

Bases de Datos

Ingeniería Civil Informática Segundo semestre 2012

Profesor: Jorge Maturana
jorge.maturana@inf.uach.cl
3° piso, Edificio Informática



Programa del curso

- Introducción a las bases de datos
 - Conceptos generales
 - Tipos de BDs
 - Historia
 - Ciclo de Vida
 - Tutorial introductorio
- Modelamiento de datos
 - Modelos Entidad-Relación
 - Modelo Relacional
- Implementación práctica
 - SQL
 - PL/SQL



Metodología del curso



Metodología del curso

- Naturaleza de la asignatura
 - Parte teórica: conocimientos
 - Parte práctica: ejercicios, laboratorio
- Esquema de Evaluación:
 - Un control, orientado a medir conocimientos
 - Dos pruebas, orientadas a medir competencias
 - Dos tareas, para medir competencias y prepararse para las pruebas



Evaluaciones

- Control (20%)
 - Materia de clase y lecturas complementarias
- Tarea de Modelamiento (15%)
 - Modelamiento E-R y Relacional de un problema (grupal)
- Prueba 1, Modelamiento (25%)
 - modelamiento E-R y Relacional
- Tarea de Programación (15%)
 - Implementación de bases de datos
- Prueba 2, Programación (25%)
 - Implementación de bases de datos, en laboratorio



Otras disposiciones

- Asistencia (OJO: requisito de aprobación!)
 - Clases teóricas: libre
 - Clases prácticas: 70%
- Evaluación recuperativa
 - Para quienes hayan faltado a alguna evaluación (sólo pruebas o control)
 - En caso de haber faltado a más de una, reemplaza la de mayor ponderación
- Examen
 - Carácter Global
 - Para quienes tengan una nota de presentación $n | 3.5 \leq n < 4.5$
 - Ponderación: 30%
- La nota de aprobación es 4.0
- No habrán otras evaluaciones aparte de las mencionadas.
- Puede “probar la prueba”



Bibliografía



- Ramakrishnan, Gehrke, Database Management Systems 2nd ed. Mcgraw Hill
- Simsion, Witt, Data modeling essentials, 3rd ed. Morgan Kaufmann, 2005
- Rob, Coronel, Database Systems - Design, implementation and management, 6th ed. Thomson
- Date, An Introduction to Database Systems, 8th ed., 2004
- Sumathi, Esakkirajan, Fundamentals Of Relational Database Management Systems, Springer, 2007
- Oppel, Databases demystified, McGraw-Hill, 2004
- Elmasri, Navathe, Fundamentals of database Systems, 4th ed., Addison Wesley, 2004
- García-Molina, Ullman, Widom, Database Systems. The complete Book, Prentice Hall
- Gustavo Coronel, SQL & PL/SQL, Material de Curso
- Rafael Camps, Introducción a las bases de datos
- Oracle Database, SQL Reference 10g Release 1, 2003



Y finalmente...



Introducción a las Bases de Datos

Parte 1



¿Qué es un dato?

- Expresión que describe una característica
- “átomo” de información
- Par $\{característica, valor\}$
 - Ejemplos:

característica	valor
color	rojo
Edad	21
Estado	encendido



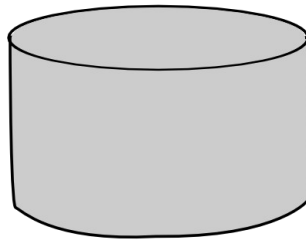
¿Qué son las Bases de datos?

- “Un conjunto de datos, que describe las actividades de una organización” (Ramakrishnan, Gehrke)
- “Colección de datos bien organizados relacionados con sentido, que pueden ser accedidos en distinto orden” (Sumathi, Esakkirajan)
- “Una colección de datos organizados” (Elmasri, Navathe)
- Dos conceptos principales:
 - Conjunto de **datos**
 - Poseen una **estructura**



DBMS

- DBMS: “Data Base Management System”
 - = “Motor de Bases de Datos”
 - = “Sistemas de bases de datos”
- “Conjunto de programas que manejan la estructura de la base de datos y controlan el acceso a los datos almacenados en ella”
(Rob, Coronel)
- Programas computacionales especializados en el manejo de bases de datos



Símbolo usado para denotar una base de datos o DBMS



DBMS vs sistema de archivos

- ¿Por qué usar DBMS?
 - ¿Por qué no usar simplemente archivos?
- Veamos un Ejemplo:

...Érase una vez una biblioteca que requería un sistema computacional para gestionar sus libros...
- Requerimientos:
 - Gestionar libros, usuarios y préstamos
 - Ingreso/ eliminación/ listado/ modificación de todo
 - Búsqueda por autor



Sistema de Biblioteca

- Solución:
 - Programa en <Python/C++/Java/...>
 - Archivo con la siguiente estructura:

```
<título> | <autor> | <editorial>  
...  
#  
<nombre> | <fono> | <RUT> | <libro1> | <libro2> | ...  
...
```

```
El principito | Antoine de Saint-Exupery | Zig-Zag  
El código da Vinci | Dan Brown | Planeta  
Fábulas | Esopo | Andrés Bello  
Subterra | Baldomero Lillo | Andrés Bello  
#  
Juan Pérez | 347562 | 15.378.538-6 | El Principito | Fábulas  
María Pardo | 478539 | 14.836.946-k |  
Marcela Díaz | 739458 | 16.836.745-2 | Subterra
```



Sistema de Biblioteca

- Procedimientos:
 - Ingreso/ Eliminación / Listado / Modificación de libros
 - Ingreso/ Eliminación / Listado / Modificación de usuario
 - Ingreso/ Eliminación / Listado / Modificación de préstamo
 - Búsqueda por autor: algoritmo simple, $O(n)$
- Se contrata a un programador que cree el sistema
- El sistema funciona bien durante cuatro meses. El bibliotecario se acostumbra a usarlo y le parece cómodo
- ... luego, él piensa que el sistema podría hacer otras cosas...



Sistema de Biblioteca

- ¡Nuevo requerimiento!
 - Los libros se empastan de vez en cuando, por lo que no se pueden prestar. Se requiere controlar la fecha de envío.
- Solución:
 - Modificar el archivo, agregar “estado” a los libros
 - **D** para disponible o **E** para cuando esté en empaste.
 - Agregar fecha de envío a empaste

```
<título> | <autor> | <editorial> | <D|E> | <fecha>  
...  
#  
<nombre> | <fono> | <RUT> | <libro1> | <libro2> | ...  
...
```



Sistema de Biblioteca

```
<título> | <autor> | <editorial> | <D|E> | <fecha>  
...  
#  
<nombre> | <fono> | <RUT> | <libro1> | <libro2> | ...  
...
```

- Modificar los siguientes procedimientos, para adaptarse al nuevo formato de archivo:
 - Ingreso/ Listado / Modificación de libros
 - Ingreso de préstamo
- Agregar los siguientes procedimientos:
 - Enviar libro a empaste
 - Retorno de libro desde empaste



Sistema de Biblioteca

- La biblioteca crece y la búsqueda por autor se hace ineficiente por la cantidad de información en el archivo.
- Se necesita reemplazar el algoritmo de búsqueda por otro más eficiente:
 - Se requiere ordenar los registros alfabéticamente por nombre de autor
 - Se incorpora un encabezado con el número de línea en donde empiezan los autores que comienzan con A, B, C, ...
- Lamentablemente, el programador inicial se cambió de ciudad...
 - Se contrata otro, que descubre que el código no estaba documentado
 - Aumenta la dificultad de modificar, hay algunos bugs y aumenta el costo del desarrollo



Sistema de Biblioteca

- Nuevo formato:

```
A <linea> B <linea> C <linea> ....  
#  
<título> | <autor> | <editorial> | <D|E> | <fecha>  
...  
#  
<nombre> | <fono> | <RUT> | <libro1> | <libro2> | ...  
...
```

- Modificar todos procedimientos para manejar nuevo formato de archivo



Sistema de Biblioteca

- Se contrata un ayudante del bibliotecario, el cual tiene derecho a modificar los préstamos, pero no los usuarios ni los libros
- Como el sistema operativo permite tener permisos por usuario se decide dividir el archivo en 2:

```
A <linea> B <linea> C <linea> ....  
#  
<título> | <autor> | <editorial> | <D|E> | <fecha>  
...  
#  
<nombre> | <fono> | <RUT>  
...
```

```
<RUT> | <libro1> | <libro2> | ...  
...
```

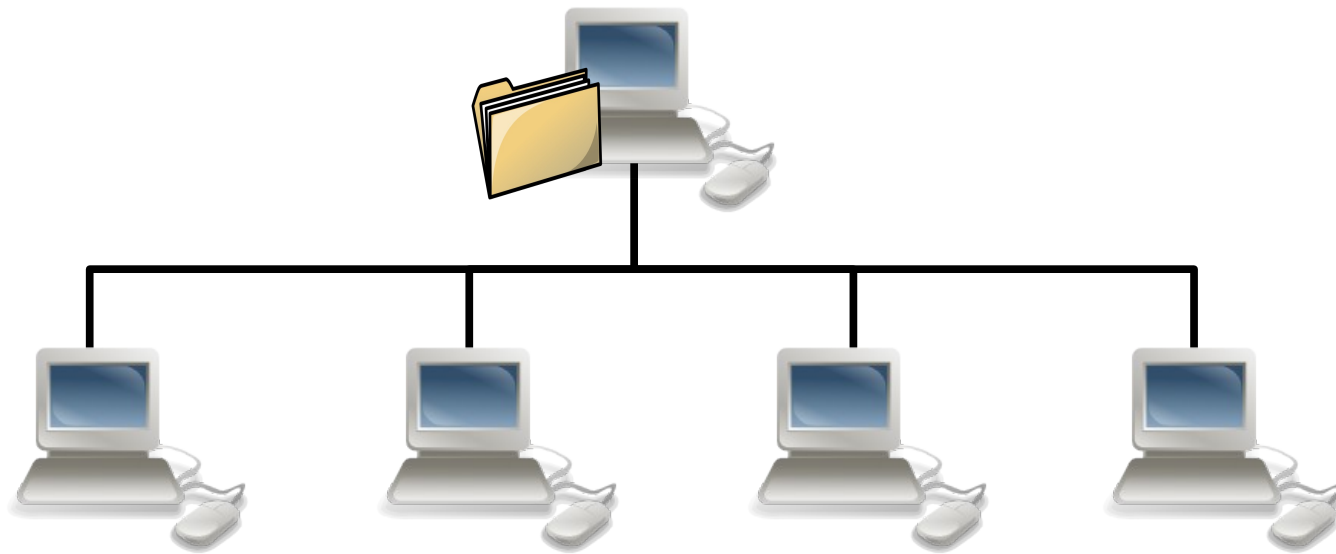
- Además, se modifican todos los procedimientos, y se incluye un login al sistema, y un archivo con los usuarios (no de la biblioteca, sino del sistema)

```
<usuario> | <password> | <RUT> | <perfil> ...  
...
```



Sistema de Biblioteca

- Para permitir a varias personas trabajar desde distintos computadores, se dejan los archivos de datos en un directorio compartido



- Aún así, la carga de datos a memoria es algo lenta...



Sistema de Biblioteca

- Más ayudantes llegan, pero no pueden usar el sistema al mismo tiempo, pues editan el mismo archivo de datos
- Solución:
 - Cada instancia del programa crea un archivo “ocupado.txt” en el directorio compartido antes de abrir el archivo de datos, y lo borra una vez que termina
 - Ninguna instancia puede abrir el archivo mientras ese archivo esté en el directorio
- Además, se modifican todos los procedimientos, para evitar escribir al mismo tiempo que otra instancia



Sistema de Biblioteca

- Un corte de luz sucede mientras se ingresaba un préstamo. El archivo abierto queda mal guardado y no se sabe quien pidió libros y quien no...
- Se “pierden” 54 libros y nadie sabe quién los tiene
- Se requiere modificar el sistema para hacerlo a prueba de fallas
- Solución:
 - Antes de modificar el archivo, se hace una copia. Una vez terminada la modificación sin problemas, ésta se borra
 - Si hubo problemas, la copia se usa como respaldo
- Se deben modificar todos los procedimientos para trabajar con el nuevo esquema de seguridad



Sistema de Biblioteca

- Durante la recuperación de la información, se descubren varios problemas con los datos
 - Algunas personas están ingresadas más de una vez:
 - En algunas líneas aparecen con préstamos y en otras sin
 - Los teléfonos no tienen un formato común:
 - Con y sin guiones, espacios y códigos de ciudad
 - Alguien editó el archivo de texto de préstamos, se sospecha que para robar libros
- Se realizó un inventario físico y se descubrió la desaparición de 132 libros. El sistema se dejó de usar pues ya no era confiable.

¿Qué hemos aprendido de esta historia?



Bases de datos vs sistema de archivos

- Problemas de trabajar con archivos:
 - Lenguajes de 3 generación -3GL- (Java, C++, Visual*, Python, etc.)
 - Se debe especificar **qué** hacer y **cómo** hacerlo
 - Gran esfuerzo en programación
 - un procedimiento para **cada** pregunta
 - Retrabajo ante nuevos requerimientos
 - Dependencia de datos: cómo éstos están organizados
 - Dependencia de estructura: organización del archivo
 - Los cambios son difíciles/lentos/caros de efectuar
 - Todo se hace “desde cero”
 - Compleja administración del sistema
 - Uso ineficiente del espacio de disco
 - Seguridad ineficiente y poco versátil
 - Consistencia de datos insuficiente



Ventajas de DBMS

- Independencia de datos
 - DBMS provee una vista abstracta de ellos
- Acceso de datos eficiente
 - Las operaciones de manipulación de datos están optimizadas para lograr una alta eficiencia
- Integridad y seguridad de datos
 - Robusto sistema de permisos y mantención de la coherencia entre los datos y la realidad
- Administración de datos
 - Centralizar la administración de datos hace ganar en eficiencia
- Acceso concurrente y recuperación ante fallos
- Reducción en tiempos de desarrollo de la aplicación
 - 4GL (enfocado a problemas específicos) facilita la programación



Enfoques del trabajo en DBMS

- **Diseño**

- ¿Cómo modelar los datos de un dominio para que representen bien la realidad?
- Involucra modelar datos, diagramas Entidad-Relación, etc.

- **Programación**

- ¿Cómo acceder a los datos directamente?
- ¿Cómo acceder a ellos desde un programa externo?
 - Involucra programar con 3GL para conectarse a DBMS, SQL, etc.

- **Implementación de DBMS**

- ¿Cómo implementar DBMS eficientes?
- Involucra trabajar con estructuras de datos, algoritmos de búsqueda y ordenamiento, estándares de seguridad, sistemas operativos, etc.



Un poco de historia

- 1960s Charles Bachman desarrolla el primer DBMS "Integrated Data Store" basado en Modelo de Red

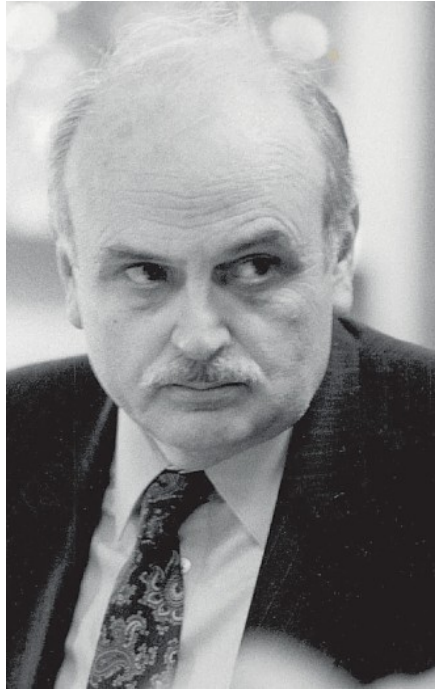


- 1960s: IBM desarrolla el "Information Management System"
 - basado en modelo de datos jerárquico
 - Sistema de reservas SABRE se desarrolla sobre él



Un poco de historia

- 1970 Edgard Codd desarrolla el modelo relacional
- 1976 Peter Chen desarrolla el diagrama de Entidad-Relación

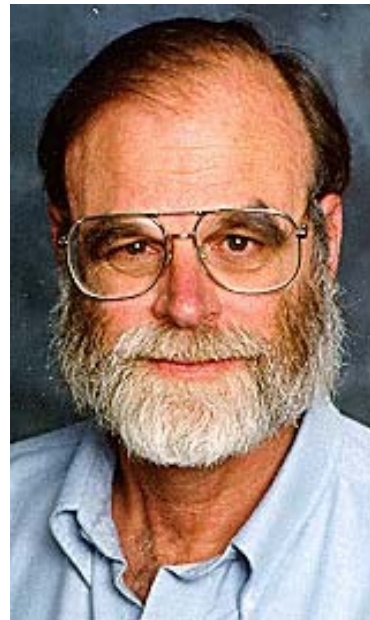


- 1973 C. Bachman recibe el 1er Turing Award



Un poco de historia

- 1980s Modelo relacional se impone
 - R Project de IBM (1974-78) desarrolla el lenguaje SQL, hoy estándar
 - La actual compañía Oracle participó en ese proyecto
 - Jim Gray desarrolla el modelo de transacciones para manejar concurrencia



Un poco de historia

- 1981 Codd recibe el Turing Award
- 1980s-1990s Desarrollo de grandes DBMS

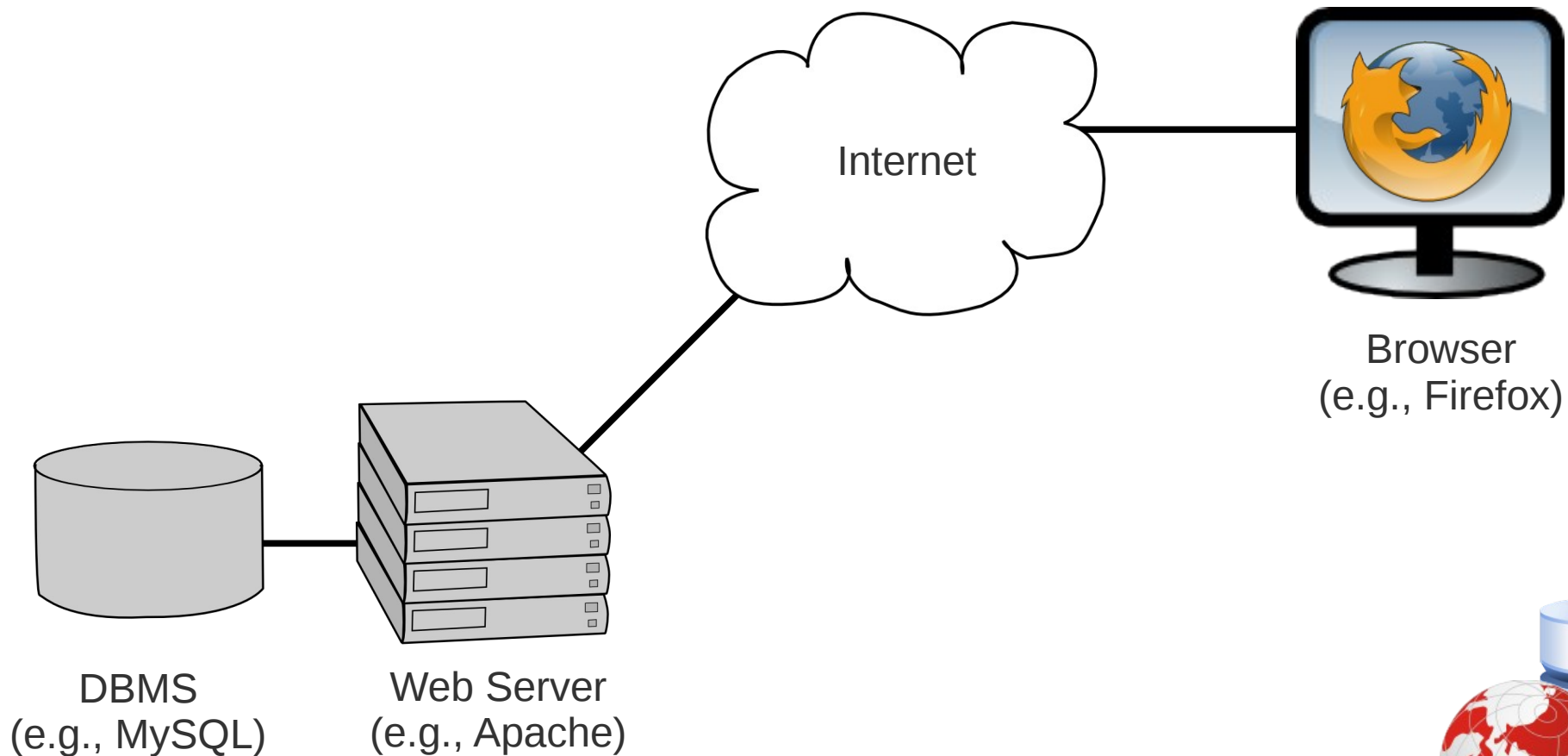


- 1980s Popularidad de lenguajes orientados a objetos impulsa el desarrollo de BD OO
- 1990s XML (eXtended Markup Language) emerge como un estándar de transferencia de datos estructurados



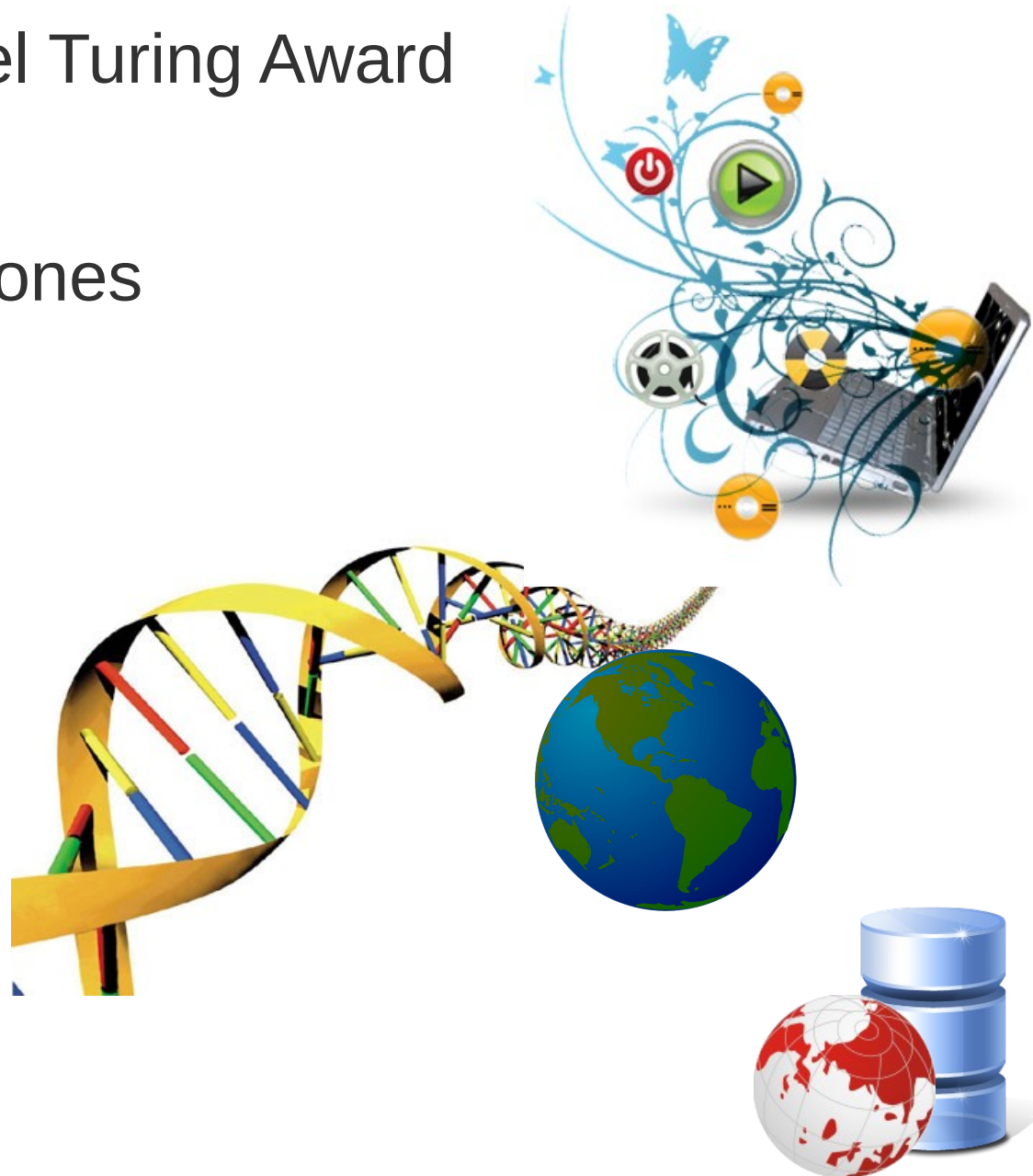
Un poco de historia

- 1990s se populariza la interconexión de DBMSs vía www: Sitios dinámicos, CMSs, etc.



Un poco de historia

- 1999 Jim Gray recibe el Turing Award
- 2000s Nuevas aplicaciones
 - BD multimedia
 - Video interactivo
 - Librerías digitales
 - BD distribuidas
 - Mapeo de genoma
 - DB geográficas
 - DB temporales



Resumen

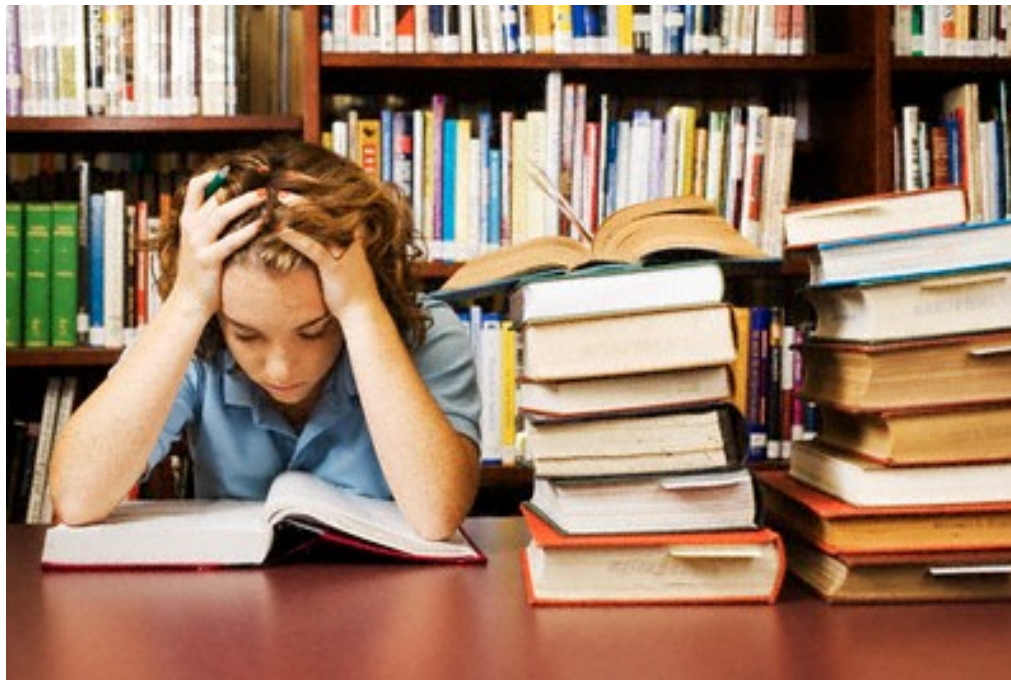
- BDs son conjuntos de datos estructurados sobre un dominio
- DBMS: programas que manejan BDs
- DBMS facilitan el buen manejo de los datos
- DBMS pueden conectarse a otros programas
- Existen diversos enfoques de trabajo en DBMS (diseño, programación, implementación de DBMS)
- BDs se vienen desarrollando desde 1960, hoy son ubicuas

¡Un informático no puede no saber sobre Bases de Datos!



Bonus!

