1.- Presentación de bases de datos.

Ideas generales y ejemplos (por ejemplo de alumnos, materias, profesores, etc.)

2.- Información en sistemas de archivos tradicionales frente a Sistemas de Gestión de Bases de Datos (SGBD)

3.- Nomenclatura básica (conjunto, tupla, etc.) y diseño de Bases de Datos.

4.- Modelos de Bases de Datos.

El modelo relacional.

5.- Desarrollo de un ejemplo concreto de aplicación.

Etapas del diseño

» ¿qué pasos se siguen en el diseño de bases de datos?

» existen una serie de diseños que se realizan en un cierto orden

- Etapas de diseño
  - Análisis de requisitos
    - *comprender* los datos a gestionar
    - necesidades del cliente: reuniones, discusiones, documentaciones, …
    - etapa clave que puede ser muy costosa

…

- Diseño conceptual
  - descripción de *alto nivel* de los datos y sus restricciones
  - modelo que representa, organiza, *clarifica* la información —habitualmente Entidad-Relación—
  - *preciso*, que permita la traducción a un modelo específico del SGBD
- Diseño lógico
  - esquema de la BD acorde al SGBD elegido
  - (*relacional*) traducir esquema ER a esquema relacional
- Refinamiento de los esquemas
  - restructuración para garantizar propiedades importantes —*normalización*—

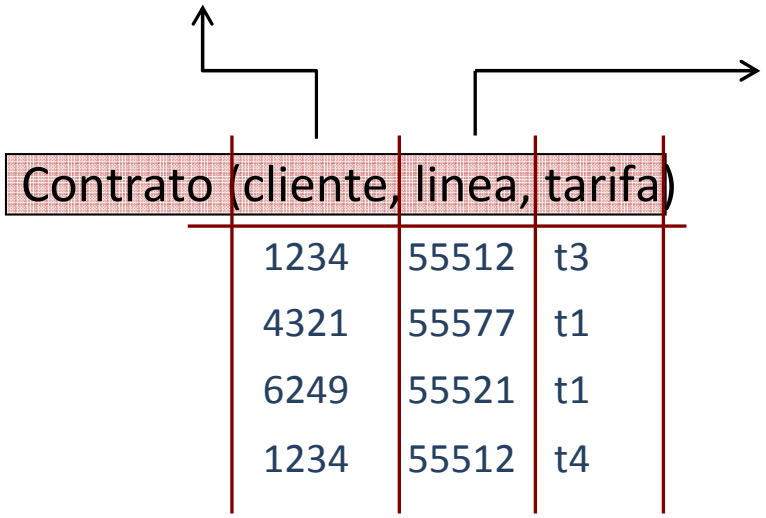
...

- Diseño físico
  - mejora de rendimiento en base a cargas típicas
  - idealmente no supone rediseño de etapas anteriores
- Diseño de aplicaciones y seguridad
  - procesos relacionados con las aplicaciones
  - tareas y flujos de trabajo
  - accesibilidad y seguridad

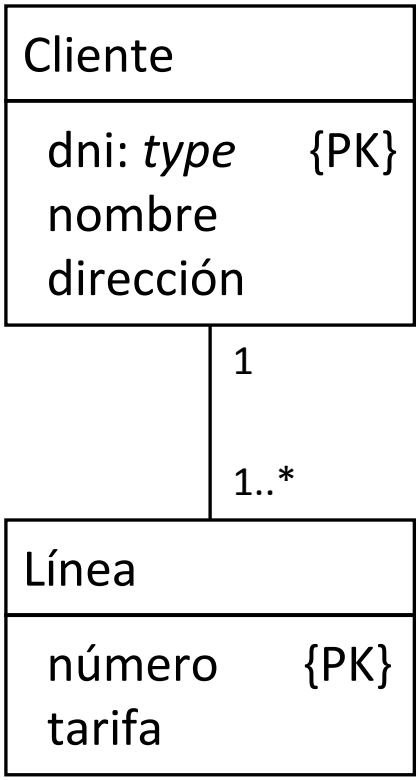
D. lógico (o conceptual) primero

» sobre el sistema de gestión telefónica

Cliente (dni, nombre, direccion)



- ¿dos clientes pueden tener la misma línea?
- ¿una línea puede tener dos tarifas?
- ¿hay que poner más tablas? ¿cuáles serían?



