INTRODUCCIÓN A LAS BASES DE DATOS RELACIONALES

Rodrigo García Carmona
Universidad San Pablo-CEU
Escuela Politécnica Superior

INTRODUCCIÓN

DEFINICIÓN DE DBMS

- Los sistemas de gestión de bases de datos (DBMS) proporcionan:
  - Almacenamiento y acceso **eficiente**, **fiable**, **conveniente** y **seguro** a cantidades **enormes** de información **persistente** por parte de **múltiples usuarios**.

CARACTERÍSTICAS DE UN DBMS

- **Eficiente:** Miles de peticiones/actualizaciones por segundo.
- **Fiable:** 99.99999% del tiempo funcionando.
- **Conveniente:**
  - Independencia de los datos físicos.
  - Lenguajes de peticiones declarativos de alto nivel.
- **Seguro:** A nivel de *hardware*, *software*, energía, usuarios.
- **Cantidades enormes de información:** Terabytes o más.
- **Persistente:** Información que perdura.
- **Múltiples usuarios:** Control de accesos concurrentes.

ESTRUCTURA EN CAPAS

- Las aplicaciones que hacen uso de la base de datos suelen programarse para que hagan uso de *frameworks*.
- Los DBMS suelen compaginarse con *middleware* específico.
- Las aplicaciones que hacen uso de un DBMS no tienen ni por qué saber de sus características…
  - …¡o si quiera de su existencia!
- Durante este curso usaremos *middleware* y *frameworks* Java.

CONCEPTOS CLAVE

- Modelo de datos:
  - Conjunto de registros o datos tabulados.
  - Datos jerárquicos o en forma de red.
  - Información desorganizada.
- En un DBMS debemos definir esquemas y poblarlos con datos.
- Podemos interactuar con un DBMS a través de 3 lenguajes:
- **DDL:** Lenguaje de definición de datos.
  - Para definir el esquema.
- **DML:** Lenguaje de manipulación de datos.
  - Para hacer consultas y actualizaciones.
- **DCL:** Lenguaje de control de datos.
  - Para controlar el acceso de los usuarios.

ROLES CLAVE

- Varias personas interactúan con un DBMS durante su ciclo de vida:
- **Implementador del DBMS:**
  - Construye el sistema.
- **Diseñador de la base de datos:**
  - Establece el esquema.
- **Desarrollador de aplicaciones:**
  - Programas que interactúan con la base de datos.
- **Administrador de la base de datos:**
  - Manipula datos.
  - Consigue que siga funcionando.

IMPORTANCIA

- Seamos conscientes o no, estamos usando una base de datos casi constantemente…
  - Servicios de internet (email, redes sociales).
  - **TIC** y ordenadores.
  - Logística y comercio.
  - Gobierno y administración pública.
  - Bancos y finanzas.
  - Sanidad.
  - …

BASES DE DATOS RELACIONALES

EL MODELO RELACIONAL

- Utilizado por la gran mayoría de sistemas comerciales…
  - …aunque NoSQL va ganando popularidad.
- Modelo muy sencillo.
- Tablas bidimensionales.
- Construido a partir de las matemáticas: **conjuntos** y **relaciones**.
- Consultas utilizando lenguajes de alto nivel…
  - …sencillos pero expresivos.
  - SQL o SQL-like.
- Implementaciones muy eficientes.

TÉRMINOS BÁSICOS (I)

- **Base de datos:** Conjunto de **relaciones** (**tablas**).
- Cada relación tiene una conjunto de **atributos** (**campos**, **columnas**) identificadas por su nombre.
- Cada **tupla** (**registro**, **fila**) tiene un **valor** para cada atributo.
- Cada atributo tiene un **tipo** (**dominio**).
- Un atributo puede ser **único** (no se pueden repetir valores en varias tuplas).

students

id	name	score	photo
123	Anna	6.5	^_^
234	Bob	3.3	NULL
345	Mike	NULL	-_-
…	…	…	…

universities

name	city	students
CEU	Madrid	11500
UPV	Valencia	40000
MIT	Cambridge	10000
…	…	…

TÉRMINOS BÁSICOS (II)