- Lecturas para control:
 - Rafael Camps Paré, "Introducción a las bases de datos", UOC
 - Simsion, Witt "What is Data Modeling?" (Ch. 1)
 - Oppel, "The Database Life Cycle" (Ch. 5)



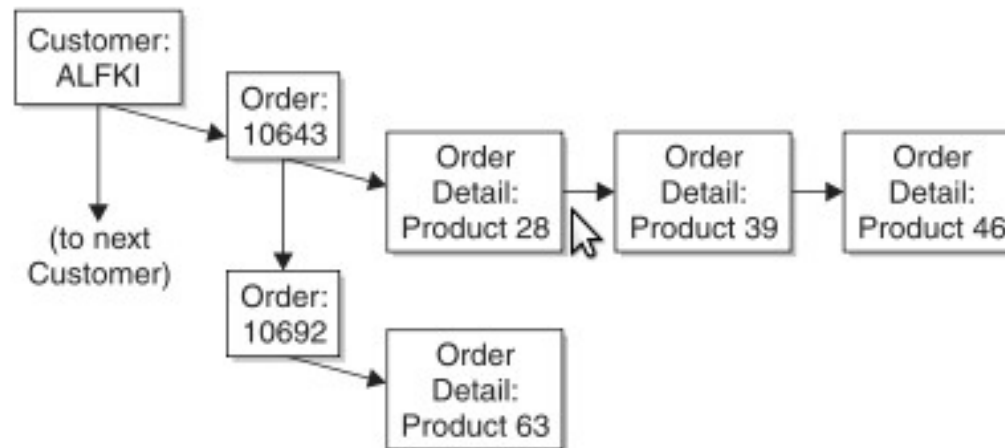
Introducción a las Bases de Datos

Parte 2



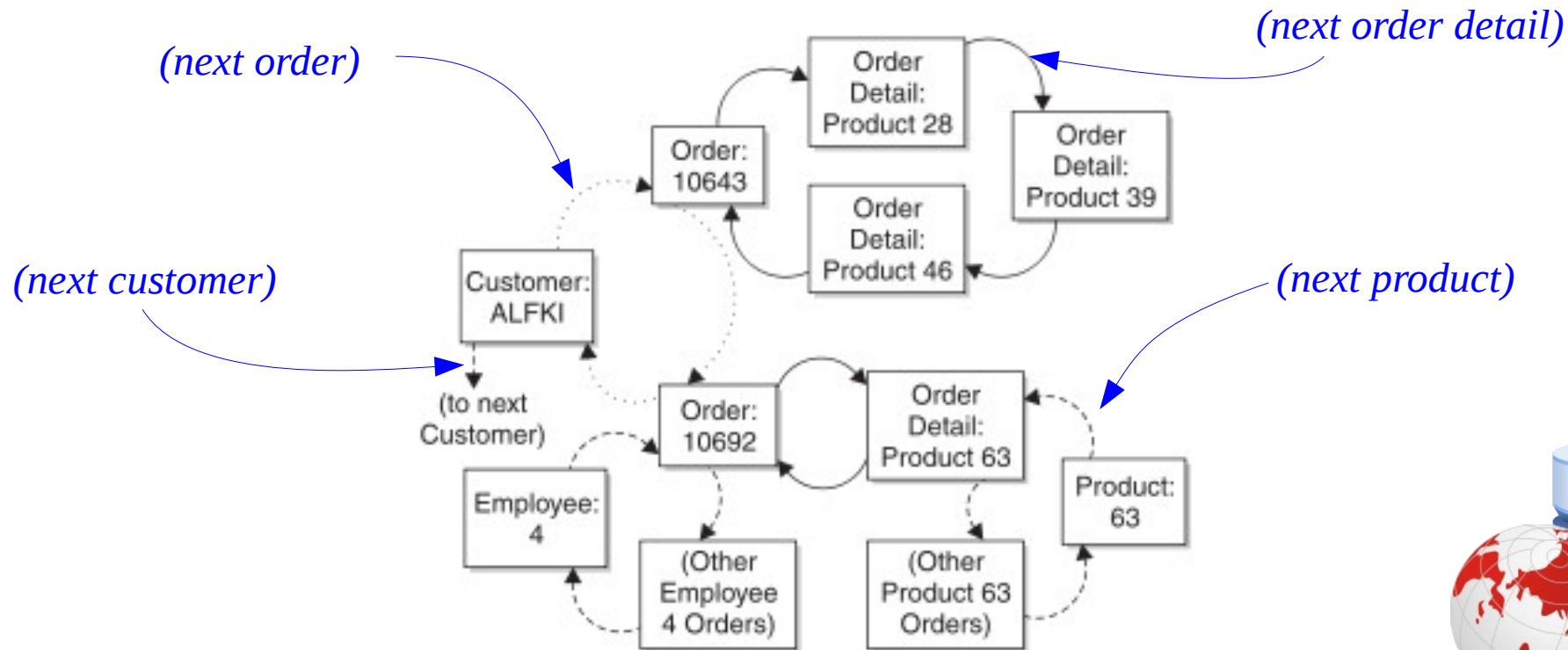
Modelos de DBMS

- La forma en que los DBMS operan puede clasificarse según diferentes criterios
- De acuerdo a la organización de los datos (“database model”):
 - Archivos “planos”
 - archivos del sistema operativo
 - En realidad no son BDs, sin embargo, los DBMS los usan para almacenar la información
 - Modelo Jerárquico
 - El más antiguo paradigma
 - Evolución de estructuras de datos de 3GL (ejemplo: structs/nodos en C)



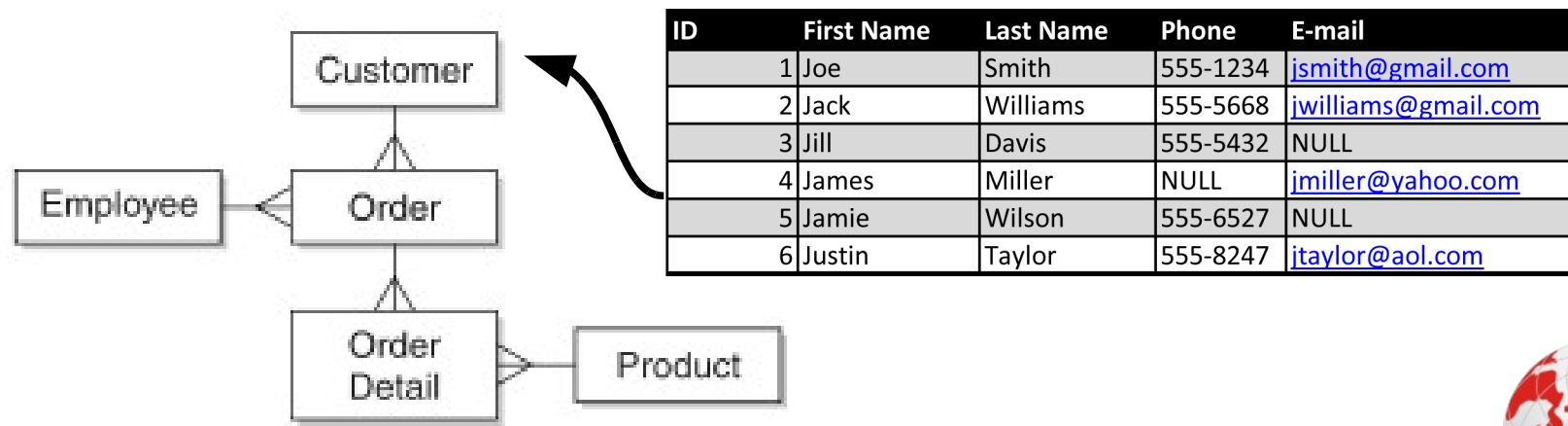
Modelos de DBMS

- ...de acuerdo a la organización de los datos (“database model”):
 - Modelo de Red
 - Similar al modelo jerárquico (desarrollado históricamente en paralelo)
 - Las relaciones entre registros (“punteros”) son diferenciadas
 - Más versátil, pero más complejo
 - Referencias circulares permiten ciclos de búsqueda (a veces infinitos)



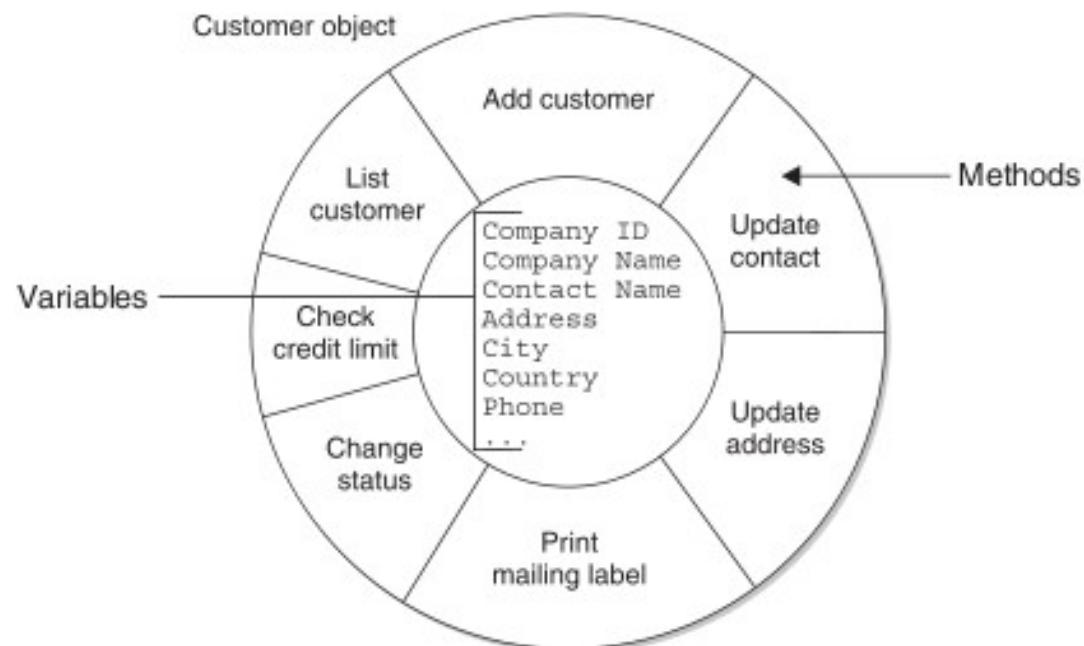
Modelos de DBMS

- ...de acuerdo a la organización de los datos (“database model”):
 - Modelo Relacional
 - Predominante en la actualidad
 - Intuitivo: los datos se guardan en tablas bidimensionales, como en una planilla
 - Se define la relación entre **conjuntos de registros** (tablas) en vez de entre **registros individuales**
 - Se pueden establecer relaciones entre registros después de haber generado el modelo
 - Más simple que los anteriores
 - Permite programar algoritmos más genéricos (búsquedas, inserciones, etc.)



Modelos de DBMS

- ...de acuerdo a la organización de los datos (“database model”):
 - Modelo Orientado a Objetos
 - Creado en 1970s, pero no considerado hasta 1990s
 - “Impedance mismatch” guardar en BD relacionales datos de programas orientados a objetos
 - Sigue el paradigma objetual: clase, objeto, herencia, etc.
 - Encapsulación estricta (objetos sólo se acceden a través de métodos)



Modelos de DBMS

- ...de acuerdo a la organización de los datos (“database model”):
 - Modelo Objeto-Relacional
 - En BDs relacionales se pueden hacer **queries** (consultas), de manera simple, lo cual no se puede hacer en una BD OO
 - Para muchos profesionales el enfoque relacional es natural, y muchas aplicaciones existentes lo utilizan
 - Buscando no perder mercado, algunos fabricantes mezclan ambos paradigmas, objetual y relacional
 - Nombre original: “universal databases”
 - Las grandes bases de datos actuales tienen algún grado de “OR”
 - Correspondencias:
 - Tablas asociadas a clases
 - Registros (tuplas) asociados a objetos



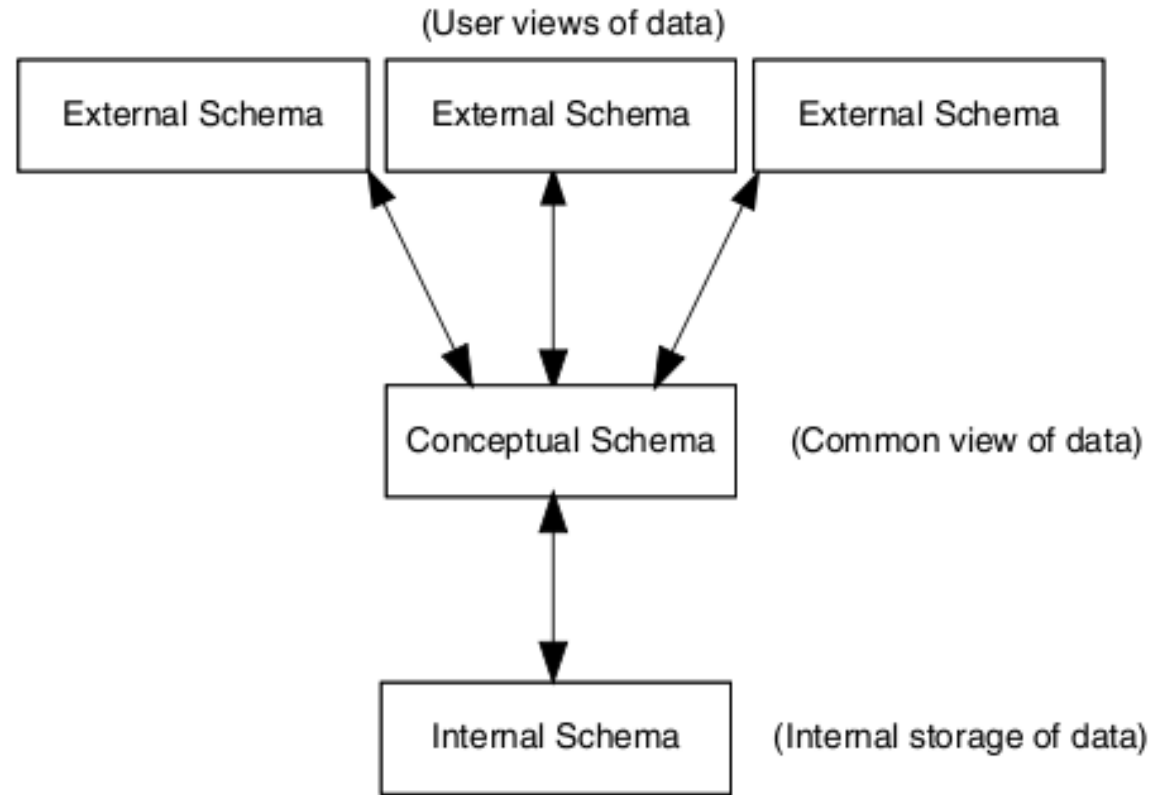
Modelos de DBMS

- De acuerdo al uso:
 - Transaccional
 - También llamada “de producción”
 - Diseñada para almacenar operaciones del día a día (registro de ventas, pasajes de aviones, log de usuarios, etc.)
 - Pobladas (carga de datos) desde los eventos diarios
 - Data warehouse
 - Diseñada para tomar decisiones de nivel táctico o estratégico (predicciones de ventas, análisis de mercado, etc.)
 - Pobladas desde BD transaccionales
 - “Destilado” de la información más importante
- De acuerdo al número de usuarios:
 - Monousuario
 - Multiusuario



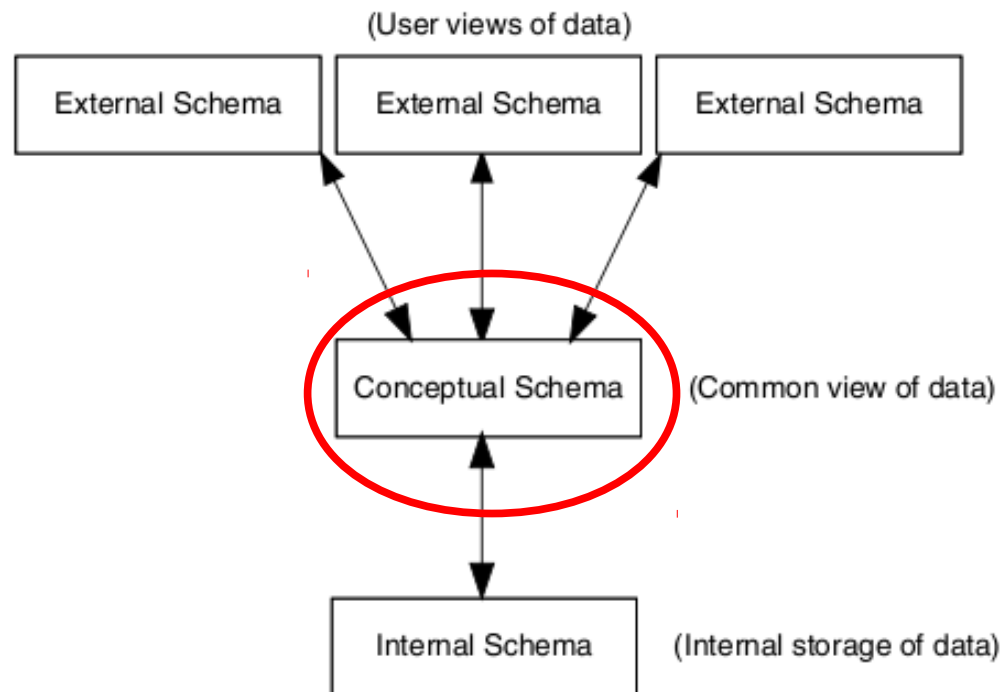
Niveles de abstracción en DBs

- Los datos en una BD pueden ser tratados en tres niveles de abstracción (ANSI/SPARK Data Model, 1978)
- La abstracción permite obviar los detalles de implementación de las capas inferiores, facilitando al manipulación de los datos



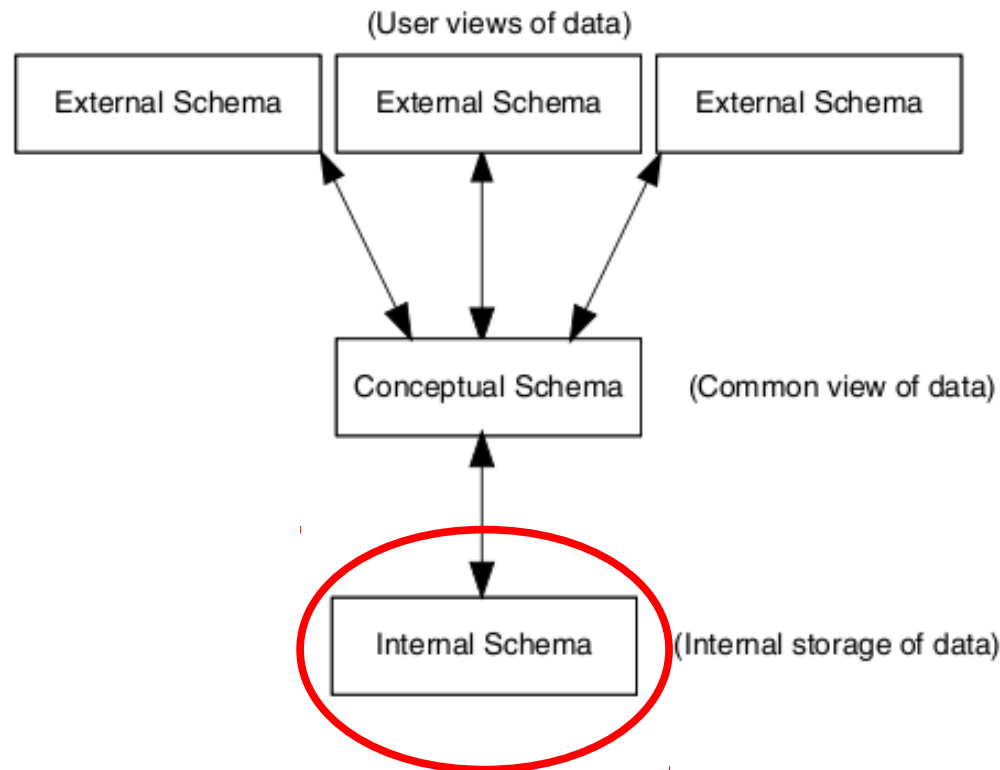
Niveles de abstracción en DBs

- Esquema conceptual:
 - Estructura lógica que representa el dominio (Biblioteca: libros, préstamos, multas, etc.)
 - Lo constituyen tablas, relaciones, etc.



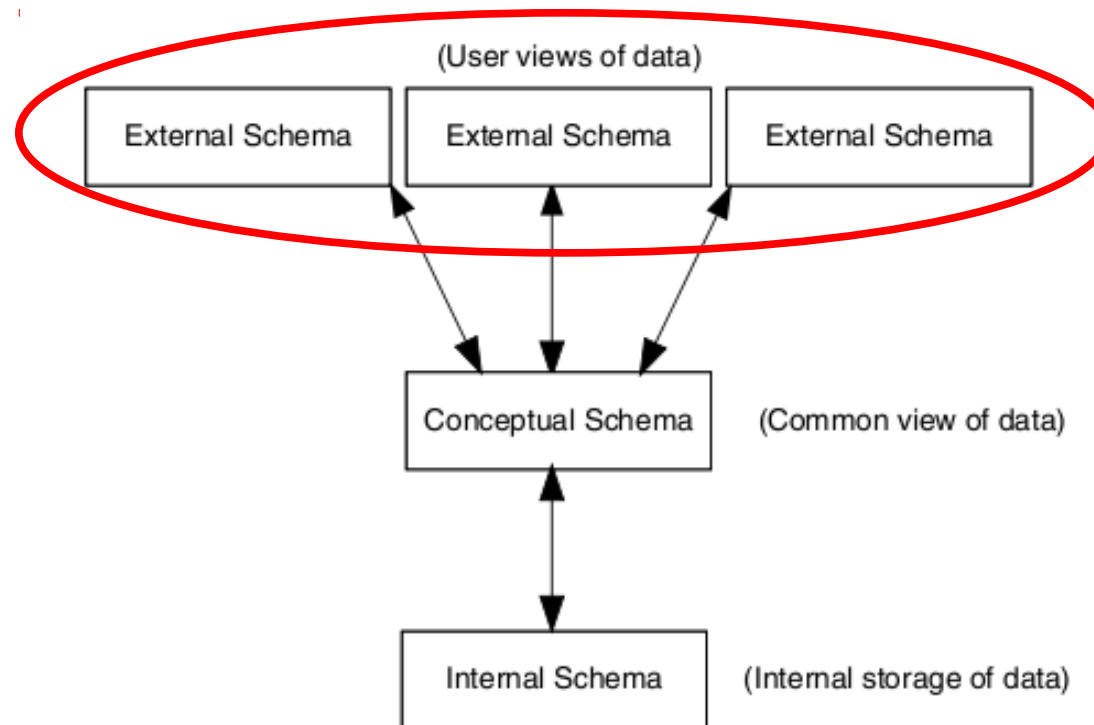
Niveles de abstracción en DBs

- Esquema físico
 - Cómo los datos están guardados en el disco
 - Lo constituyen archivos, índices, algoritmos de ordenamiento, protocolos de seguridad, etc.



Niveles de abstracción en DBs

- Esquema Externo
 - Como el usuario **ve** los datos almacenados en la BD
 - Subconjunto de datos “útil” para un usuario en particular
 - Tantas vistas como usuarios y consultas
 - Lo constituyen las vistas (**views**), especie de filtros de datos

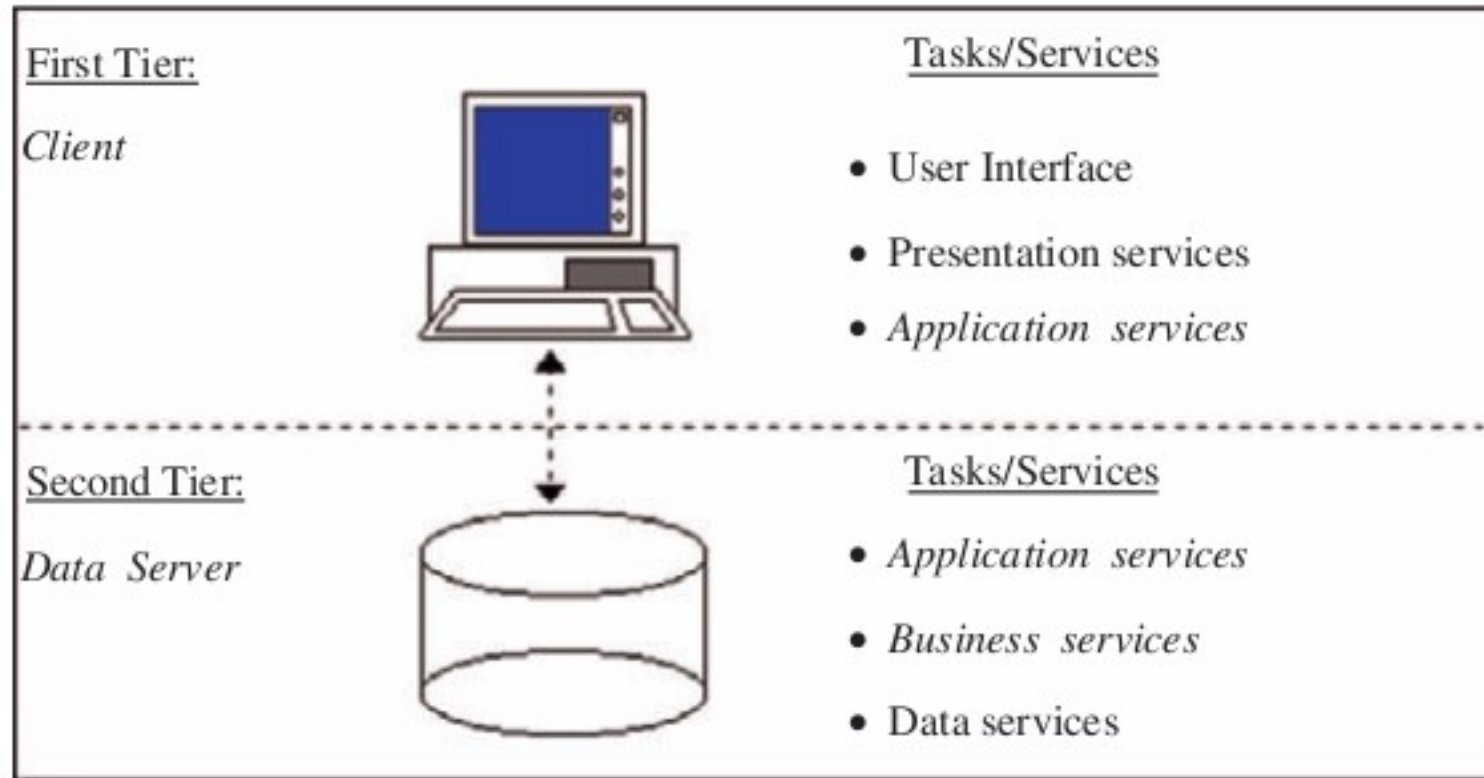


Despliegue de Sistemas DBMS

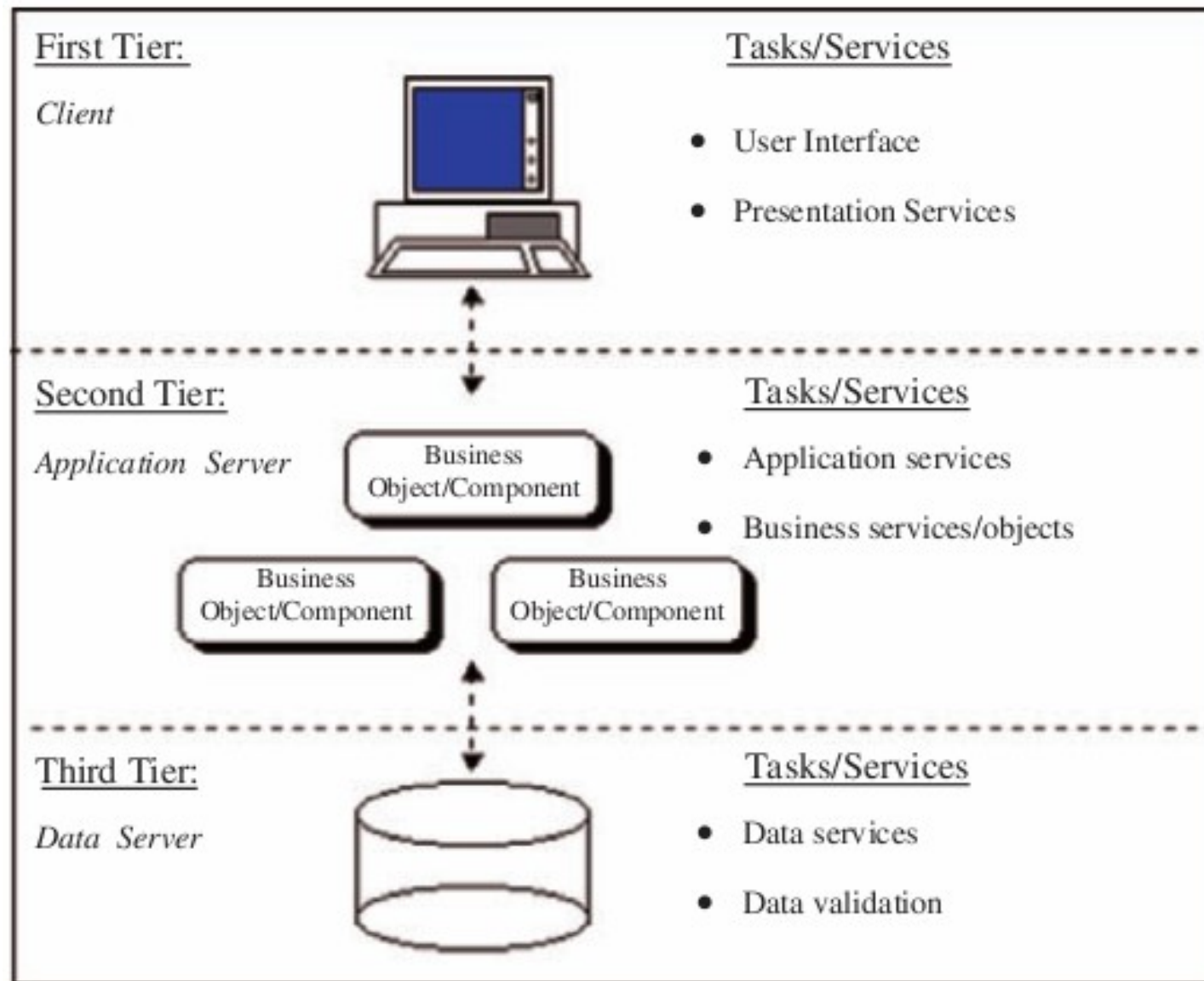
- Define cómo se “enchufa” un DBMS en el contexto de un sistema completo
- Se denomina arquitectura, y se organizan por **capas** (tiers)
- Se distinguen las siguientes:
 - 2-Tier (Cliente/Servidor)
 - 3-Tier
 - Multi-Tier



