Análisis y Diseño de Bases de Datos

Introducción

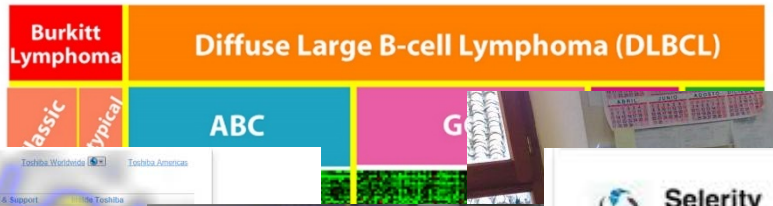
Datos

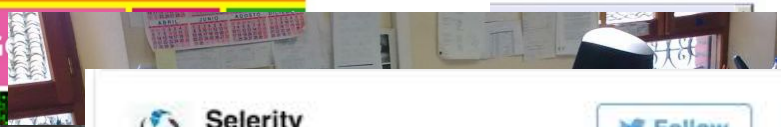
- Representación o codificación de algún tipo de información o conocimiento
  - Datos: valores que se guardan y analizan
  - Información: datos analizados que sirven para tomar decisiones (interpretación)
  - Conocimiento: compresión de la información en base a experiencia sobre una materia
- Dato: registro de alguna acción
  - Cualquier cosa puede registrarse
  - Distintos tipos de datos

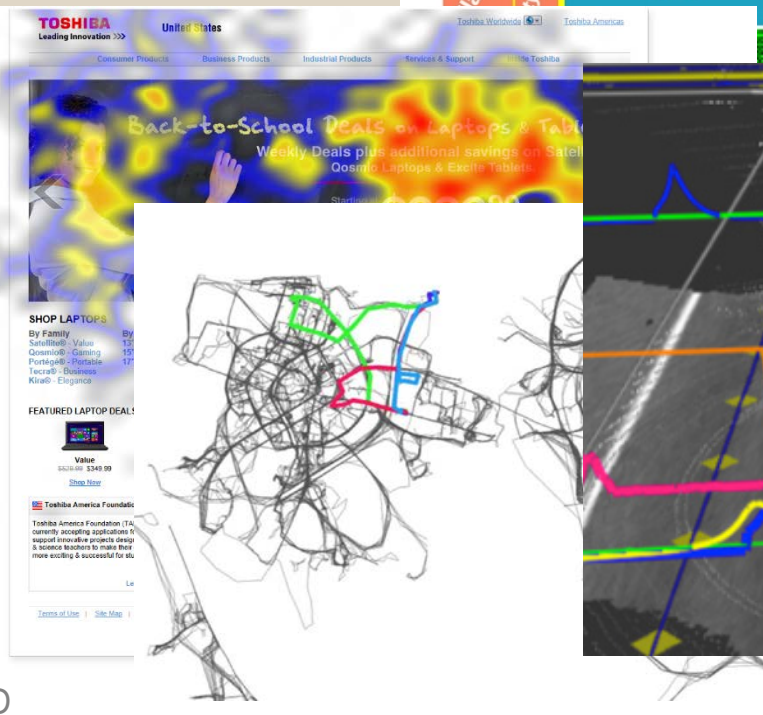
...

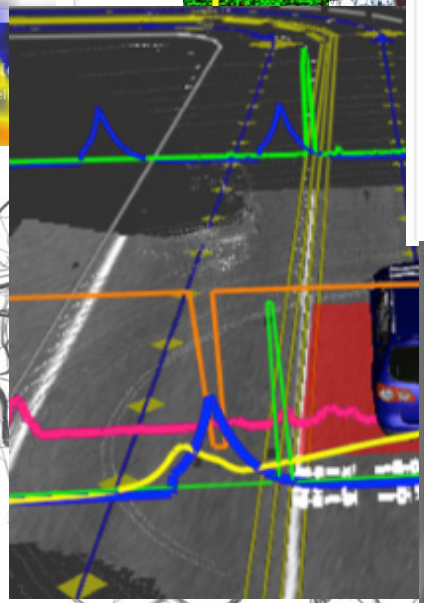
Table 7 Top 10 cancer incidence rates in cancer registration areas in 2009

Rank	Both					
	Site	Incidence rate ($1/10^5$)	%	ASIRC* ($1/10^5$)	Site	
1	Lung (C33-34)	53.57	18.74	25.34	Lung (C33-34	
2	Stomach (C16)	36.21	12.67	17.85	Stomach (C16	
3	Colon, rectum	29.44	10.30	14.21	Liver (C22)	









Preguntas clave

- Qué queremos que haga la BD con estos datos en términos de guardarlos y analizarlos?
  - solo nos interesan los datos *persistentes* en memoria secundaria
  - guardarlos puede ser relativamente estándar (*ficheros binarios*), pero analizarlos es tan variado como los propios datos que tratemos
- Se puede construir un sistema de base de datos que pueda hacer esto para cualquier tipo de dato que nos encontremos?
  - no existe un único sistema que pueda gestionar cualquier cosa

Por dónde empezamos?

- Por los datos más *fáciles*
  - hay muchos de estos? sí, pero cada vez más de los complejos —*revolución big data*
  - esto valdrá de algo cuando queramos tratar los complejos? sí, pero con limitaciones

Table 7 Top 10 cancer incidence rates in cancer registration areas in 2009

Rank	Both				Male				Female			
	Site	Incidence rate ($1/10^5$)	%	ASIRC* ($1/10^5$)	Site	Incidence rate ($1/10^5$)	%	ASIRC* ($1/10^5$)	Site	Incidence rate ($1/10^5$)	%	ASIRC* ($1/10^5$)
1	Lung (C33-34)	53.57	18.74	25.34	Lung (C33-34)	70.40	22.14	34.75	Breast (C50)	42.55	16.81	23.16
2	Stomach (C16)	36.21	12.67	17.85	Stomach (C16)	49.61	15.60	25.37	Lung (C33-34)	36.34	14.36	16.41
3	Colon, rectum (C18-21)	29.44	10.30	14.21	Liver (C22)	41.99	13.21	22.49	Colon, rectum (C18-21)	26.42	10.44	12.29
4	Liver (C22)	28.71	10.04	14.78	Colon, rectum (C18-21)	32.38	10.18	16.23	Stomach (C16)	22.50	8.89	10.62
5	Esophagus (C15)	22.14	7.74	10.88	Esophagus (C15)	30.44	9.57	15.62	Liver (C22)	15.11	5.97	7.11
6	Breast (C50)	21.21	7.42	11.64	Prostate (C61)	9.92	3.12	4.34	Esophagus (C15)	13.64	5.39	6.27
7	Pancreas (C25)	7.28	2.55	3.35	Bladder (C67)	9.78	3.08	4.70	Cervix (C53)	12.96	5.12	7.42
8	Lymphoma (C81-85, 88, 90, 96)	6.68	2.34	3.75	Pancreas (C25)	8.24	2.59	4.01	Thyroid (C73)	10.09	3.99	6.50
9	Bladder (C67)	6.61	2.31	3.03	Lymphoma	7.71	2.42	4.46	Uterus (C54-55)	8.77	3.46	4.69
10	Thyroid (C73)	6.56	2.29	4.21	Kidney (C64-66, 68)	7.07	2.22	3.82	Ovary (C56)	7.95	3.14	4.54
	Top 10	218.40	76.39	109.05	Top 10	267.55	84.14	135.81	Top 10	196.32	77.57	99.01

*Age-standardized incidence rate (China population)

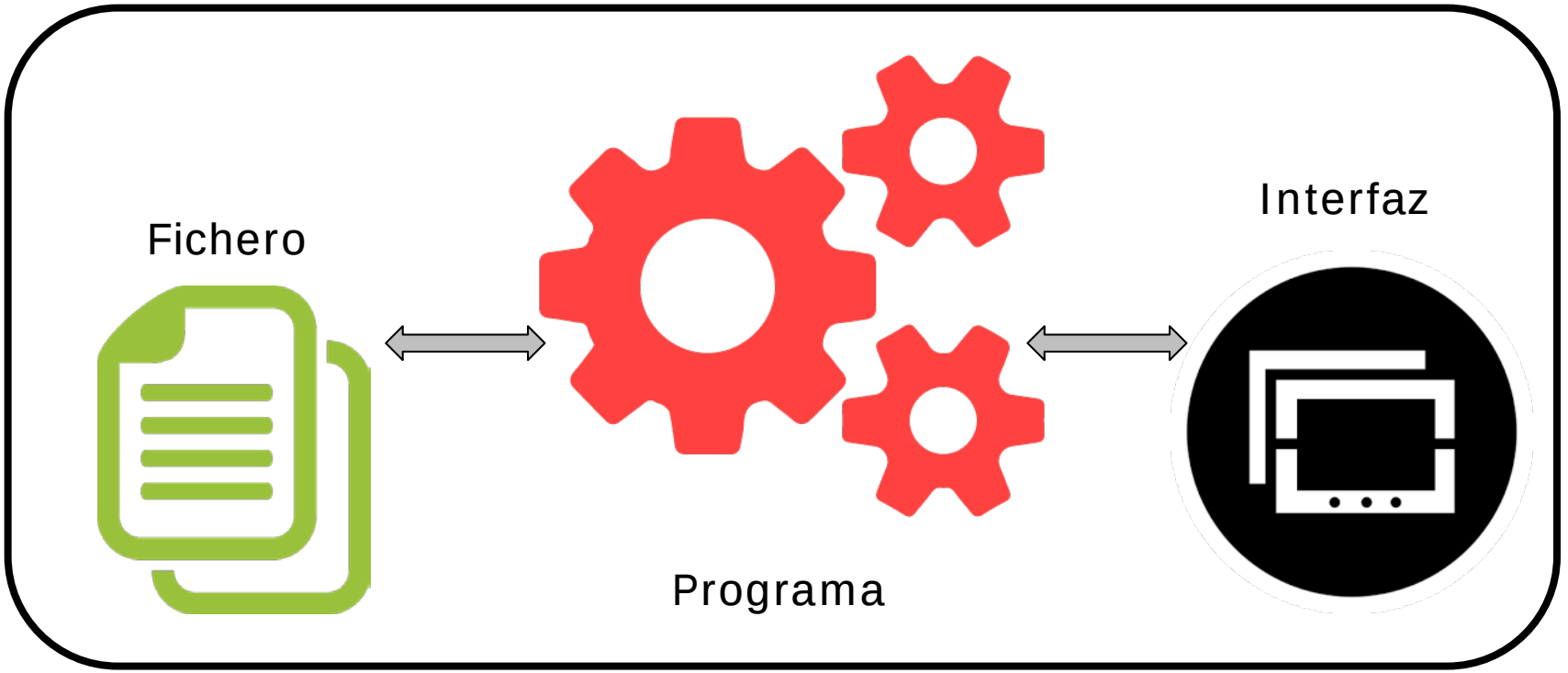
...

- Fáciles = tabla = *estructurados*
  - guardar: estructuras de datos orientadas a registros con campos, etc.
  - analizar: recuperar los datos, incluyendo condiciones lógicas, ordenaciones, agrupaciones, operaciones aritméticas

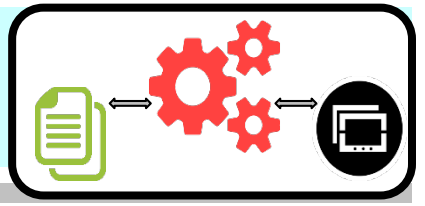
  $\Rightarrow$ sistema de base de datos general que pueda tratar (*casi*) cualquier tipo de datos estructurado, acompañado de un lenguaje de consulta

Sin BDs ...

- fichero con los datos; programa con acceso y lógica sobre los datos; interfaz de usuario



...



- Problemas
  - tipo de fichero: secuencial, tabla?
    eficiencia de las búsquedas?
  - programa con acceso a fichero: abrir/cerrar,
    acceso a bloques; acceso a índices
    si cambio tipo de fichero, tengo que reprogramar
  - soporta accesos concurrentes?
  - como implemento la seguridad/privacidad/etc?
  - qué ocurre con los datos erróneos?
  - y cuando hay un fichero y varios programas?

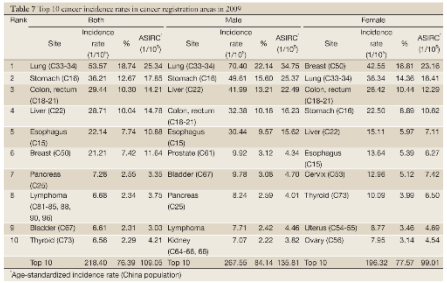
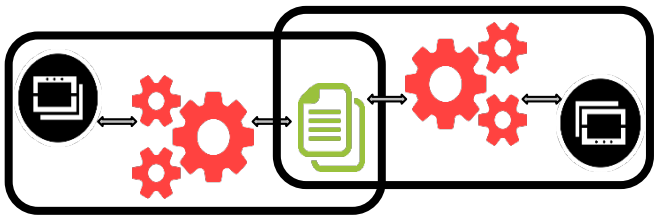


Table 7 Top 10 cancer incidence rates in cancer registration areas in 2009

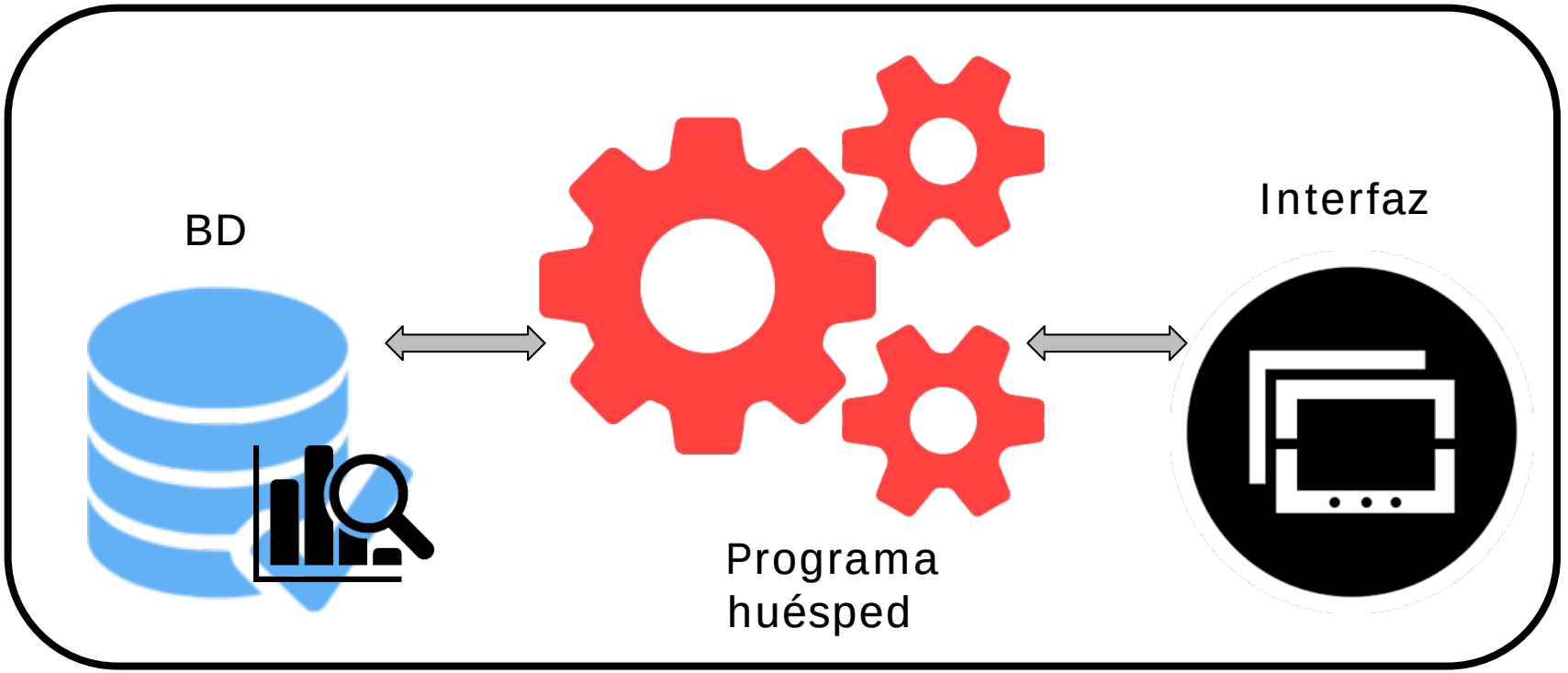
Rank	Both				Male				Female			
	Site	Incidence rate ($1/10^5$)	%	ASIRC ($1/10^5$)	Site	Incidence rate ($1/10^5$)	%	ASIRC ($1/10^5$)	Site	Incidence rate ($1/10^5$)	%	ASIRC ($1/10^5$)
1	Lung (C33-34)	53.57	18.74	25.34	Lung (C33-34)	70.40	22.14	34.75	Breast (C50)	42.55	16.81	23.16
2	Stomach (C16)	36.21	12.67	17.85	Stomach (C16)	49.61	15.60	25.37	Lung (C33-34)	36.34	14.36	16.41
3	Colon, rectum (C18-21)	29.44	10.30	14.21	Liver (C22)	41.99	13.21	22.49	Colon, rectum (C18-21)	26.42	10.44	12.29
4	Liver (C22)	28.71	10.04	14.78	Colon, rectum (C18-21)	32.38	10.18	16.23	Stomach (C16)	22.50	8.89	10.82
5	Esophagus (C15)	22.14	7.74	10.88	Esophagus (C15)	30.44	9.57	15.62	Liver (C22)	15.11	5.97	7.11
6	Breast (C50)	21.21	7.42	11.64	Prostate (C61)	9.92	3.12	4.34	Esophagus (C15)	13.64	5.39	6.27
7	Pancreas (C25)	7.26	2.55	3.39	Bladder (C67)	9.78	3.08	4.70	Cervix (C53)	12.96	5.12	7.42
8	Lymphoma (C81-85, 88, 90, 96)	6.68	2.34	3.73	Pancreas (C25)	8.24	2.59	4.01	Thyroid (C73)	10.09	3.99	6.50
9	Bladder (C67)	6.61	2.31	3.03	Lymphoma	7.71	2.42	4.46	Uterus (C54-55)	8.77	3.46	4.80
10	Thyroid (C73)	6.56	2.29	4.21	Kidney (C64-66, 68)	7.07	2.22	3.82	Ovary (C56)	7.55	3.14	4.54
	Top 10	218.40	76.39	109.05	Top 10	267.55	84.14	136.81	Top 10	196.32	77.57	99.01

Age-standardized incidence rate (China population)



Con BDs ...