- **Esquema:** Descripción estructural de una base de datos.
- **Instancia:** Contenidos de una base de datos en un instante de tiempo determinado.
- **Clave primaria (primery key):** Atributo o combinación de atributos que identifican una fila.
- **Clave ajena (foreign key):** Atributo que referencia una clave primaria de otra tabla.
- **NULL:** Valor especial que significa "desconocido" o "no identificado".
- Un atributo puede ser **NOT NULL** (no puede tomar el valor NULL).

students

id	name	score	photo
123	Anna	6.5	^_^
234	Bob	3.3	NULL
345	Mike	NULL	-_-
…	…	…	…

universities

name	city	students
CEU	Madrid	11500
UPV	Valencia	40000
MIT	Cambridge	10000
…	…	…

CÓMO CREAR Y USAR BASES DE DATOS RELACIONALES

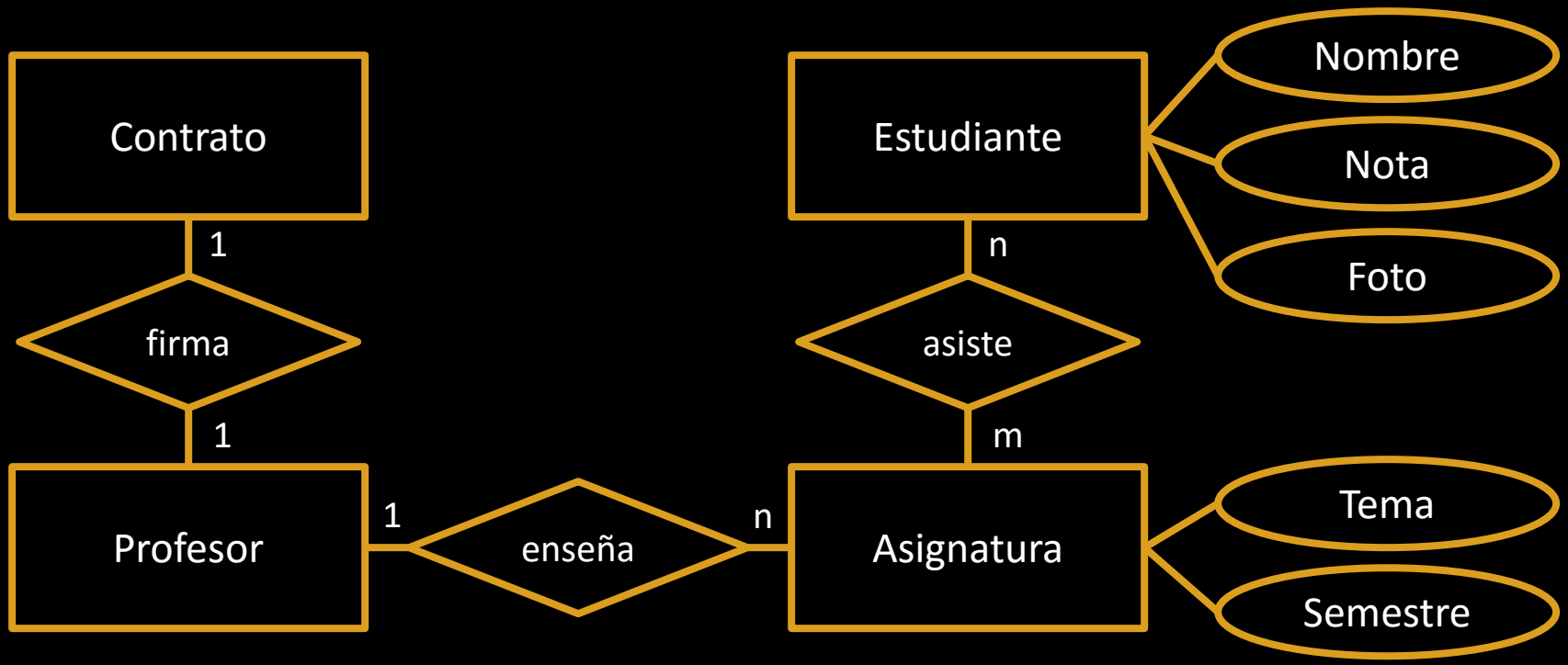
- **Para crear:**
  1. Diseñar el esquema (¡sobre el papel!)
  2. Escribir el esquema usando **DDL**
  3. Cargar el conjunto de datos inicial usando **DML**.
- **Para usar:**
  1. Pensar en la consulta que queremos hacer (¡sobre el papel!)
  2. Escribir la consulta usando **DML**