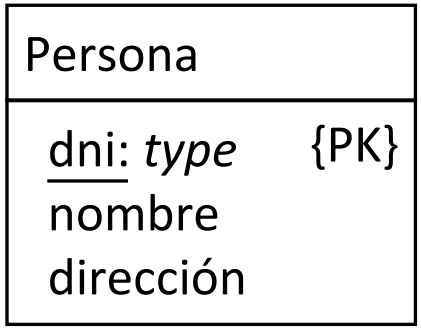
- ¿ER o UML?
- ER en notación UML

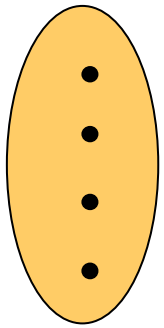
Diseño conceptual ER

» descripción de cómo se *estructuran* los datos

» varias alternativas para un mismo escenario

- Entidades y conjuntos de entidad
  - objetos que engloban los datos de interés. Ejemplo, una persona, un coche, un usuario, etc.
  - se describen como colecciones de entidades similares
  - se describen mediante atributos y *propiedades* adicionales
  - instancias como elementos de un conjunto

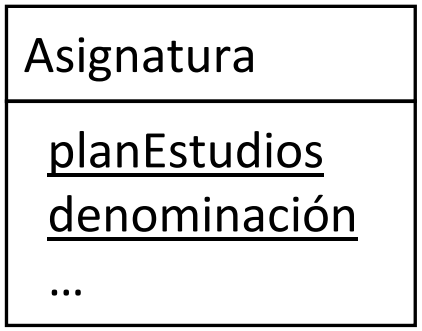


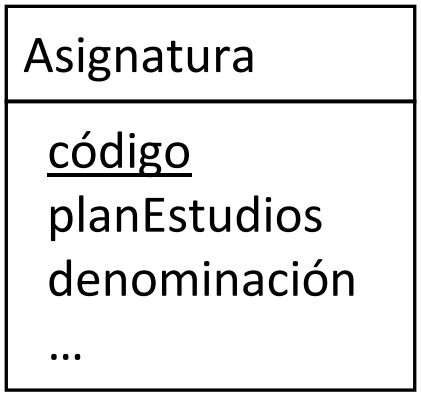


[123321,Santiago,'Plaza Mayor 1']

...

- Atributos (:dominios) y claves
  - Una entidad se representa por un conjunto de atributos (propiedades descriptivas) del mismo.
  - definición de datos para cada instancia
  - el dominio (opcional) es el conjunto posible de valores; predefinidos (INT, TEXT, DATE, etc.) o posibilidad de definición de tipos.
  - clave: conjunto mínimo de atributos que *identifican* a cada entidad (valor mínimo sin repetición). Las entidades deben ser distinguibles de alguna forma.





- identificación más sencilla
- ¿cómo se asegura la no repetición plan + denominación?
- claves alternativas

Relaciones

•Una relación es una asociación entre dos o más entidades, por ejemplo, “el usuario XXX ha pedido el producto YYY” (usuario y producto son entidades tipo, “ha pedido” es la relación).

- definición de datos para cada instancia
- grado de una relación es el número de tipos de entidades que participan en la relación (binaria, ternaria, …) .
- atributos de una relación: una relación puede tener atributos que describen las propiedades de una relación. Estos atributos no están ligados a las entidades participantes sino a la propia relación. Una relación no tiene atributos clave. La identificación de una relación concreta se hace mediante las claves de las entidades participantes.

Restricciones en las Relaciones

Limitan el número de posibles combinaciones de entidades que pueden participar en un conjunto de relaciones. Hay dos tipos de restricciones: cardinalidades y restricciones de participación.

Cardinalidad:

- Muchos a muchos (defecto). Por ejemplo, un empleado puede trabajar en varios departamentos (≥0) y un departamento puede tener varios trabajadores.
- Muchos a uno:
- Uno a muchos
- Uno a uno

Elementos del modelado ER

- Decisiones de diseño
  - Modelar un concepto como entidad o como atributo
  - Modelar un concepto como entidad o como relación
  - Identificación de relaciones: binarias o ternarias
  - Uso de la agregación
- Restricciones en el modelo ER
  - Los datos están acompañados de condiciones de validez
  - Algunas de estas restricciones no se pueden capturar en diagramas ER
  - Inclusión de restricciones en notas
  - Lenguajes de restricciones

Modelo relacional

Modelo Relacional: Introducción

Es un modelo sencillo y potente. Una BD es un conjunto de tablas con filas y columnas. Las tablas pueden estar relacionadas unas con otras.