- No es un entorno general de programación / computación



Base de datos, BD

- BD: colección organizada de datos, que
  - modela aspectos relevantes de la realidad, ej.: clientes, llamadas, tarifas
  - da soporte a procesos de información: ej.: 10% clientes con menor ratio fijo/móvil
- SGBD: sistema gestor de BD
  - sistema software que gestiona los datos
- Gestión: almacenamiento, extracción, modificación, borrado, búsqueda, seguridad, integridad, compartición, ...

Nivel lógico

PATIENTS						
YPL-320	SMITH	m	38	176	1	124
GLI-532	JOHNSON	m	43	163	0	109
PNI-258	WILLIAMS	f	38	131	0	122
MIJ-579	JONES	f	40	133	0	117
XLK-030	BROWN	f	49	119	0	122
TFP-518	DAVIS	f	46	142	0	121
LPD-746	MILLER	f	33	142	1	130
ATA-945	WILSON	m	40	180	0	115

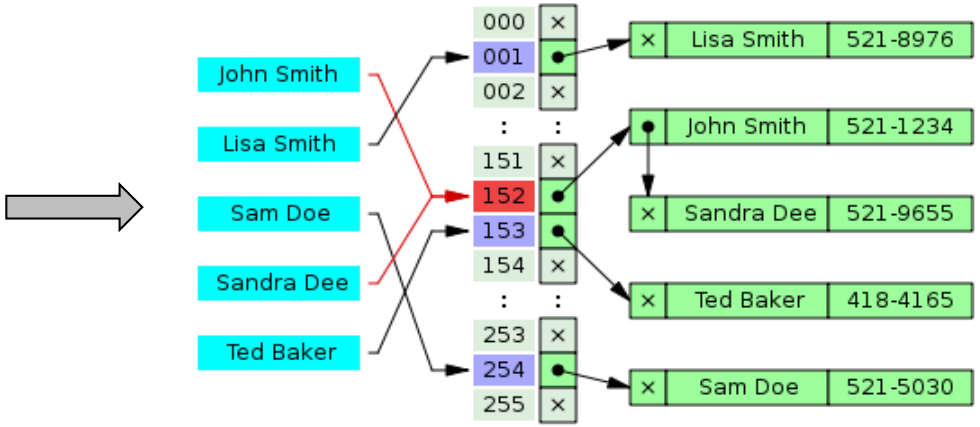
definimos, representamos y manipulamos datos como si fueran tablas —visión 'usuario'— aunque internamente/físicamente la BD los gestiona de otra forma

- la *traducción* al esquema interno la hace la BD → ventajas !!
- comprensible por agrupación de columnas
- estructura de tabla = entidades, propiedades y relaciones
- la línea 2 es especial = esquema de los datos
- problemas para almacenar otros datos en esa misma tabla

Nivel físico

PATIENTS

id	name	sex	age	wgt	smoke	sys
YPL-320	SMITH	m	38	176	1	124
GLI-532	JOHNSON	m	43	163	0	109
PNI-258	WILLIAMS	f	38	131	0	122
MIJ-579	JONES	f	40	133	0	117
XLK-030	BROWN	f	49	119	0	122
TFP-518	DAVIS	f	46	142	0	121
LPD-746	MILLER	f	33	142	1	130
ATA-945	WILSON	m	40	180	0	115



- estructuras de datos en memoria secundaria: bloques, índices, etc; métodos de acceso y coste asociado
- independencia lógico-física: automatizado por BD; se puede cambiar el diseño físico y mantener nuestros 'programas'

Usuario (N. Lógico)

1. abrir tabla
2. seleccionar sys>120 & smoke
3. recuperar solo id,name
4. adjuntar conteo y promedio(sys)

BD (N. físico)

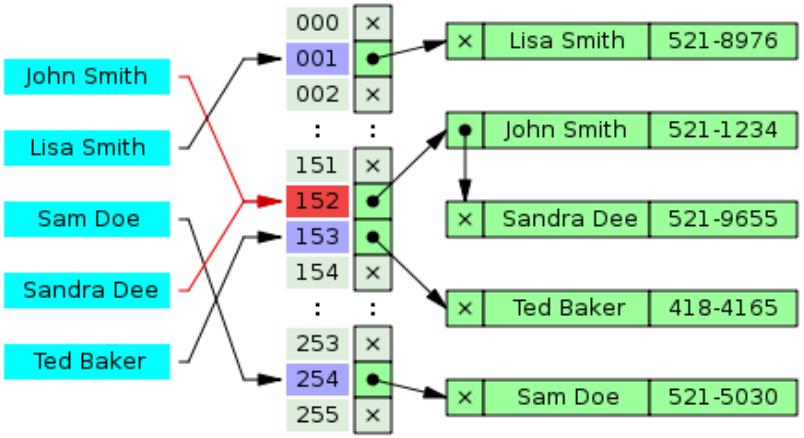
1. abrir archivo
2. abrir índice, recorrer y comprobar registros sys, smoke
3. recuperar bloques y registros id, name
4. hacer cálculos y formatear salida

INDEPENDENCIA LÓGICO-FÍSICA

Essentials

PATIENTS

id	name	sex	age	wgt	smoke	sys
YPL-320	SMITH	m	38	176	1	124
GLI-532	JOHNSON	m	43	163	0	109
PNI-258	WILLIAMS	f	38	131	0	122
MIJ-579	JONES	f	40	133	0	117
XLK-030	BROWN	f	49	119	0	122
TFP-518	DAVIS	f	46	142	0	121
LPD-746	MILLER	f	33	142	1	130
ATA-945	WILSON	m	40	180	0	115



- traducción automatizada por la BD
- se puede cambiar el diseño físico, manteniendo los programas de usuario

Nivel almacenamiento

- almacenamiento físico para persistencia
- la base de datos hará las peticiones de acceso al sistema operativo

Estructura vs Complejidad

- añadir diversas mediciones sys

sys
124, 129, 117, …

grupo repetitivo: nº indeterminado de valores en una celda

—las BD relacionales no lo permiten por eficiencia

YPL-320	SMITH	…	124
YPL-320	SMITH	…	129
YPL-320	SMITH	…	117

repetición de datos e inconsistencia:

YPL-320	SMITH	…
YPL-320	SMITZ	…

- añadir doctor(es) responsable(s) —nombre,especialidad—

YPL-320	SMITH	…	DrX,Alerg
PNI-258	WILLIAMS	…	DrX,Alerg
TFP-518	DAVIS	…	DrX,Card

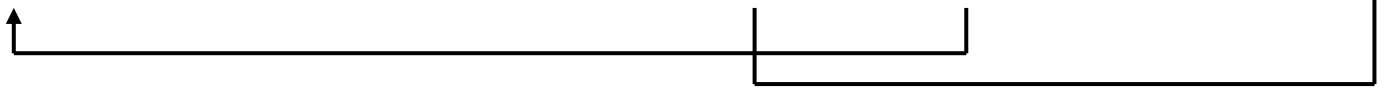
repeticiones, inconsistencia

- Utilizar varias tablas

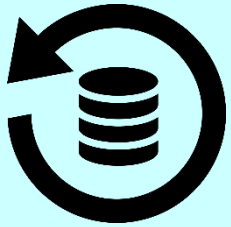
PATIENTS						
id	name	sex	age	wgt	smoke	dr
YPL-320	SMITH	m	38	176	1	DrX
GLI-532	JOHNSON	m	43	163	0	DrY
PNI-258	WILLIAMS	f	38	131	0	DrX

SYS	
pat	val
YPL-320	124
YPL-320	129
PNI-258	157

DOCTOR	
name	special
DrX	Alerg
DrY	Card



GRUPO REPETITIVO

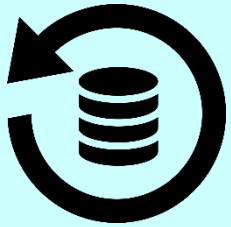


Essentials

- Número indeterminado de valores en un atributo de una entidad

id	sys
GLI-532	124, 129, 117, ...

INCONSISTENCIA



Essentials

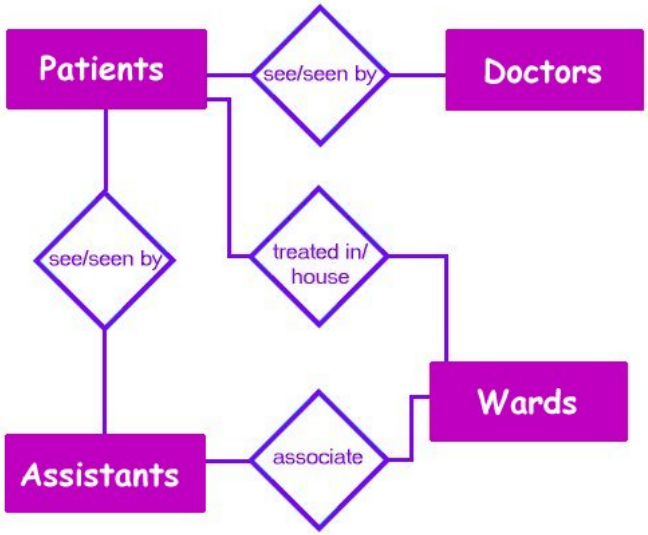
- Información repetida que da lugar a no coincidencia

id	name
YPL-320	SMITH
YPL-320	SMITZ

Nivel conceptual

PATIENTS

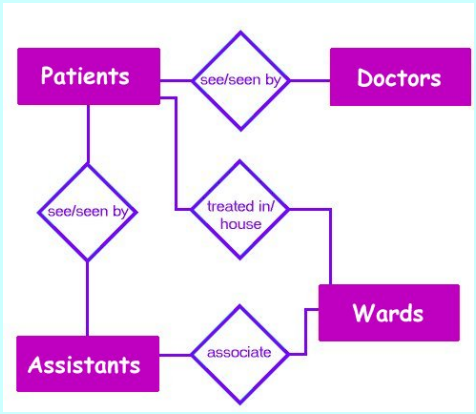
id	name	sex	age	wgt	smoke	sys
YPL-320	SMITH	m	38	176	1	124
GLI-532	JOHNSON	m	43	163	0	109
PNI-258	WILLIAMS	f	38	131	0	122
MIJ-579	JONES	f	40	133	0	117
XLK-030	BROWN	f	49	119	0	122
TFP-518	DAVIS	f	46	142	0	121
LPD-746	MILLER	f	33	142	1	130
ATA-945	WILSON	m	40	180	0	115



- no necesitamos ver las *instancias* de los datos
- modelado de la estructura de los datos
- visión global de alto nivel, basada en entidades, relaciones y propiedades
- o en clasificadores, asociaciones y atributos
- aclaración sobre los modelos/esquemas/diseños conceptuales y lógicos

NIVELES / DISEÑOS

Essentials

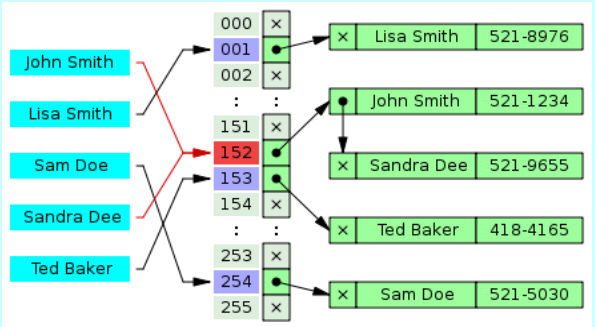


Conceptual

PATIENTS

id	name	sex	age	wgt	smoke	sys
YPL-320	SMITH	m	38	176	1	124
GLI-532	JOHNSON	m	43	163	0	109
PNI-258	WILLIAMS	f	38	131	0	122
MIJ-579	JONES	f	40	133	0	117
XLK-030	BROWN	f	49	119	0	122
TFP-518	DAVIS	f	46	142	0	121
LPD-746	MILLER	f	33	142	1	130
ATA-945	WILSON	m	40	180	0	115

Lógico



Físico

Almacenamiento

Funcionalidades adicionales

1. Integridad (corrección?) y seguridad de los datos
   - el SGBD asegura las restricciones de integridad, en vez de los múltiples programas de acceso
   - controles de acceso a datos visibles para cada usuario
2. Centralización de los datos
   - minimizar la redundancia; evitar inconsistencias
3. Acceso concurrente y recuperación
4. Reducción del tiempo de desarrollo y mantenimiento de aplicaciones

Control de datos incorrectos

- Erróneo vs incorrecto

dni	nombre	fdn
147	ADELA	24/06/2000

en realidad es erróneo, Adela nació el 24/07/2000

—pero es correcto (para la BD)

dni	nombre	fdn
147	ADELA	24/07/3000

ahora ya es incorrecto porque la BD puede comprobarlo (fecha>actual)

—esto se denomina integridad

- Integridad implícita
  1. Dominio

sensor	temp (K)
S089	-52

  2. Identificación

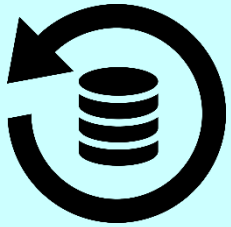
dni	nombre	fdn
147	ADELA	24/06/2000
147	TOMAS	02/12/1999

  3. Referencial

PATIENTS		
id	name	...
YPL-320	SMITH	...
GLI-532	JOHNSON	...
PNI-258	WILLIAMS	...