DEFINICIÓN DEL ESQUEMA

MODELO ENTIDAD-RELACIÓN

- El **modelo E-R (Entidad-Relación)** es una herramienta de análisis usada para crear bases de datos:
- **Entidad:** un tipo de objeto en el mundo real.
  - Se representa con una caja rectangular.
  - Cada entidad puede tener propiedades.
    - Representadas mediante óvalos.
- **Relación:** cómo dos entidades están relacionadas.
  - Se representa con un rombo con dos líneas.
  - **Tipos de relación:**
    - Uno-a-uno.
    - Uno-a-muchos.
    - Muchos-a-muchos.

MODELO E-R DE EJEMPLO



NORMALIZACIÓN DE BASES DE DATOS

- **La normalización de bases de datos** es el proceso utilizado para organizar sus atributos y tablas y minimizar redundancias. Transformamos una tabla grande en varias tablas pequeñas.
- Con la normalización buscamos:
  - **Minimizar redundancias:** La información no debería estar duplicada.
  - **Aislar la información:** Las inserciones, modificaciones y eliminaciones sólo deberían afectar a una única tabla.
  - **Evitar el perder información:** Las relaciones entre tablas se representan mediante claves ajenas..
- **Formas normales:**
  - **1ª Forma Normal (1NF):** Sólo un atributo por campo.
    "Sin filas duplicadas"
  - **2ª Forma Normal (2NF):** Se permite la dependencia funcional transitiva.
    "Los valores se determinan a partir de la clave o de un valor determinado por la clave"
  - **3ª Forma Normal (3NF):** Sólo se permite la dependencia funcional no-transitiva.
    "Los valores sólo se determinan a partir de la clave"

BASE DE DATOS SIN NORMALIZAR

invoice id	date	client id	client name	product id	product name	product price	tax rate	amount
001	2014-09-17	C01	Dracotienda	ISH01	La Puerta de Ishtar	38.00€	4%	10
				ISH02	Ishtar: Pantalla	19.50€	21%	5
002	2014-09-17	C02	Tesoros de la Marca	ISH01	La Puerta de Ishtar	38.00€	4%	7
				ABL01	Ablaneda	14.00€	4%	3
003	2015-01-10	C01	Dracotienda	ABL01	Ablaneda	14.00€	4%	10
				ABL02	Relatos de Ablaneda	9.50€	4%	5

1ª FORMA NORMAL

invoice id	date	client id	client name
001	2014-09-17	C01	Dracotienda
002	2014-09-17	C02	Tesoros de la Marca
003	2015-01-10	C01	Dracotienda

invoice id	product id	product name	product price	tax rate	amount
001	ISH01	La Puerta de Ishtar	38.00€	4%	10
001	ISH02	Ishtar: Pantalla	19.50€	21%	5
002	ISH01	La Puerta de Ishtar	38.00€	4%	7
002	ABL01	Ablaneda	14.00€	4%	3
003	ABL01	Ablaneda	14.00€	4%	10
003	ABL02	Relatos de Ablaneda	9.50€	4%	5

2ª FORMA NORMAL

invoice id	product id	amount
001	ISH01	10
001	ISH02	5
002	ISH01	7
002	ABL01	3
003	ABL01	10
003	ABL02	5

product id	product name	product price	tax rate
ISH01	La Puerta de Ishtar	38.00€	4%
ISH02	Ishtar: Pantalla	19.50€	21%
ABL01	Ablaneda	14.00€	4%
ABL02	Relatos de Ablaneda	9.50€	4%

invoice id	date	client id	client name
001	2014-09-17	C01	Dracotienda
002	2014-09-17	C02	Tesoros de la Marca
003	2015-01-10	C01	Dracotienda