La idea es antigua (Codd, CACM 1970) y es el modelo utilizado más ampliamente. Las razones de su uso se pueden resumir en:

- La representación de los datos es sencilla.
- Es muy fácil formular consultas en este modelo.

Ejemplos de SGBD de estas características:

- Ingres/Postgres, Informix (IBM), Sybase (SAP), MySQL (Oracle), SQL Server (Microsoft), Oracle, DB2 (IBM), …

Modelo Relacional: Partes

Una Base de datos relacional es un conjunto de relaciones.

Una relación tiene dos partes:

- Esquema: nombre de la relación, nombre y tipo de cada columna (atributo, campo)
- Instancia: valores de la tabla (registro, tupla). Todas las tuplas son distintas.

Ejemplo: Películas (nombre de la Tabla) Las columnas son los atributos y Cada fila es una tupla

(En un sistema de fichero sería *campo* para las columnas y *registro* para las filas)

Título	Año	duración	género
Lo que el viento se llevó	1939	231	Drama
Star Wars	1977	124	C. Ficción
El mundo de Wayne	1992	95	Comedia

Modelo Relacional: Definiciones

Las columnas representan atributos (título, año, duración, género). Cada fila representa la información que considera el modelo para cada película. Se llama *tupla*. Una *tupla* tendrá un *componente* (valor) para cada atributo.

El nombre de una relación y el conjunto de atributos de la misma se llama *esquema*. Para la relación Películas el esquema es:

Películas (título, año, duración, género)

Los atributos son un conjunto y no una lista, aunque a veces se habla de orden estándar de los atributos.

El esquema para la relación Películas se puede representar por:

Películas (título: string, año: date, duración: int, género: string)

Se pueden listar los elementos de la relación en cualquier orden.

Modelo Relacional: Presentaciones de una relación

título	año	duración	género
Lo que el viento se llevó	1939	231	Drama
Star Wars	1977	124	C. Ficción
El mundo de Wayne	1992	95	Comedia

Distintas presentaciones de la relación Películas

año	género	título	duración
1939	Drama	Lo que el viento se llevó	231
1977	C. Ficción	Star Wars	124
1992	Comedia	El mundo de Wayne	95

género	título	año	duración
Drama	Lo que el viento se llevó	1939	231
C. Ficción	Star Wars	1977	124
Comedia	El mundo de Wayne	1992	95

Modelo Relacional: Creación de relaciones

- Creación de la relación/tabla, con sus atributos/campos (sentencia SQL:)
- Los tipos/dominios se especifican para cada campo
- El tipado (tipos de datos) es asegurado por el SGBD cuando se añaden o modifican tuplas/registros

```
CREATE TABLE Películas (
   título     CHAR(50),
   año        DATE,
   duración   INTEGER,
   genero     CHAR(20)
)
```

- Es habitual la existencia de diversas tablas que, eventualmente, estarán relacionadas

```
CREATE TABLE Directores (
   título     CHAR(50),
   director   CHAR(50)
)
```

Nota: las tablas indicadas son un ejemplo, no necesariamente la mejor opción.

Modelo Relacional: Destrucción/ modificación

- La destrucción de una relación implica su borrado del esquema, y el borrado de todas sus tuplas

```
DROP TABLE Películas
```

- La modificación de un esquema se puede hacer añadiendo, borrando o modificando campos

- Para campos añadidos, todas las tuplas existentes son extendidas con valor null es ese campo

```
ALTER TABLE Películas
   ADD COLUMN
(nacionalidadCHAR(20)
)
```

Modelo Relacional: Destrucción/ modificación

- Se puede insertar una tupla en la relación

```
INSERT INTO Películas (título, año, duración, género)
VALUES ('El hombre tranquilo',1951,129,'comedia')
```

- El borrado se realiza indicando la condición que cumplirán todas las tuplas a ser borradas

```
DELETE
FROM Películas E
WHERE E.título='Star
Wars'
```

- La modificación se realiza igualmente con una condición de selección de tuplas

```
UPDATE Películas E
SET
E.duración=E.duración+1
WHERE E.título='El
hombre tranquilo'
```