SYS	
pat	val
YPL-320	124
YPL-320	129
BCT-330	157

INTEGRIDAD



Essentials

- Corrección de datos comprobable automáticamente por la BD

dni	nombre	fdn
147	Adela	24/07/3000

Niveles de abstracción

- Esquema externo
  - *vista* (tabla) que combina datos para una presentación específica de usuario
  - se define sobre el conceptual
- Esquema conceptual
  - incluye todas las tablas/relaciones con información sobre entidades y relaciones
- Esquema interno
  - esquema físico con los detalles de almacenamiento

Lenguajes de consulta

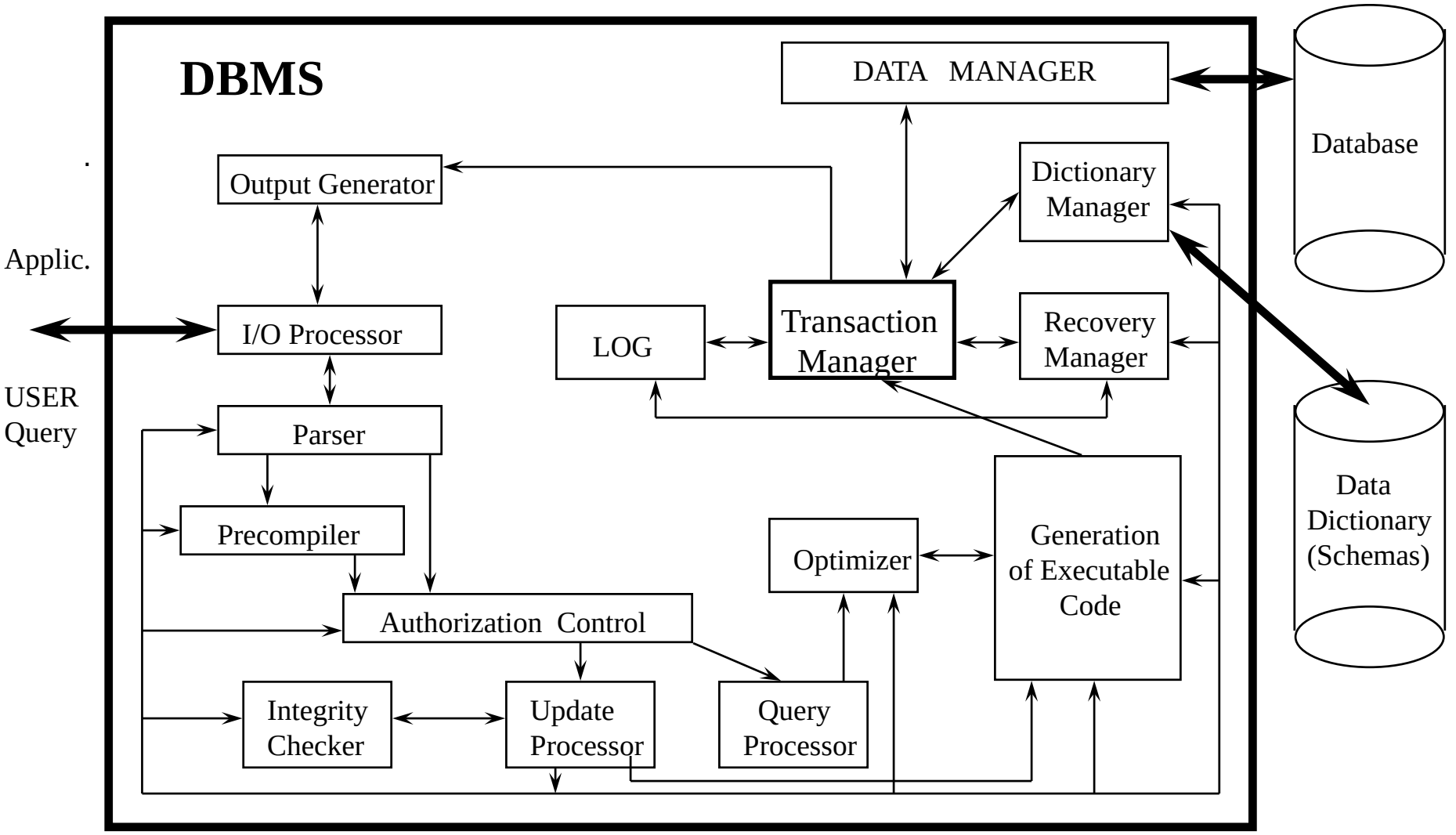
» formulación de consultas a través de un lenguaje

- Definición de esquemas (DDL), manipulación de datos (DML), control
- Formalización a través de cálculo/álgebra relacional (utilidad?)
- Optimización de la eficiencia
- SQL estándar
- Lenguaje anfitrión

Arquitectura de un SGBD

- Bloques que desarrollan las funciones encargadas al SGBD
  - interacción con usuario/aplicación y almacenamiento secundario
  - secuencia operativa
  - interdependencia
  - configuración para mejorar eficiencia
  - trabajo del administrador de la BD

...


DBMS
DATA MANAGER
Database
Output Generator
Dictionary Manager
Applic.
I/O Processor
LOG
Transaction Manager
Recovery Manager
USER Query
Parser
Data Dictionary (Schemas)
Precompiler
Optimizer
Generation of Executable Code
Authorization Control
Integrity Checker
Update Processor
Query Processor

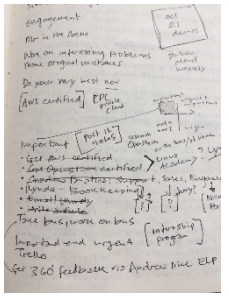
Diseño de Bases de Datos

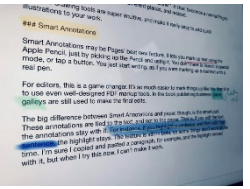
Introducción al Diseño de Bases de Datos

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
    - necesidades del cliente: reuniones, discusiones, documentaciones, ...
    - *comprender* los datos a gestionar
    - 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*—

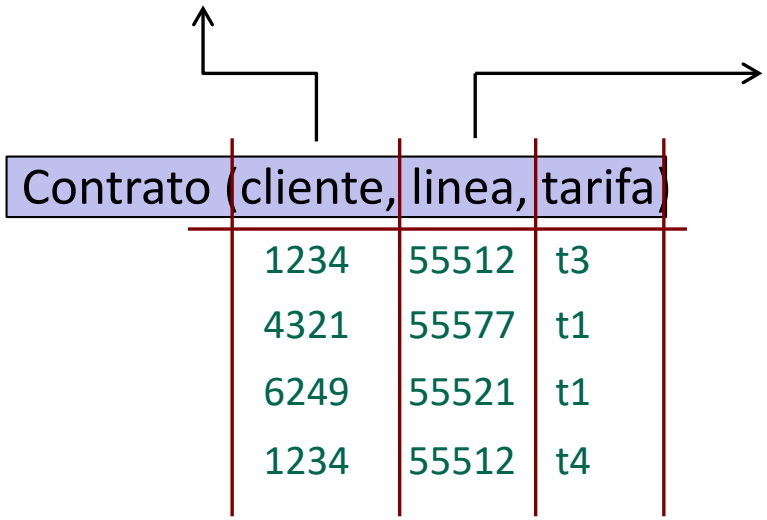
...

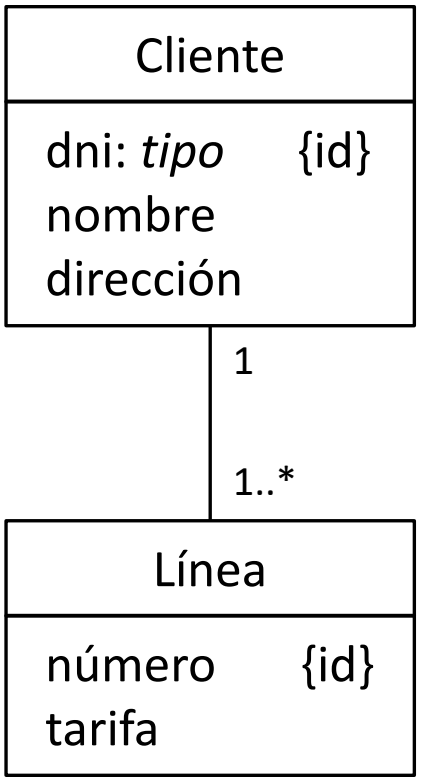
- Diseño físico
  - mejora de rendimiento en base a cargas típicas
  - idealmente no supone rediseño de etapas anteriores
- Diseño de aplicaciones y seguridad
  - procesos relacionados con las aplicaciones
  - tareas y flujos de trabajo
  - accesibilidad y seguridad

D. lógico (o conceptual) primero

» sobre un sistema de gestión telefónica

Cliente (dni, nombre, direccion)





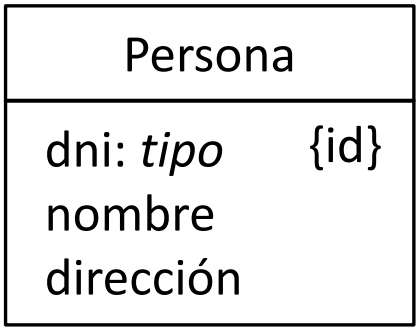
- ¿dos clientes pueden tener la misma línea?
- ¿una línea puede tener dos tarifas?
- ¿hay que poner más tablas? ¿cuáles serían?

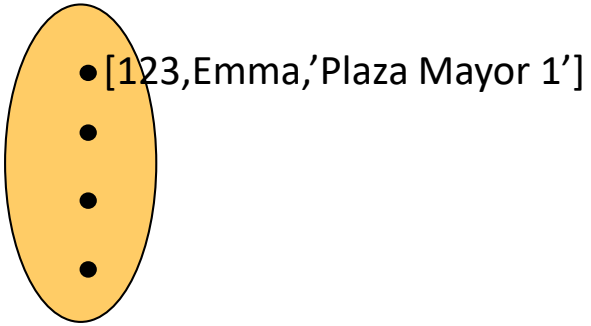
- ¿ER o UML?
- ER en notación UML

Diseño conceptual ER

» descripción de cómo se *estructuran* los datos
» *repaso* a partir de vista estructural UML
» varias alternativas para un mismo escenario

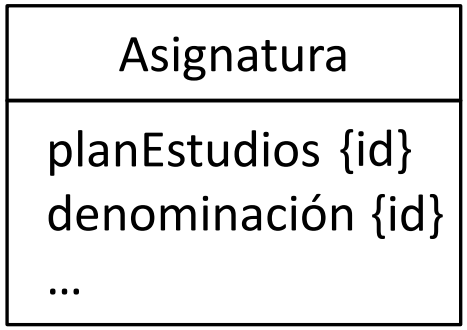
- Entidades y conjuntos de entidad
  - objetos que engloban los datos de interés
  - se describen como colecciones de entidades similares
  - se describen mediante atributos y *propiedades* adicionales
  - instancias como elementos de un conjunto

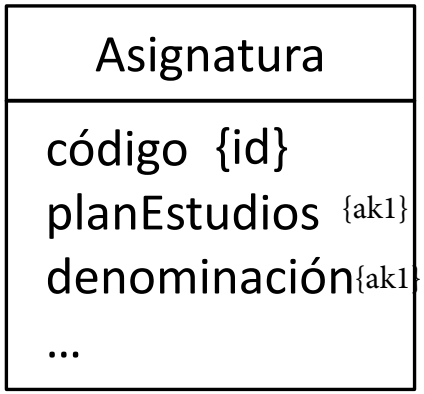




...

- Atributos (:dominios) y claves
  - definición de datos para cada instancia (incluye dominio)
  - clave: conjunto mínimo de atributos que *identifican* a cada entidad (valor mínimo sin repetición)
  - atributo → clase (ej. asignatura - plan de estudios)
    - grupo repetitivo: asignatura en un número *indeterminado* de planes de estudios
    - información asociada: año de comienzo del plan de estudios

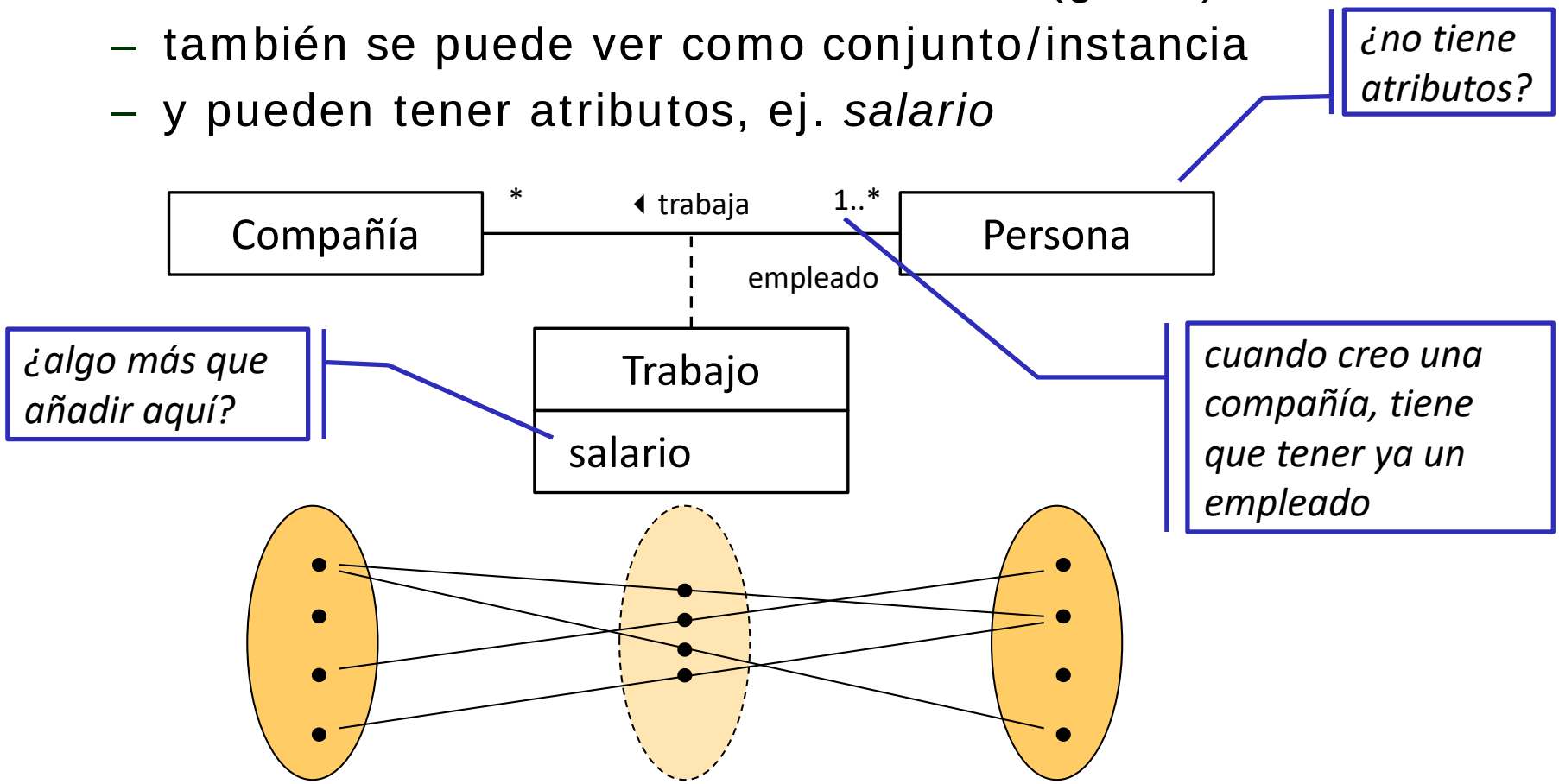




- identificación más sencilla
- ¿cómo se asegura la no repetición plan + denominación?
- claves alternativas

...

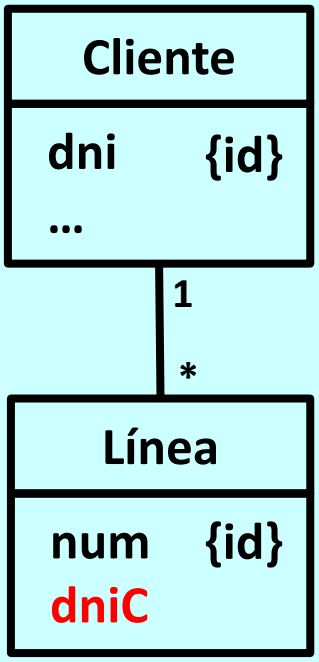
- Relaciones y conjuntos de relaciones
  - *asociación* entre dos o más entidades (grado)
  - también se puede ver como conjunto/instancia
  - y pueden tener atributos, ej. *salario*



CLAVE FORÁNEA EN ERD?