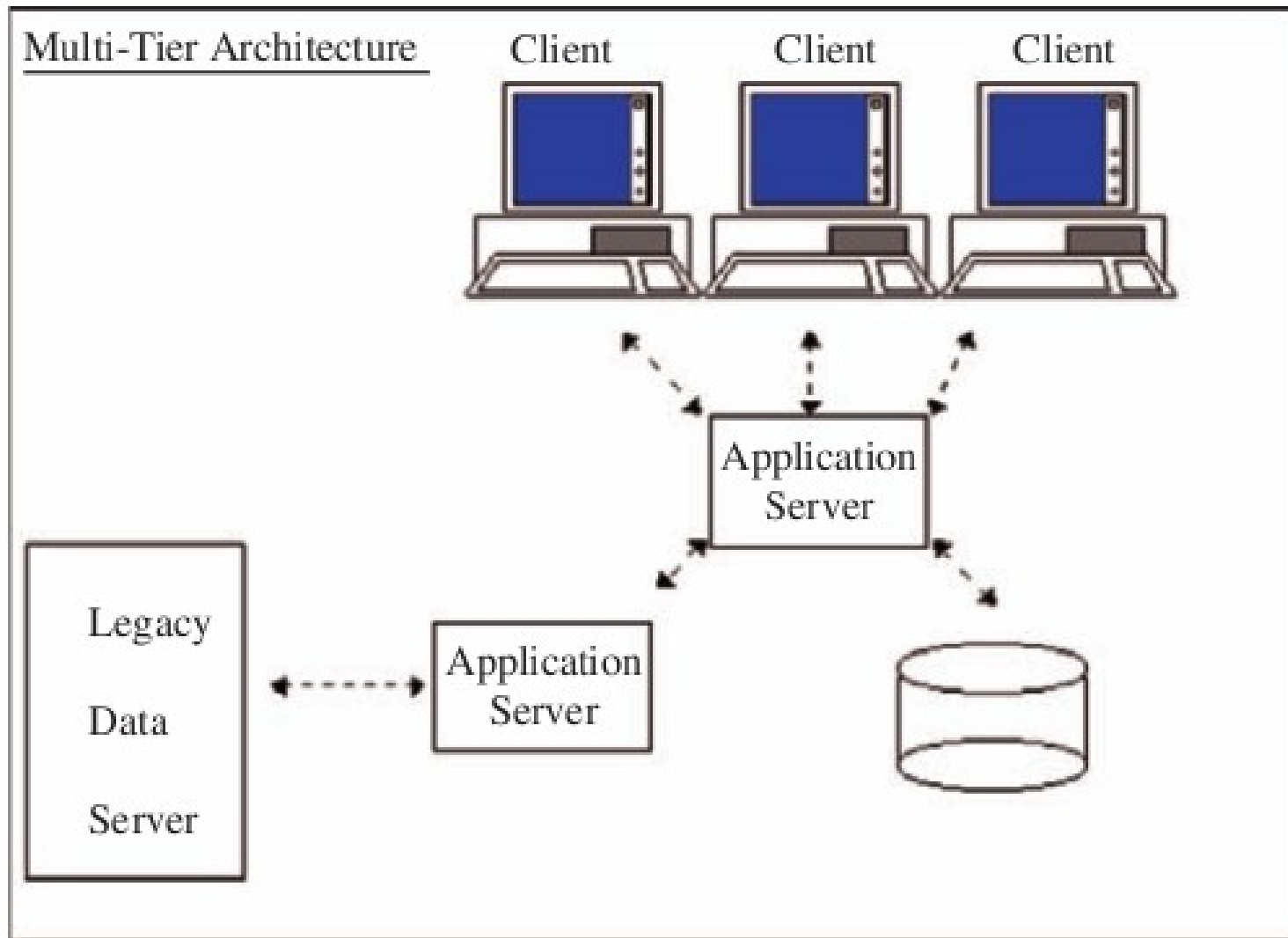
Arquitectura Cliente/Servidor



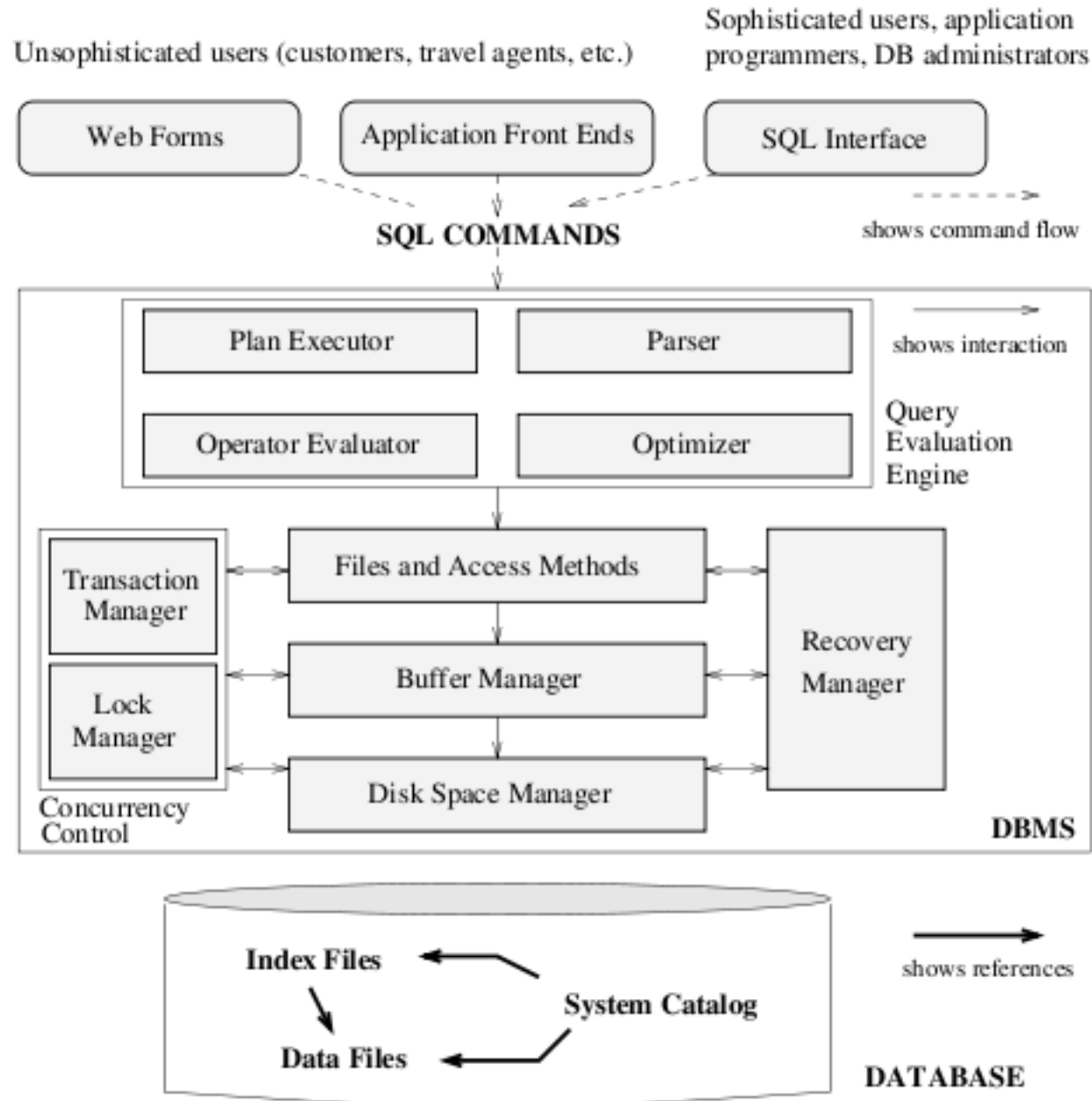
Arquitectura 3-Tier



Arquitectura Multi-Tier



Estructura de un DBMS



Resumen

- Existen múltiples modelos de BDMS (Relacional, Objetual, etc.)
- El más utilizado actualmente es el relacional
- Los datos de una BD pueden considerarse en 3 niveles de abstracción diferentes, llamados "esquemas"
- Existen diferentes arquitecturas para integrar BDs en sistemas de información (n-Tier)
- Los BDMS son sistemas complejos, con distintos componentes, encargados entre otros, de la concurrencia, acceso a datos y seguridad ante fallas



Introducción a las Bases de Datos

Parte 3

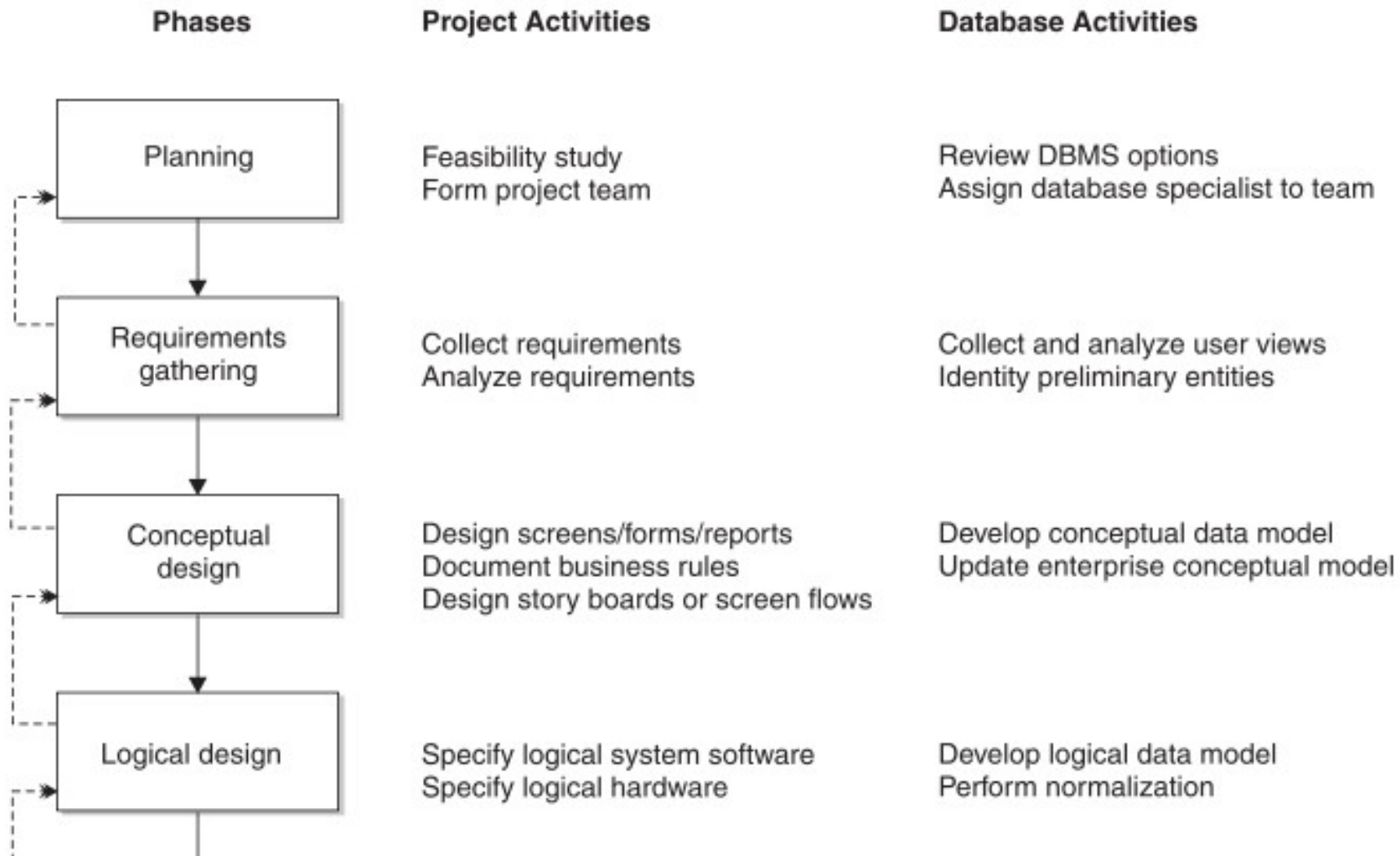


Ciclo de Vida de una BD

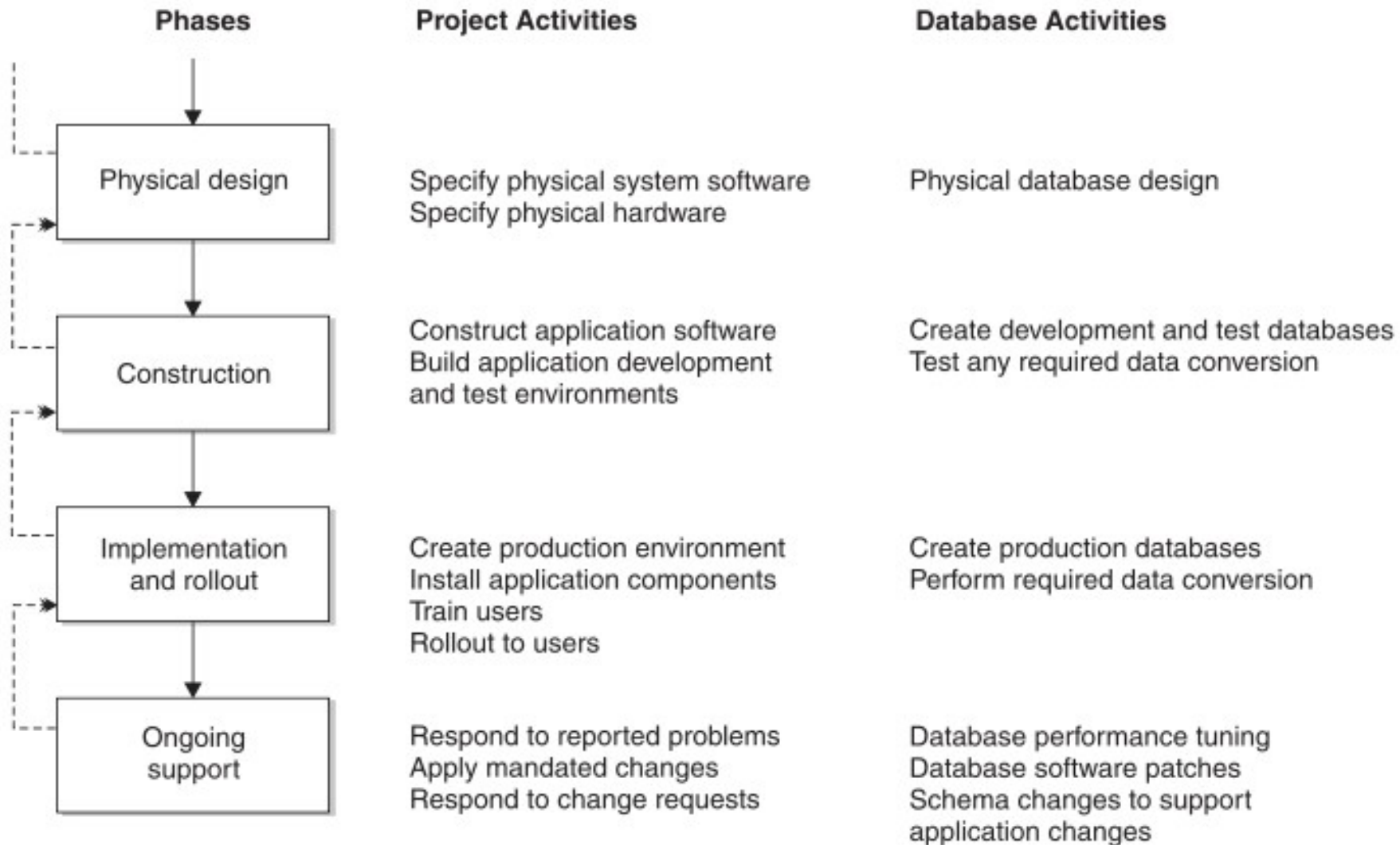
- Ciclo que se extiende desde que se detecta la necesidad de contar con una BD hasta el cese de sus funciones.
- Relacionado con la Ingeniería de Software
- Contempla una serie de ***pasos a seguir***
- Al igual que otras construcciones ingenieriles (autos, edificios, etc.) puede hacerse de distintas maneras
- Consiste, básicamente, en recopilar los requerimientos de los usuarios, para hacer un diseño ordenado antes de programar el sistema. Posteriormente viene la puesta en marcha y la mantención y adaptación (soporte)
- A continuación veremos el ciclo tradicional, en el cual los pasos son secuenciales (con posibilidad de devolverse al paso anterior)



Ciclo de Vida de una BD



Ciclo de Vida de una BD



Otros ciclos de desarrollo

- El ciclo tradicional ordena el desarrollo de software, pero tiene dos inconvenientes principales dada su secuencialidad:
 - Aumento de tiempos y costos al no poder traslapar fases
 - Rigidez, en particular respecto a nuevos requerimientos
- Ejemplos de -familias de- ciclos no tradicionales:
 - Prototipos (familia de métodos)
 - Se crean prototipos, ya sean de interfaz o del sistema completo
 - Se involucra al cliente, el cual refina sus requerimientos a lo largo del desarrollo
 - El producto obtenido tiene mayor aceptación
 - El paralelismo dificulta la gestión del proyecto, atención al control de cambios
 - RAD (Rapid Application Development)
 - Combina diferentes metodologías de desarrollo enfatizando la rapidez del desarrollo
 - Alterna creación de modelos de datos y negocios preliminares con prototipos
 - Otros ciclos
 - Espiral, evolutivo, iterativo, etc. (La mayoría basados en ciclos)



Especificación de Requerimientos

- Todo sistema informático es desarrollado con un propósito y dentro de un contexto.
- Esta fase busca establecer o refinar la definición del sistema a construir
 - (Entrevistas, prototipos, estudio de documentos, etc.)
- A este contexto (sistema) se le denomina “***Dominio***” aka “***Universo de Discurso***”
 - Biblioteca de la universidad, club de fútbol, acelerador de partículas, en general, un “*negocio*” [neg-ocio]
- “***Reglas de negocio***”: conjunto de reglas que definen cómo funciona el sistema en el cual la solución se implantará
 - Pueden ser restricciones, definiciones u operaciones, generalmente expresadas en lenguaje natural



Dominio y Reglas de Negocio

- Ejemplo:

“La Universidad Austral es una institución de educación superior. En ella participan alumnos y funcionarios. Estos últimos pueden ser académicos o administrativos.

Los estudiantes se matriculan en carreras, las cuales están compuestas de un conjunto de asignaturas y un trabajo de título. Para obtener un título profesional el estudiante debe aprobar todas las asignaturas y desarrollar exitosamente su trabajo de título.

Existen también estudiantes de postgrado, los cuales optan a grados académicos de magíster o doctorado. Sólo profesionales pueden ser estudiantes de postgrado.

Por su parte, los académicos imparten clases y realizan investigación. Las clases reúnen grupos de estudiantes, y se organizan en asignaturas, mientras que la investigación se organiza en proyectos. Los académicos pueden desempeñar cargos administrativos, como Decanos o directores de Institutos.”

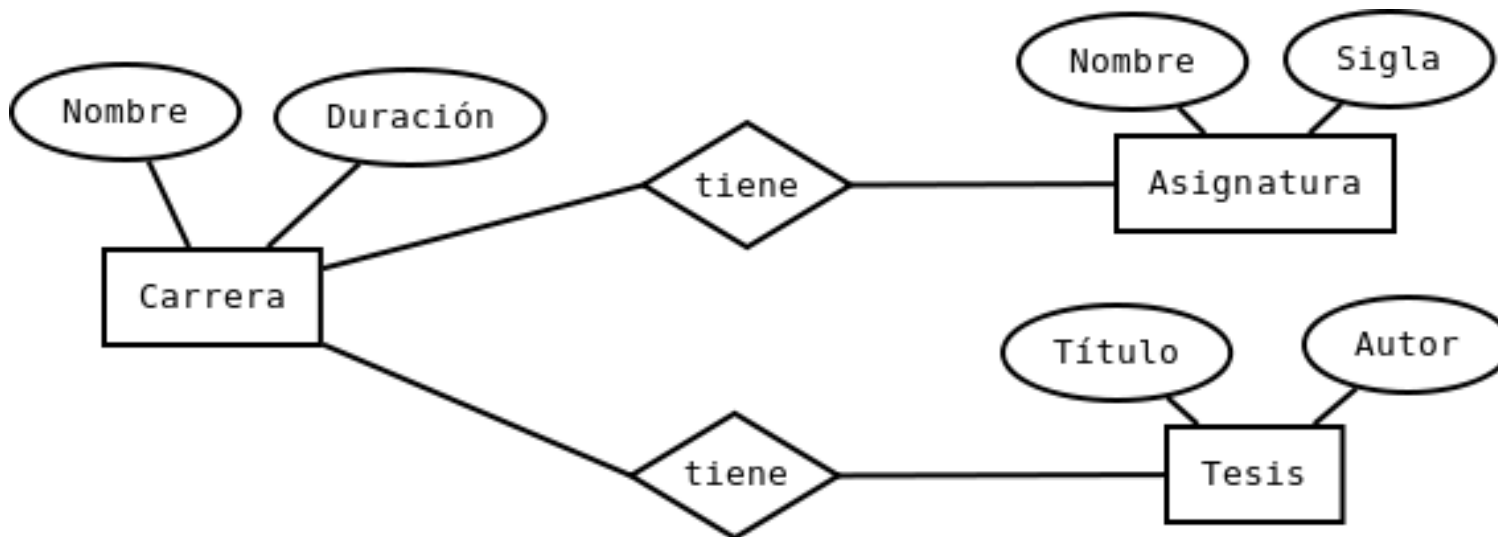
- (En la “vida real” la especificación de requerimiento es un documento mucho más extenso)



Diseño Conceptual

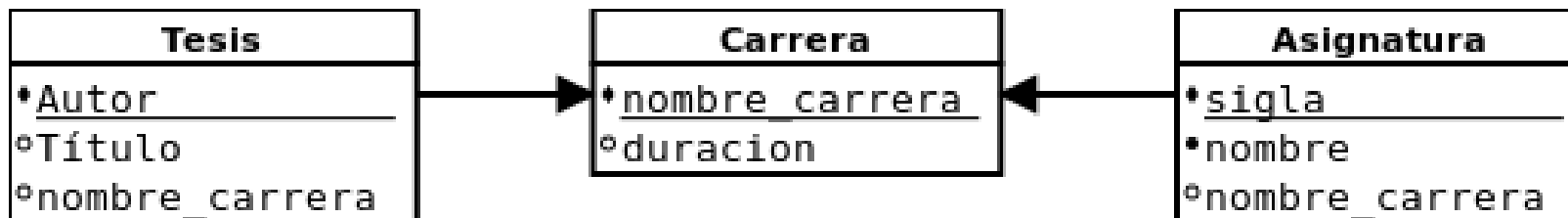
- Una vez que se definen -o refinan- los requerimientos, se diseña la solución
- Los datos se diseñan mediante **Diagramas de Entidad-Relación**
 - Modelo conceptual que expresa los elementos y sus relaciones
- Ejemplo:

"[...] carreras, las cuales están compuestas de un conjunto de asignaturas y un trabajo de título"



Diseño Lógico

- El Modelo ER es una representación abstracta. Debe ser traducida al paradigma que utilice la BD en la cual se implementará el sistema
- Al traducir se opta por un modelo de DBMS (relacional, objetual, jerárquico, etc.)
- En este paso también se **normalizan** las tablas
 - Criterios para aprovechar mejor el espacio utilizado y aumentar la mantenibilidad de la BD
- Ejemplo:
 - modelo relacional correspondiente al ER anterior



Diseño Físico

- Finalmente, el diseño lógico se expresa en un lenguaje que permite la creación de la BD. Programas se denominan **scripts**
- Para BD relacionales, es **SQL** (Structured Query Language), 4GL
- SQL permite:
 - Crear la BD (**DDL**: Data Definition Language)
 - Modificar la información (**DML**: Data Modification Language)
 - Hacer consultas a la BD (**DQL**: Data Query Language)
 - Gestionar el acceso a la BD (**DCL**: Data Control language)
- Ejemplo:
 - Script de creación de la tabla Carrera del ejemplo anterior

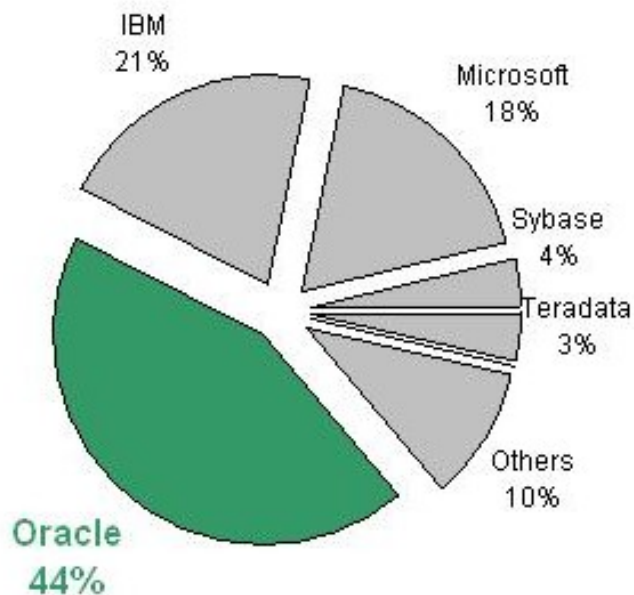
```
CREATE TABLE Carrera (  
    nombre_carrera varchar2,  
    duracion number(2),  
    constraint ca_pk primary key (nombre_carrera)  
);
```



¿Por qué enfocarse en Relacional?

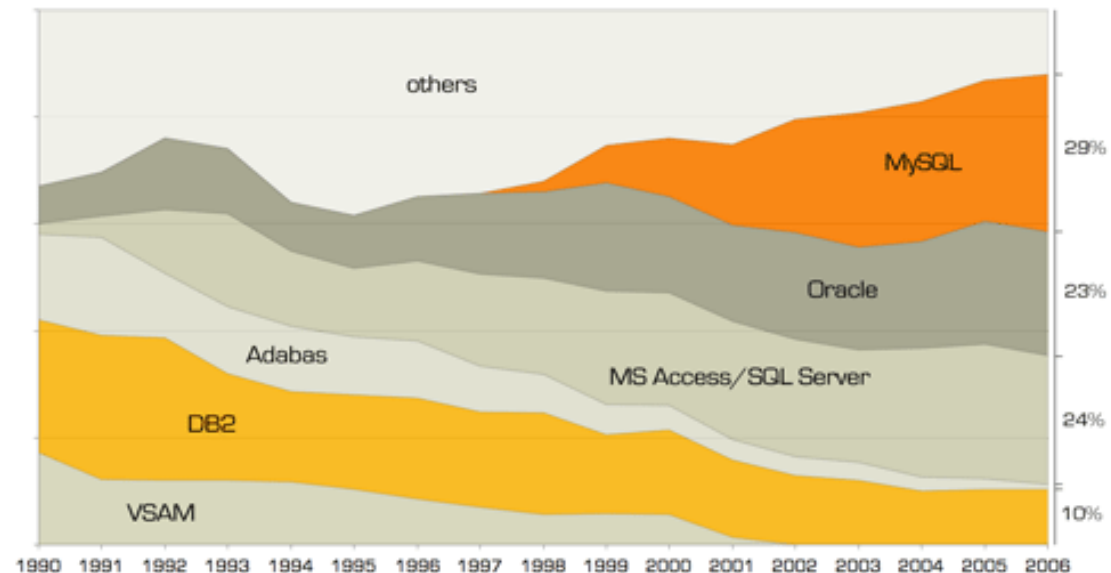
- Modelo relacional es el más utilizado en la actualidad
- Madurez de RDBMSs
- Estándar asentado: “aprender una RDBMS es aprenderlas todas”
- Baja complejidad de aprendizaje

RDBMS, Market Share



Source: IDC (2007), NQ Logic (2009)

Instalaciones de DBMS



¿Quiénes manejan bases de datos?