Restricciones de integridad

- RI = condición de validez
  - Condiciones que tienen que cumplirse para cualquier instancia de la base de datos
  - Se especifican cuando se define el esquema
  - Se comprueban cuando se modifican las relaciones (DBMS)
- Instancia legal = satisface todas las RI
- Semántica del mundo real descrito
- Formas de poner restricciones por : Dominio, clave primaria, clave foránea
- También se soportan otras más generales

Claves primarias y candidatas

- Clave (candidata)
  - Identificación única de cada tupla, por medio de un subconjunto mínimo de campos
- Condiciones
  - No existen dos tuplas con los mismos valores en todos los campos de la clave —implicación sobre los *null*
  - Ningún subconjunto de la clave es identificador único —superclave: {nia, nombre}
- Una candidata debe ser elegida como clave primaria (o principal)
  - ¿Quién hace esta elección?

Estudiante (nia,nombre,login,edad,notam)

...

- Las claves candidatas se especifican con `UNIQUE`, y la que es elegida como primaria con `PRIMARY KEY`

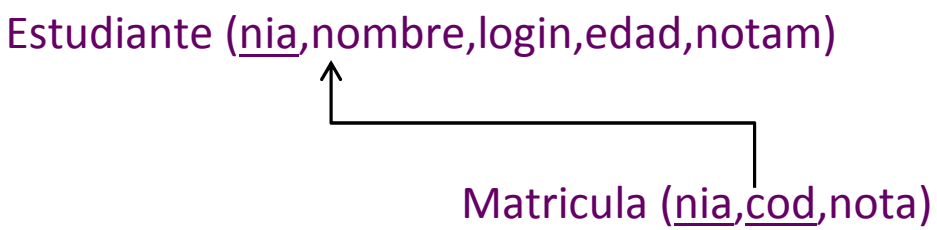
```
CREATE TABLE Matricula (
   nia   INTEGER,
   cod   CHAR(20),
   nota  REAL,
   PRIMARY KEY (nia,cod)
)
```

```
CREATE TABLE Estudiante (
   nia     INTEGER,
   nombre  CHAR(20),
   login   CHAR(10),
   edad    INTEGER,
   notam   REAL,
   UNIQUE (nombre,edad),
   CONSTRAINT claveEst PRIMARY KEY (nia)
)
```

nombrado de la restricción, lo que permite identificar errores

Claves foráneas; integridad ref.

- Clave foránea (o externa)
  - Conjunto de campos en una relación que sirven para *referenciar* tuplas en otra relación —puntero de asociación
  - La referenciada debe ser clave primaria, para asegurar que se hace referencia a una única tupla



- Integridad referencial
  - Imposición de las restricciones referenciales: sólo los estudiantes listados en Estudiante son admitidos en la Matrícula de cursos

...

```
CREATE TABLE Matricula (
   nia   INTEGER,
   cod   CHAR(20),
   nota  REAL,
   PRIMARY KEY (nia,cod),
   FOREIGN KEY (nia) REFERENCES Estudiante
) :
```
La clave foránea (nia) cuya referencia está en la tabla Estudiante

- Una clave foránea puede hacer referencia a la misma relación en la que se encuentra
- El *null* —desconocido/no-aplicable— cumple la restricción de clave foránea, pero no la de clave primaria

Cumplimiento de la integridad ref.

- Inserción con referencia no existente
  - `NO ACTION`: rechazar la inserción de una matrícula con *nia* no existente en la tabla de estudiantes
- Borrado de una tupla referenciada
  - (default) `NO ACTION`: rechazar el borrado de un estudiante que tiene entradas en matrícula
  - `CASCADE`: borrar el estudiante, y todas las matrículas que le referencian
    (pe borrar al director: borra todas sus peliculas)
  - `SET DEFAULT`: asignar esas matrículas a un *nia* por defecto
  - (en SQL) `SET NULL`: asignar el *nia* de esas matrículas al valor especial *null* —conflicto con PK
- Actualización de la clave primaria
  - Mismo funcionamiento

...

```
CREATE TABLE Matricula (
   nia   INTEGER,
   cod   CHAR(20),
   nota  REAL,
   PRIMARY KEY (nia,cod),
   FOREIGN KEY (nia) REFERENCES Estudiante
       ON DELETE CASCADE
       ON UPDATE SET DEFAULT
)
```

- La acción 'cascade' se propaga *en cascada* por siguientes relaciones que referencian a la tupla borrada o actualizada
- ¿Se puede utilizar en una relación que se autoreferencia?
- ¿Podríamos *diferir* una comprobación?