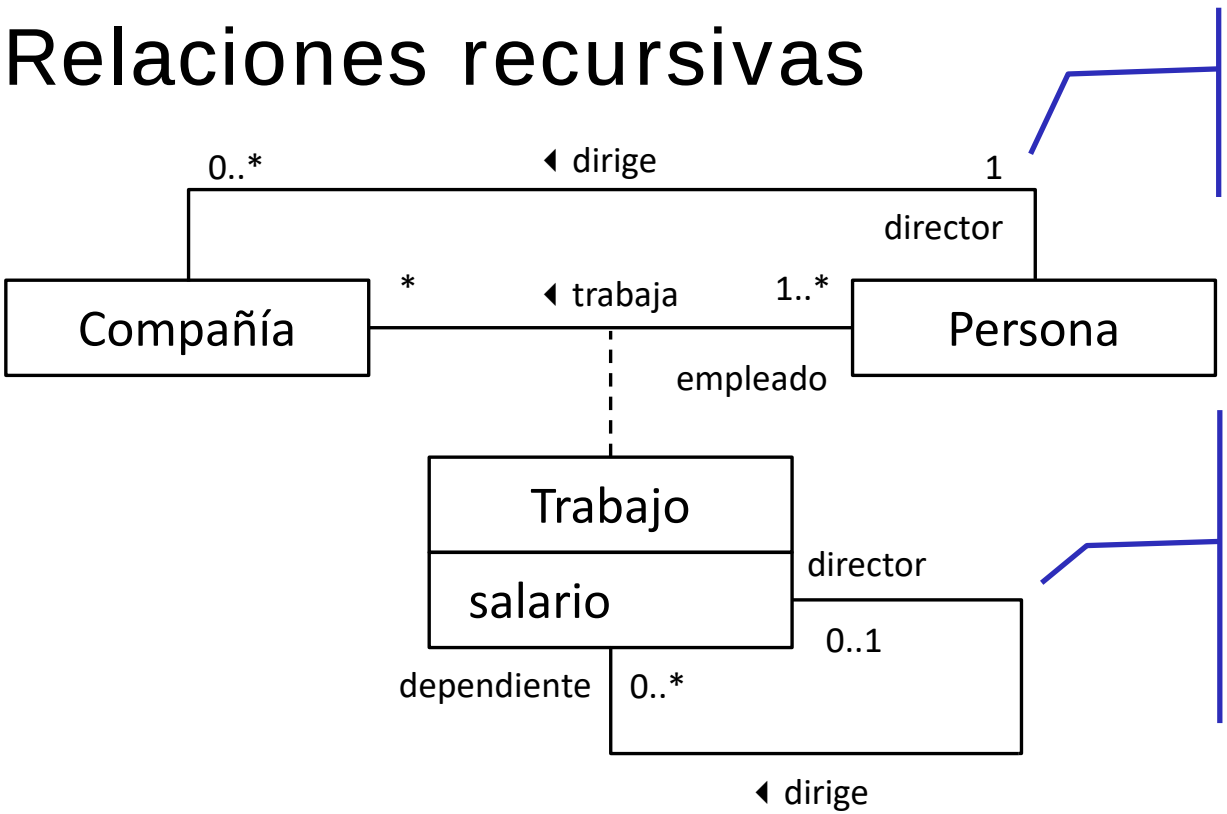
Essentials

- En ERD, no se copian atributos de una clase a otra para representar asociaciones
- Ya representado con la línea de asociación



...

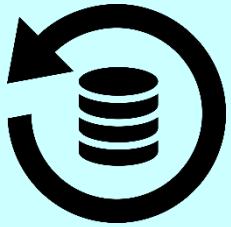
- Cardinalidad y roles
  - cardinalidad (multiplicidad) simplificadam. 1:1, 1:M, N:M
  - (mínima,máxima); mínima como *opcionalidad*
- Relaciones recursivas



¿y si el director tiene que ser uno de los empleados?

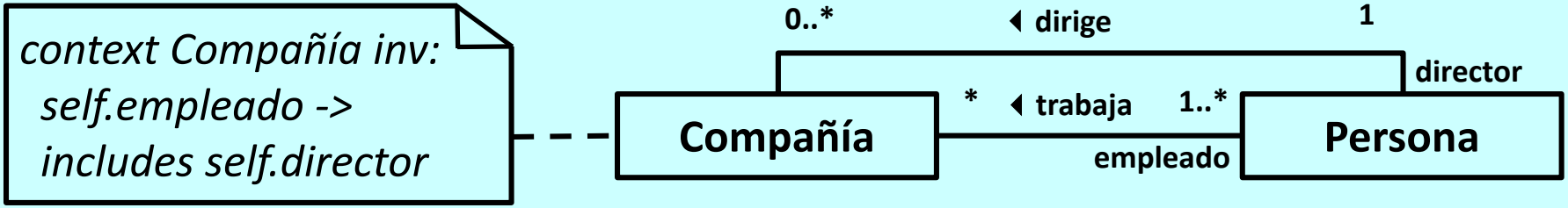
¿y si los trabajos dependientes tienen que involucrar siempre a la misma compañía?

RESTRICCIONES



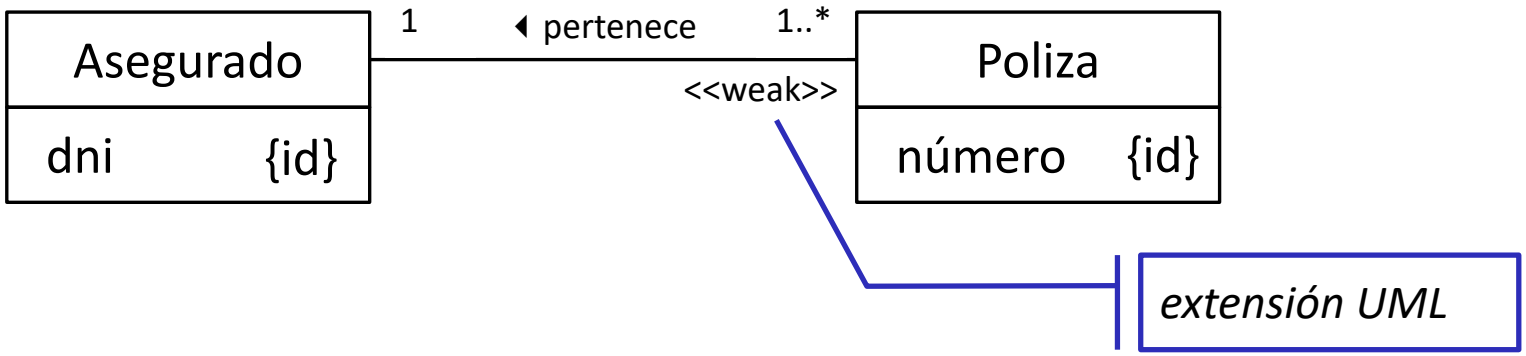
Essentials

- Condiciones de integridad específicas del ejemplo modelado
- Imprescindibles como requisitos
- Habrá que *introducirlas* en la BD para que pueda comprobarlas



...

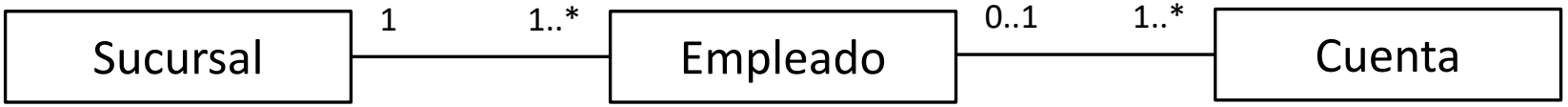
- Entidades débiles
  - *dependiente (subordinada)* se identifica considerando el id de otra *propietaria (dominante)*
  - relación obligatoria en el otro extremo de la relación



Trampas de conexión

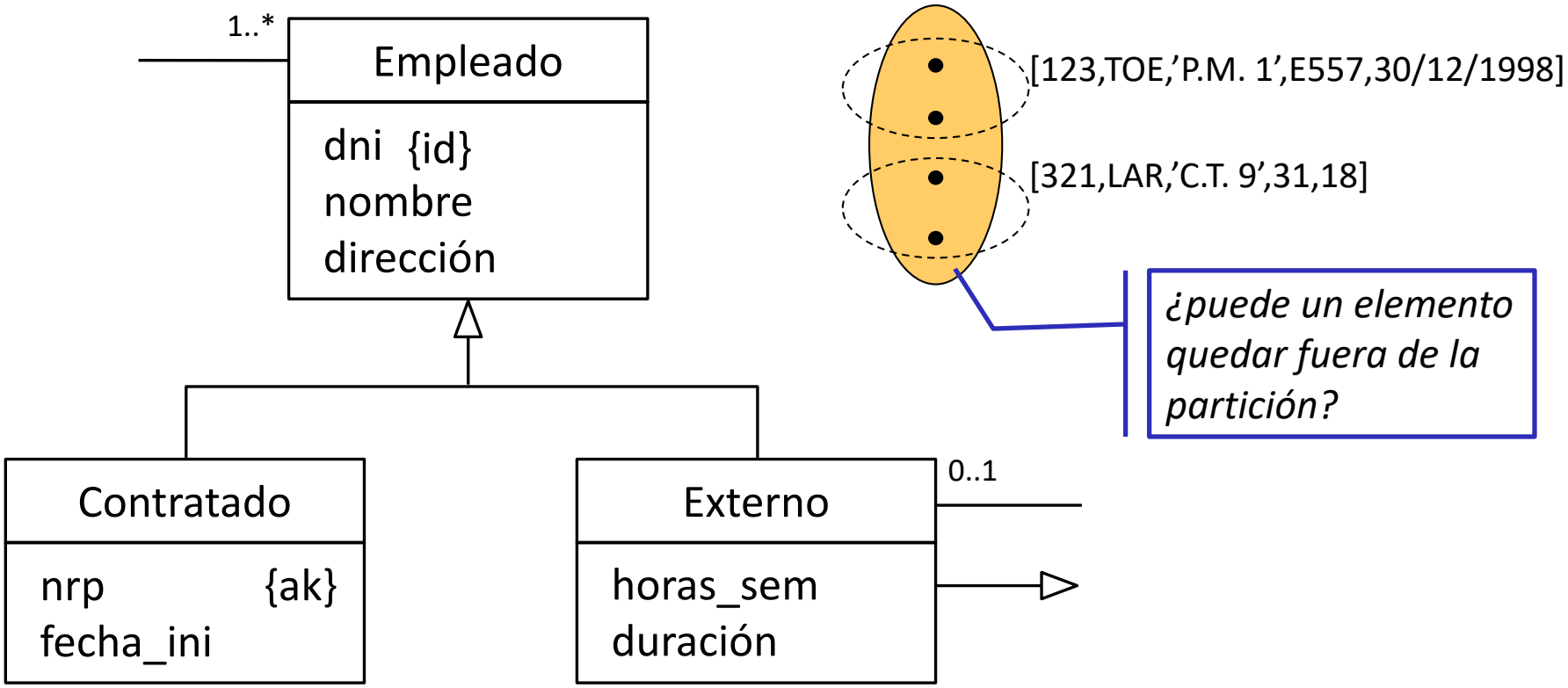
- Problemas por mal uso de relaciones
  - *Fan traps*

  - *Chasm traps*



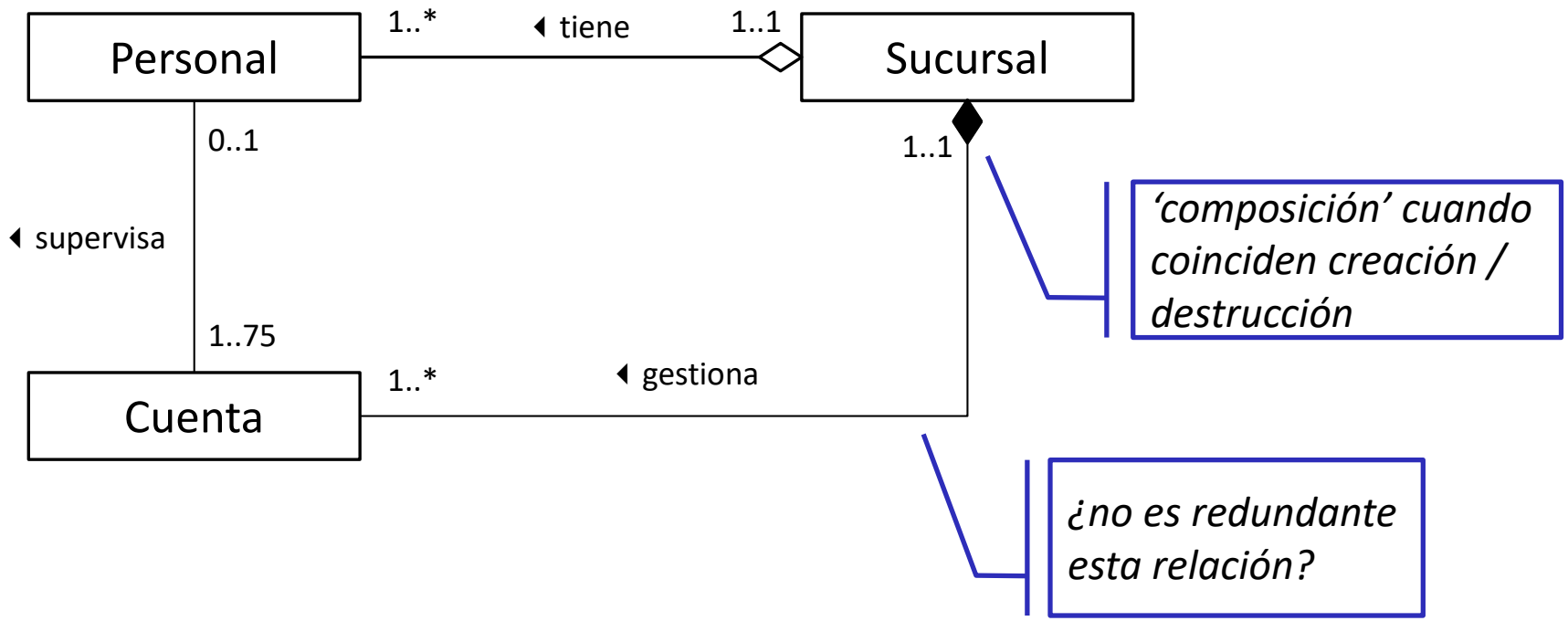
...

- Generalización/especialización (*'is_a'*)
  - especialización de entidades con propiedades específicas + todas las comunes generales, que son *heredadas*



...

- Agregación
  - representa la composición o agregación de *partes* en un *todo*
  - funciona como una asociación con un significado adicional



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

Diseño de Bases de Datos

Modelo Relacional

Origen e importancia

» sencillo y potente: BD como conjunto de tablas con filas y columnas; las tablas están asociadas

- Origen en los 70
  - Edgar F. Codd [CACM 1970]: definición y consultas
  - Substituye a modelos más antiguos: jerárquico, en red
- Modelo más ampliamente utilizado
  - Sencilla representación de datos
  - Facilidad para formular consultas
- El mundo de los RBDMS
  - Oracle, MySQL (Oracle), Postgres, SQL Server (Microsoft), Informix, DB2 (IBM), Sybase (SAP), ...
  - [map]

...

- Modelos competidores
  - Modelo objeto-relacional
  - Renovación de ‘viejas ideas’: datos jerárquicos
- Lenguaje relacional
  - definición (DDL) y manipulación (DML) de datos
  - el modelo relacional soporta consultas sencillas y potentes
    IMPORTANTE: *semántica* precisa de consulta relacional
  - el DBMS optimiza la operación en términos de eficiencia
  - SQL-92 (revisión importante)
    SQL-99 (extensiones importantes; estándar actual)

Definiciones

- BD Relacional: conjunto de relaciones
- Relación: dos partes
  - Esquema: nombre de la relación, y nombre y tipo de cada columna (atributo, campo)

    Estudiante (nia:int,nombre:string,login:string,edad:int,notam:real)
  - Instancia: valores de tabla (registro, tupla); todas las tuplas son distintas

nia	nombre	login	edad	notam
2354	García	garcia@med	21	7,2
9625	Aragón	aragon@eii	18	6,7
5557	Pozo	pozo@inf	23	8,9

Creación de relaciones

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

```
CREATE TABLE Estudiante (
   nia      INTEGER,
   nombre   CHAR(20),
   login    CHAR(10),
   edad     INTEGER,
   notam    REAL
)
```

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

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

Destrucción/modificación

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

```
DROP TABLE Estudiante
```

- La modificación de un esquema se puede hacer añadiendo, borrando o modificando campos
- Para campos añadidos, todas las tuplas son extendidas con valor *null* es ese campo

```
ALTER TABLE Estudiante
   ADD COLUMN matr  DATE
```

Añadir, borrar y modificar tuplas

- Se puede insertar una tupla en la relación

```
INSERT INTO Estudiante (nia,nombre,login,edad,notam)
VALUES (5557,'Rojo','rojo@cie',23,8.0)
```

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

```
DELETE
FROM Estudiante E
WHERE
E.nombre='Ledesma'
```

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

```
UPDATE Estudiante E
SET E.edad=E.edad+1
WHERE E.nia=5557
```

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
- 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`

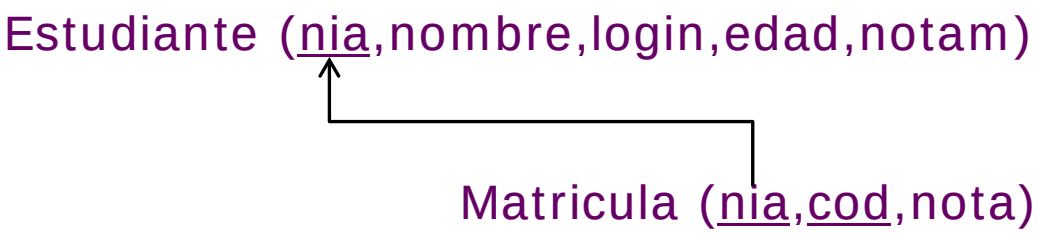
```
CREATE TABLE Matricula (
   nia   INTEGER,
   cod   CHAR(20),
   nota  REAL,
   PRIMARY KEY (nia,cod)
)
```

```
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
)
```

- 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
  - `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?