- BD son centrales en informática, es casi imposible no trabajar con ellas
- Diversos roles:
 - Administrador de BD (“DBA”)
 - Autoriza accesos, Coordina y monitorea su uso, adquiere recursos de HW y SW según necesidad, Backup & Recovery, etc.
 - Diseñador de BD
 - Identifica datos a ser almacenados en BD, elige las estructuras más adecuadas
 - Usuarios (dos categorías)
 - Programadores de aplicaciones: escriben programas que consultan la BD
 - Usuarios finales: consulta la BD directamente



Fin de la Introducción

- ¿Qué viene a continuación?
 - Diseño Conceptual (Modelos ER)
(Cómo expresar requerimientos de manera estructurada)
 - Diseño Lógico (Modelos Relacionales)
(Cómo expresar el Modelo ER para RDBMSs)
 - Diseño Físico (SQL y PL/SQL)
(Creación y manipulación de BDs en la práctica)



Resumen

- Las BDs existen en un contexto de negocio (dominio)
- Su desarrollo está ligado al desarrollo del sistema completo
- Existen diferentes ciclos de vida, secuenciales y cíclicos
- Fases principales:
 - Especificación de requerimientos: QUÉ hacer
 - Diseño: CÓMO hacerlo
 - Conceptual: relación entre conceptos (Modelo ER)
 - Lógico: adaptar al modelo de la DBMS que se usará (Relacional)
 - Físico: expresarlo en lenguaje del DBMS (SQL)
- Foco en Modelo relacional debido a su popularidad y conveniencia académica
- Distintos profesionales interactúan con DBMSs



Diseño Conceptual

Parte 2



Entidades fuertes y débiles

- Las entidades pueden clasificarse según la fortaleza de sus atributos identificadores (clave). Existen dos tipos
- Entidades ***Fuertes***
 - Tienen “vida propia”, es decir, no dependen de otra entidad para existir
 - Tienen una clave que las identifica por completo
- Entidades ***Débiles***
 - Dependen de una entidad fuerte para existir, solas no tienen sentido (*compra sin artículo, habitación sin hotel*, etc.)
 - Necesitan incluir la clave de la entidad fuerte de la que dependen para poder identificarse totalmente
 - Pueden no tener un atributo clave propio



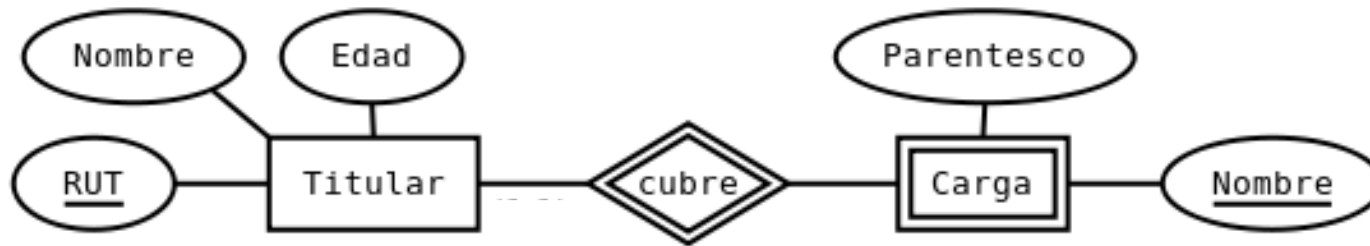
Entidades débiles

- Ejemplos:

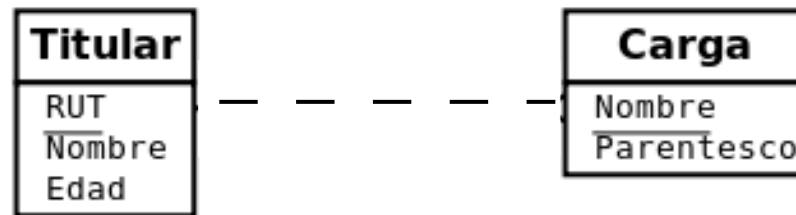
- “En una Isapre, los familiares del titular (cargas) están cubiertos por el plan de salud de éste”
- Las cargas no pueden existir si no existe el titular

- Notación:

- Chen:



- Crow's Feet:



Ejercicio de la clase anterior

- Ejercicio: crear diagramas de E/R correspondiente al siguiente texto:
 - *“Se desea crear una BD de alumnos de la UACH. De cada uno se conoce su nombre, su RUT, la carrera que estudia, y la fecha de nacimiento. Cada carrera posee un nombre, una duración, y un director, y está asociada a un instituto. Las carreras están compuestas de ramos, los cuales tienen un nombre, una sigla y una cantidad de créditos. Los ramos son dictados por al menos un profesor, los cuales están identificados por su nombre y RUT. Un profesor puede dictar hasta 4 ramos por semestre”*



Un modelo más detallado

- Dudas del modelo anterior:
 - ¿Instituto debe ser una entidad o un atributo de Carrera?
 - ¿El director de la carrera es un profesor?
 - Si es así, ¿debería existir una relación entre carrera y profesor, en vez de un atributo en carrera?
 - En la relación Ramo-Profesor, ¿cómo se expresa que...
 - ...un ramo es impartido por “**al menos un**” profesor?
 - ... un profesor imparte un “**hasta 4 ramos**”?
 - Profesores y alumnos son personas ¿deberían estar agrupados bajo una entidad “Persona”?

A estas preguntas nos dedicaremos a partir de hoy



Cardinalidad de Relaciones

- Las entidades asociadas a través de relaciones son conocidas como participantes
- Los participantes pueden relacionarse en distinto número, esto es llamado **cardinalidad** de una relación
- Básicamente, existen tres tipos:
 - Uno a uno (1:1)
 - Uno a varios, aka uno a muchos (1:N)
 - Varios a Varios, aka muchos a muchos (N:N, N:M)



Relación 1:N (o N:1)

- Ejemplos:
 - “Una carrera se compone de varias asignaturas”
 - “Un reloj se compone de varias piezas”
 - “Una biblioteca tiene muchos libros”
- Notación:
 - Chen:



- Crow's Foot



Relación N:M

- Ejemplos:

- “Un estudiante toma varios cursos, un curso es tomado por varios estudiantes”
- “Un supermercado vende varios productos, un producto es vendido en varios supermercados”

- Notación:

- Chen:

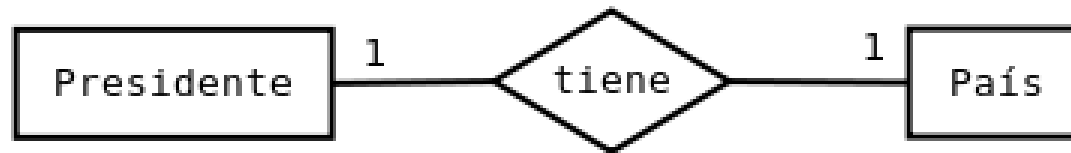


- Crow's feet

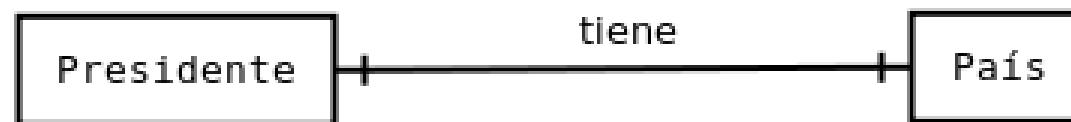


Relación 1:1

- Menos utilizadas que las anteriores
- Ejemplos:
 - “Un país tiene sólo un presidente, un presidente preside sólo un país”
 - “En un matrimonio, dos personas está ligadas exclusivamente”
- Notación:
 - Chen:

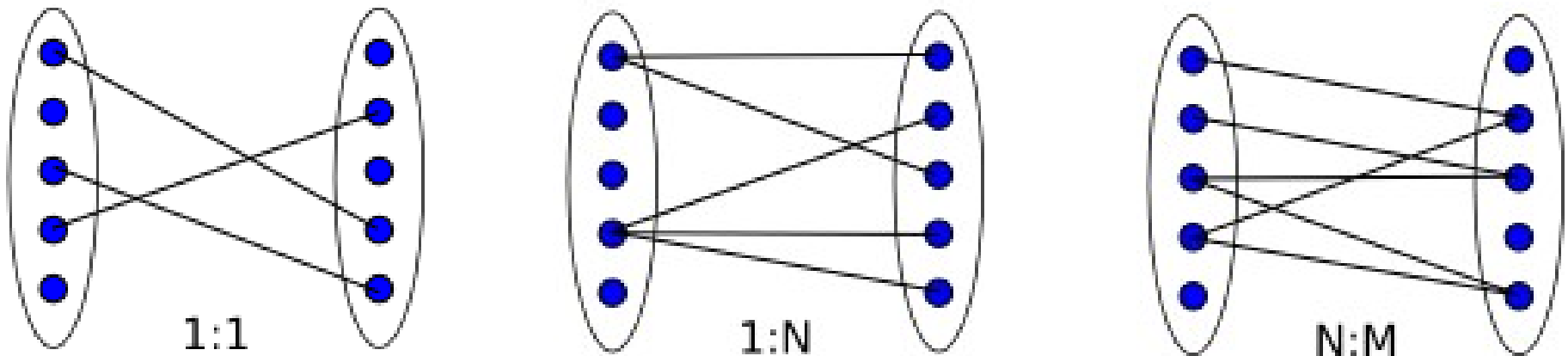


- Crow's feet:



Relaciones entre conjuntos

- Los conjuntos de entidades y sus relaciones no son otra cosa que conjuntos y relaciones matemáticas
 - Diagramas de relaciones:



- N.B. En estos ejemplos no se exige que todas las entidades participen de la relación. Esta exigencia podría ser impuesta, como veremos a continuación



Participación

- En general, se distinguen dos niveles de participación:
 - **Opcional**: puede o no haber entidades relacionadas (min=0)
 - **Obligatoria**: debe haber al menos una entidad asociada (min=1)
- En algunos casos, se puede exigir la participación de un número determinado de entidades en una relación
- Este número puede estar:
 - Acotado a un mínimo: “al menos ____”
 - Acotado a un máximo: “a lo más ____”



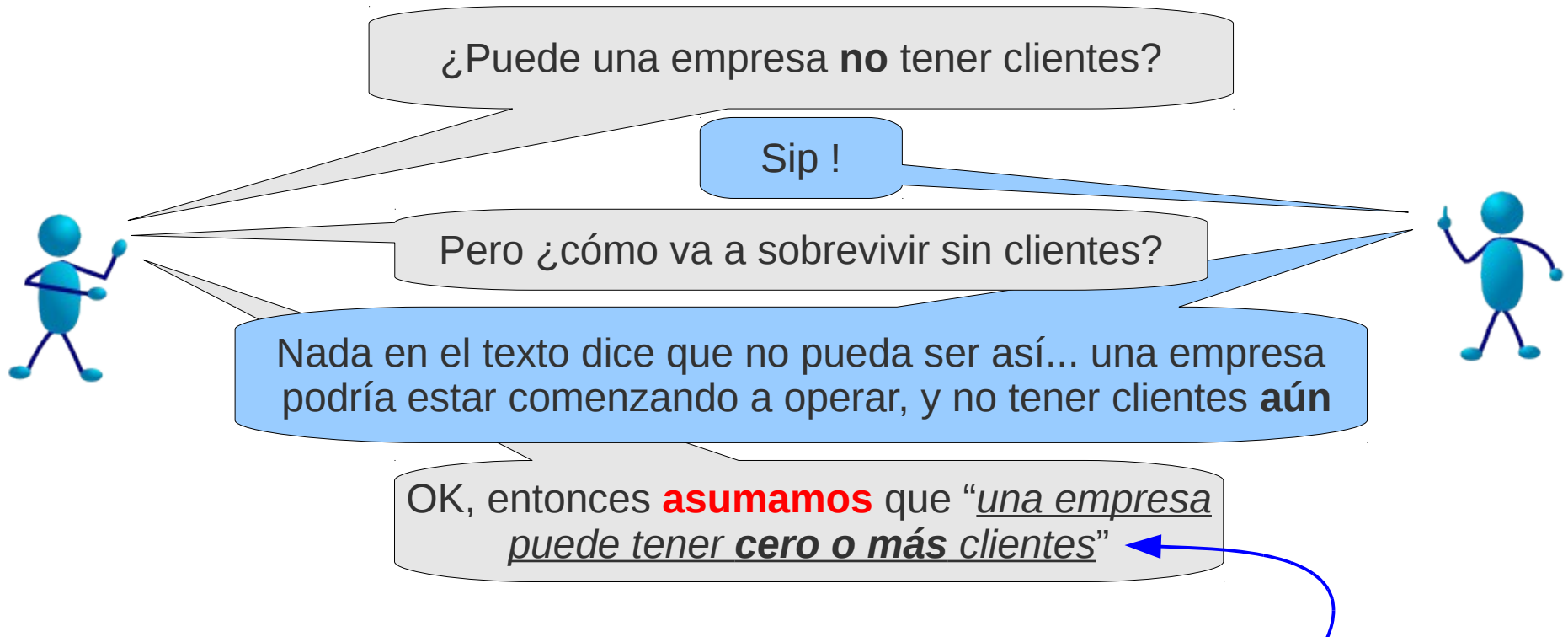
Participación opcional

- Ejemplos:
 - “Una persona puede estar casada con otra”
 - “Un árbol puede tener varios nidos”
- En estos casos, no hay problema si las entidades **no** se relacionan:
 - Una persona puede **no** estar casada
 - Un árbol puede **no** tener nidos



Participación opcional

- Dependiendo del dominio, se puede asumir por defecto:
 - e.g. “una empresa tiene varios clientes”:



- La suposiciones se explicitan como “**supuestos semánticos**”



Participación obligatoria

- Se requiere que al menos una entidad se relacione
 - “La sociedad debe estar constituida por al menos un responsable”
 - “La demanda debe estar presentada por al menos una persona natural o jurídica”
 - “Un mail debe estar dirigido por lo menos a un destinatario”
- Un caso especial es cuando se requiere un mínimo mayor a 1:
 - “debe haber al menos 5 personas inscritas para que un curso se dicte”
 - “se necesitan al menos 11 jugadores para formar un equipo de fútbol”



Cardinalidad + Participación

- Ejemplo:
 - “Una asignatura es dictada sólo por un profesor, un profesor puede dictar hasta 4 asignaturas”
- Notación:
 - Chen: Se usa notación (min,max)



- Crow's feet: no puede expresar mínimos diferentes de 0 ó 1, ni máximos

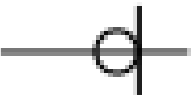

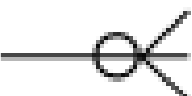
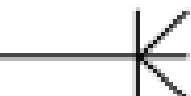


Resumen de notaciones

- Notación:

Crow's Foot

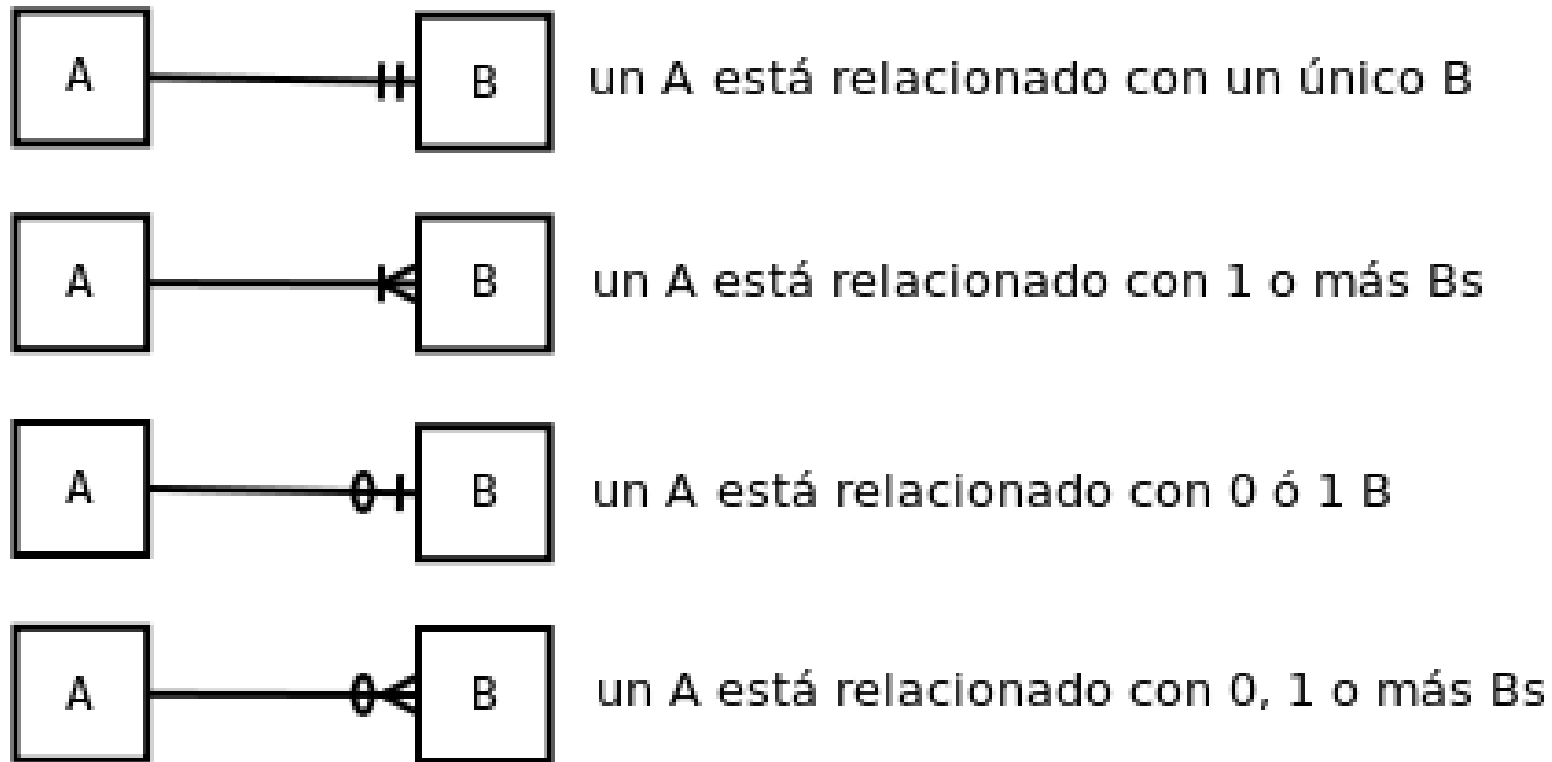
Chen

		Participación	
		Opcional	Obligatoria
Cardinalidad	Uno	 (0,1)	 (1,1)
	Muchos	 (0,N)	 (1,N)



Relaciones en Crow's Feet

- Dicho de otra manera:



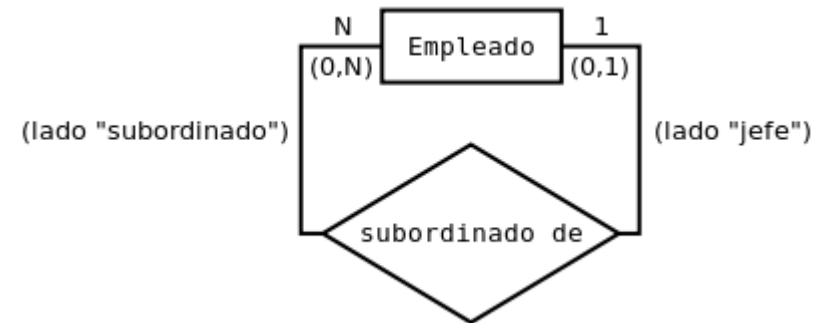
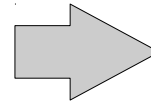
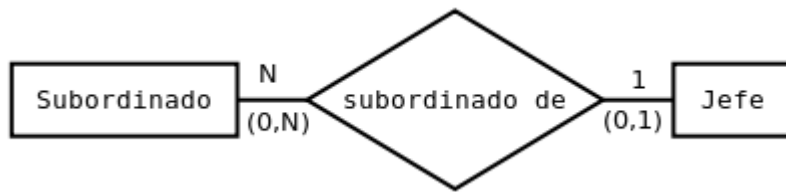
Tipos de Relaciones

- Hasta ahora sólo hemos visto relaciones entre dos conjuntos de entidades
- Las relaciones se pueden clasificar según el número de participantes:
 - Unarias (una entidad)
 - Binarias (dos)
 - Ternarias (tres)
 - N-arias (N, en general)

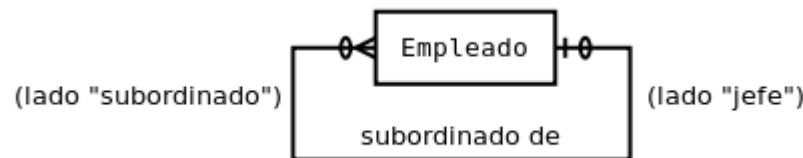


Relaciones unarias

- Se establecen entre entidades del mismo conjunto
- Ejemplos:
 - “Los empleados de la empresa tienen un superior jerárquico”
 - “El matrimonio un contrato establecido entre dos personas
- Notación:
 - Chen:

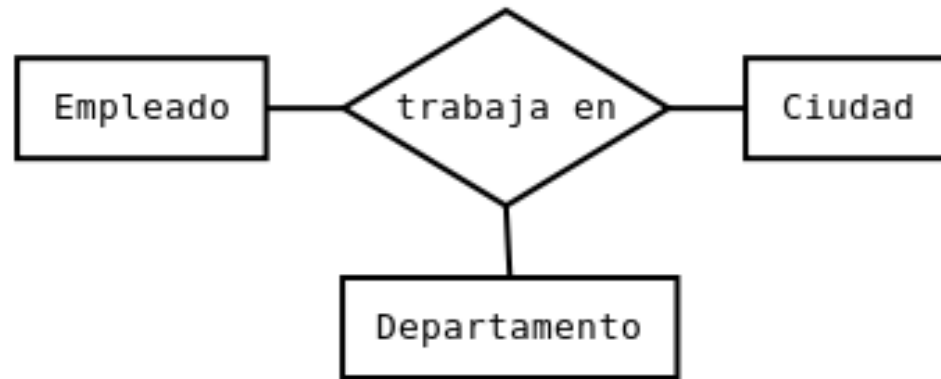


- Crow's feet



Relaciones ternarias

- Involucran 3 entidades
- Ejemplo:
 - “Un empleado trabaja en un departamento de la empresa, y desarrolla sus actividades en una ciudad”



- (La generalización a N-arias es trivial)



Ejercicio

- ¿Cuál es la diferencia entre estos dos modelos?
- ¿Cuál está correcto? ¿Existe algún error?



Ejercicio

- Ejercicio: completar el diagrama de E/R del ejercicio anterior:
- *“Se desea crear una BD de alumnos de la UACH. De cada uno se conoce su nombre, su RUT, la carrera que estudia, y la fecha de nacimiento. Cada carrera posee un nombre, una duración, y un director, y está asociada a un instituto. Las carreras están compuestas de ramos, los cuales tienen un nombre, una sigla y una cantidad de créditos. Los ramos son dictados por al menos un profesor, los cuales están identificados por su nombre y RUT. Un profesor puede dictar hasta 4 ramos por semestre”*



Resumen

- Entidades Fuertes siempre tienen clave propia, débiles no necesariamente
- Las relaciones poseen cardinalidades, 1:1, 1:N, N:M
- Las relaciones pueden requerir participación
 - Opcional u obligatoria
- Chen permite especificar número de participantes en relaciones
 - (minimo,maximo)
- Crow's foot representa cardinalidad y participación con extremos de líneas
- Las relaciones pueden involucrar diferente número de entidades:
 - Unarias, binarias, ternarias o N-arias
- Ternarias y N-arias se pueden expresar como conjunto de binarias



Bonus

- No se pierda la próxima clase... cita con Hannibal Lecter



Diseño Conceptual

Parte 1



Diseño conceptual

- La forma de formalizar el diseño es mediante un **modelo**
- Diversos autores lo llaman al modelo de diferentes formas:
 - Modelo semántico
 - Modelo entidad/relación
 - Modelo objetual
 - Modelo de entidades
 - Modelo de datos
- “conceptual” y “semántico” son los nombres más genéricos
- Un modelo **semántico** es aquel que busca capturar el **sentido** de un sistema, no sólo las **partes** sino que también cómo éstas se **conectan**.
- En este curso, nos enfocaremos en los modelos Entidad/Relación como herramienta de diseño conceptual



Hay Buenas y Malas noticias...

- **Malas:** Los peores días del liceo están de vuelta...



- **Buenas:** ... pero de manera somera y práctica.
- Sólo hay que saber reconocer:
 - Sujeto, Predicado
 - Sustantivos, verbos
- ¿Por qué?
 - Para construir modelos a partir de textos de requerimientos



Conceptos Básicos

- Entidad:
 - Objeto reconocible e identificable
 - e.g. mi auto, asignatura Base de Datos, Graciela Asencio, etc.
- Conjuntos de entidades:
 - Agrupación de entidades por categorías
 - e.g. Automóviles, asignaturas, Alumnos, etc.
- Atributos:
 - Aspectos conocidos de las entidades
 - e.g. Patente, color, créditos, nombre, dirección, etc.
- Relaciones:
 - Relaciones entre entidades
 - e.g. Alumno **curso** una asignatura, alumno **posee** un auto, etc.



Ejemplo

- Qué entidades, conjuntos de entidades, atributos y relaciones vemos en esta imagen?

Conjuntos de Entidades: Cosas similares agrupadas

Relaciones: conexión entre cosas

Entidades: Cosas



Atributos: características de las cosas



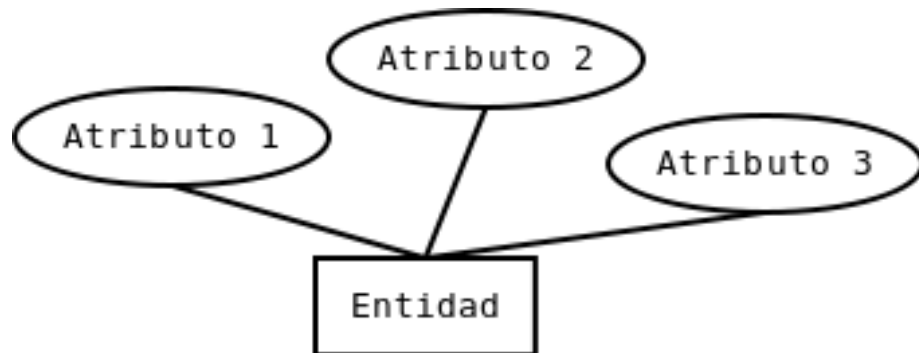
Notaciones

- Se han desarrollado distintas notaciones para expresar diagramas ER
 - Unas más concisas, otras más expresivas
 - Ponen énfasis en diferentes aspectos
- En este curso, veremos dos de ellas:
 - **Chen**: Propuesta por el inventor de los diagramas ER. Utilizada comúnmente en textos sobre BDs.
 - **Crow's Foot**: Usada ampliamente en herramientas Software de diagramas y herramientas CASE. Más concisa (e incompleta) que Chen, y más cercana al modelo Relacional.
- Son las notaciones más ampliamente utilizadas hoy en día
- **Atención**: Ambas notaciones varían ligeramente dependiendo del libro o herramientas de diseño que se utilice.



Entidades y atributos: Notación

Chen:



Crow's Foot:

Entidad
Atributo 1
Atributo 2
Atributo 3

- Los conjuntos de entidades se nombran con un ***sustantivo común singular***:
 - Buenos ejemplos: “Alumno”, “Curso”, “Contrato”
 - Malos ejemplos: “Lista de alumnos”, “alumnos”, etc.)