Otras restricciones

- Restricciones de columna
  - NOT NULL: no admite nulos
  - CHECK: condición de integridad

```
CREATE TABLE Estudiante (
   ...
   notam   REAL not null,
   ...
   CONSTRAINT notampos CHECK (notam>=0)
)
```

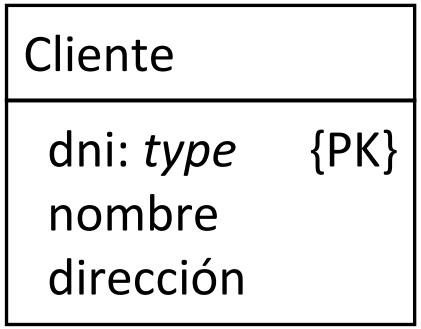
- Aserciones
- Disparadores (nivel superior)

necesitaremos conocer más sobre la potencia de las consultas en SQL

Diseño lógico: ER -> Relacional

- ERD

(lenguaje conceptual)



- ER (lenguaje intermedio)

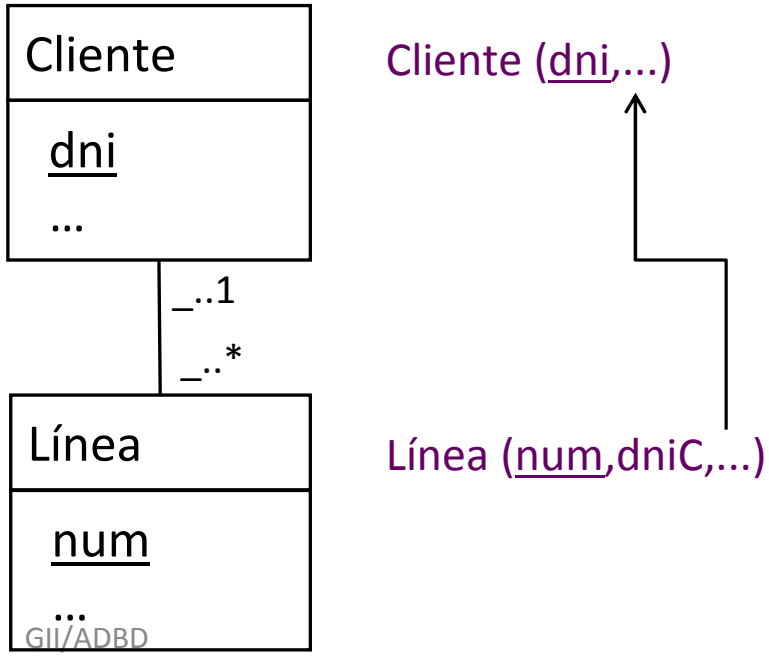
Cliente (dni,nombre,dirección)

- SQL (lenguaje del a máquina

```
CREATE TABLE Cliente (
   dni         CHAR(10),
   nombre      CHAR(20),
   dirección   CHAR(50),
   PRIMARY KEY (dni))
```

Tipos de relación -> tablas

- Cardinalidad máxima: 1—1
  - Comprobamos si se pueden poner en términos de una única entidad
- 1—*
  - Referenciamos en el lado *, el valor (único) del lado 1



Cliente (dni,...)

Línea (num,dniC,...)

```
CREATE TABLE Cliente (
   dni      CHAR(10), ...,
   PRIMARY KEY (dni))

CREATE TABLE Linea (
   num      INTEGER,
   dniC     CHAR(10), ...,
   PRIMARY KEY (num)
   FOREIGN KEY (dniC)
     REFERENCES Cliente)
```

...

Empleado_Cont (dni,nombre,dirección,nrp,fechaIni,FK1)

Empleado_Exte (dni,nombre,dirección,horasSem,duración,FK1)

- ¿es un problema la repetición de lo común?
- ¿qué ocurre si llega una relación *—1 a empleado?
- ¿y si quiero saber el número total de empleados?