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

necesitaremos conocer más sobre la potencia de las consultas en SQL

Tipos de datos

```
<predefined type>    ::=
     <character string type> [...]
   | <national character string type>
   | <binary large object string type>
   | <bit string type>
   | <numeric type>
   | <boolean type>
   | <datetime type>
   | <interval type>
```

SQL-99

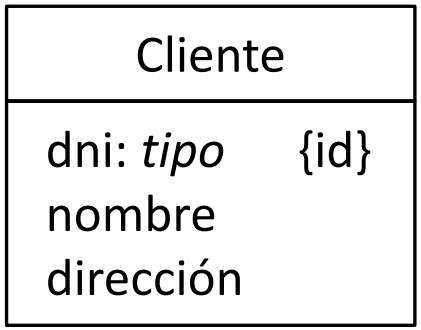
SQL-99

```
<domain definition>    ::=
   CREATE DOMAIN <domain name> [ AS ] <data type> [...]
       [<domain constraint>] [...]

CREATE DOMAIN color AS VARCHAR(10)
     CHECK (VALUE IN ('rojo',‘verde',‘azul'));
```

Diseño lógico: ER -> Relacional

- ERD



- ER

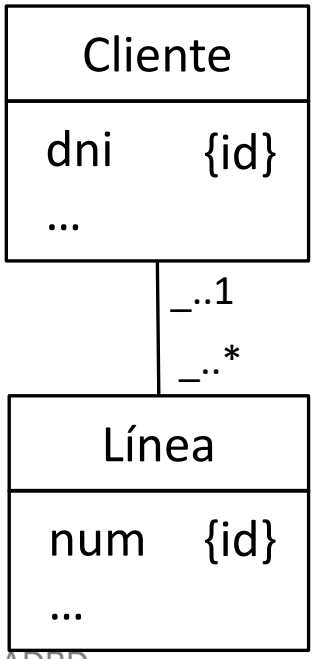
Cliente (<u>dni</u>,nombre,dirección)

- SQL

```
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,...)

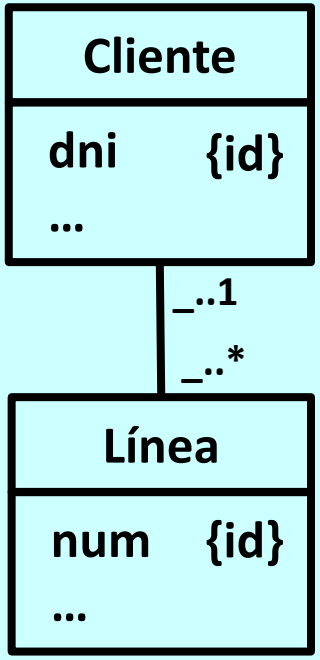
```
CREATE TABLE Cliente (
   dni     CHAR(10), ...,
   PRIMARY KEY (dni))

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

CLAVE FORÁNEA EN RELACIÓN 1—*

Essentials

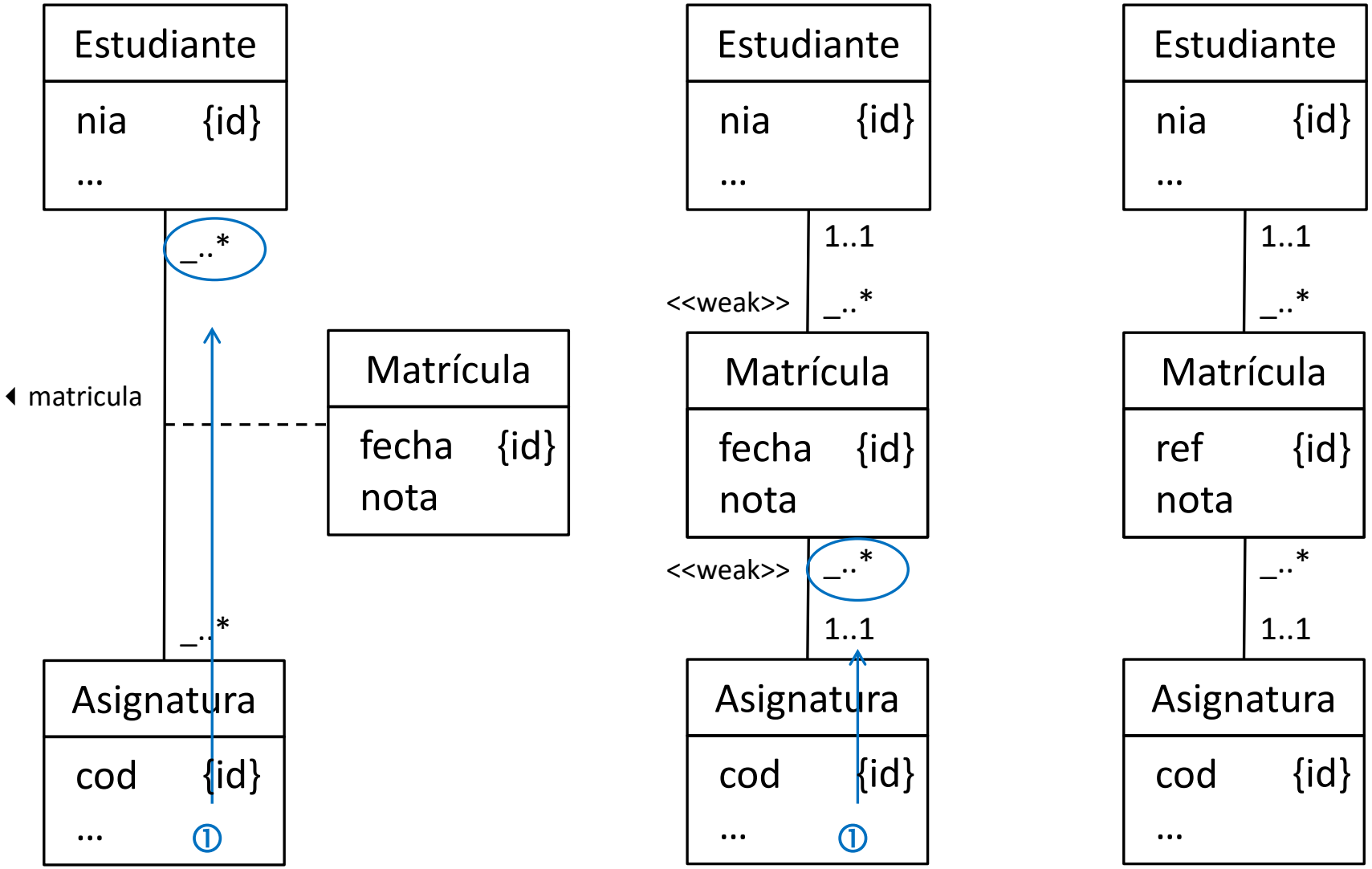
- En el lado *, apuntando al lado 1 (valor único)



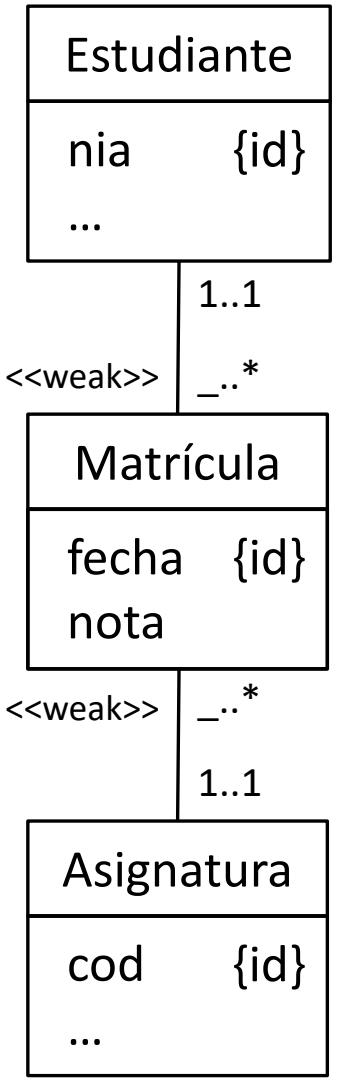
Cliente (<u>dni</u>,...)

Línea (<u>num</u>, dniC,...)

___


Estudiante
nia {id}
...
_..*
matricula
Matrícula
fecha {id}
nota
_..*
Asignatura
cod {id}
...
①
Estudiante
nia {id}
...
1..1
<<weak>>
_..*
Matrícula
fecha {id}
nota
<<weak>>
_..*
1..1
Asignatura
cod {id}
...
①
Estudiante
nia {id}
...
1..1
_..*
Matrícula
ref {id}
nota
_..*
1..1
Asignatura
cod {id}
...

...



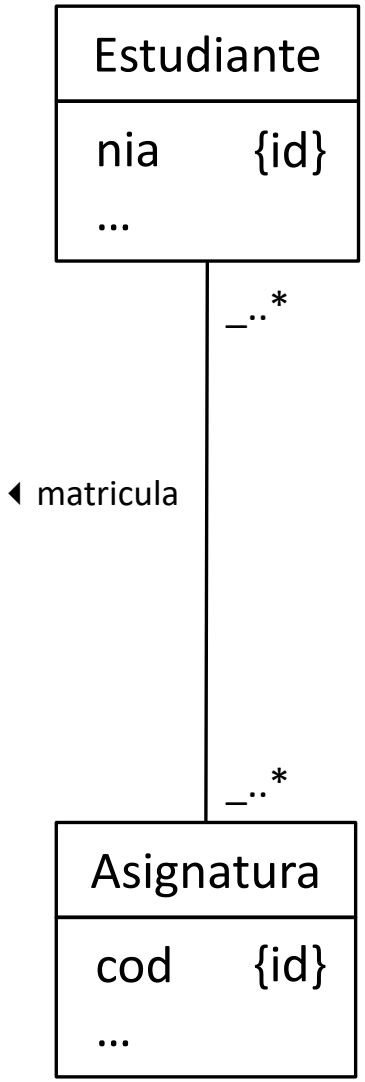
Estudiante (nia,...)

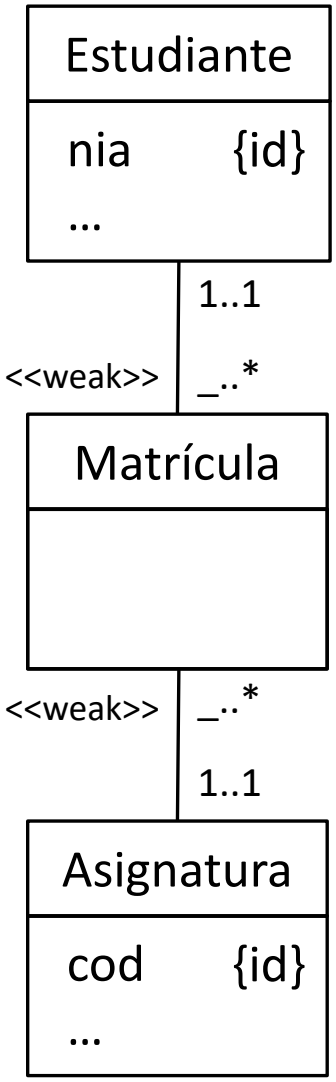
Matrícula (niaE,codA,fecha,nota)

Asignatura (cod,...)

```
CREATE TABLE Estudiante (...)
CREATE TABLE Asignatura (...)
CREATE TABLE Matricula (
   niaE    INTEGER,
   codA    INTEGER,
   fecha   DATE,
   nota    REAL
   PRIMARY KEY (niaE,codA,fecha),
   FOREIGN KEY (niaE) REFERENCES Estudiante,
   FOREIGN KEY (codA) REFERENCES Asignatura)
```

...





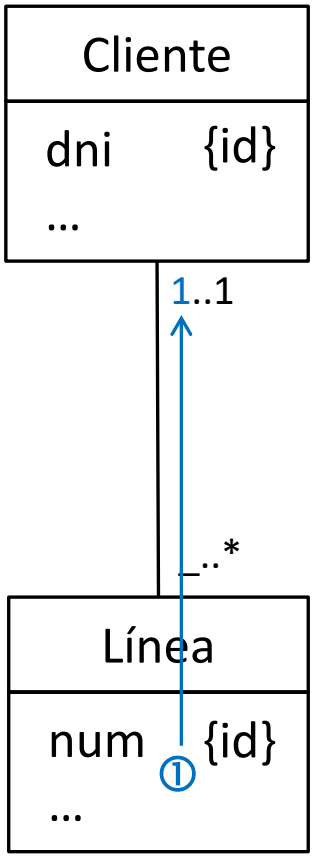
Estudiante (nia,...)

Matrícula (niaE,codA)

Asignatura (cod,...)

```
CREATE TABLE Estudiante (...)
CREATE TABLE Asignatura (...)
CREATE TABLE Matricula (
   niaE    INTEGER,
   codA    INTEGER,
   PRIMARY KEY (niaE,codA),
   FOREIGN KEY (niaE)
      REFERENCES Estudiante,
   FOREIGN KEY (codA)
      REFERENCES Asignatura)
```

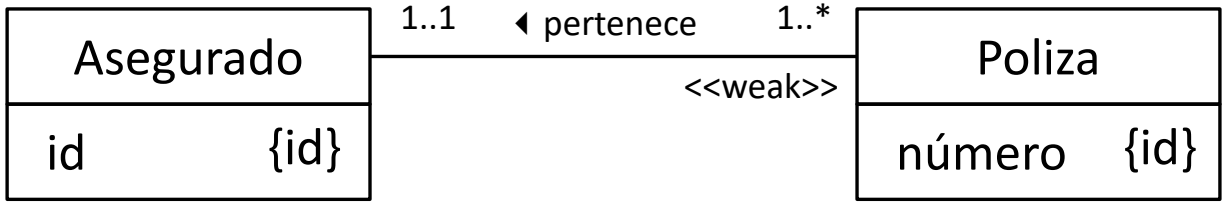
Traducción de la opcionalidad




```
CREATE TABLE Linea (
   num     INTEGER,
   dniC    CHAR(10) not null,
   ...,
   PRIMARY KEY (num),
   FOREIGN KEY (dniC)
     REFERENCES Cliente)
       ON DELETE NO ACTION)
```

- la opcionalidad 0.._ supone la existencia de nulos (*default*)
- ¿representación explícita en el modelo relacional?
- la obligatoriedad cliente — (1.._) línea debe tratarse con restricciones `CHECK`

Entidades débiles



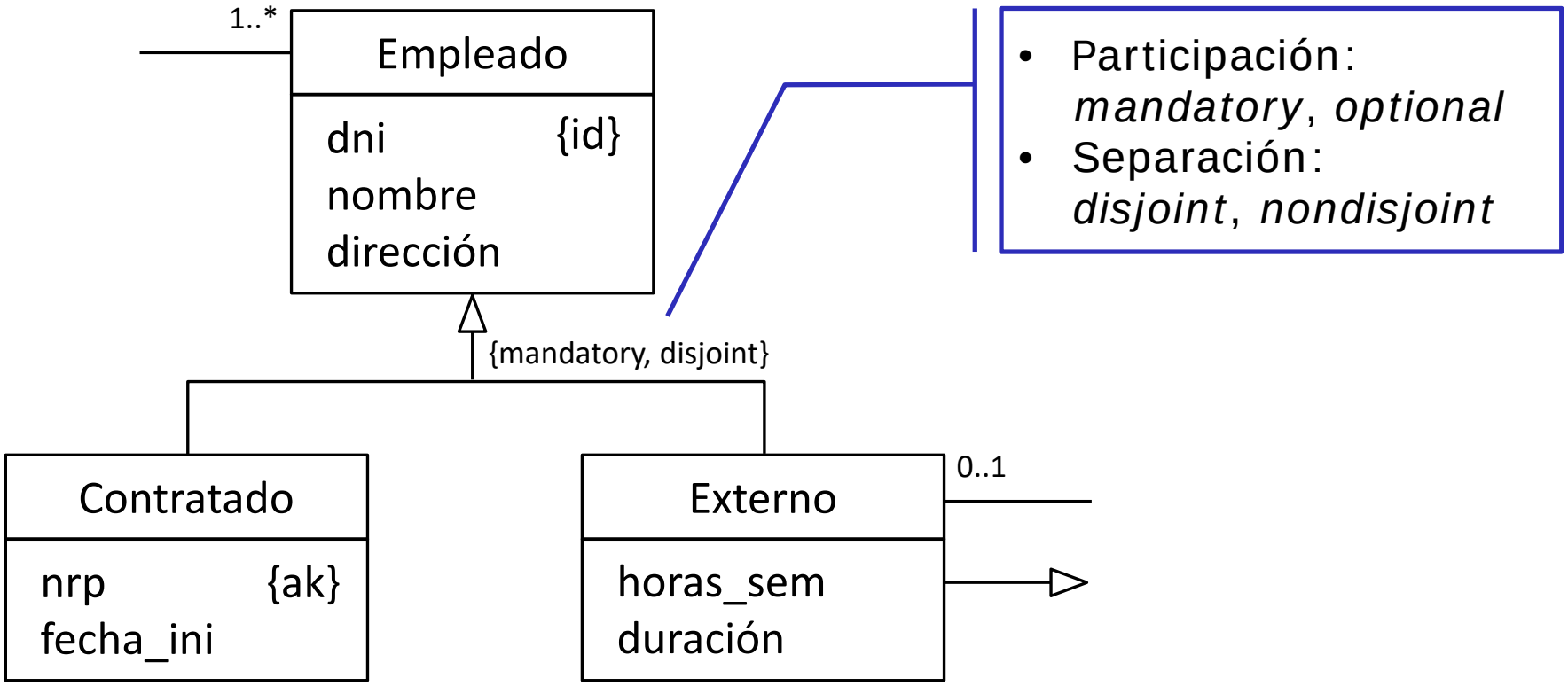
- combinación de: (1) identificación con campos de otra entidad, (2) relación obligatoria
- FK no nula que forma parte de la PK

Asegurado (id,...)