Entidades y atributos

- Para reconocer entidades, nos fijamos en los sustantivos
- Ejemplo:
 - “Se desea crear una BD de alumnos de la UACH. De cada uno se conoce su nombre, su RUT, la carrera que estudia, y la fecha de nacimiento”
- ¿Cuáles son las entidades y los atributos?

Entidades: objetos

Atributos: sus características



Clave

- Nos interesa **individualizar** las entidades (“Ivo Cuq”, “Daniela Améstica”, etc.) al interior del conjunto de entidades (“Alumnos”)
- Podríamos identificar cada entidad mediante la suma de todos sus atributos

ENTIDAD 1: “Ivo Cuq, 2? años, RUT ??.???.???-?, Calle XX número YY, ...”

ENTIDAD 2: “Daniela Améstica, 2? años, RUT ??.???.???-?, Calle XX número YY, ...”

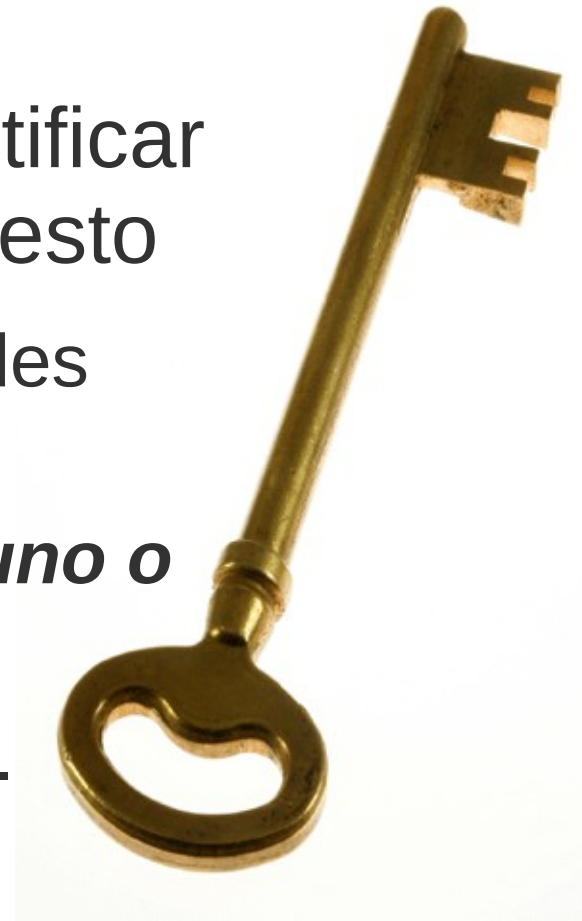
...

- ...sin embargo esto es muy engorroso!
- Una mejor manera es identificar la mínima información posible que no permita confundir registros
- A este atributo (o conjunto de atributos) se le denomina **clave**



Clave (Key)

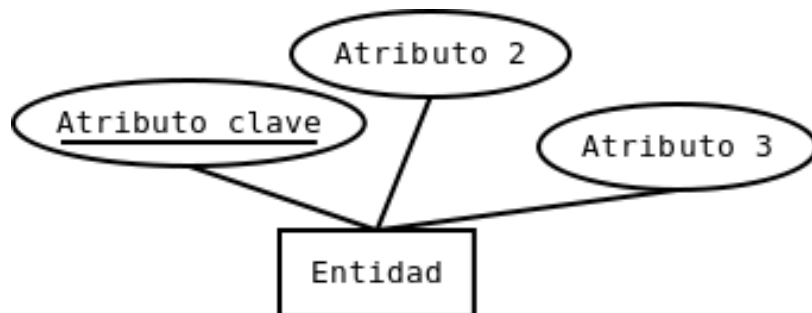
- Un atributo clave es el conjunto de atributos **mínimo** que permite identificar inequívocamente una entidad del resto
 - (casi) **todos** los conjuntos de entidades deben tener una clave
 - Una clave puede estar formada por **uno o varios** atributos
 - Pueden haber varias claves posibles. Si es así se elige **una** y el resto es tratado como atributos normales



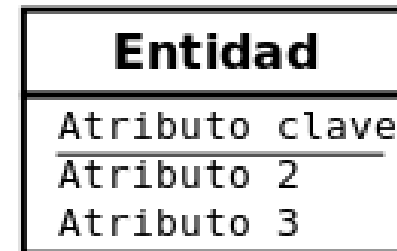
Atributo clave

- Notación: la clave se subraya

Chen



Crow's Foot



Atributo clave

- Ejemplos:
 - Clave compuesta (de varios atributos):
 - [Nombre de canción, Autor]
 - [Película, Director]
 - [Nombre de Polola, año]
 - Clave simple (sólo un atributo):
 - RUT de persona
 - Patente de automóvil
 - Sufijo de país en Internet (.cl, .fr, .uk, ...)
 - ID ad-hoc (no existe naturalmente, pero se crea para la BD)



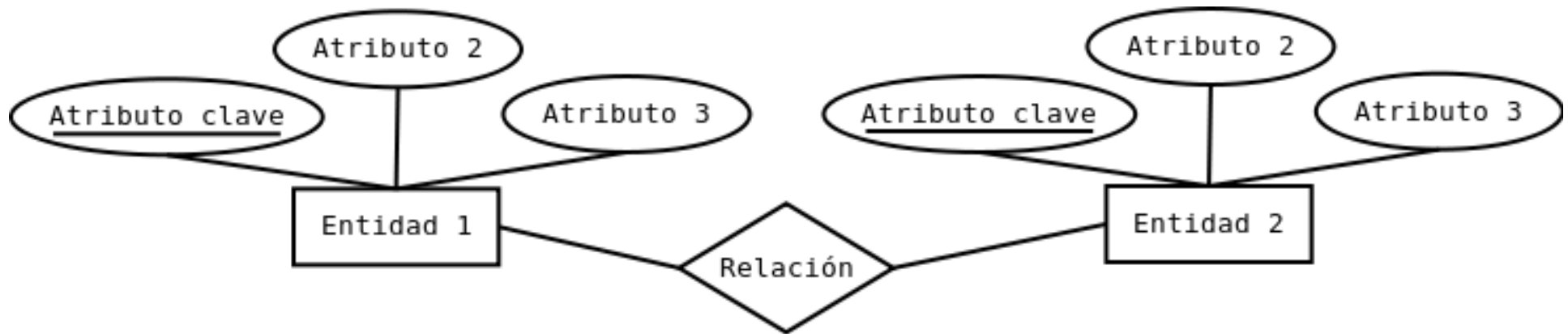
Entidades y atributos

- Otro Ejemplo:
 - *“Se desea crear un registro de mascotas. Cada mascota tiene un nombre, pertenece a una especie y posee un amo humano. El humano posee un nombre, un RUT y una dirección”*
- ¿Cuáles son las entidades y los atributos? ¿cuáles las claves?



Relaciones: Notación

- Chen



- Crow's Foot



Relaciones

- Las entidades usualmente se **relacionan** entre ellas
- En un texto, éstas se expresan normalmente como verbos
- Se nombran como verbos (“posee”, “contiene”), eventualmente con alguna preposición (“pertenece a”, “reporta desde”, etc.)
- Ejemplo:
 - *“Se desea crear un registro de mascotas. Cada mascota tiene un nombre, pertenece a una especie y posee un amo humano. El humano posee un nombre, un RUT y una dirección”*
- ¿Cuáles son las relaciones?



Ejercicio

- Diagramas ER (Chen y Crow's Foot) para lo identificado previamente



Ejercicio

- Ejercicio: crear diagramas de E/R correspondiente al siguiente texto:
 - *“Se desea crear una BD de alumnos de la UACH. De cada uno se conoce su nombre, su RUT, la carrera que estudia, y la fecha de nacimiento. Cada carrera posee un nombre, una duración, y un director, y está asociada a un instituto. Las carreras están compuestas de ramos, los cuales tienen un nombre, una sigla y una cantidad de créditos. Los ramos son dictados por al menos un profesor, los cuales están identificados por su nombre y RUT. Un profesor puede dictar hasta 4 ramos por semestre”*



Resumen

- Modelo Conceptual puede hacerse de distintas maneras
- Una de ellas es mediante un Diagrama ER
- Distintas Notaciones (Chen, Crow's Foot) y variantes de ellas
- Elementos principales:
 - Entidades: “cosas”
 - Conjuntos de entidades: “agupaciones de cosas”
 - Atributos: “lo que se sabe de las cosas”
 - Relaciones: “lo que une a las cosas”
- Hay aspectos que no se pueden expresar sólo con estos elementos. Los veremos la próxima clase.



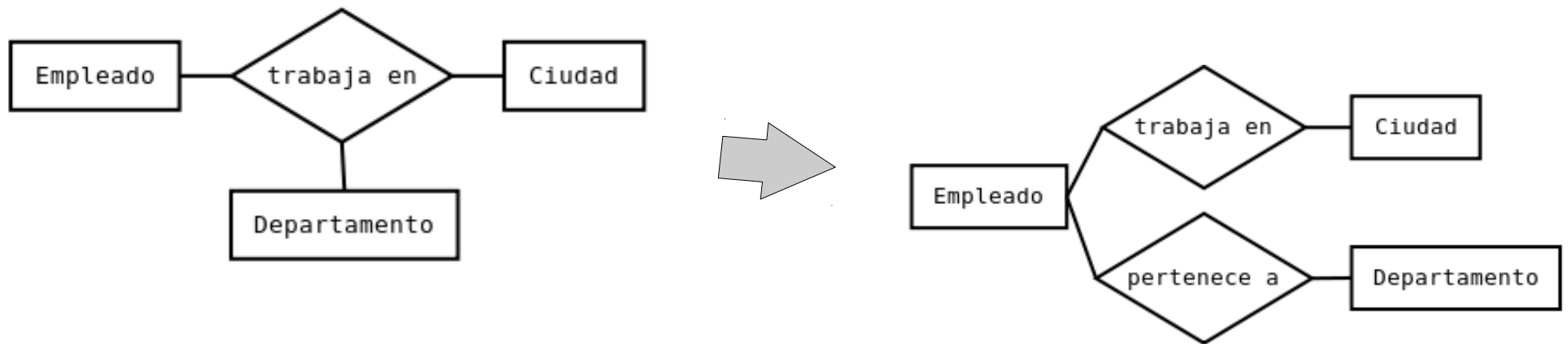
Diseño Conceptual

Parte 3



Relaciones ternarias a binarias

- Las relaciones ternarias o superiores son difíciles de manejar
 - ¿A qué participante se refieren las cardinalidades?
 - No se pueden implementar en BDs relacionales
- Sin embargo, pueden expresarse como varias relaciones binarias
 - Es necesario transformarlas



BD Policial

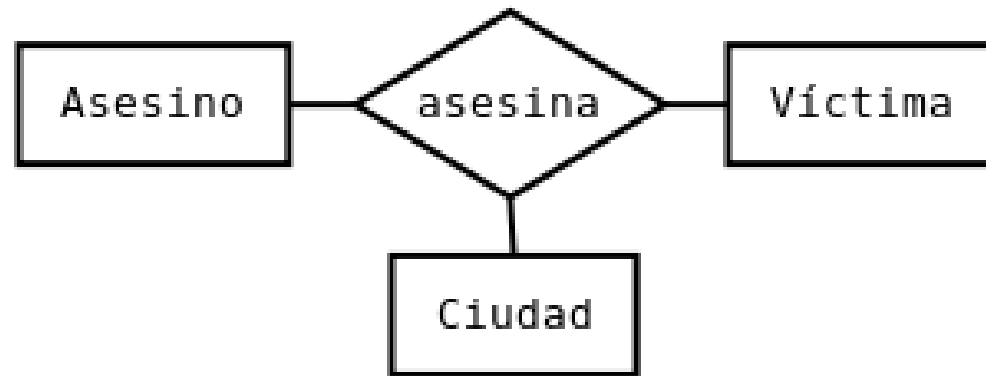


- La policía necesita una BD para almacenar información sobre asesinos en serie.
 - Existe relación ternaria que asocia al asesino con su víctima y el lugar de los hechos
 - Un asesino ha cometido uno o más asesinatos
 - Cada asesinato es perpetrado por un asesino y ha ocurrido en una ciudad
 - En una ciudad han ocurrido diversos asesinatos, perpetrados eventualmente por distintos asesinos



BD Policial

- Se cuenta con la siguiente relación ternaria:

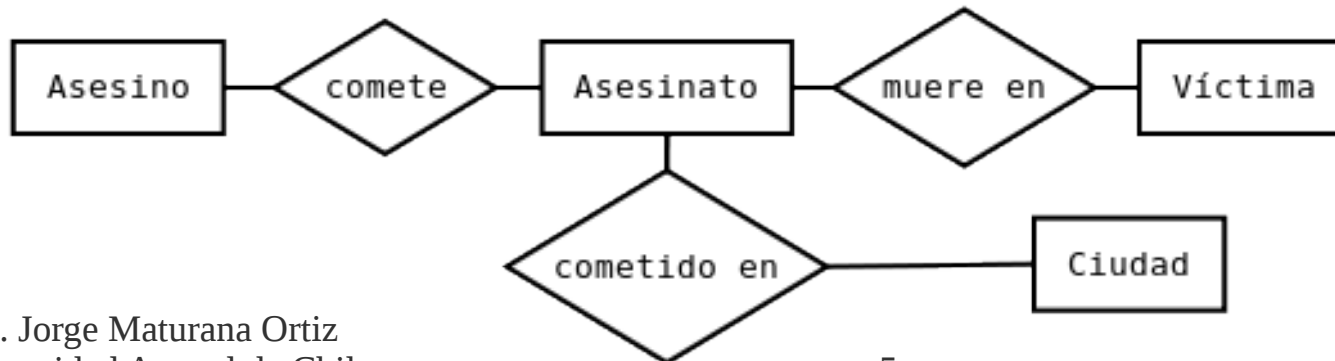
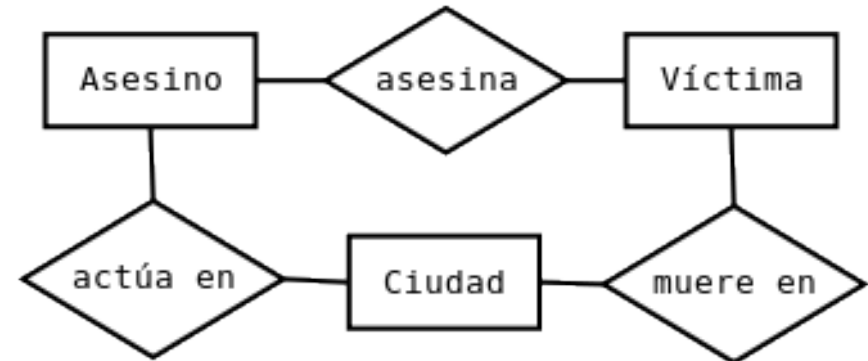
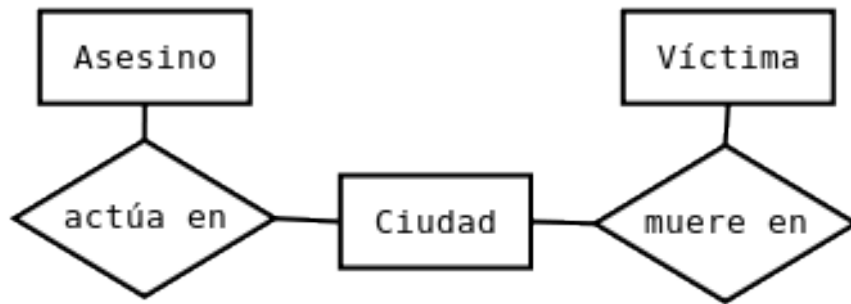
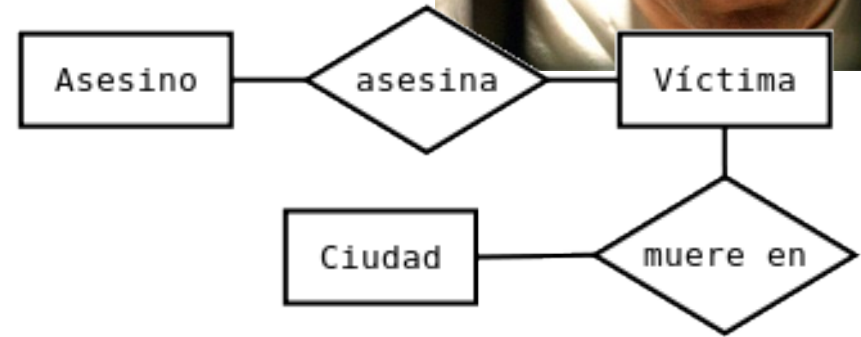
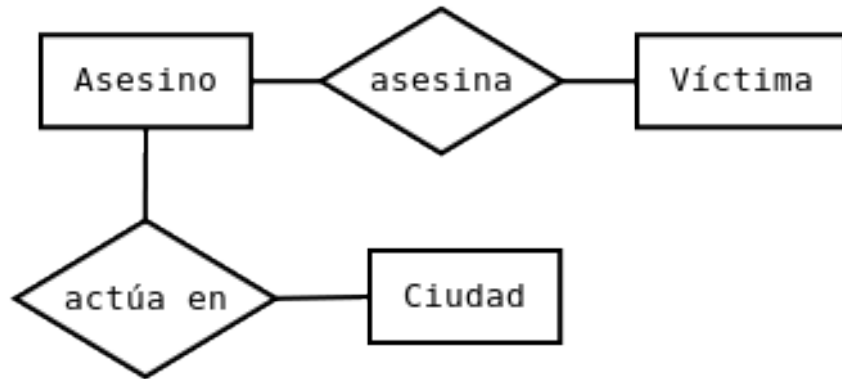


- Se le solicita transformarla a su equivalente utilizando sólo relaciones binarias
 - Sin perder la capacidad a responder preguntas
 - ¿Qué opciones existen?

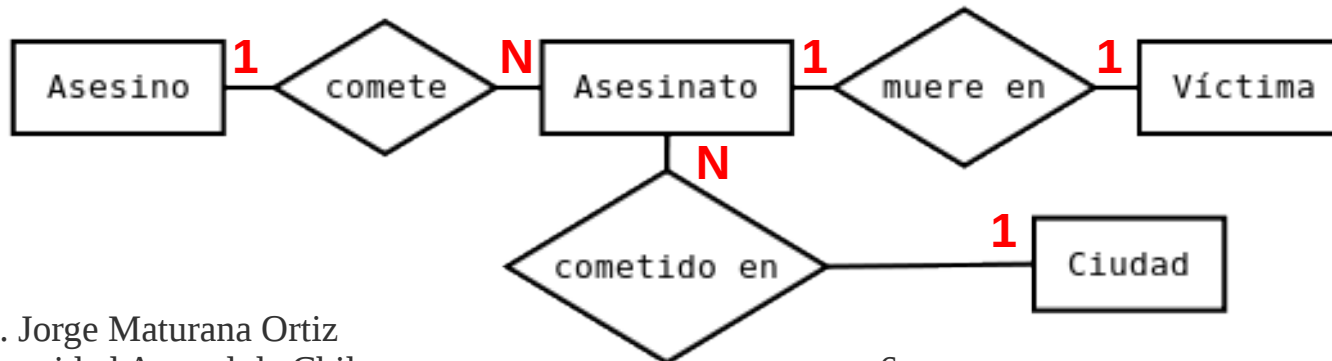
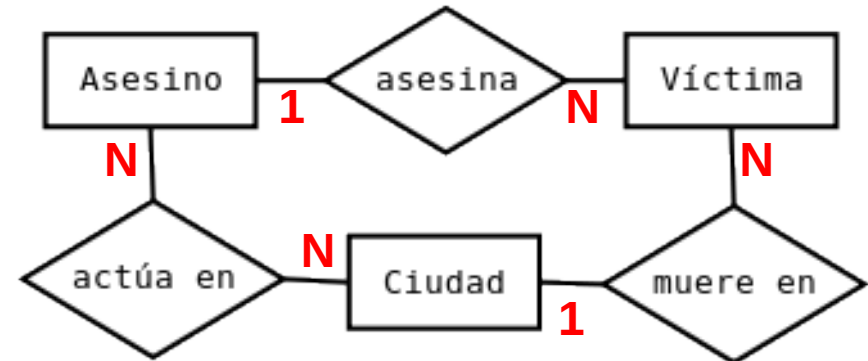
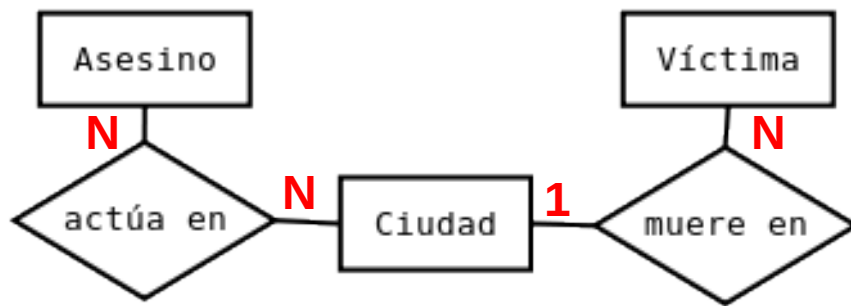
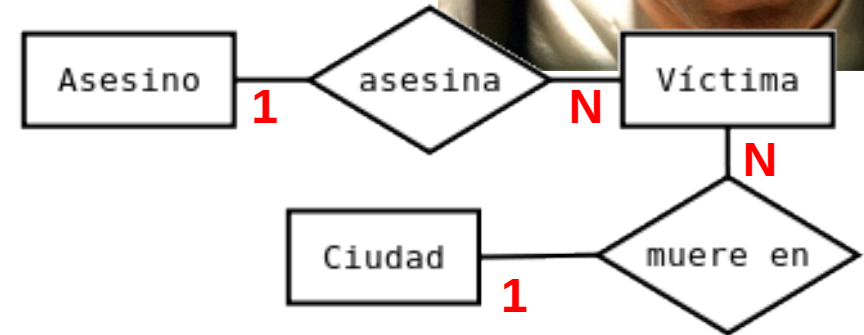
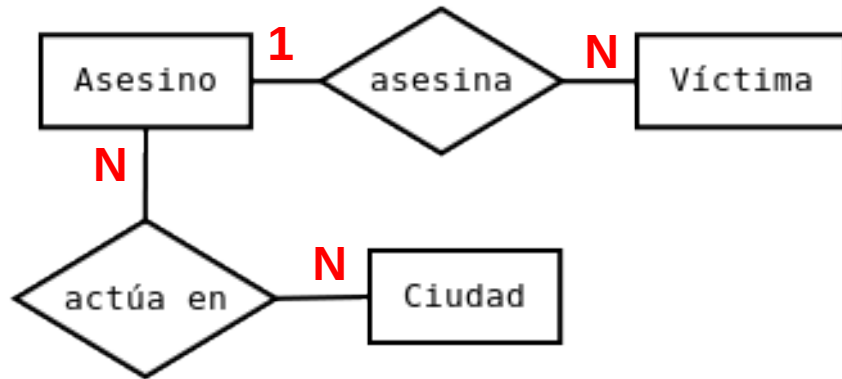


Varias opciones...

Are you confused?



Varias opciones...



Ternarias a binarias

- En general, crear una nueva entidad para reemplazar la relación funciona bien
 - Pero no es la única opción
- Evitar relaciones que generen preguntas en sentido $1 \rightarrow N$
 - Pues son imposibles de responder



Especialización / Generalización

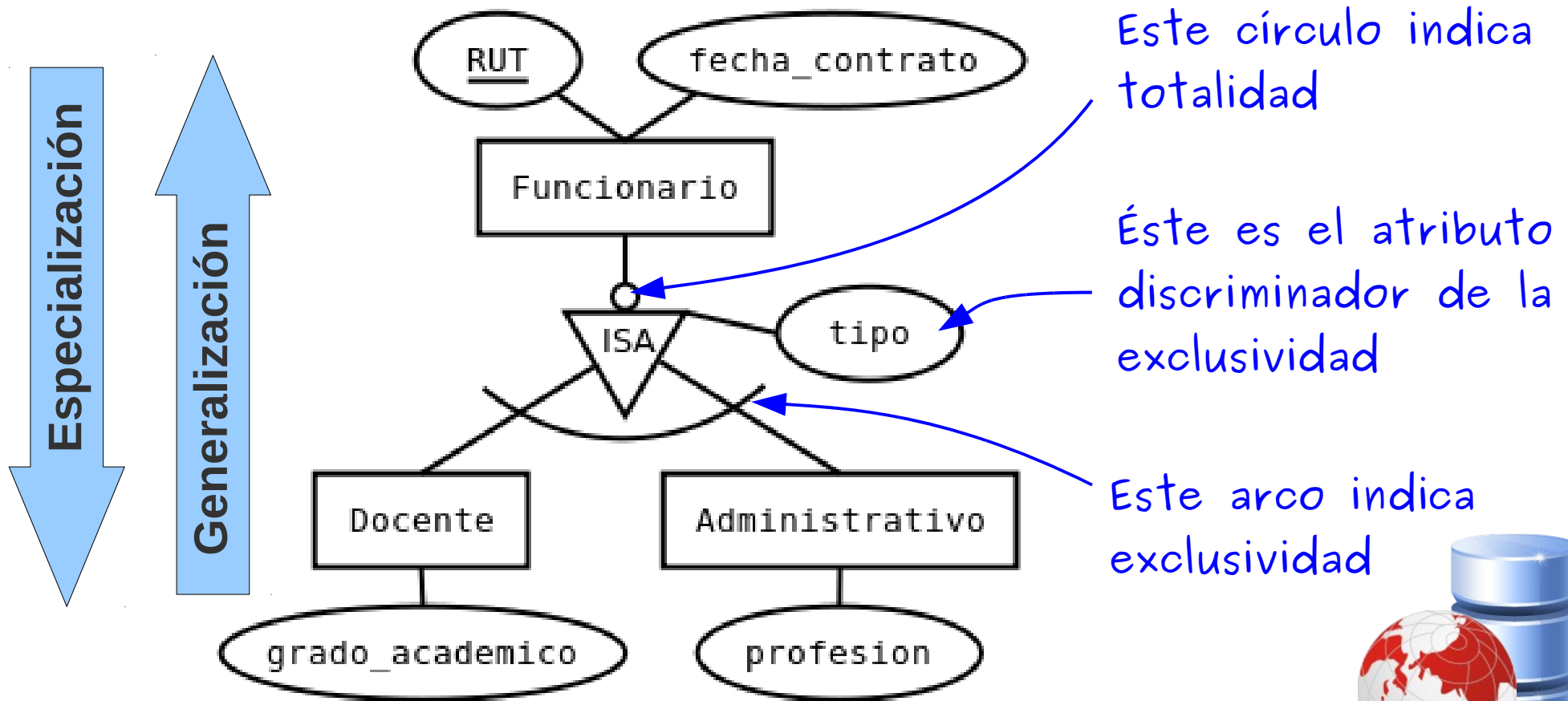
“Los funcionarios de la universidad pueden ser docentes o administrativos. De los primeros interesa saber su grado académico, y de los segundos su profesión”

- Docentes y administrativos son ambos funcionarios, ¿cómo reflejar este hecho?
- Se puede crear una jerarquía de entidades, agrupándolas en sub o super tipos
- Similar a herencia en diagrama de clases
- Esta relación se denomina ISA (“is a ____”)



Especialización / Generalización

- La entidad genérica mantiene los atributos comunes
- Las entidades especializadas mantienen los atributos que le son propios
- La entidad genérica mantiene la clave, que es “heredada” a las especializadas



Tipos de especialización

- Una especialización puede ser:
 - **Total**: todos los miembros de la superclase deben pertenecer a alguna subclase
 - **Parcial**: puede existir algún miembro de la superclase que no pertenezca a ninguna subclase
 - **Exclusiva**: las subclases son disjuntas
 - **Solapada**: una entidad puede pertenecer a más de una subclase (“superpuesta”)



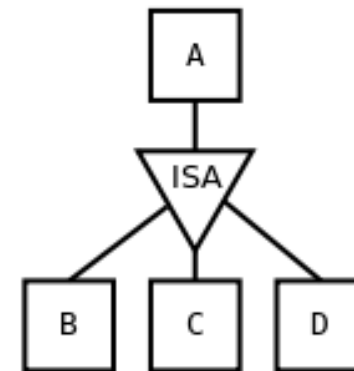
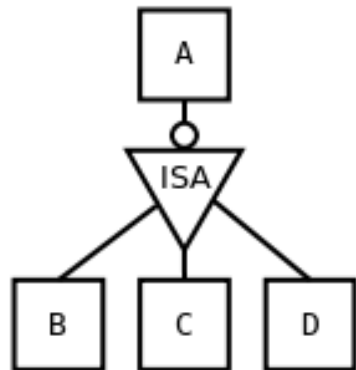
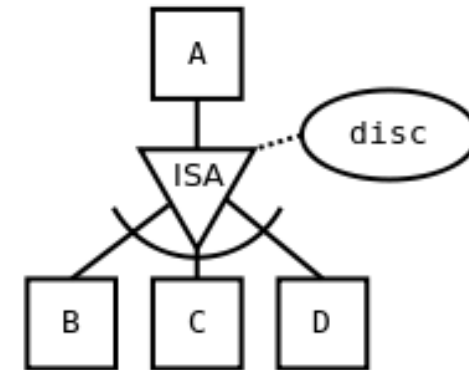
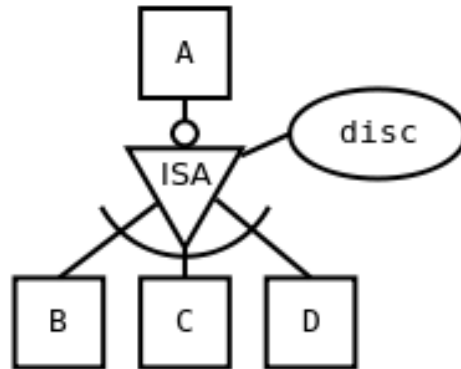
Notación de especializaciones

Exclusiva

Solapada

Total

Parcial



¿Cuándo Especializar?