Empleado (dni,nombre,dirección,FK1)

Empleado_Cont (dni,nrp,fechaIni)

Empleado_Exte (dni,horasSem,duración)

- ¿es un problema la repetición del *dni*?
- ¿qué ocurre si quiero saber el nombre de un *nrp*?

Vistas y seguridad

- Definición de la vista
  - Tabla que no almacena tuplas, las cuales se obtienen a partir de una *definición* sobre tablas base
  - Permite restructurar el esquema lógico base, e implementar políticas de acceso restringido a los datos

```
CREATE VIEW EstudianteMH (nia,notam)
   AS SELECT E.nia,E.notam
      FROM Estudiantes E
      WHERE E.notam >= 9.0
```

dando acceso a través de la vista, no se pueden ver los nombres, login o edades

no se puede acceder al resto de alumnos

Modelo Relacional: Ejemplo

Relaciones correspondiente a los datos climatológicos de Segovia. La parte de año es un resultado elaborado no datos iniciales.

Mes	T	TM	Tm	R	H	DR	DN	DT	DF	DH	DD	I
Enero	4.3	8.2	0.3	38	74	6.9	-	0.0	3.9	14.5	4.5	124
Febrero	5.8	10.4	1.1	31	66	6.0	3.1	0.0	2.0	10.0	4.8	152
Marzo	8.6	13.9	3.2	30	59	5.9	1.7	0.1	1.3	6.1	5.4	203
Abril	9.7	15.1	4.2	44	59	8.0	1.5	0.8	0.6	4.2	3.4	213
Mayo	14.0	19.7	8.2	66	57	10.1	0.2	3.7	0.7	0.5	3.3	250
Junio	19.0	25.8	12.1	43	48	4.8	0.0	4.2	0.6	0.0	6.9	314
Julio	22.2	29.7	14.6	17	39	2.7	0.0	2.6	0.2	0.0	12.5	358
Agosto	22.1	29.4	14.8	20	40	3.0	0.0	3.8	0.1	0.0	10.4	328
Septiembre	17.7	24.0	11.4	28	50	4.7	0.0	2.1	0.4	0.0	5.8	246
Octubre	13.0	18.0	7.9	59	63	8.9	0.1	0.6	1.0	0.7	4.0	177
Noviembre	7.6	11.8	3.4	52	72	8.6	1.6	0.0	2.7	5.8	3.9	126
Diciembre	5.1	8.8	1.3	46	75	8.2	2.2	0.0	4.1	11.7	3.6	110
Año	**12.4**	**17.9**	**6.9**	**479**	**59**	**78.6**	**-**	**18.1**	**17.6**	**53.1**	**-**	**-**

T	Temperatura media mensual/anual (°C)
TM	Media mensual/anual de las temperaturas máximas diarias (°C)
Tm	Media mensual/anual de las temperaturas mínimas diarias (°C)
R	Precipitación mensual/anual media (mm)
H	Humedad relativa media (%)
DR	Número medio mensual/anual de días de precipitación superior o igual a 1 mm
DN	Número medio mensual/anual de días de nieve
DT	Número medio mensual/anual de días de tormenta
DF	Número medio mensual/anual de días de niebla
DH	Número medio mensual/anual de días de helada
DD	Número medio mensual/anual de días despejados
I	Número medio mensual/anual de horas de sol