Póliza (num,idA,...)

```
CREATE TABLE Asegurado (...)
CREATE TABLE Poliza (
   num     INTEGER,
   idA     CHAR(10) not null,
   ...,
   PRIMARY KEY (num,idA),
   FOREIGN KEY (idA)
      REFERENCES Asegurado
        ON DELETE CASCADE)
```

Jerarquías ISA



ISA, opciones

- 1 tabla

Empleado (dni,nombre,dirección,nrp,fechaIni,horasSem,duracion,FK1)

  - ¿cómo distingo a los contratados de los externos?
  - ¿qué pongo en nrp cuando introduzco un externo?

- 2 tablas

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?

ISA, opciones

- 3 tablas

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*?

- 2* tablas

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

Empleado_Exte (dni,horasSem,duración)

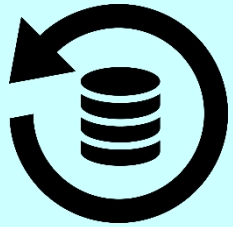
  - ¿cuándo sería esto adecuado?

...

» es una guía de elección, pero puede haber otros factores (cantidad de nulos y de tuplas, relaciones, consultas / combinación de tablas)

- Tabla para la superclase
  - si tiene asociaciones 1:* (lado 1); las tablas que incluyen las claves foráneas deben apuntar a una única tabla
  - si es *optional,* para los que solo pertenecen a la superclase
  - si es *nondisjoint,* para evitar repeticiones
  - discriminador: explícito o implícito
- Tablas para las subclases
  - diferencias entre ellas; todas en una tabla daría lugar a nulos y restricciones
  - diferencia con la superclase; puede habler tablas para cada combinación superclase/subclase

TABLAS PARA ISA



Essentials

- Tabla para la superclase
  - si es apuntada por clave foránea
  - si es optional y/o nondisjoint
- Tablas para las subclases
  - si hay diferencias que implicarían *demasiados* nulos y restricciones

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

```
CREATE VIEW Persona AS
   SELECT * FROM Persona_Fisica
   UNION
   SELECT * FROM Persona_Juridica
```

¿Qué nos falta?

- Principalmente, la manipulación y consulta de datos de una BD
  - Necesitaremos lenguajes de consulta (QL)
  - QLs
    - *"reales"* para la implementación (SQL)
    - formales para la conceptualización (*RA,RC*)
  - ¡ Primero los conceptos !
    - *fundamentar* lo que significa manipular relaciones para obtener los resultados de las consultas
  - ¡ Después las implementaciones !

Diseño de Bases de Datos

Álgebra/Cálculo Relacional

Ejemplo: reservas de vehículos

C (clientes)

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

R (reservas)

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

V (vehículos)

Vehiculo

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

- Clientes (idc, nombre) con categoría 3
- Clientes con categoría 3 o 4
- Clientes (nombre) que en algún momento hayan reservado el vehículo 101
- Cliente (*) con la mayor categoría
- Número de veces que cada coche ha sido reservado

Lenguajes de consulta (QL)

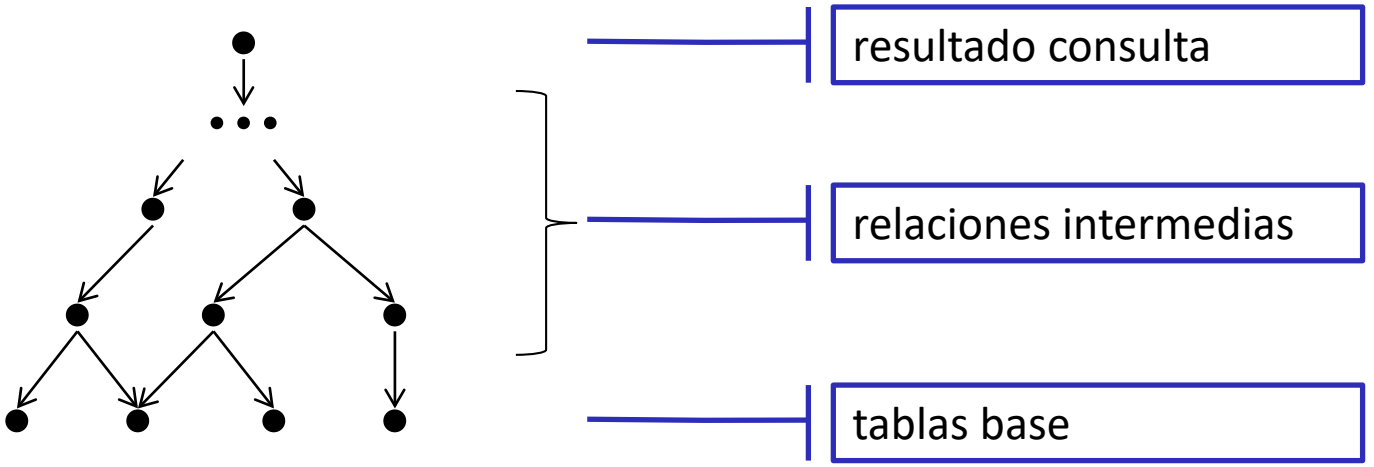
- Leguajes de consulta
  - permiten la manipulación y consulta de datos de las BD
- Lenguajes relacionales *formales*
  - manipulación de relaciones para obtener una relación resultado, que es el resultado de una consulta
  - fundamentos formales basados en la lógica
  - permiten una muy potente optimización
  - base de los lenguajes *reales* de implementación (SQL)
- QL no es un lenguaje de programación
  - sólo está pensado para soportar el acceso a los datos
  - no para cálculos complejos / computación

QLs relacionales formales

- Álgebra relacional
  - operacional/imperativo —*cómo hago que X ocurra*
  - muy útil para representar *planes de ejecución:* secuencia de operaciones que hacen que se produzca el resultado
  - base de la ejecución interna de una consulta, lo cual es muy útil para un usuario avanzado / administrador (optimización)
- Cálculo relacional
  - declarativo —*cómo reconozco que X ha ocurrido*
  - permite a los usuarios describir lo que buscan: declaración de los resultados en los que se está interesado

Álgebra relacional

- Consulta = árbol relacional
  - tomando como punto de partida las tablas base —que nunca son modificadas,
  - se aplican una secuencia de operadores —unarios o binarios,
  - cada uno de los cuales devuelve una relación intermedia,
  - cuya relación resultante final es el resultado de la consulta



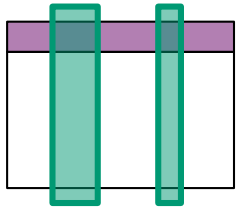
Operadores

» cada operador devuelve una relación, por lo que pueden ser *compuestos*

- Básicos
  - selección: σ, selecciona un subconjunto de filas
  - proyección: π, extrae columnas
  - unión: $\cup$, tuplas en relación1 y relación2
  - diferencia: $-$, tuplas en relación1, pero no en relación2
  - producto cartesiano: $\times$, combina dos relaciones
- Adicionales, se pueden definir en término de los básicos, pero son de gran utilidad y uso (esp. *join*)
  - intersección: $\cap$, tuplas en ambas relación1 y relación2
  - división: $/$, *cociente* de relaciones con *factores* comunes
  - join: $\bowtie$, combinación condicionada

Proyección y selección

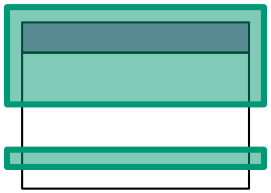
$\pi_{\text{listaAtrib}}(\text{Rel})$ » mantiene los campos que están en la lista de proyección



– elimina los duplicados (conjuntos de tuplas)

– los sistemas reales sólo eliminan duplicados si se pide

$\sigma_{\text{cond}}(\text{Rel})$ » selecciona las tuplas con cumplen la *condición de selección*



– no hay duplicados en el resultado

– el esquema es igual que el de la relación original

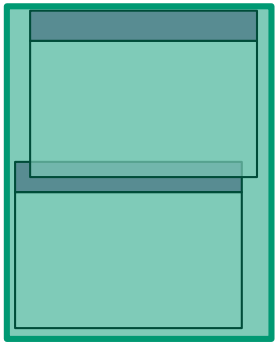
– composición de operadores

$\pi_{nombre,cat}\ (\sigma_{\text{cat}>7}(\text{c2}))$

nombre	cat
Izarra	8
Pozo	10

Unión, intersección y diferencia

R1 ∪ R2

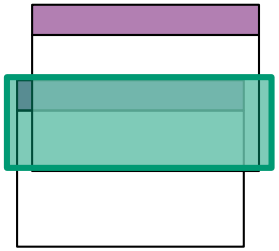


» operaciones clásicas de unión, intersección y diferencia de conjuntos (de tuplas, en este caso)

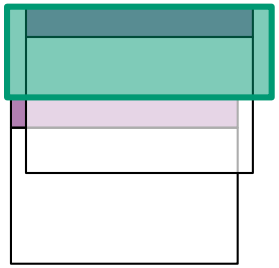
– en estos operadores, el esquema tiene que ser *compatible*, manteniéndose en el resultado

– eliminación de duplicados

R1 ∩ R2



R1 – R2

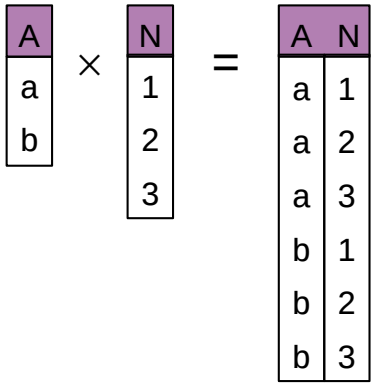


c1 ∪ c2

idc	nombre	cat	edad
29	Rojo	6	23
33	Izarra	8	42
41	Heredia	2	23
57	Pozo	10	23
23	García	7	31
96	Aragón	8	58

Producto cartesiano

R1 × R2



» combinación de todos los posibles pares de tuplas de las dos relaciones

– esquema combinado de las dos relaciones, con renombrado de atributos —(atr)— para evitar coincidencias

– producto de tuplas, suma de esquemas

c1 × r1

idc	nombre	cat	edad	(idc)	matr	fecha
23	García	7	31	23	101	10/07/2011
23	García	7	31	57	201	01/10/2011
96	Aragón	8	58	23	101	10/07/2011
96	Aragón	8	58	57	201	01/10/2011
57	Pozo	10	23	23	101	10/07/2011
57	Pozo	10	23	57	201	01/10/2011

¿qué tipo de resultado nos gustaría obtener en realidad?

Joins: combinaciones cond.

R1 ⋈R2

A	B
a	1
b	2

⋈

B	C
1	x
1	y
3	z

=

A	B	C
a	1	x
a	1	y

- » combinación por igualdad de campos coincidentes —join *natural*
- soporta gran parte de las consultas que involucran a dos o más tablas
- combina datos de entidades que residen en varias tablas (normalización, p.e.)
- operación difícil de implementar eficientemente por un DBMS, lo que genera un problema intrínseco de rendimiento

c1 ⋈ r1

idc	nombre	cat	edad	matr	fecha
23	García	7	31	101	10/07/2011
57	Pozo	10	23	201	01/10/2011

...

- Join-theta, R1 $\bowtie_{cond}$ R2
  - producto cartesiano que se restringe con una condición adicional
- Equijoin, R1 $\bowtie_{=}$ R2
  - la condición sólo involucra igualdades
  - los campos que se igualan sólo aparecen una vez en el esquema resultante
- Natural, R1 $\bowtie$ R2
  - equijoin sobre todos los campos comunes

...

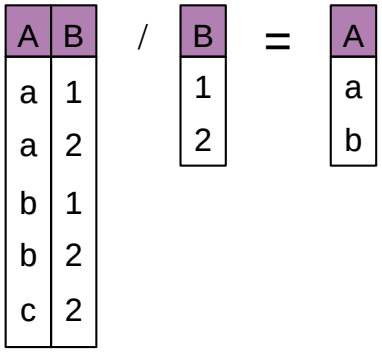
- Semijoin, R1 $\triangleright$ R2
  - tuplas de R1 que participan en un join con R2
  - π_{atrR1} (R1 $\bowtie$ R2)
- Outer (left,right), R1 $\bowtie$ R2
  - (left) las tuplas de R1 que no tienen correspondencia en los campos comunes con R2, también son incluidas en el resultado, completando con nulos

c1 $\bowtie$ r1

idc	nombre	cat	edad	matr	fecha
23	García	7	31	101	10/07/2011
96	Aragón	8	58	*null*	*null*
57	Pozo	10	23	201	01/10/2011

División

R1 / R2



A	B
a	1
a	2
b	1
b	2
c	2

/

B
1
2

=

A
a
b

» (proyección de) tuplas de R1 que figuran en combinación con todas las tuplas de R2

– esquema reducido a los campos que combinan con los de la otra relación
– se puede expresar en términos de los operadores básicos
– útil para consultas del tipo: *clientes que han reservado todos los coches ...*

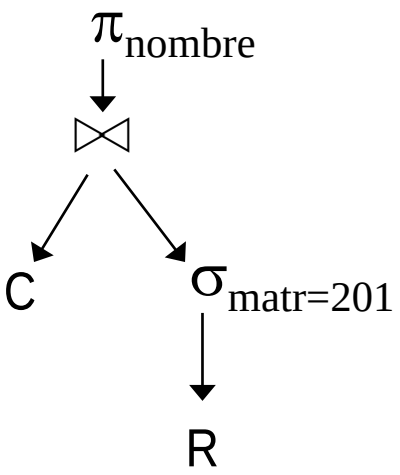
Ejemplos de consultas

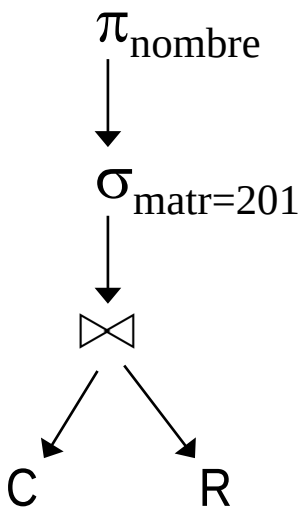
Cliente (idc,nombre,cat,edad)