- Especializar complejiza el modelo. ¿Cuándo vale la pena hacerlo?
 - (Cuando distintos tipos de entidades tienen atributos diferentes OR
 - Cuando distintos tipos de entidades se relacionan de manera diferente)
 - AND
 - (Cuando existen atributos en común para todas las entidades OR
 - Cuando todas las entidades tienen relaciones en común)
- En resumen, cuando se necesita tratarlas, en el mismo modelo, como conjunto para algunas cosas e individualmente para otras.



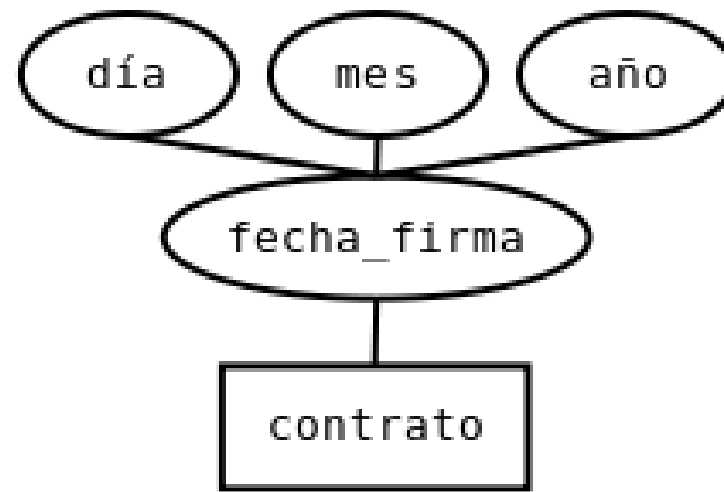
Tipos de atributos

- Simple
 - Cuando están compuestos de un dato simple, que no interesa ser subdividido
 - Edad de una persona en años
 - Estado de una orden de compra (cursada, en espera, terminada)
- Compuesto
 - Cuando interesa subdividirlo en otros atributos
 - RUT (número y dígito verificador)
 - Fecha (desglose en mes/día/año)
 - Dirección (Calle, número, población, ciudad)
 - N.B. No confundir con **clave compuesta** (Ejemplo: RUT)
- Casos vagos: “INFO-261”, ¿simple o compuesto?
 - Respuesta. ¿se necesitan las partes individualmente?



Atributos compuestos

- Notación:
 - Agrupar los atributos facilita la lectura del diagrama



- En este ejemplo no hay 4 atributos, sino 3
 - (fecha_firma es la agrupación del resto)

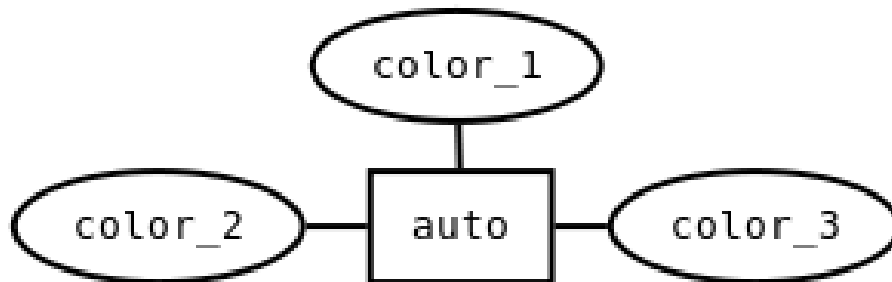


Atributos multivaluados

- Cuando pueden poseer más de un valor
 - Notación.

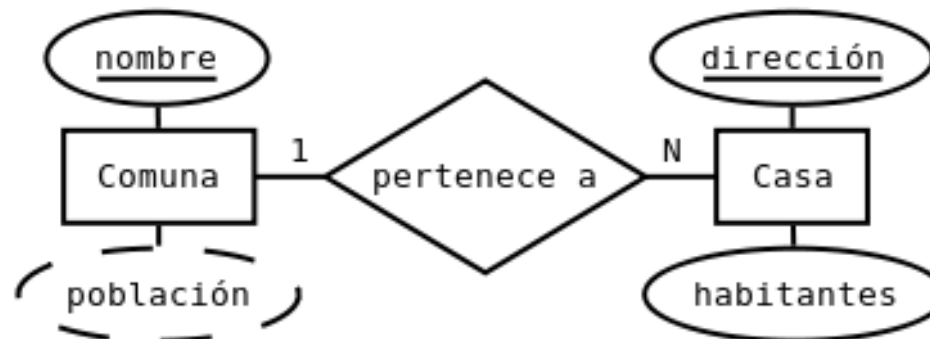
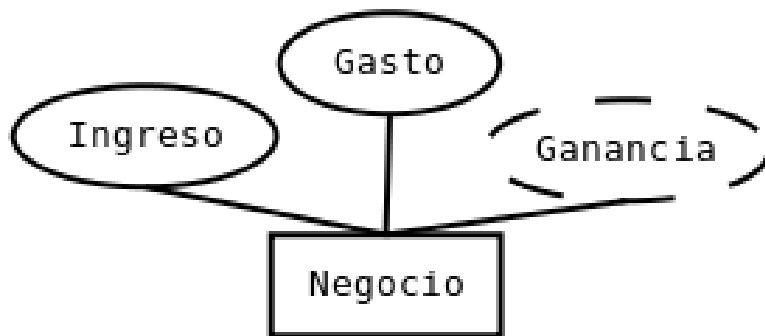


- Alternativas:
 - ¿Pros?, ¿Contras?



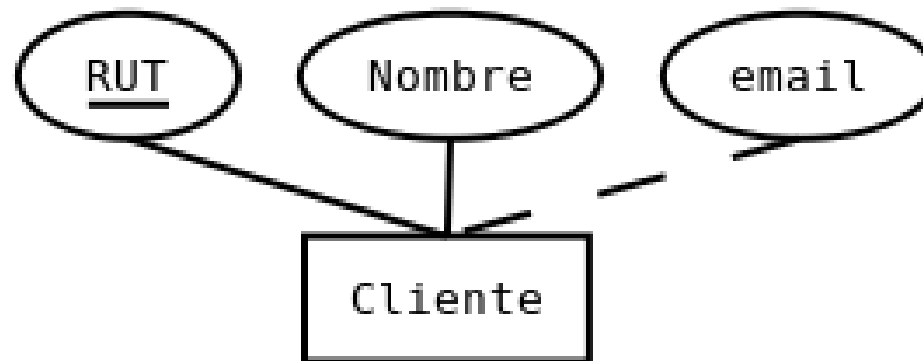
Atributos derivados

- Cuando pueden ser obtenidos a partir de otros atributos
- Ya sea de la misma entidad, de otra(s), y/o de algún dato del sistema



Atributos opcionales

- A veces, es necesario explicitar que un atributo es prescindible
 - e.g., al llenar un formulario una persona puede no tener email
- Notación:



- N.B. ¡Los atributos que pertenecen a la clave no pueden ser opcionales!
 - (Pues de ellos depende la identificación de la entidad)



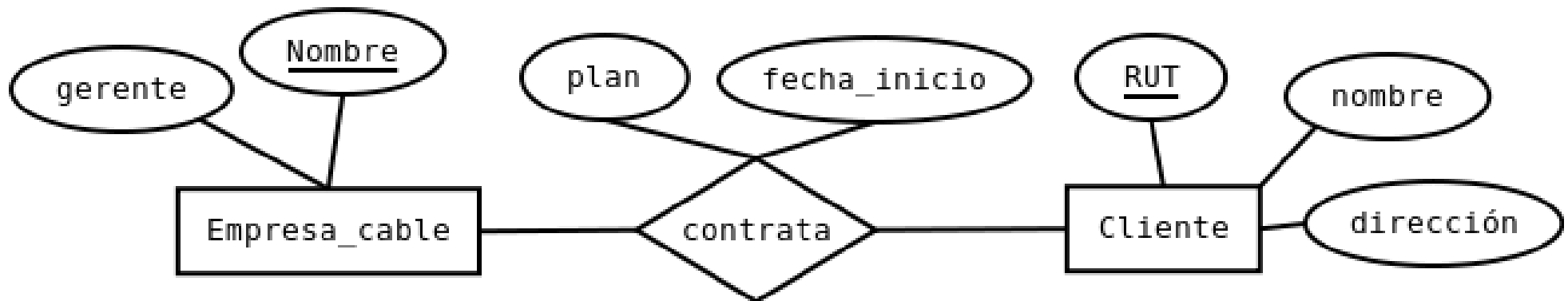
Valor NULL

- Los atributos opcionales pueden no tener valor
- Un atributo sin valor es denominado nulo (NULL)
- Un valor NULL es:
 - Un valor que ***no se conoce***
 - Un valor que ***no existe***
 - En suma, es un ***no-valor***
- N.B. Un valor ***por defecto*** no es necesariamente nulo
 - Pero un valor nulo puede ser un valor por defecto
 - A propósito: usar valores por defecto con precaución
 - Pueden generar información inexacta



Relaciones con atributos

- Una relación puede tener atributos
 - Son producto de la relación, y no propios de las entidades que participan
- Ejemplo:
 - “Cuando un cliente contrata TV cable, interesa conocer qué plan contrató y cual es la fecha de inicio del contrato”

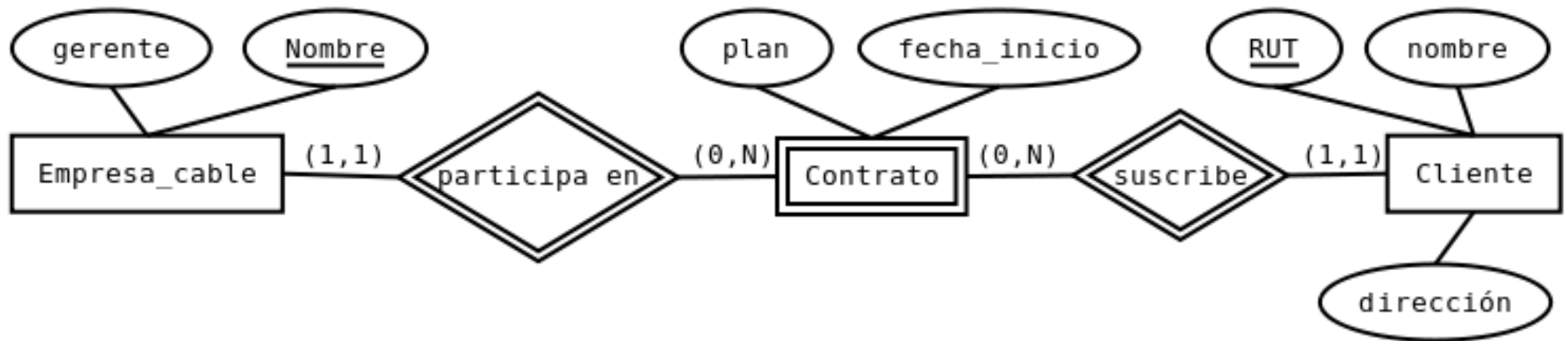


- En Crow's feet no existe notación equivalente



Alternativa a relación con atributos

- Transformar una relación con atributos en una entidad débil
 - El verbo se nominaliza (“contrata” → “contrato”)



- N.B. Participación obligatoria (1,1) desde entidad Contrato a otros participantes: se obliga que **ambos** (empresa y cliente) participen



Resumen

- Paso de relaciones ternarias a conjunto de binarias
- Entidades fuertes y débiles
- Especialización/Generalización
 - total o parcial
 - exclusiva o solapada
- Tipos de atributos
 - simples o compuestos
 - multivaluados
 - derivados
 - opcionales
- Relaciones con atributos
- Fin de modelo conceptual



Bonus!

- Próxima clase: **clase práctica**
 - (Se controlará asistencia)
- Repasar la materia vista hasta hoy



Diseño Lógico

Parte 1



Diseño lógico

- Una vez terminado el diseño conceptual, éste se puede “traducir” a un diseño más cercano al modelo (relacional, objetual, etc.) de la DBMS a utilizar
- En nuestro caso, nos interesará traducirlo al modelo relacional, por lo que nuestro diseño lógico será un ***modelo relacional***
- Existen dos tareas principales:
 - Traducción de modelo ER a Relacional
 - Normalización de modelo Relacional



Componentes del modelo relacional

- (Desarrollado por Edgard Codd en 1970)
- El elemento central es la **Tabla**
 - aka **relación** (de ahí el nombre), no confundir con las del modelo ER
- Una tabla se compone de:
 - Un nombre
 - Un conjunto de atributos (sólo **un** valor, **orden irrelevante**)
 - Tuplas (una instancia de elementos de la tabla, **orden irrelevante**)
- Si pensamos “objetualmente” esto equivale a:
 - Tabla → Clase
 - Tupla → Objeto



Ejemplo de tabla

nombre

atributos

CUSTOMER

ID	First Name	Last Name	Phone	E-mail
1	Joe	Smith	555-1234	jsmith@gmail.com
2	Jack	Williams	555-5668	jwilliams@gmail.com
3	Jill	Davis	555-5432	NULL
4	James	Miller	NULL	jmiller@yahoo.com
5	Jamie	Wilson	555-6527	NULL
6	Justin	Taylor	555-8247	jtaylor@aol.com

una tupla

(descripción por extensión)

(descripción por intención)

CUSTOMER(ID, First_Name, Last_Name, Phone, E-mail*)



Integridad

- Nuestro principal interés es mantener información que sea consistente con la realidad
- Esto requiere que los datos mantengan su integridad
- Ejemplos:
 - Que no existan dos personas con el mismo RUT
 - No almacenar edades negativas
 - Formatos claros: ¿ “3/5/2010”: 3 de mayo o 5 de marzo?
 - Evitar automóviles sin dueño
- Para mantener esta integridad se definen restricciones, que son reglas que el DBMS debe hacer cumplir



Restricciones

- Almacenar los datos en tablas no basta para mantener la consistencia de la información.
- Adicionalmente, se definen ciertas restricciones sobre ellas, a saber:
 - Restricciones de clave primaria
 - Restricciones de clave foránea
 - Reglas de modificación/eliminación
 - Restricciones de dominio, rango y formato
 - Restricciones de unicidad
 - Restricciones de obligatoriedad
- Veremos cada una a continuación



Clave Primaria

- Equivalente a la **clave** en el modelo ER
 - (Conjunto mínimo de atributos que permiten identificar una entidad)
- Pueden haber distintas posibilidades de clave, por ejemplo:
 - ALUMNO(RUT,rol, nombre, e-mail)
 - PERSONA(nombre,fecha_nacimiento,dirección)
 - Éstas son llamadas claves candidatas
- De entre ellas, se escoge una y se la designa como **clave primaria**

Primary key

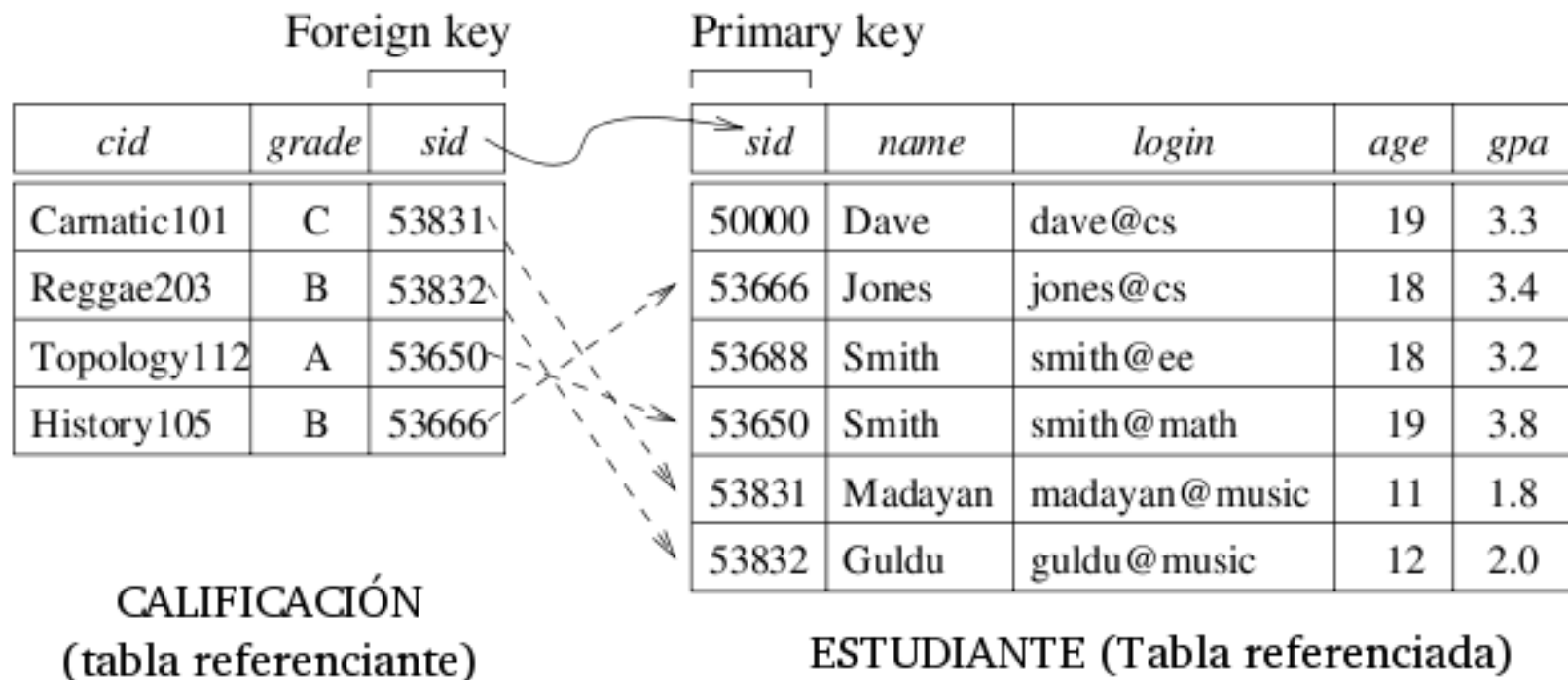
<i>sid</i>	<i>name</i>	<i>login</i>	<i>age</i>	<i>gpa</i>
50000	Dave	dave@cs	19	3.3
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	18	3.2
53650	Smith	smith@math	19	3.8
53831	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0

Students

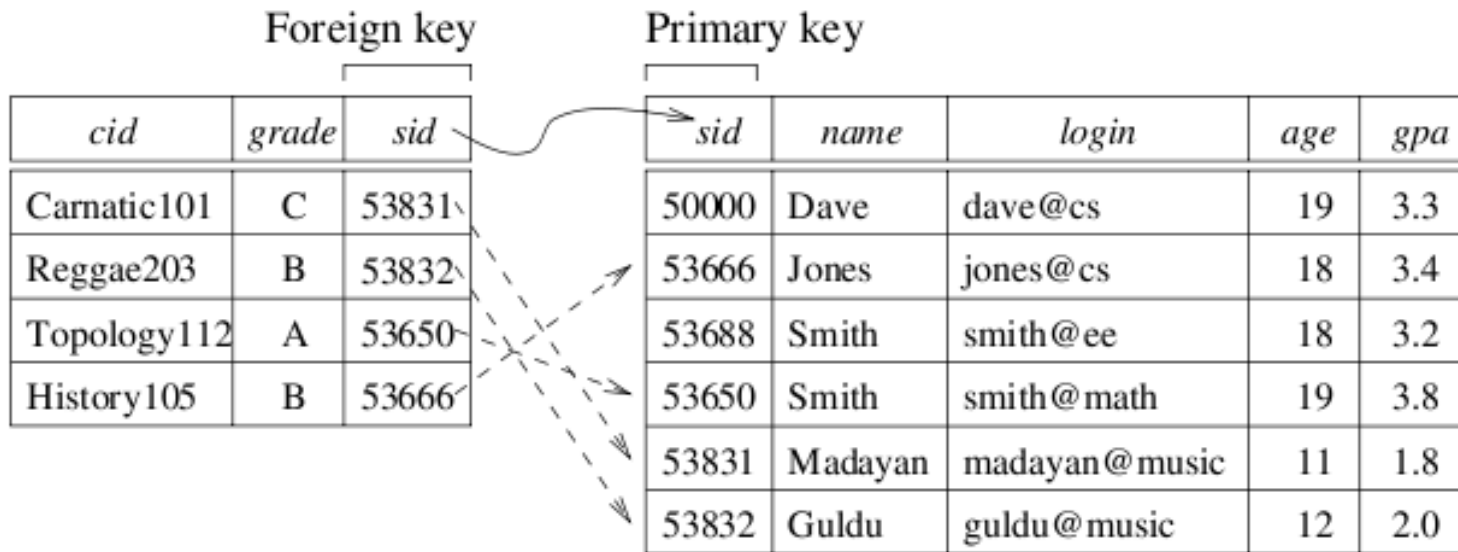


Clave foránea

- Enlaza una tabla con otra, incluyendo la clave primaria de la otra tabla como un atributo propio → **Integridad referencial**
- Ejemplo: tablas de alumnos y calificaciones:



Integridad Referencial

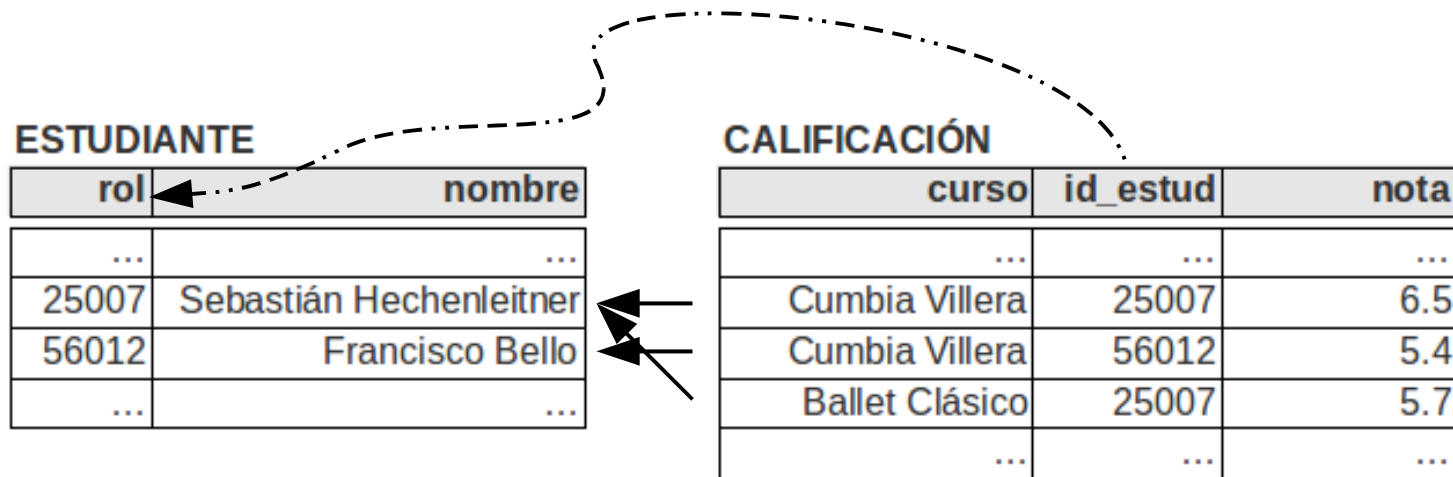


- Todas las tuplas en la tabla CALIFICACIÓN deben tener un *sid* que exista en la tabla ESTUDIANTE, siempre.
- No es necesario que el atributo común ("*sid*" en el ejemplo) se llame igual en ambas tablas
 - ¿Que pasa si deseo ingresar la calificación (BD,A,12345)?
 - ¿Qué pasa si deseo ingresar la calificación (Reggae203,A,53832)?
 - ¿Qué pasa si deseo eliminar el estudiante (53666,Jones,jones@cs,18,3.4)?
 - ¿Qué pasa si deseo actualizar la edad de Jones?
 - ¿Qué pasa si deseo actualizar el *sid* de Dave o de Guldu?



