa.
- Generador de documentación
- Interfaz de usuario



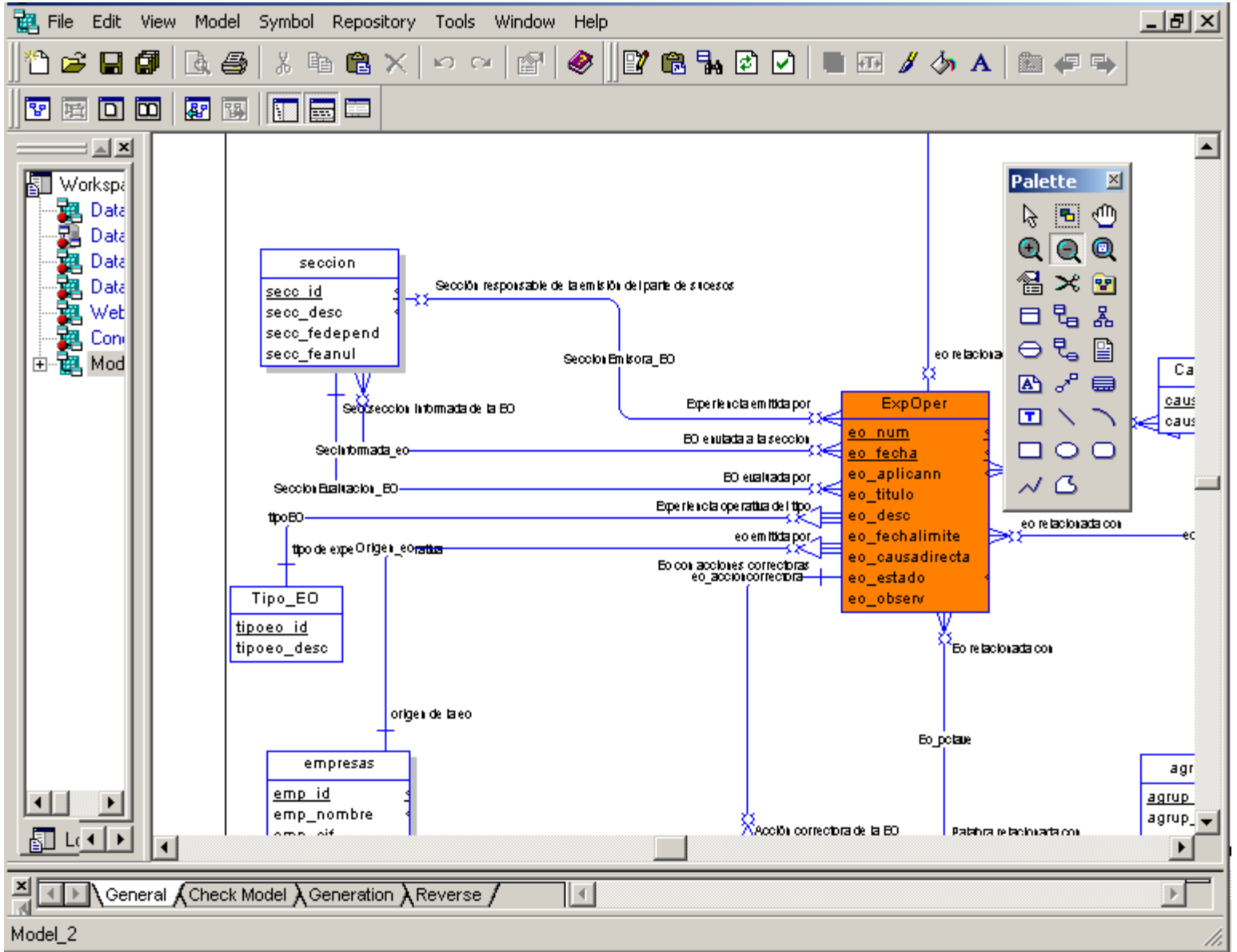
Herramientas CASE: productos

- IBM InfoSphere
- ERWin
- PowerDesigner (Sybase)
- EasyCASE
- Oracle designer (Discoverer)
- Visio (Microsoft)

HERRAMIENTAS CASE: VISIO



HERRAMIENTAS CASE: DATA ARCHITECH



Elección de la herramienta de diseño de bases de datos

- Multiplataforma
- Trabajo en grupo
- Aspectos de seguridad
- Software Open Source / licencia (precio)
- Esquema de BD para diferentes gestores. Comprobación de restricciones
- Sincronización con el gestor
- Ingeniería inversa
- Generación de documentación
- Interfaz gráfica cómoda e intuitiva
- Capacidad de representación respecto a la notación teórica