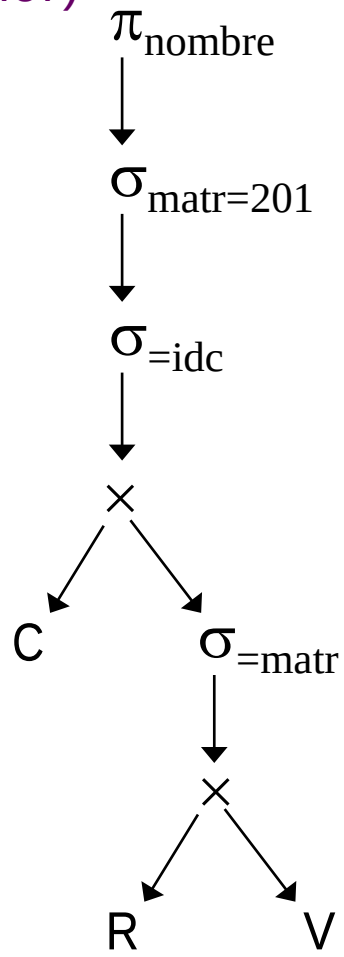
Vehículo (matr,marca,color)

Reserva (idc,matr,fecha)

- Clientes que han reservado el coche 201
  - $\pi_{nombre}\ (\sigma_{matr=201}\ (\ \sigma_{=idc}\ (\ C \times (\ \sigma_{=matr}\ (\ R \times V\ )))))$
  - $\pi_{nombre}\ (\sigma_{matr=201}\ (\ C \bowtie R\ ))$
  - $\pi_{nombre}\ (C \bowtie \sigma_{matr=201}\ (R))$



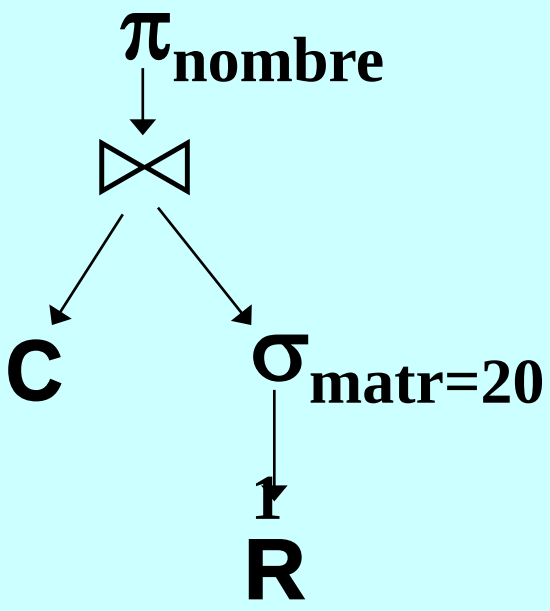




(AR) CONSULTA = ÁRBOL RELACIONAL

Essentials

- Secuencia de operadores sobre tablas; raíz = datos buscados



= Clientes que han reservado el coche de matrícula 201

...

» Clientes que han reservado un coche rojo

- π_{nombre} (($\sigma_{color='rojo'}$ V $\bowtie$ R) $\bowtie$ C)
- π_{nombre} (π_{idc} (π_{matr} $\sigma_{color='rojo'}$ V $\bowtie$ R) $\bowtie$ C)

  – el optimizador se encarga de encontrar la mejor forma de resolver la consulta
  – (en principio) no es dependiente de la forma del SQL
  – el color sólo está en la relación Vehículo, por lo que es necesario en join extra

...

» Clientes que han reservado un coche rojo o verde

- ρ(rvVeh, ($\sigma_{color='rojo' \text{ v } color='verde'}$ V))
- π_{nombre} ((rvVeh $\bowtie$ R) $\bowtie$ C)

– operador de renombrado
– también se podría haber utilizado la unión

...

» Clientes que han reservado un coche rojo y verde

- ρ(rRes, π_{idc} (($\sigma_{color='rojo'}$ V) $\bowtie$ R))
- ρ(vRes, π_{idc} (($\sigma_{color='verde'}$ V) $\bowtie$ R))
- π_{nombre} ((rRes $\cap$ vRes) $\bowtie$ C)

– el planteamiento anterior (o), ¡no funciona!
– está basado en la identificación de clientes según sus reservas de vehículos
– la intersección debe hacerse sólo sobre los *idc;* en caso contrario la intersección sería vacía

...

- Clientes que han reservado todos los vehículos
  - ρ(idcs, ($\pi_{idc,matr}$ R) / (π_{matr} V))
  - π_{nombre} (idcs $\bowtie$ C)
- Clientes que han reservado todos los BMW
  - ρ(idcs, ($\pi_{idc,matr}$ R) / (π_{matr} ($\sigma_{marca='BMW'}$ V)))
  - π_{nombre} (idcs $\bowtie$ C)

...

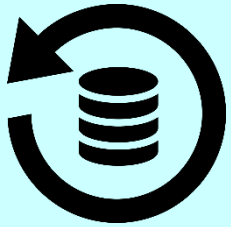
- Mayor categoría
  - C(cat) - π_{cat1} ($\sigma_{cat1<cat2}$ (C(cat1) x C(cat2)))

- Número de veces que cada coche ha sido reservado
  - operadores de agrupamiento ...

Cálculo relacional

- Consulta = condición (predicado)
  - sobre las tuplas que deben incluirse en la contestación a la consulta —aquellas que evalúan la condición a *true*
  - $\{<x_1,\ldots,x_n> \mid p(<x_1,\ldots,x_n>)\}$
  - las expresiones son fórmulas el algún tipo de *lógica de predicados de primer orden*, con variables a las que puedo asignar tuplas (TRC) o elementos de tuplas (DRC)
  - se utilizan: variables, constantes, comparadores, conectores lógicos y cuantificadores
  - las fórmulas se definen recursivamente: pertenencia de tuplas a relaciones, comparación de valores, conectivas lógicas
  - entorno útil para restricciones / aserciones

(CR) CONSULTA = PREDICADO



Essentials

- Condición que satisfacen (solo) las datos buscados

$$\{ c : \text{Cliente} \mid \exists\, r : \text{Reserva} \mid c.idc = r.idc \wedge r.mr = 201 \}$$

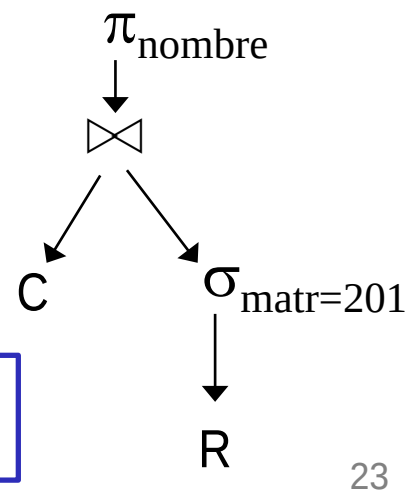
= Clientes que han reservado el coche de matrícula 201

DRC

- Variables de campos/dominios
  - las variables son asignadas con elementos del dominio de los campos de las tuplas
  - $\{<i,n,c,e> : Cliente \mid c>7\}$
  - las variables i,n,c,e se *vinculan* a los campos de tuplas Cliente
  - no es una restricción para todas las tuplas cliente (aserciones), sino una condición de consulta
  - $\{<i,n,c,e> : Cliente \mid c>7 \wedge \exists <ir,mr,f> : Reserva \mid ir=i \wedge mr=201\}$

encuentra una reserva que coindice (*join*) con el cliente en consideración

comparar con el álgebra relacional



Lenguajes de restricciones

- ¿Alguna relación con OCL?
  - no serán restricciones del esquema (aserciones), sólo restricciones de las tuplas resultantes de una consulta
  - OCL utiliza la *navegación* en vez de la combinación explícita de relaciones por igualdad de campos
  - no se representa la proyección
  - context Cliente inv:
    self.reserva -> exists (r|r.matr=201)
  - context Cliente inv:
    self.reserva.vehículo -> exists (v|v.color='rojo' $\vee$ v.color='verde')
  - context Cliente inv:
    self.reserva.vehículo -> exists (v1,v2| v1$\neq$v2 $\wedge$
    v1.color='rojo' $\wedge$ v2.color='verde')

...

» Clientes que han reservado todos los BMW

– $\{ \langle i,n,c,e \rangle : \text{Cliente} \mid \forall \langle m,ma,c \rangle : \text{Vehículo} \mid ma \neq \text{'BMW'} \vee \exists \langle ir,mr,f \rangle : \text{Reserva} \mid ir = i \wedge mr = m \}$

sólo se *declara* una condición de pertenencia al resultado;
en contraste con el álgebra, donde se *detalla* la secuencia de operaciones

» Y en OCL?

– context Cliente inv:
self.reserva.vehículo ->
forAll(v | v.marca='BMW')

NO: todos los reservados son BMW !!

– context Cliente inv:
self.reserva.vehículo =
Vehículo.allInstances -> select(v|v.marca='BMW')

la notación punteada evita las igualdades de combinación (join)

Potencia expresiva

- Capacidad de expresar consultas
  - cualquier consulta que se pueda expresar en AR, se puede expresar en CR
  - ¿y viceversa?
    - CR permite fórmulas *inseguras,* con infinitos resultados en dominios distintos al referenciado
      $\{<i,n,c,e> \mid \neg(<i,n,c,e> \in \text{Cliente})\}$
    - cualquier consulta segura en CR, se puede expresar en AR
- Relacionalmente completo
  - potencia expresiva del álgebra/cálculo relacional
  - los lenguajes de consulta prácticos (SQL) deben ser relacionalmente completos
  - de hecho, permiten expresiones que no permite el AR

Diseño de Bases de Datos

SQL. Consultas y Restricciones

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, with, boolean, ...), 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

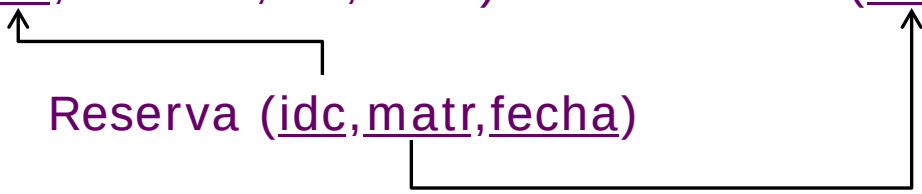
- Ya vista una parte de DDL
- Nos quedan partes fundamentales:
  - Consultas: operativa mayoritaria en BD
  - Restricciones
  - Disparadores
- Respecto al Álgebra Relacional
  - es relacionalmente completo
  - utiliza *multiconjuntos*
  - construcciones adicionales

Datos ejemplo

Cliente (idc,nombre,cat,edad)

Vehículo (matr,marca,color)

Reserva (idc,matr,fecha)



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

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

Vehiculo

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

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

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)

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
...
```

...

Empleado

id	nombre	dpto
11	Jiménez	ventas
12	Blanco	compras
13	Gracia	ventas
14	Fonseca	compras

Departamento

dpto	jefe
ventas	Carcedo
compras	Arias

- join natural:

```
SELECT *
FROM Empleado E
     NATURAL JOIN Departamento D;
```

Empleado ⋈ Departamento

id	nombre	dpto	jefe
11	Jiménez	ventas	Carcedo
12	Blanco	compras	Arias
13	Gracia	ventas	Carcedo
14	Fonseca	compras	Arias

Join de tablas —explícito

- Natural join

```
SELECT C.nombre
FROM Cliente C NATURAL JOIN Reserva R
WHERE ...;
```

solo sobre atributos con el mismo nombre (no tiene por qué ocurrir)

se puede ver como una tabla *temporal* construida como el join de las dos, pero sin nombre propio

- Inner (equi-) join

```
SELECT C.nombre
FROM Cliente C INNER JOIN Reserva R
     ON C.idc=R.idc
WHERE ...;
```

igualdad con el mismo nombre, `USING (idc)` —en vez de `ON ...`

- Outer join

```
SELECT C.nombre
FROM Cliente C LEFT OUTER JOIN Reserva R
     ON C.idc=R.idc
WHERE ...;
```

`LEFT | RIGHT | FULL`

Expresiones

» Nueva categoría para los clientes que hayan reservado dos coches diferentes el mismo día

```
SELECT C.nombre, C.cat+1 AS scat
FROM Cliente C, Reserva R1, Reserva R2
WHERE C.idc=R1.idc AND C.idc=R2.idc
      AND R1.fecha=R2.fecha AND R1.matr<>R2.matr;
```

```
+----------+------+
| nombre   | scat |
+----------+------+
|  Davila  |    8 |
|  Davila  |    8 |
+----------+------+
```

- expresiones en la lista de selección
- habrá que utilizar `DISTINCT`
  —¿por qué la repetición?
- repetición de Reserva en `FROM`
  para referirse a los dos roles de búsqueda

Strings

- las cadenas de caracteres se pueden comparar (=,>,<,...) utilizándose el orden alfabético
- la comparación por patrones se realiza con `LIKE`, y los símbolos comodín para caracteres arbitrarios `%` (0+), `_` (1)

» Clientes cuyo nombre empieza por ‘A’, y tienen menos de 30 años

```
SELECT C.nombre, C.edad
FROM Cliente C
WHERE C.nombre LIKE ‘A%’ AND C.edad < 30;
```

```
+--------+------+
| nombre | edad |
+--------+------+
| Arnaud |   25 |
+--------+------+
```

- `SIMILAR` (SQL99) permite una gran variedad de expresiones regulares

Manipulación de conjuntos

- `UNION`, `INTERSECT`, `EXCEPT`
  - hay sistemas con sólo la unión, teniendo que recurrirse a otro tipo de construcciones
  - hay sistemas con `MINUS` en lugar de `EXCEPT`

—para el resto de operaciones se introducen las *subconsultas* (más inspirado en el cálculo relacional)

- `IN`, *op* `ANY`, *op* `ALL`
  - pertenencia y comparaciones con conjuntos
- `EXISTS`
  - comprobación de conjunto vacío
- `NOT` (`IN`, `EXISTS`)
  - modificación del significado por negación

Unión

» Clientes que han reservado un coche rojo o verde

```
SELECT DISTINCT C.idc
FROM Cliente C, Vehiculo V, Reserva R
WHERE C.idc=R.idc AND R.matr=V.matr
      AND (V.color='rojo' OR V.color='verde');
```

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

```
SELECT C.idc
FROM Cliente C, Vehiculo V, Reserva R
WHERE C.idc=R.idc AND R.matr=V.matr AND V.color='rojo'
UNION [ALL]
SELECT ... V.color='verde';
```

- los operadores sobre conjuntos del AR están disponibles en SQL —para *multiconjuntos*
- los conjuntos tienen que ser compatibles

Intersección

» Clientes que han reservado un coche rojo y verde

```
SELECT ... V.color='rojo'
INTERSECT
SELECT ... V.color='verde';
```

```
+-----+
| idc |
+-----+
|  22 |
|  31 |
+-----+
```

– no todos los DBMS lo incluyen (aunque es SQL-92)
– la otra opción es más compleja, basada en roles

```
SELECT DISTINCT C.idc, C.nombre
FROM Cliente C, Vehiculo V1, Reserva R1,
                Vehiculo V2, Reserva R2
WHERE C.idc=R1.idc AND R1.matr=V1.matr AND
      C.idc=R2.idc AND R2.matr=V2.matr AND
      V1.color='rojo' AND V2.color='verde';
```

Consultas anidadas

» tabla *temporal*, *calculada* construida como el resultado de un select, y

» utilizada en otro select habitualmente en:
(1) como una tabla en el `FROM`
(2) en alguna condición del `WHERE`

- Utilización en `FROM`

```
SELECT T.at1, T.at2
FROM Tabla T
WHERE ...;


SELECT ...
FROM (SELECT ...) Ttemp, ...
WHERE ...;
```

```
+-----+-----------+
| at1 | at2       |
+-----+-----------+
|  -- |  -------  |
|  -- |  -------  |
|  -- |  -------  |
|  -- |  -------  |
+-----+-----------+
```

sin nombre y, por tanto sin poder referenciarla

nombre que permite referenciar `Ttemp.at1, Ttemp.at2, ...`

Consultas anidadas

» en la cláusula `WHERE` aparece en construcciones más complejas

```
SELECT C.nombre
FROM Cliente C
WHERE C.idc IN (SELECT R.idc
                FROM Reserva R
                WHERE R.matr IN (SELECT V.matr
                                 FROM Vehiculo V
                                 WHERE V.color='rojo'));
```

aquí no es necesario dar nombre a las tablas temporales

- `IN` comprueba la pertenencia; el complementario de la consulta se consigue con `NOT IN`
- se evitan los *joins* en base a subconsultas
- evaluación *anidada*: para cada tupla del nivel exterior, calcular la subconsulta —tiene esto sentido para subconsultas independientes del nivel exterior?

Anidamiento correlacionado

» subconsulta depende de la tupla que se esté examinando en la consulta exterior