3ª FORMA NORMAL

invoice id	product id	amount
001	ISH01	10
001	ISH02	5
002	ISH01	7
002	ABL01	3
003	ABL01	10
003	ABL02	5

product id	product name	product price	tax rate
ISH01	La Puerta de Ishtar	38.00€	4%
ISH02	Ishtar: Pantalla	19.50€	21%
ABL01	Ablaneda	14.00€	4%
ABL02	Relatos d Ablaneda	9.50€	4%

client id	client name
C01	Dracotienda
C02	Tesoros de la Marca

invoice id	client id	date
001	C01	2014-09-17
002	C02	2014-09-17
003	C01	2015-01-10

CÓMO CREAR TABLAS USANDO SQL

```
CREATE TABLE students (
id INTEGER NOT NULL,
name VARCHAR(255) UNIQUE NOT NULL,
score FLOAT,
photo BLOB,
PRIMARY KEY(id)
);

CREATE TABLE universities (
name VARCHAR(255) NOT NULL,
city VARCHAR(255) NOT NULL,
students INTEGER,
PRIMARY KEY(name)
);
```

CÓMO CARGAR DATOS USANDO SQL

```
INSERT INTO students (id, name, score, photo)
VALUES (123, ‘Anna’, 6.5, ^_^);
INSERT INTO students (id, name, score)
VALUES (234, ‘Bob’, 3.3);
INSERT INTO students (id, name, photo)
VALUES (345, ‘Mike’, -_-);

INSERT INTO universities (name, city, students)
VALUES (‘CEU’, ‘Madrid’, 11500);
INSERT INTO universities (name, city, students)
VALUES (‘UPV’, ‘Valencia’, 40000);
INSERT INTO universities (name, city, students)
VALUES (‘MIT’, ‘Cambdrige’, 10000);
```

CONSULTAS

LENGUAJE NATURAL VS. LENGUAJE DE CONSULTAS

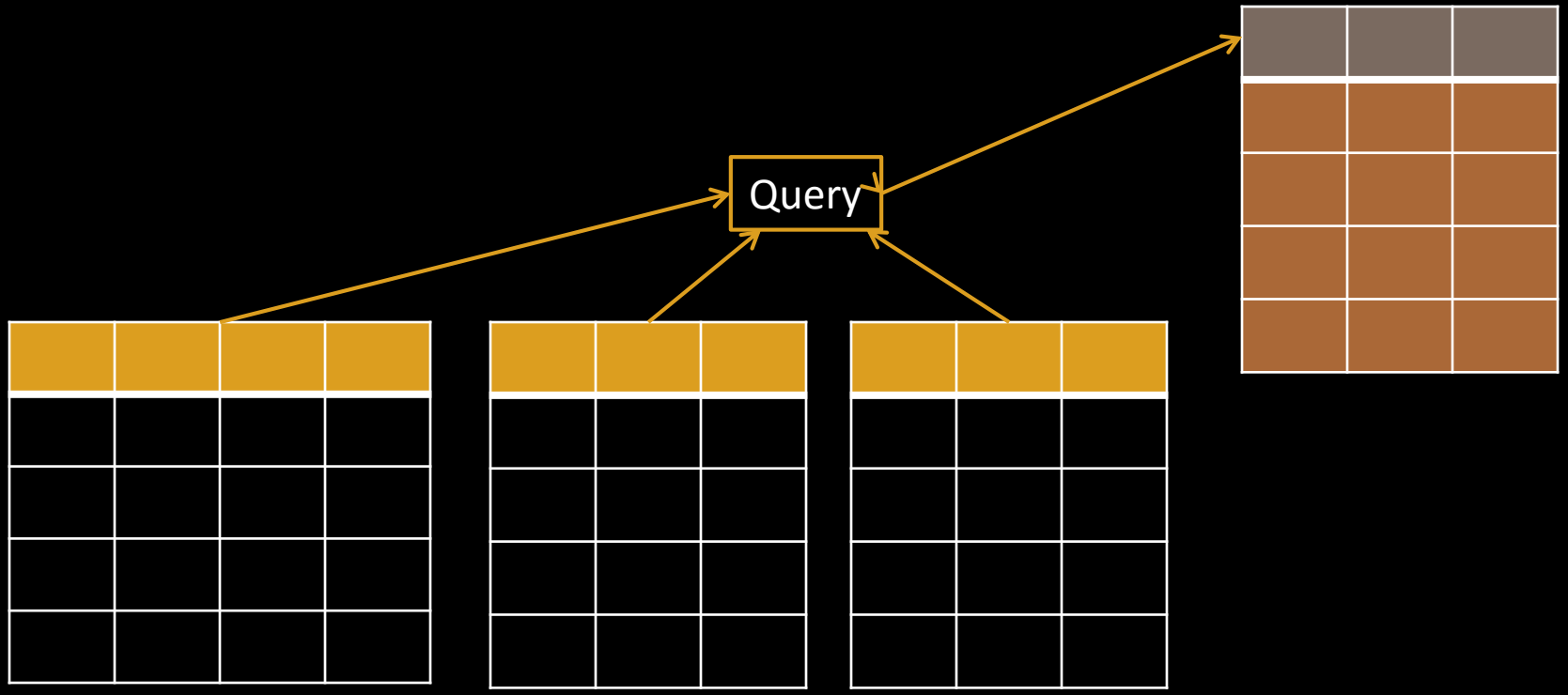
- Ejemplos de consultas usando lenguaje natural:
  - *Los nombres de todas las universidades en Madrid que tengan más de 20000 estudiantes.*
  - *Todos los estudiantes con una nota de menos de un 8.5 que quieran estudiar en el MIT.*
  - *La universidad con la nota media de sus alumnos admitidos más alta.*
- Estas mismas consultas usando lenguaje de consultas:
  - ```
    SELECT name FROM universities
    WHERE city IS Madrid AND students > 20000
    ```
  - ```
    SELECT * FROM students, applications
    WHERE students.id=applications.id
    AND students.score < 8.5
    AND applications.university IS ‘MIT’
    ```
  - Fuera del alcance de este tema…