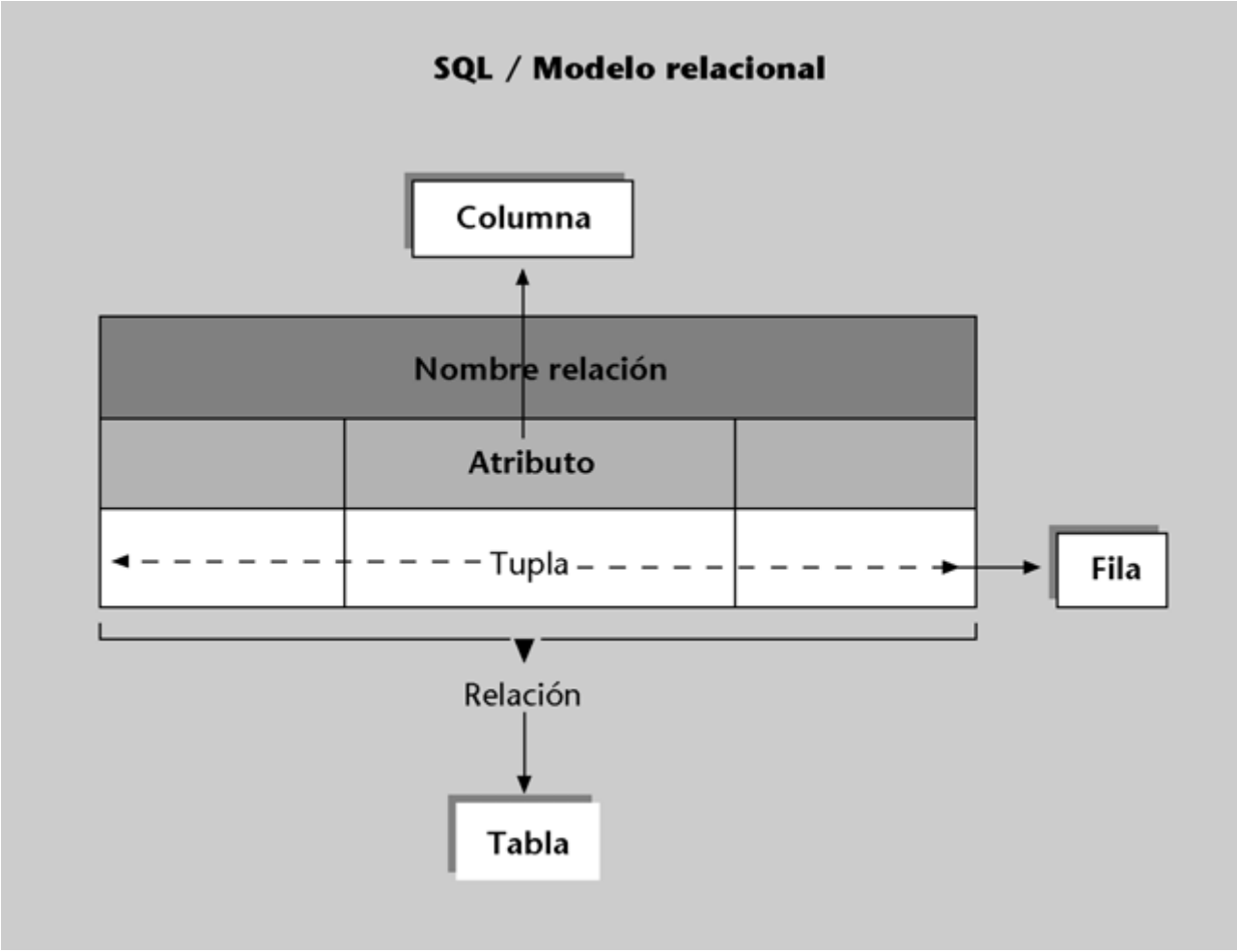
SQL Structured Query Language

Structured Query Language

» lenguaje relacional diseñado para la gestión de datos

- Propuesta
  - uno de los primeros lenguajes comerciales
  - el más utilizado en el uso de las bases de datos (aunque existen otras propuestas)
- Estandarización
  - SQL-86 (primera ANSI), SQL-92 (revisión importante), SQL-99 (exp.reg., recursión, triggers, ...), SQL-2003 (XML, ...), SQL-2008 (...)
  - los sistemas comerciales tienen, al menos, SQL-92
  - incluyen más elementos, algunos específicos
  - los específicos van en contra de la portabilidad, aunque pueden ser muy útiles

SQL – modelo relacional

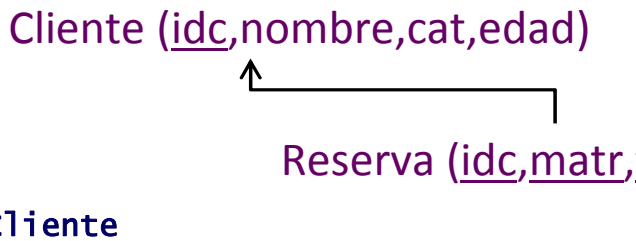


Structured Query Language

El lenguaje de definición de datos (DDL) de SQL permite la especificación de información sobre relaciones, incluyendo:

- El esquema de cada relación.
- El dominio de valores asociados con cada atributo.
- Restricciones de integridad.
- Otra información como:
  - El conjunto de índices que serán mantenidos por cada una de las relaciones.
  - Información de seguridad y autorización para cada relación.
  - La estructura de almacenamiento físico de cada relación en disco.