Reglas de modificación/eliminación

- Ante eliminaciones o modificaciones que atenten contra la integridad referencial existen distintas políticas:
 - Ejemplo: se desea **eliminar** a Hechenleitner de la tabla estudiante

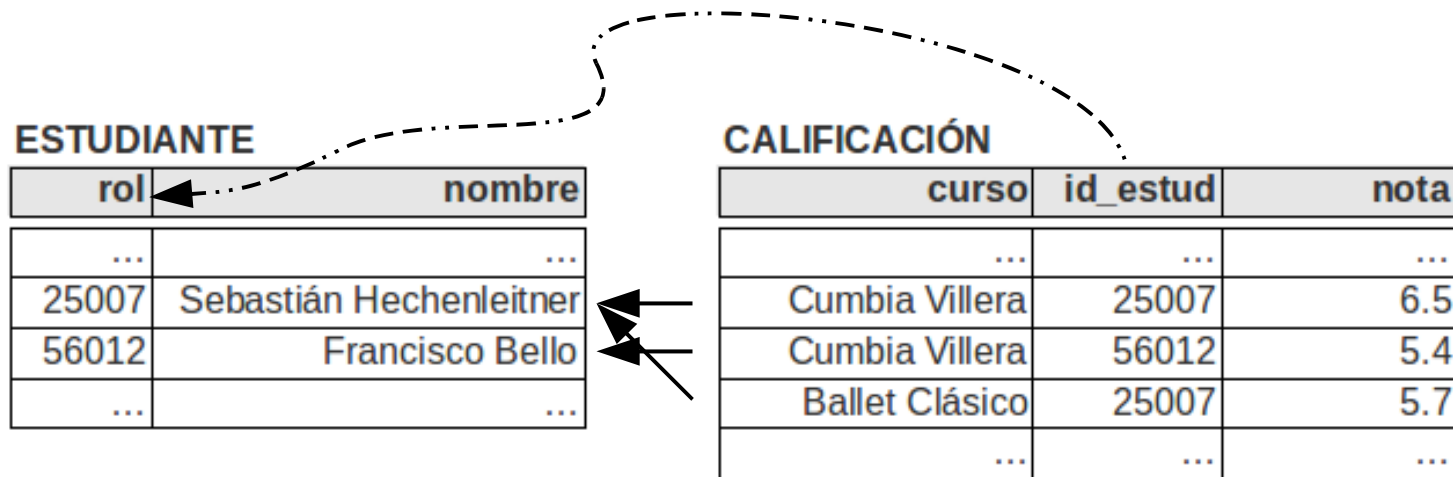


- Opciones:
 - **Impedir** la eliminación, pues hay calificaciones que quedarían “sueltas”
 - Eliminar **“en cascada”** todas las calificaciones de Hechenleitner
 - Asignar las calificaciones de Hechenleitner a otro estudiante (valor por defecto)
 - Asignar a los valores de **id_estud** asociados a Hechenleitner el **valor NULL**



Reglas de modificación/eliminación

- Ante eliminaciones o modificaciones que atenten contra la integridad referencial existen distintas políticas:
 - Ejemplo: se desea **modificar** el rol de Hechenleitner de la tabla estudiante



- Opciones:
 - **Impedir** la modificación, pues hay calificaciones que quedarían “seltas”
 - Modificar “**en cascada**” todas sus calificaciones, asignándoles el nuevo rol
 - Asignar las calificaciones de Hechenleitner a otro estudiante (valor por defecto)
 - Asignar a los valores de id_estud asociados a Hechenleitner el **valor NULL**



Restricciones de dominio, rango y formato

- Estas restricciones se aplican sobre el valor de un atributo
- **Dominio:** El valor del atributo a un conjunto de valores posibles.
 - Estado civil: {soltero, casado, divorciado, viudo}
- **Rango:** El valor debe pertenecer a un rango
 - Edad > 0
- **Formato:** El valor debe ajustarse a un formato
 - dd/mm/aa, hh:mm, hh:mm:ss



Restricción de unicidad

- Cuando se necesita que el valor de un atributo no aparezca en más de una tupla
- Se marca el atributo como UNIQUE
- El caso más común es la clave primaria
- Pero pueden haber atributos únicos que no sean clave primaria:
 - PERSONA(RUT, Nombre, **RUT_conyuge**)
(El RUT de un cónyuge sólo puede aparecer una vez en **toda** la tabla, pues el cónyuge sólo puede casarse una vez)



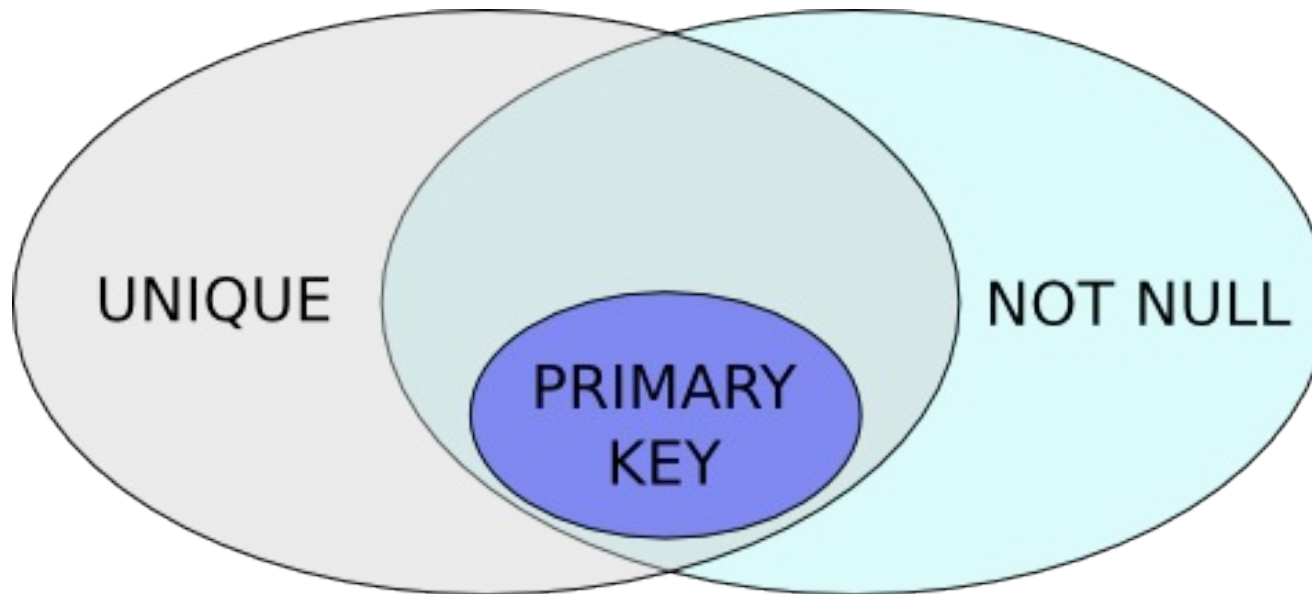
Restricción de obligatoriedad

- Algunos atributos son obligatorios, por lo que son marcados como NOT NULL
- e.g., en un call center, se tiene la tabla cliente:
 - CLIENTE (RUT, Nombre, **teléfono**)
(¿Es admisible que en una empresa cuyo contacto con el cliente es vía telefónica no se cuente con el teléfono de los clientes?)
- Usualmente, en vez de marcar como NOT NULL, se marca como NULLABLE (i.e., lo contrario)



PK \Rightarrow UNIQUE y NOT NULL

- **Toda** Clave Primaria es UNIQUE y NOT NULL
- Pero **no todos** los atributos UNIQUE y NOT NULL son Claves Primarias



Notación

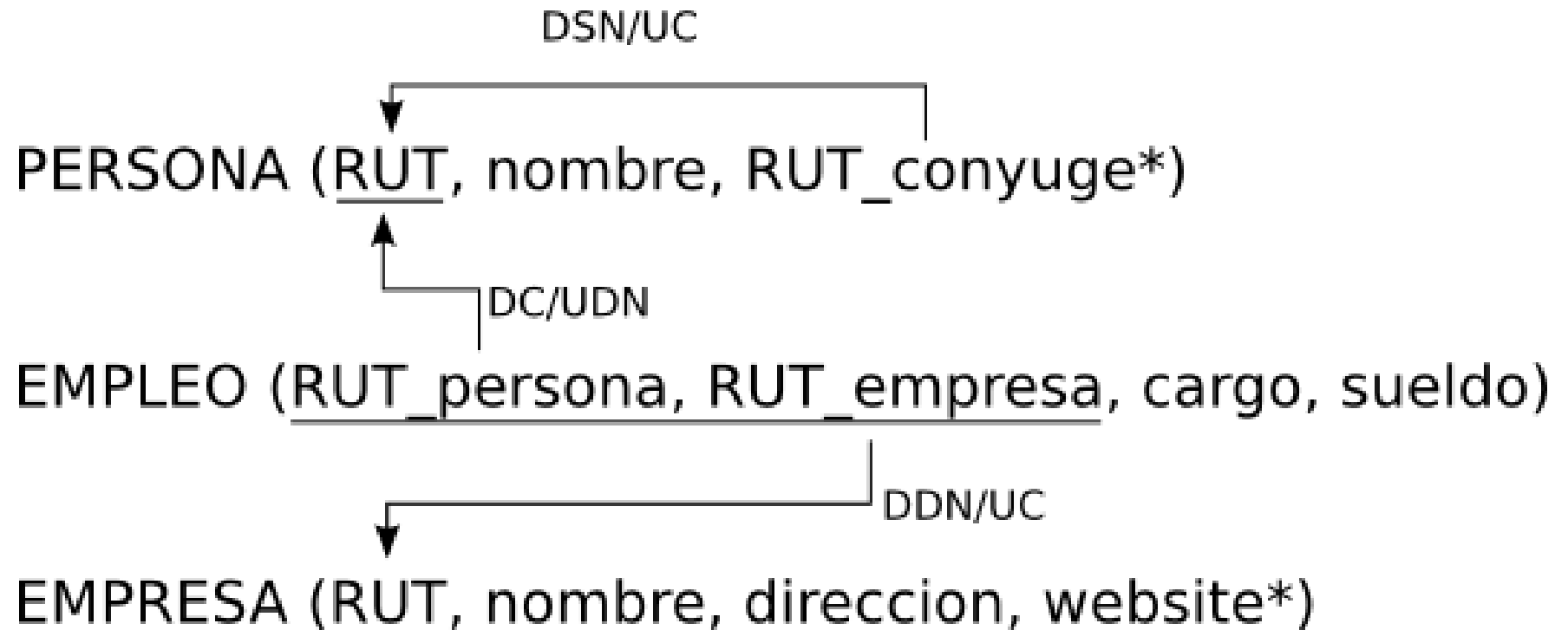
- Es menos estándar que la del modelo ER
(Algunos libros ni siquiera definen una)
- En este curso, usaremos las siguientes convenciones:
 - PK: subrayado
 - FK: flecha desde la tabla que referencia hacia la referenciada, agregando a veces un código para las reglas de modificación y eliminación:

Acción	Do Nothing	Cascade	Set NULL	Set Default
Delete	DDN	DC	DSN	DSD
Update	UDN	UC	USN	USD

- NULLABLE: asterisco al final*



Ejemplo de Notación



Resumen

- El diseño lógico "acerca" el modelo conceptual hacia un modelo de DBMS
- Nosotros estudiaremos el modelo relacional, cuyos componentes principales son las tablas:
 - Nombre + conjunto de atributos + tuplas
- Para asegurar la consistencia, se definen restricciones de integridad:
 - Clave primaria: identificador
 - Clave foránea: integridad referencial
 - Restricciones de dominio, rango y formato
 - Restricciones de unicidad y obligatoriedad
- PK => UNIQUE y NOT NULL



Diseño Lógico

Parte 2



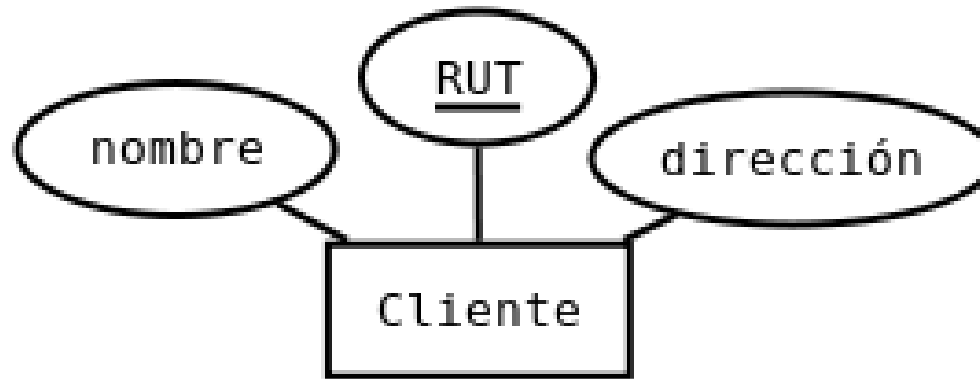
Maapeo de modelos

- Una vez conocidos los elementos del modelo relacional, nos interesa “traducir” un modelo conceptual (ER) a uno lógico (Relacional)
- Esta equivalencia entre dos modelos se denomina ***maapeo***.
- Este maapeo se realiza mediante algunas reglas, que veremos en esta clase
- La aplicación de las reglas es la mayor parte de las veces mecánica, pero también depende del criterio del diseñador.



Mapeo de entidades y atributos

- Toda entidad se mapea a una tabla, y los atributos univaluados se mapean a atributos de la tabla
- La clave se transforma en la clave primaria
(Ya sea simple o compuesta)

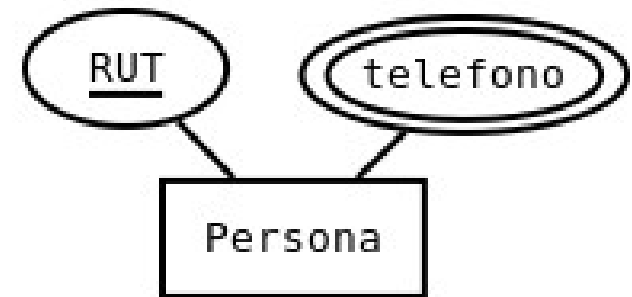


CLIENTE (RUT, nombre, dirección)



Mapeo de atributos multivaluados

- Existen dos opciones:
 - Crear una nueva tabla, para recoger los posibles valores



PERSONA (RUT)
↑
TELEFONO (RUT_persona, numero)

PERSONA (RUT)
↑
TELEFONO (RUT_persona, numero)

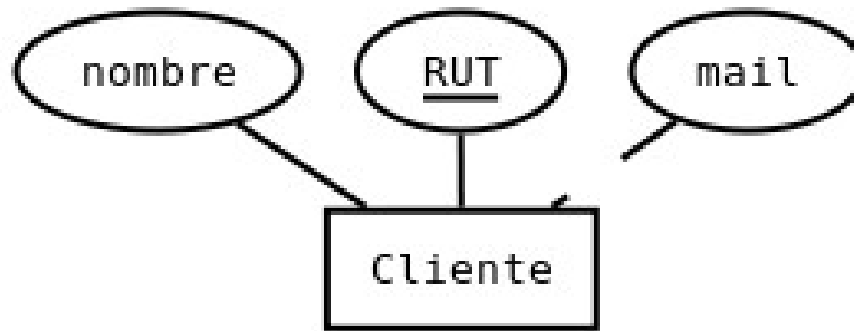
- Si se conoce el número de atributos, crear varios atributos en la tabla

PERSONA (RUT, telefono1, telefono2)



Atributos obligatorios y opcionales

- Atributos obligatorios pueden marcarse como NOT NULL
- Sin embargo, dado que es más común que los atributos sean obligatorios, usualmente se marcan los opcionales como NULLABLE

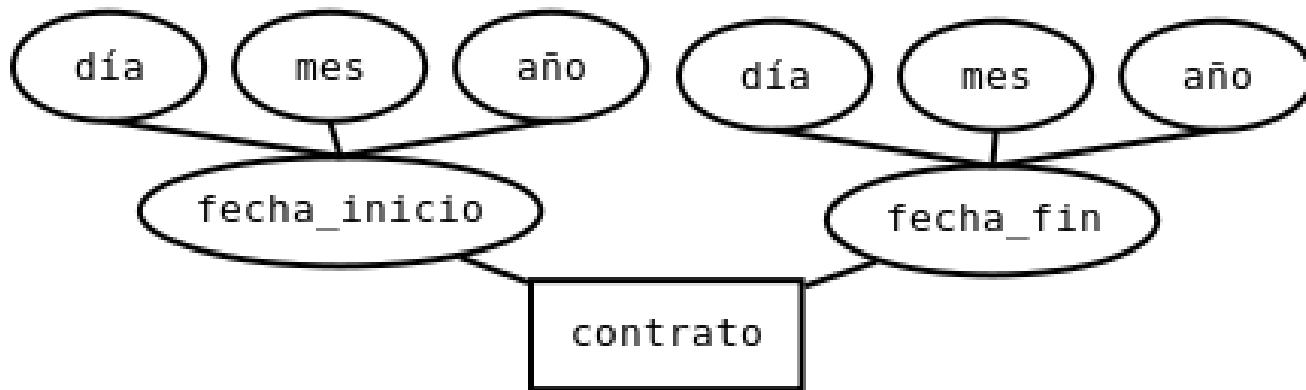


CLIENTE(RUT, nombre, mail*)



Maapeo de atributos compuestos

- Se crean tantos atributos como elementos haya en la composición

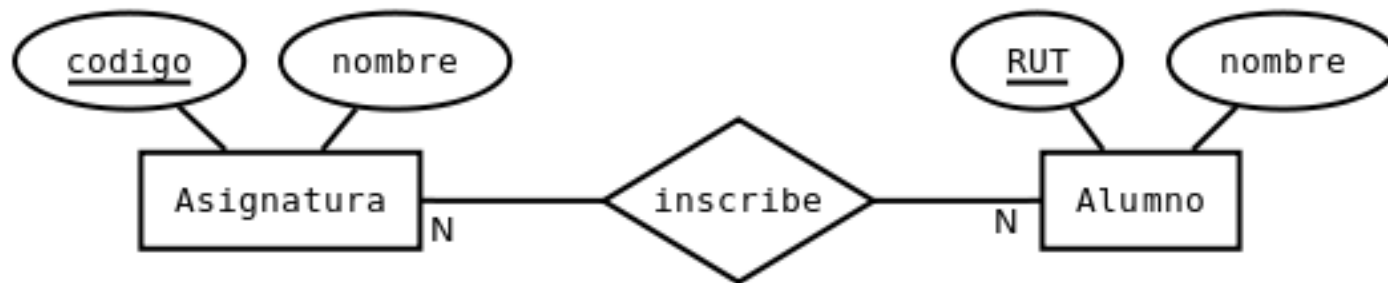


CONTRATO(dia_i,mes_i,año_i, dia_f, mes_f, año_f)



Mapeo de relaciones N:M

- Las relaciones N:M se mapean a una tabla
 - Las ex-entidades participantes propagan sus claves (ahora PKs) como FK en la tabla que mapea la relación
 - La nueva tabla tiene como PK las dos FK de las tablas referenciadas



ALUMNO (RUT, nombre)

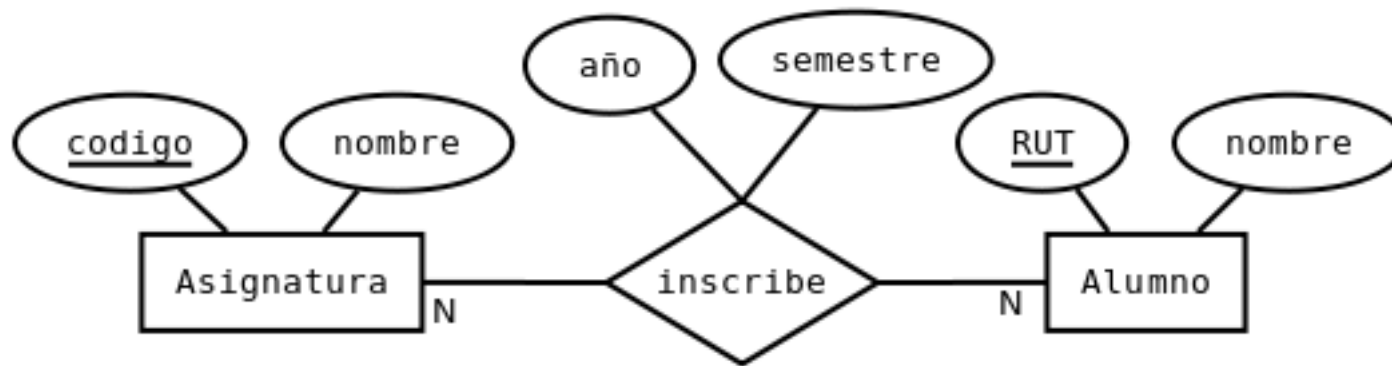
INSCRIPCION (RUT_alumno, cod_asignatura)

ASIGNATURA (codigo, nombre)



Mapeo de relaciones N:M

- Si una relación N:M tiene atributos, simplemente se incluyen en la nueva tabla, como si fuera una entidad débil



ALUMNO (RUT, nombre)

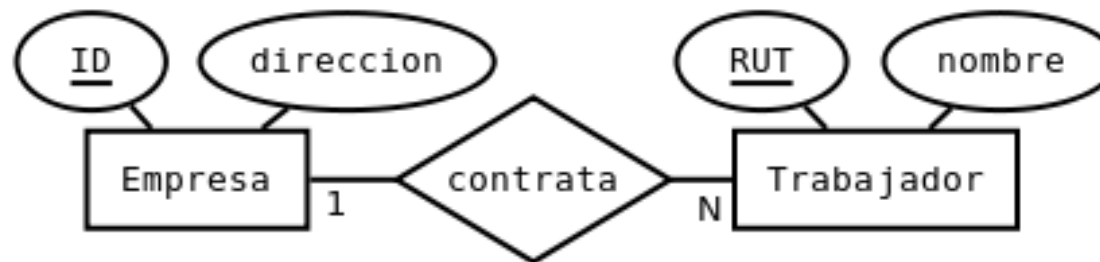
INSCRIPCION (RUT_alumno, cod_asignatura, año, semestre)

ASIGNATURA (codigo, nombre)



Mapeo de relaciones 1:N

- Existen dos opciones:
 - Propagar la PK desde el lado 1 al lado N
 - Mapear la relación a una nueva tabla con PK = PK del lado N
- 1a opción: Propagar la PK desde el lado 1 al lado N:



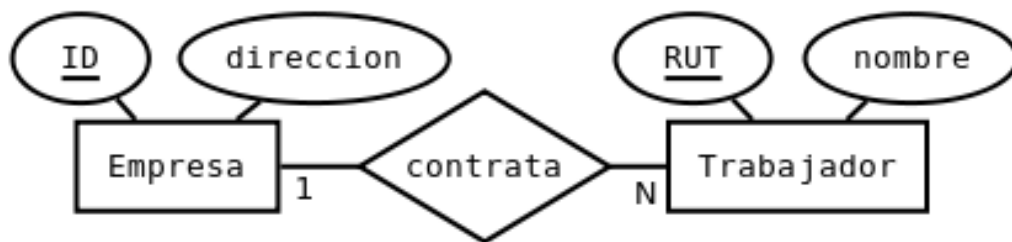
EMPRESA (ID, direccion)

TRABAJADOR (RUT, nombre, ID_empresa)



Mapeo de relaciones 1:N

- 2a opción: Mapear la relación a una tabla con PK = PK del lado N



- ¿Por qué sólo propagar sólo una PK?

- Supongamos que fueran ambas:
(PK compuesta)

EMPRESA (ID, direccion)

CONTRATO (ID_EMPRESA, RUT_trabajador)

TRABAJADOR (RUT, nombre)

E	T	C (c/ 2 PK)
1	A	1,A
2	B	1,B
	C	1,C
		2,A
		2,B
		2,C

Si la PK de C fuera sólo la de T esto no sería posible

(Significa que A trabaja en dos empresas: 1 y 2)



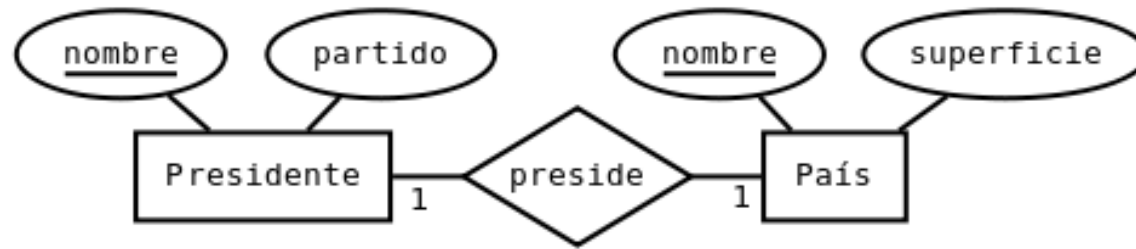
Mapeo de relaciones 1:N

- ¿Cómo decidir cuál alternativa usar?
- Aspectos a considerar:
 - Simpleza del modelo
 - ¿Es necesario agregar otra tabla?
 - Claridad del modelo
 - ¿Es razonable poner el contrato en la tabla persona?
 - Posibles requerimientos futuros del modelo
 - ¿Se agregarán atributos al contrato? ¿habrá relaciones con la entidad contrato?
 - Existencia de atributos en la relación
 - ¿Existen ya atributos para el contrato?



Mapeo de relaciones 1:1

- Ambas tablas están al mismo nivel de importancia
 - Propagar la PK de una como FK de la otra (=> 2 alternativas)



PRESIDENTE (nombre, partido)

PAIS (nombre, superficie, nombre_presidente)

PRESIDENTE (nombre, partido, nombre_pais)

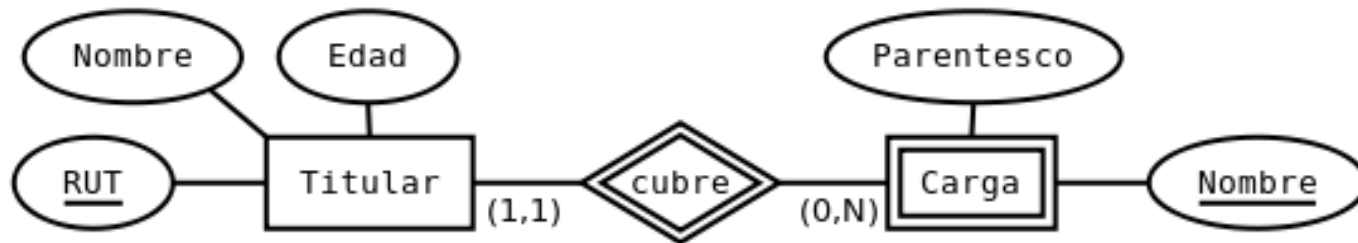
PAIS (nombre, superficie)

- ¿Cuál elegir?
 - La idea es que la “menos importante” dependa de la “más importante”
 - ¿Es una BD de presidentes o de países?
 - Pista: Nombre de la relación: ¿“preside” o “es gobernado por”?



Mapeo de entidades débiles

- Para asegurar que una entidad débil no pueda existir sin la entidad fuerte asociada, Se propaga la PK de la tabla fuerte como PK y FK de la tabla débil



TITULAR (RUT, nombre, edad)

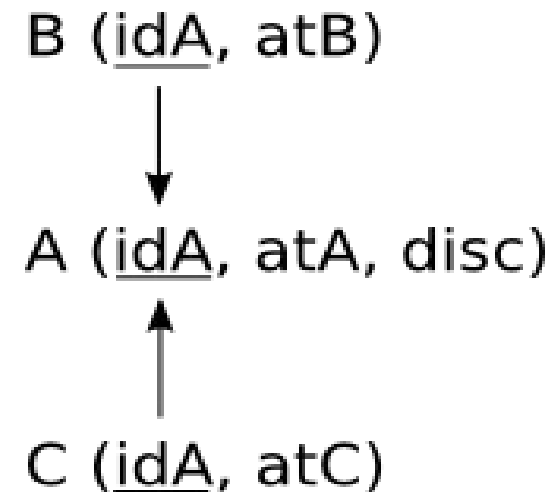
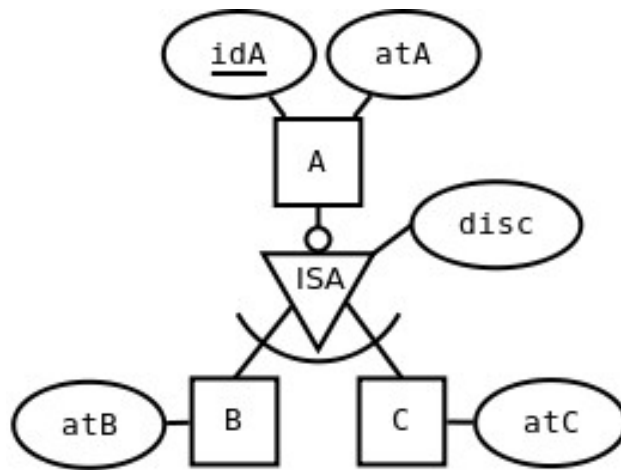
CARGA (RUT_titular, nombre, parentesco)

- N.B. No se puede tener regla DSN o USN (pues no sería realmente dependiente) Generalmente se utiliza DC y UC



Mapeo de generalizaciones

- Se crea una tabla para el supertipo y una para cada subtipo



- Control de Exclusividad:
 - Si es total: hay discriminante y es NOT NULL
 - Si es parcial: hay discriminante y es NULLABLE
 - En cualquier caso, el atributo discriminante se incluye en la tabla del supertipo
 - La integridad del discriminante se asegura externamente
- Control de Totalidad:
 - Mediante código (poner supuesto)



Mapeo de relaciones de grado > 2

- No hay manera de mapearlas directamente
- Es necesario transformarlas a binarias, y luego mapearlas como binarias



Do you remember me?



Documentación de BD

- Metadata: datos sobre los datos
- Diccionario de datos:
 - herramienta de comunicación entre diseñadores y programadores
 - Incluye explicaciones sobre tablas y atributos
- Qué incluir (referencial):
 - Descripción del atributo
 - Tipo de dato
 - Unidad en que se mide
 - Formato
 - Límites (tamaños, valores, si aplica)
 - Otros (PK, FK, regla de derivación, etc.)