LENGUAJES DE CONSULTAS

- Las consultas pueden ser más difíciles o fáciles de:
  - Para el usuario: expresar.
  - Para la base de datos: ejecutar eficientemente.
  - **No existe correlación** entre estos dos puntos.
- El "lenguaje de consultas" (**DML**) se usa también para modificar datos, no sólo par acceder a ellos.
- Hay varios lenguajes de consultas
  - **SQL:**
    - El más usado. Trasparencia anterior.
  - **Álgebra Relacional:**
    - Muy formal. Poco usado en la práctica.
    - $\pi_{id}\sigma_{score<8.5 \wedge university='MIT'}(students * applications)$

INFORMACIÓN DEVUELTA

- Tras hacer una consulta, el DBMS devuelve **relaciones**.



OPERACIONES DE EXTRACCIÓN DE DATOS

- Las bases de datos relacionales utilizan las reglas de la matemática discreta.
- Las operaciones de extracción de datos (consultas) se construyen sobre las teorías de conjuntos y relaciones.
- **Operaciones sobre conjuntos:**
  - **Unión:** Tablas con el mismo esquema.
  - **Diferencia:** Tablas con el mismo esquema.
  - **Intersección:** Tablas con el mismo esquema.
  - **Producto cartesiano:** Tablas con el mismo o distintos esquemas.
- **Operaciones sobre relaciones:**
  - **Proyección:** Extrae una columna.
  - **Selección:** Extrae una fila.
  - **Unión (Join):** Construye una tabla a partir de otras dos y una condición de unión.
  - **División:** Tomando dos tablas, extrae las filas de la primera que también estén en la segunda, pero sólo las columnas que no estén en la segunda.

SENTENCIAS DML

- Hemos llamado a todas las sentencias “queries”, pero las hay de varios tipos:
  - **Selección:** Para extraer datos.
    - SELECT en SQL.
  - **Inserción:** Para añadir datos nuevos.
    - INSERT en SQL.
  - **Modificación:** Para cambiar datos ya existentes.
    - UPDATE en SQL.
  - **Borrado:** Para eliminar datos ya existentes.
    - DELETE en SQL.