Datos ejemplo



Cliente

idc	nombre	cat	edad
22	Davila	7	45
29	Bravo	1	32
31	Lorenzo	8	24
32	Alvarez	8	31
58	Rubio	4	67
64	Huerga	2	18
71	Zurro	10	45
74	Huerga	9	35
85	Arnaud	3	25
95	Benitez	3	63

Vehiculo

matr	marca	color
101	BMW	azul
102	Lancia	rojo
103	Seat	verde
104	BMW	rojo

Reserva

idc	matr	fecha
22	101	2009-11-07
64	101	2010-12-28
22	102	2010-04-21
31	102	2012-01-14
64	102	2010-04-11
22	103	2009-11-07
31	103	2011-07-19
74	103	2009-09-13
22	104	2012-01-01
31	104	2006-12-09

Estrategia de evaluación

- Obtención de resultados
  1. producto cartesiano de las tablas
  2. eliminar las filas que no cumplen la condición
  3. eliminar los atributos que no están en la lista
  4. eliminar duplicados si es necesario
- Eficiencia – optimizador
  - estrategia menos eficiente para computar la consulta (obtener los resultados)
  - el optimizador encontrará estrategias más eficientes (que devuelven los mismos resultados)

Consulta básica

```
SELECT [DISTINCT] lista-atributos
FROM lista-tablas
WHERE condición
```

- Lista tablas
  - tablas involucradas en los datos objeto de la búsqueda
  - posiblemente con un nombre de *correlación* cada una
- Lista atributos
  - atributos de la lista de tablas que son de interés
- Condición
  - expresiones (booleanas) sobre los datos que deben formar parte del resultado de la consulta
- `[DISTINCT]` es opcional para no incluir duplicados; por defecto no son eliminados

Join de tablas —implícito

```
SELECT C.nombre
FROM Cliente C, Reserva R
WHERE C.idc=R.idc AND R.matr=103;
```

- Nombre correlación / variable de rango
  - sólo son realmente necesarios cuando una relación aparece dos veces en la claúsula `FROM`
  - se recomienda utilizarlos siempre

```
SELECT nombre
FROM Cliente, Reserva
WHERE Cliente.idc=Reserva.idc AND matr=103;
```

```
+-----------+
| nombre    |
+-----------+
|  Davila   |
|  Lorenzo  |
|  Huerga   |
+-----------+
```

- Condición de *join*
  - p.e., igualdad de los campos de combinación

...

» Clientes que han reservado al menos un coche

```
SELECT C.nombre
FROM Cliente C, Reserva R
WHERE C.idc=R.idc;
```

```
+-----------+
| nombre    |
+-----------+
|  Davila   |
|  Davila   |
|  Davila   |
|  Davila   |
|  Lorenzo  |
|  Lorenzo  |
|  Lorenzo  |
|  Huerga   |
|  Huerga   |
|  Huerga   |
+-----------+
```

```
+-----------+
| nombre    |
+-----------+
|  Davila   |
|  Lorenzo  |
|  Huerga   |
+-----------+
```

②

```
+-----+
| idc |
+-----+
|  22 |
|  31 |
|  64 |
|  74 |
+-----+
```

③

```
+-----+-----------+
| idc | nombre    |
+-----+-----------+
|  22 |  Davila   |
|  31 |  Lorenzo  |
|  64 |  Huerga   |
|  74 |  Huerga   |
+-----+-----------+
```

④

- habrá que poner `DISTINCT` ②
- mejor por idc's distintos ③
  (¿interviene Cliente?)
- podemos añadir el nombre ④

```
SELECT DISTINCT C.idc, C.nombre
...
```

Referencias

T. Connolly, C. Begg. *Sistemas de bases de datos*. 5a edición. Addison-Wesley. 2010. Existe una edición de 2015.

A.Silberschatz, H. F. Korth, S. Sudarshan. *Fundamentos de bases de datos. 6a edición*. McGraw-Hill. 2010. Material en: http://codex.cs.yale.edu/avi/db-book/db6/slide-dir/

R. Elmasri, S. Navathe. *Fundamentos de sistemas de bases de datos*. 5a edición. Addison-Wesley. 2007

Manuel Barrio. Apuntes clase Base de Datos. (colegui de Pablo)