Diccionario de datos

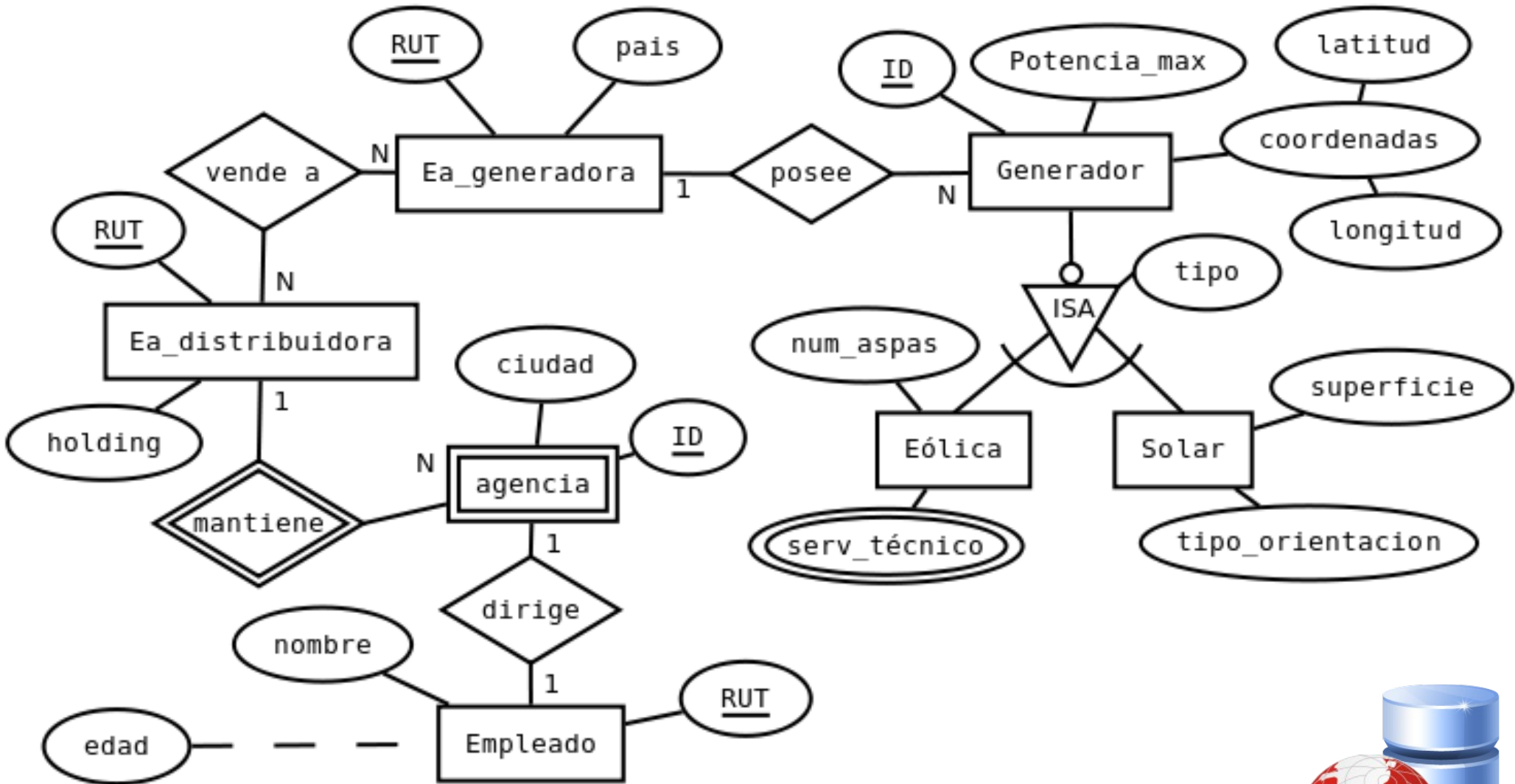
- Ejemplo:

ESP_VEGETAL(denom_cientifica, floracion, incio_fl, fin_fl, presencia)

Nombre tabla	ESP_VEGETAL		
Descripción	Almacena detalles sobre especies vegetales		
Nombre Campo	Descripción	Tipo	Unidad
DENOM_CIENTIFICA	denominación científica de la especie (PK)	Varchar2(30)	
FLORACION	Indica si florece o no ('S'/'N')	Varchar2(1)	
INICIO_FL	fecha de inicio de floración (si floracion vale 'S')	DATE	
FIN_FL	fecha de fin de floración (si floracion vale 'S')	DATE	
PRESENCIA	presencia de la especie en el parque	NUMBER	hectáreas



Ejercicio: Mapear el modelo ER



Resumen

- Mapeo: traducción de un modelo a otro
 - En nuestro caso, ER \rightarrow Relacional
- Mapeo de:
 - Entidades fuertes y atributos univaluados
 - Atributos multivaluados, opcionales y compuestos
 - Mapeo de relaciones N:M, 1:N, 1:1
 - Mapeo de entidades débiles
 - Mapeo de generalizaciones
 - Mapeo de relaciones n-arias



Diseño Lógico

Parte 3



Problemas de diseño

- La manera más metódica de crear un modelo relacional es haciendo el ER antes
- Podría hacerse directamente, pero el riesgo de cometer errores aumenta
- En cualquier caso, a veces es necesario verificar formalmente si el modelo presenta problemas
 - Incapacidad de almacenar hechos
 - Redundancia que puede permitir inconsistencias
 - Ambigüedades
 - etc.



Problemas debidos a mal diseño

- Ejemplo: una multitienda tiene la siguiente tabla en su BD:

CLIENTE (RUT,nombre, num_tarjeta, descuento, plan)

- “num_tarjeta” es único y asignado a cada cliente
 - “descuento” es un porcentaje de descuento sobre sus compras
 - “plan” es el plan de puntos al cual pertenece (GOLD, SILVER, etc.)
 - El descuento está asociado al plan: GOLD: 15%, SILVER: 10%
-
- ¿Qué problemas puede presentar esta tabla?
 - Veamos una instancia...



Problemas debidos a mal diseño

RUT	nombre	num_tarjeta	descuento	plan
1	Yasna Barrientos	334	10%	Silver
2	Patricia Bello	513	15%	Gold
3	Baldomero Águila	889	10%	Silver
4	Boris Sotomayor	114	15%	Gold

- ¿Cuál debería ser la Clave primaria?
- ¿Qué sucede si elimino a Patricia y a Boris? ¿Aún conozco el descuento para el plan Gold? ¿Dónde está esa información?
- ¿Qué pasa si deseo ingresar (5, “Jonathan Muster”, 15%, “Silver”)?



Ejemplos de anomalías

RUT	nombre	num_tarjeta	descuento	plan
1	Yasna Barrientos	334	10%	Silver
2	Patricia Bello	513	15%	Gold
3	Baldomero Águila	889	10%	Silver
4	Boris Sotomayor	114	15%	Gold

- Anomalía de inserción:
 - Si ingreso a Jonathan Muster con plan Platinum ¿De cuánto es el descuento?
- Anomalía de modificación:
 - Descuento para Gold se aumenta en un 20%, pero sólo para Patricia
- Anomalía de eliminación:
 - Si se elimina a Yasna y Baldomero, se pierde el descuento asociado al plan Silver



Normalización

- Una vez mapeado el modelo ER al relacional, pueden persistir problemas
- Normalización: proceso formal para decidir cómo se deben agrupar los datos
 - i.e., cómo deben estar conformadas las tablas
- Un modelo puede tener problemas porque:
 - No se modeló tan bien como era necesario
 - Caso común: tabla creada a partir de un formulario o planilla existentes
 - Hubo cambios en las reglas del negocio que el modelo no consideraba
- Se articula en torno a conjuntos de restricciones, a su vez agrupadas en ***formas normales***



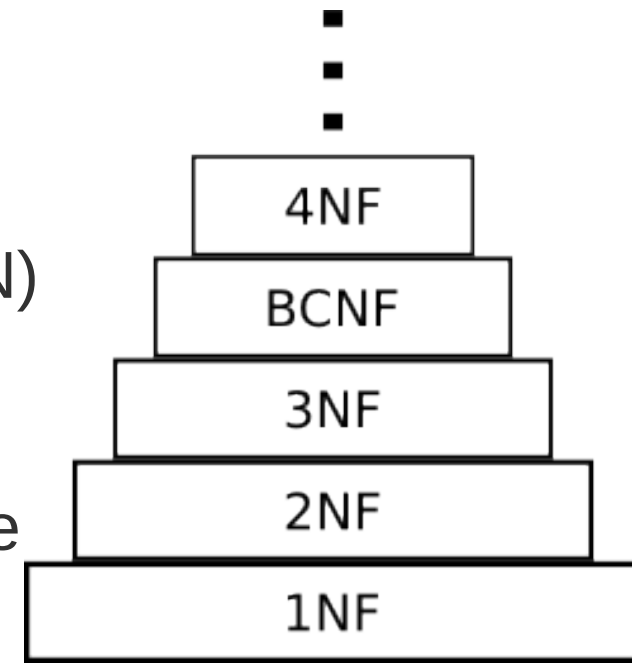
Objetivos de la Normalización

- Evitar la redundancia, almacenando cada hecho en un sólo lugar
- Estructurar los datos para facilitar su modificación
- Evitar anomalías
- Facilitar el cumplimiento de restricciones de datos
- Evitar programación innecesaria
 - Tratar que el modelo resguarde la integridad por sí solo en la medida de lo posible, evitando que procedimientos almacenados y triggers deban velar por ella
- En general, hacer que ***datos de distinta naturaleza se almacenen en tablas diferentes***



Formas Normales

- Para asegurar la inexistencia de distintos tipos de problemas, se definen ciertas restricciones que las tablas deben cumplir
- Estas restricciones se agrupan en “categorías”, llamadas formas normales (FN)
- Las FNs se organizan en un jerarquía en la que una NF de orden superior supone el cumplimiento de las restricciones de FNs de menor orden
- Existen varias formas normales. Veremos las cuatro primeras, que usualmente son suficientes en la práctica



Claves candidatas

- Recuerde que una tabla puede tener varias alternativas de PK
- A cada una de ellas se les conoce como “***claves candidatas***”
- Ejemplo:

PERSONA (nombre, fecha_nac, dirección)

- Claves candidatas:
 - nombre
 - nombre + fecha_nac
 - nombre + dirección
 - nombre + fecha_nac + dirección



1FN

- La primera forma normal de una tabla es la más sencilla
- Simplemente se exige que no existan “grupos repetitivos”
 - que los atributos sean **univaluados**
 - (incluyendo campos univaluados “largos” que acogen varios valores)
- Ejemplos de tablas que no están en 1FN (Según Codd)

<u>Nombre</u>	dirección	teléfono
Charlotte	22 Acacia Av.	345 534, 342 443
Camille	Calle Morgue s/n	223 657

<u>Nombre</u>	dirección	teléfono
Charlotte	22 Acacia Av.	345 534, 342 443
Camille	Calle Morgue s/n	223 657



1FN

- Date extiende esta definición excluyendo la repetición en varias columnas que almacenan la misma información:
- De esta manera, la siguiente tabla no está en 1FN:
 - PERSONA (Nombre, dirección, telefono1, telefono2)
- Pero esta sí:
 - PERSONA (Nombre, dirección, tel_casa, tel_oficina)
- Problemas asociadas a la primera:
 - ¿De quién es el número 349 546? (Se debe consultar en varias columnas)
 - ¿Cómo se controla que ambos sean diferentes?
 - Si un cliente sólo tiene un número, ¿se guarda en teléfono1 o en teléfono2?
 - Si tiene dos, ¿en qué orden se colocan?



1FN: Solución

- Separar en dos (o más) tablas
 - Una para asociar los datos repetidos a la clave
 - La otra con el resto de la información

<u>Nombre</u>	dirección	teléfono
Charlotte	22 Acacia Av.	345 534, 342 443
Camille	Calle Morgue s/n	223 657

<u>Nombre</u>	dirección
Charlotte	22 Acacia Av.
Camille	Calle Morgue s/n

<u>Nombre</u>	<u>teléfono</u>
Charlotte	345 534
Charlotte	342 443
Camille	223 657



2FN

- Una tabla está en 2FN ssi:
 - Está en 1FN
 - Todo atributo que no pertenece a la clave candidata, entrega información sobre la **totalidad** de ella
- Ejemplo de tabla que no está en 2FN:

OPERACION(id_hospital, número_operación, nombre_operación, **nombre_hospital**, nombre_cirujano)

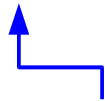
- (número_operación es correlativo en cada hospital)
- nombre_hospital sólo entrega información sobre id_hospital, no sobre **toda** la PK (id_hospital, número_operación)



2FN: Solución

- OPERACION(id_hospital, número_operación, nombre_operación, nombre_hospital, nombre_cirujano)
- Separar en dos (o más) tablas:
 - En una, como PK los atributos a los que se refieren los atributos problemáticos
(“id_hospital” en nuestro caso)
 - En la otra, el resto de los atributos, con la PK original y FK a la nueva tabla

HOSPITAL(id_hospital, nombre_hospital)



OPERACION(id_hospital, número_operación, nombre_operación, nombre_cirujano)



3FN

- Una tabla está en 3FN ssi:
 - Está en 2FN
 - Todo atributo que no está en la clave candidata entrega información **exclusivamente** sobre ella
- Ejemplo de tabla que no está en 3FN:

ALUMNO(RUT, nombre, ciudad, región)

- Región también entrega información sobre la ciudad
- Es decir, si la ciudad es “Valdivia”, no puedo poner otra región que “Los Ríos”
- Dicho de otra manera, la región **depende** de la ciudad, o la ciudad **implica** la región



3FN: Solución

ALUMNO(RUT, nombre, ciudad, región)

- Separar en dos (o más) tablas:
 - Una con PK el campo apuntado por el problemático
 - Otra con la PK original, y FK a la nueva tabla

ALUMNO (RUT, nombre, ciudad)

↓

CIUDAD (nombre, región)



Dependencia funcional

- Hemos dicho que un atributo puede “depender” de otro
- El concepto de ***dependencia funcional*** nos ayudará a comprender mejor la definición de la 2 y 3 FN

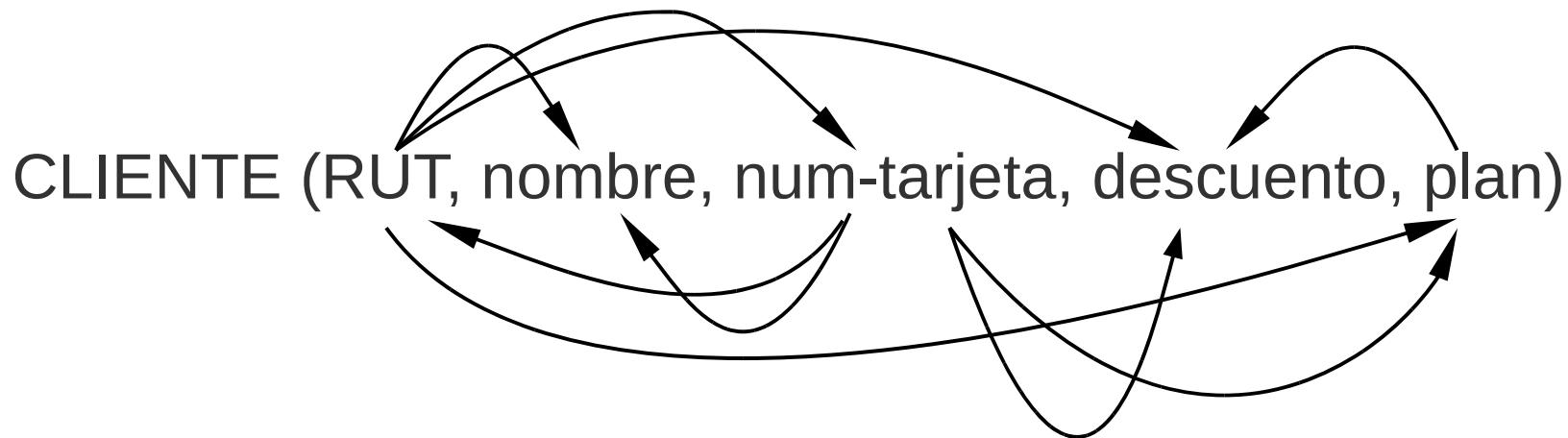
RUT	nombre	num_tarjeta	descuento	plan
1	Yasna Barrientos	334	10%	Silver
2	Patricia Bello	513	15%	Gold
3	Baldomero Águila	889	10%	Silver
4	Boris Sotomayor	114	15%	Gold

- En este caso, el descuento ***depende*** del plan, o equivalentemente, el plan ***implica*** (o ***determina***) el descuento
 - Notación: **plan → descuento**



Dependencia funcional

- En la dependencia $X \rightarrow Y$
 - X es llamado “determinante” o “implicante”
 - Y es llamado “implicado”
- Se puede definir un grafo de dependencias entre los atributos:

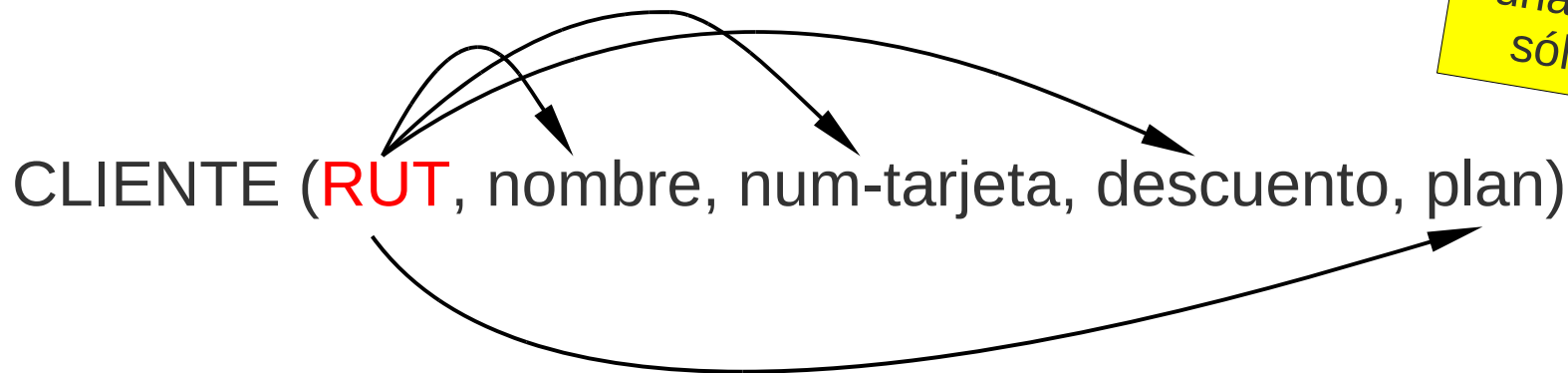


- **N.B. No confundir con flechas de FK !**



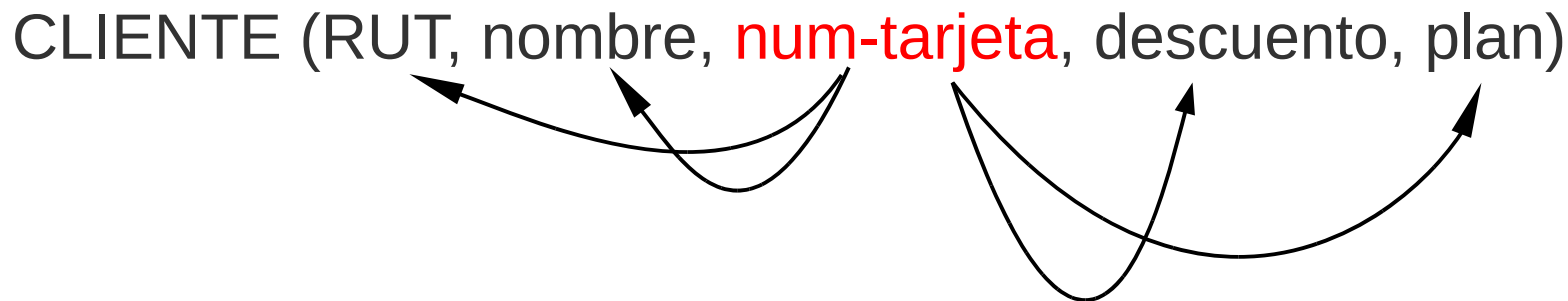
Analizando en detalle

Si supiera el RUT, puedo deducir:



Obs: se asume que una persona tiene sólo **una** tarjeta

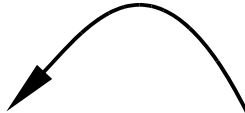
Si supiera en número de tarjeta, puedo deducir:



Analizando en detalle

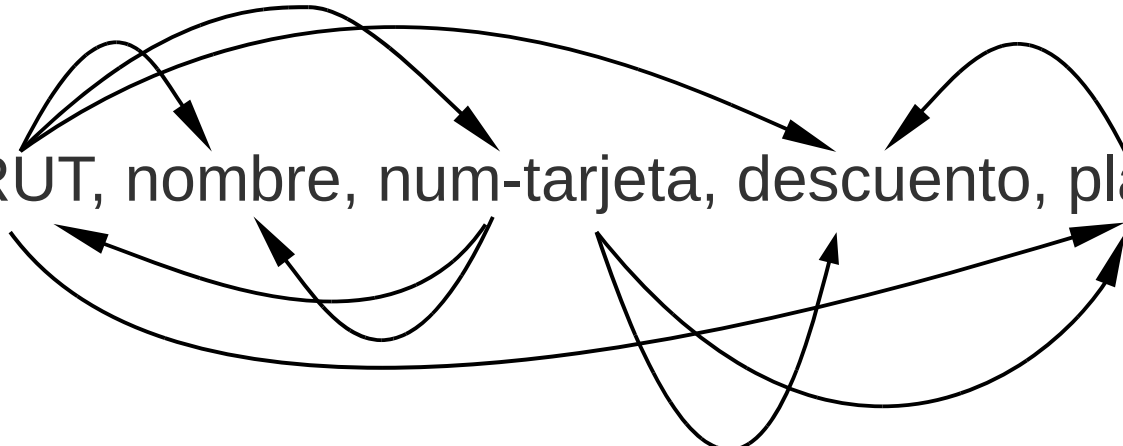
Si supiera el plan, puedo deducir:

CLIENTE (RUT, nombre, num-tarjeta, descuento, **plan**)



De nuevo, todo junto:

CLIENTE (RUT, nombre, num-tarjeta, descuento, plan)



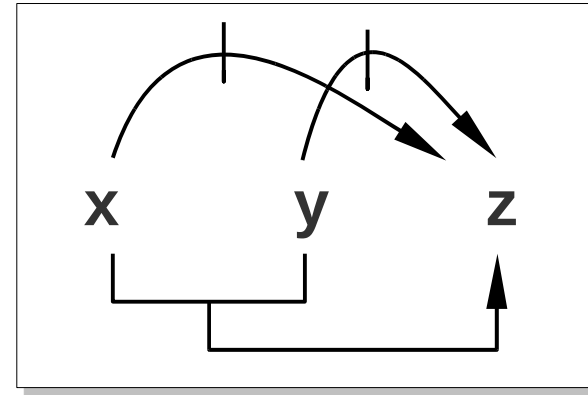
Dependencia Funcional Completa

- Sea una tabla $T(x, y, z)$

- Si se cumple que:

- $(x, y) \longrightarrow z$
- $x \not\longrightarrow z$ (x no implica z)
- $y \not\longrightarrow z$

- Entonces se dice que z tiene **dependencia funcional completa (o plena)** de (x, y)



- De esta manera, una tabla está en **2FN** ssi:

- Está en 1FN
- Todos los atributos no-clave tienen dependencia funcional completa de la clave

(“Todo atributo que no pertenece a la clave candidata, entrega información sobre la **totalidad** de ella”)



Dependencia Funcional Transitiva

- Sea una tabla $T(x, y, z)$

- Si se cumple que:

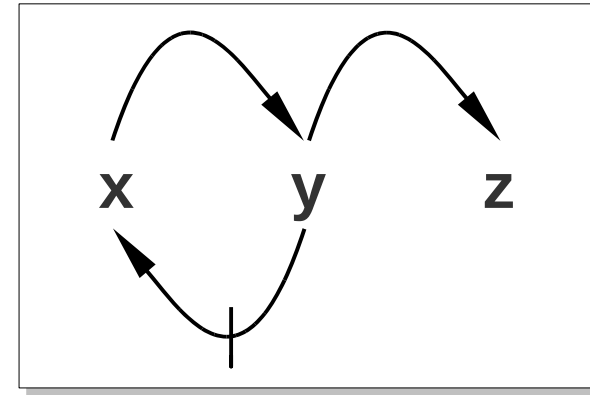
- $x \longrightarrow y$

- $y \longrightarrow z$

- $y \not\longrightarrow x$

- Entonces se dice que Z tiene **dependencia funcional transitiva** de x a través de y

- Notación: $x \longrightarrow z$



(“Todo atributo que no está en la clave candidata entrega información **exclusivamente** sobre ella”)

- De esta manera, una tabla está en **3FN** ssi:

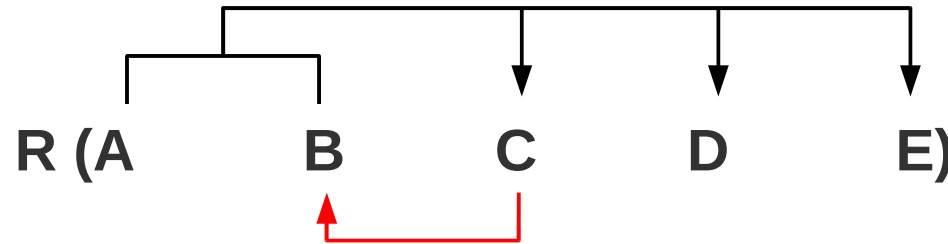
- Está en 2FN

- No existen atributos no-clave que tengan dependencia transitiva de la clave



FN de Boyce-Codd

- Existe una reformulación de la 3FN, conocida como Forma Normal de Boyce-Codd (BCNF)
- Se dice que una tabla esta en BCNF ssi:
 - Está en 3FN
 - Todos sus determinantes son claves candidatas
- Es decir, se trata de evitar situaciones como la siguiente:

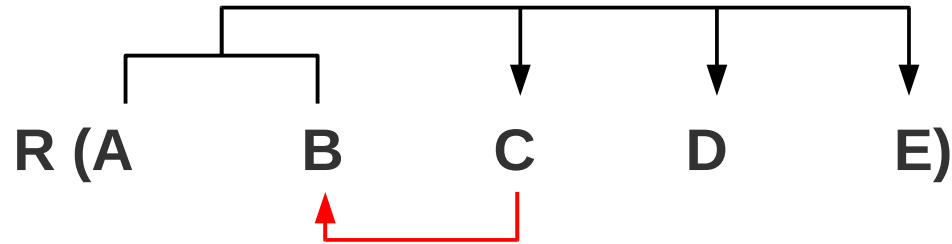


- Determinantes de R (partes izquierdas de dependencias):
 - A + B (puede ser PK, pues determina a todos los otros atributos)
 - C (no puede ser PK, pues sólo determina a B y no al resto)
(Si C determinara a todo el resto no habría problema, sería clave candidata)

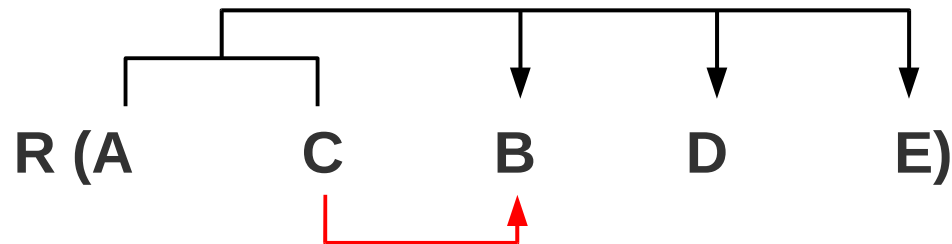


BCNF: Solución

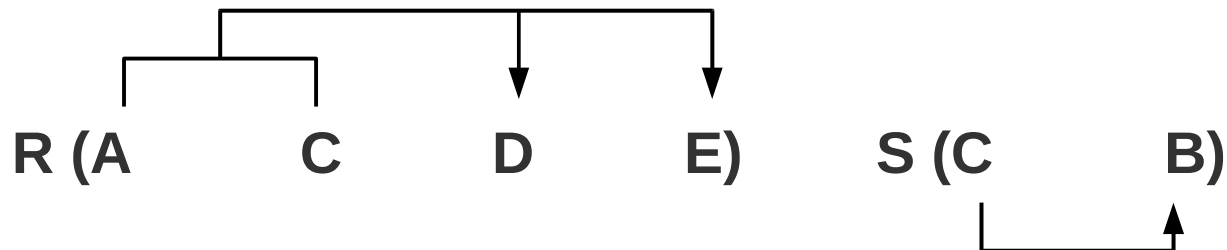
- La solución se realiza en dos pasos:



- Primero, cambiar la clave a $A+C$ (queda tabla en 1FN, debido a la dependencia parcial $C \rightarrow B$):



- Luego separar para eliminar dependencia parcial:



Ejercicio

- Hacer el grafo de dependencias y normalizar a 3FN la siguiente tabla:

INSCRIPCION (alumno, ramo, créditos, unidades, sala, edificio, plan, nota)

- En donde:
 - créditos es el número de créditos del ramo
 - unidades son los contenidos del ramo
 - plan indica a qué plan pertenece un ramo. puede tomar los valores “bachillerato”, “licenciatura” o “profesional”
 - La sala se encuentra en un edificio
 - Nota es la nota del alumno en el ramo



Resumen

- La normalización es un método que permite evitar problemas debidos a un mal diseño
- Ayuda a poner "cada cosa en su lugar" dividiendo las tablas que almacenan datos de cosas diferentes
- Se organiza en formas normales:
 - 1FN: evitar grupos repetitivos
 - 2FN: evitar dependencias parciales
 - 3FN: evitar dependencias transitivas
 - BCNF: todos los determinantes son claves candidatas
- El análisis de dependencias ayuda a visualizar qué atributos están implicados por otros