```
SELECT C.nombre
FROM Cliente C
WHERE EXISTS (SELECT *
              FROM Reserva R
              WHERE R.matr=103 AND R.idc=C.idc);
```

- la aparición de `C` en la subconsulta denota la dependencia
- `EXISTS` comprueba que el conjunto no está vacío; se puede anteponer `NOT`
- el uso del `*` se considera aquí buen estilo de programación

Otros comparadores

- comparación (<,<=,=,<>,>=,>) que se cumplirá cuando alguno (ANY) o todos (ALL) los elementos lo cumplan

```
SELECT C.idc
FROM Cliente C
WHERE C.cat >= ALL (SELECT C2.cat
                    FROM Cliente C2);
```

```
+-----+
| idc |
+-----+
|  71 |
+-----+
```

  - IN es equivalente a =ANY, y NOT IN a <>ALL

Otras consultas anidadas

- alternativa a `INTERSECT`

```
SELECT C.nombre
FROM Cliente C, Vehiculo V, Reserva R
WHERE C.idc=R.idc AND R.matr=V.matr AND V.color='rojo' AND
      C.idc IN (SELECT C2.idc
                FROM Cliente C2, Vehiculo V2, Reserva R2
                WHERE C2.idc=R2.idc AND R2.matr=V2.matr
                      AND V2.color='verde');
```

—encontrar los idc's de clientes que hayan reservado un coche 'rojo' y, además, que estos idc's están incluidos en el conjunto de quienes hayan reservado un 'verde'

- alternativa a `EXCEPT` (ej. 'rojos', pero no 'verdes')

```
SELECT C.nombre FROM ... WHERE ... AND
       C.idc NOT IN (...);
```

División

- ej.: clientes que han reservado todos los vehículos
- sin la utilización de EXCEPT

```
+--------+
| nombre |
+--------+
| Davila |
+--------+
```

```
SELECT C.nombre
FROM Cliente C
WHERE NOT EXISTS (SELECT V.matr
                  FROM Vehiculo V
                  WHERE NOT EXISTS (SELECT R.matr
                                    FROM Reserva R
                                    WHERE R.matr=V.matr AND
                                          R.idc=C.idc));
```

- cliente tal que ...
- no exista un vehículo ...
- sin que exista una reserva de ese cliente con el vehículo

Operadores de agregación

- importante extensión al álgebra relacional
- COUNT, SUM, AVG, MAX, MIN

```
SELECT COUNT(*)
FROM Cliente C;
```

```
+----------+
| COUNT(*) |
+----------+
|       10 |
+----------+
```

```
SELECT AVG(DISTINCT C.edad)
FROM Cliente C
WHERE C.cat>5;
```

```
+----------------------+
| AVG(DISTINCT C.edad) |
+----------------------+
|              33.7500 |
+----------------------+
```

```
SELECT C.nombre, C.cat
FROM Cliente C
WHERE C.cat= (SELECT MAX(C2.cat)
              FROM Cliente C2);
```

```
+--------+------+
| nombre | cat  |
+--------+------+
| Zurro  |   10 |
+--------+------+
```

Cláusulas GROUP BY y HAVING

- operadores de agregación sobre grupos de tuplas
- particionado de tuplas con los atributos de GROUP BY (un grupo para cada valor)
- eliminación de grupos que no cumplan HAVING

```
SELECT C.cat, COUNT(*)
FROM Cliente C
WHERE C.edad>18
GROUP BY C.cat;
```

```
+------+----------+
| cat  | COUNT(*) |
+------+----------+
|    1 |        1 |
|    3 |        2 |
|    4 |        1 |
|    7 |        1 |
|    8 |        2 |
|    9 |        1 |
|   10 |        1 |
+------+----------+
```

```
SELECT C.cat, MIN(C.edad) AS edadmin
FROM Cliente C
WHERE C.edad>18
GROUP BY C.cat
HAVING COUNT(*)>1;
```

```
+------+---------+
| cat  | edadmin |
+------+---------+
|    3 |      25 |
|    8 |      24 |
+------+---------+
```

...

- Número de reservas de cada coche ‘rojo’

```
SELECT V.matr, COUNT(*) AS nres
FROM Reserva R, Vehiculo V
WHERE R.matr=V.matr
GROUP BY V.matr
HAVING V.color=‘rojo’;
```

```
ERROR xxxx: Unknown column
'V.color' in 'having clause'
```

  - solo pueden aparecer en `HAVING` los campos que aparecen en `GROUP BY`

```
SELECT V.matr, COUNT(*) AS nres
FROM Reserva R, Vehiculo V
WHERE R.matr=V.matr AND V.color=‘rojo’
GROUP BY V.matr;
```

```
+------+------+
| matr | nres |
+------+------+
|  102 |    3 |
|  104 |    2 |
+------+------+
```

Cláusula ORDER BY

» operador de order —en la salida

```
SELECT C.cat
FROM Cliente C
WHERE C.edad>18
ORDER BY C.cat ASC;
```

```
+------+
| cat  |
+------+
|    1 |
|    3 |
|    4 |
|    7 |
|    8 |
|    9 |
|   10 |
+------+
```

Valores nulos

» cuando el valor es desconocido / inaplicable

- Comparaciones
  - la comparación de *desconocidos* es *desconocida*
  - comparadores específicos `IS [NOT] NULL`
- Conectivas lógicas
  - lógica de tres valores: *verdadero, falso, desconocido*
- Estructuras de SQL
  - en cláusulas `WHERE` el desconocido no es seleccionado
  - aritmética con nulos devuelve nulos; las operaciones de agregación los descartan —salvo `COUNT(*)`
- Outer joins
  - se incluyen tuplas sin correspondencia, añadiendo nulos

Restricciones de integridad

- » condiciones que debe cumplir toda instancia *legal* de la BD
- » se utilizan para asegurar la semántica de la aplicación

- Tipos
  - Implícitas: propias del modelo (relacional)
    - dominio, clave primaria, clave foránea
  - Generales: condiciones de la aplicación
    - cualquier condición impuesta por los requisitos —ej.: *"no se puede contratar TV sin ADSL de alta velocidad"*
    - pueden afectar a una o a varias tablas

Restricciones sobre una tabla

- » se comprueban cuando hay una actualización de esa tabla
- » pueden hacer referencia a otras tablas

```
CREATE TABLE Reserva (
  idc ..., matr ..., fecha ...,
  PRIMARY KEY ..., FOREIGN KEY ...,
  CHECK (fecha > ...),
  CONSTRAINT nocliente
    CHECK ( 'Meloquedo' <> ( SELECT C.nombre
                             FROM Cliente C
                             WHERE C.idc=idc )));
```

– se pueden utilizar consultas para expresar restricciones

Restricciones sobre varias tablas

- Aserciones
  - implica a dos o más tablas, sin asociación concreta a una de ellas
  - se comprueba cuando se modifica alguna de ellas

```
CREATE ASSERTION limite
CHECK ( (SELECT COUNT(*) FROM Cliente C) +
        (SELECT COUNT(*) FROM Vehiculo V) < 100 );
```

  - se puede implementar con un `CHECK` pero sería engorroso y con el problema de tener la tabla de clientes vacía, p.e.

Disparadores / *triggers*

» procedimiento que se dispara automáticamente si se producen cambios específicos (SQL-99)

- Partes
  1. Evento: activa el disparador
  2. Condición: instrucción (falso|verd.) o consulta (vacía|no)
  3. Acción: procedimiento que se ejecuta cuando se activa el disparador y la condición es verdadera

```
CREATE TRIGGER actualizarTitularActual
  AFTER INSERT ON HistorialLicencia
REFERENCING NEW AS nuevoTraspaso
FOR EACH ROW
  UPDATE Licencia L
    SET L.titularActual = nuevoTraspaso.dniTitular
    WHERE L.nro = nuevoTraspaso.nroLicencia;
```

Diseño de Bases de Datos

Refinamiento de esquemas y Normalización

El problema de la redundancia

» el punto de partida es el esquema relacional, que incluye las restricciones de integridad

- Redundancia: almacenar información en más de un sitio
  - Almacenamiento redundante
  - Posibilidad de inconsistencia
  - Anomalías
    - actualización
    - inserción
    - borrado

...

Empleado

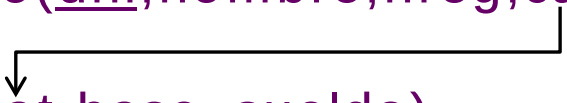
dni	nombre	nreg	cat	horas_sem	base_sueldo
2354	García	48	8	40	10
9625	Aragón	22	8	30	10
5557	Pozo	35	5	35	7
2121	Sainz	35	5	40	7

- la combinación 8 -> 10 y 5 -> 7 es redundante
- actualización de base_sueldo: en todas las tuplas
- inserción de empleado: sólo si se sabe la base de su categoría —dejar *null* ?
- borrado de todas las tuplas de una categoría: se pierde la base_sueldo

Descomposición

» sustituir una relación por otras dos (o más) cada una con un subconjunto de campos y, conjuntamente, incluyendo todos los originales

Empleado(dni,nombre,nreg,cat,horas_sem)



Sueldo(cat,base_sueldo)

- Requisitos descomposición
  - Reunión sin pérdida
  - Conservación de las dependencias
  - Consultas pueden obligar a la reunión de las relaciones descompuestas
    - penalización en el rendimiento
    - ¿y si la eficiencia resultante no fuera aceptable?

Dependencias funcionales

» generaliza el concepto de clave

DF, $X \rightarrow Y$

$\forall\ t1,t2 : R \mid t1.X = t2.X \Rightarrow t1.Y = t2.Y$

DF, $AB \rightarrow C$

- y sin embargo AB no es clave

A	B	C	D
1	1	1	1
1	1	1	2
1	2	2	1
2	1	3	2

DF, clave $\rightarrow$ todos los atributos —*definición relacional*

- la dependencia no exige que la clave sea mínima

DF, $X \rightarrow$ todos los atributos

- si X no es mínimo, entonces es una *superclave*

Formas Normales

» si el esquema se encuentra en una de estas formas normales, ciertos tipos de problemas serán eliminados / minimizados

- Basadas en DF
  - 1NF, 2NF, 3NF, BCNF
  - 3NF y BCNF son importantes en el diseño de bases de datos
- Basadas en otros tipos de dependencias
  - multivaluadas (DM): 4NF, reunión (DR): 5NF

BCNF

» —*intuitivamente*— las únicas dependencias son aquellas en las que una clave determina algún atributo; no se pueden inferir valores

- Rel R, DFs F, subconj. atributos X, atrib. A
  R está en BCNF si, para cada DF X $\rightarrow$ A,
  - A $\in$ X, DF *trivial*, o
  - X contiene una clave de R
- Ej. no BCNF
  - Empleado(dni,nombre,nreg,cat,horas_sem,base_sueldo)
    DF cat $\rightarrow$ base_sueldo
  - Reserva(<u>dni</u>,<u>matr</u>,<u>fecha</u>,tarjeta), dni ^{1}—^{1} tarjeta
    DF dni $\rightarrow$ tarjeta
- Solución por descomposición

3NF

- Problema *técnico* con BCNF —múltiples claves candidatas solapadas
  - Reserva(<u>dni</u>,<u>matr</u>,<u>fecha</u>,tarjeta), dni ^{1}—* tarjeta
    - tengo que anotar la tarjeta con la que se pagó, pero
    - DF tarjeta $\rightarrow$ dni, y esto *<u>no</u>* cumple BCNF
    - <u>tarjeta</u>,<u>matr</u>,<u>fecha</u> podría ser clave primaria
- Rebajo las condiciones y permito una cierta redundancia: 3NF
  - tercera posibilidad: A es parte de alguna clave de R

Descomposición de una relación

» en dos (o más) que se *reparten* los atributos, garantizando *reunión sin pérdida* y *conservación de dependencias*

- Algoritmo de descomposición en BCNF
  1. R no en BCNF, $X \subset R$, A atributo, $X \rightarrow A$ que provoca el no cumplimiento
  2. Descomponer en R-A y XA
  3. Si R-A o XA no en BCNF, aplicar recursivamente
- Ej. Contratos con atributos CPYDRNV (idC,idProve,idProy,idDepto,idRepuesto,cantidad,valor) y DFs PD $\rightarrow$ R, Y $\rightarrow$ P

...

1. guiado por PD $\rightarrow$ R
   - PDR y CPYDNV, ahora aplicar para Y $\rightarrow$ P
   - PDR, YP y CYDNV
2. guiado por Y $\rightarrow$ P
   - YP y CYDRNV
   - la otra dependencia ya no desnormaliza

- Distintas alternativas, cuya elección se hará respecto a la semántica de la aplicación

ER -> Rel -> Rel Normalizado

- ¿Se puede directamente producir un ER libre de problemas de redundancia?
  - ER es un proceso complejo y subjetivo (piénsese en esquemas con más de 100 tablas)
  - Determinadas restricciones / dependencias no se pueden expresar en ER (al menos fácilmente)
  - Técnica formal para tratar un diseño original que da como resultado un diseño mejor

Diseño de Bases de Datos

Introducción a la Administración de BD

Tipos de usuarios

- Distintos tipos de usuarios / responsabilidades / competencias
  - Usuario de BD
    - interactúan con la BD a través de aplicaciones
  - Desarrollador (de aplicaciones)
    - diseño y desarrollo de la aplicación, incluyendo la *estructura* de la BD
    - estimación de requisitos de almacenamiento, eficiencia, seguridad, etc.
    - ¿quién *cuida* del DBMS que permita hacer todo esto de forma adecuada?

...

- Administrador de la BD (DBA)
  - instala y actualiza el DBMS y las herramientas asociadas (¿decidir el DBMS a utilizar?)
  - establece y reserva el sistema de almacenamiento
  - crea los objetos primarios (tablas, vistas, índices), una vez que los desarrolladores han realizado el diseño
  - modifica la estructura de la base de datos, integrando la información de distintas aplicaciones
  - gestiona el acceso de usuarios y sus privilegios
  - establece políticas de seguridad y las monitoriza
  - monitoriza y optimiza el rendimiento de la BD
  - establece procedimientos de copias / recuperaciones
  - gestiona el soporte técnico con la distribuidora del DBMS

Tareas típicas

- Instalar el DBMS (inc. evaluación del hw)
  - estructura de almacenamiento (Oracle): *tablespaces*, archivos de datos, discos de almacenamiento
  - efecto en el rendimiento general del sistema
- Creación, arranque y parada de BD
  - *layout* de ficheros, inicialización, tamaño de bloque, ficheros de log, ...
- Gestión de usuarios
  - establecimiento de políticas de seguridad, gestión de usuarios y recursos

...

- Implementación del diseño (lógico)
  - objetos del esquema: tablas, campos, claves, vistas, ...
  - estructuras por defecto, índices primarios, ... (d. físico)
- Optimización del rendimiento / ajuste / *tuning*
  - optimización de consultas: árboles relacionales, rutas de acceso y planes de ejecución
  - optimización sintáctica, semántica y estadística; reglas, costes y *pistas*
  - optimización de las estructuras de datos del diseño físico: índices y su influencia en las consultas

Análisis y Diseño de Bases de Datos

Técnicas avanzadas de gestión de la información

Datos estructurados

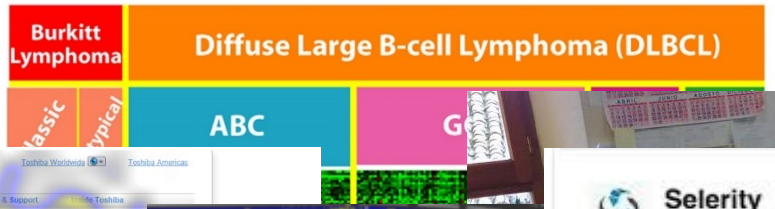
- Datos organizados en entidades (*muy*) similares
- Estructura = <entidades,atributos,relaciones>
- Esquema relacional: tablas (relación), filas (entidad), columnas (atributo), claves foráneas (asociaciones)
- Manipulación de datos: formalizada (álgebra/cálculo relacional), lenguaje (SQL)

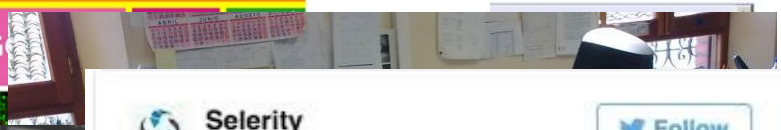
- Esquema predefinido
- Consistencia estricta
- Escalado *central* (*scaling up*)
- No es flexible / versátil

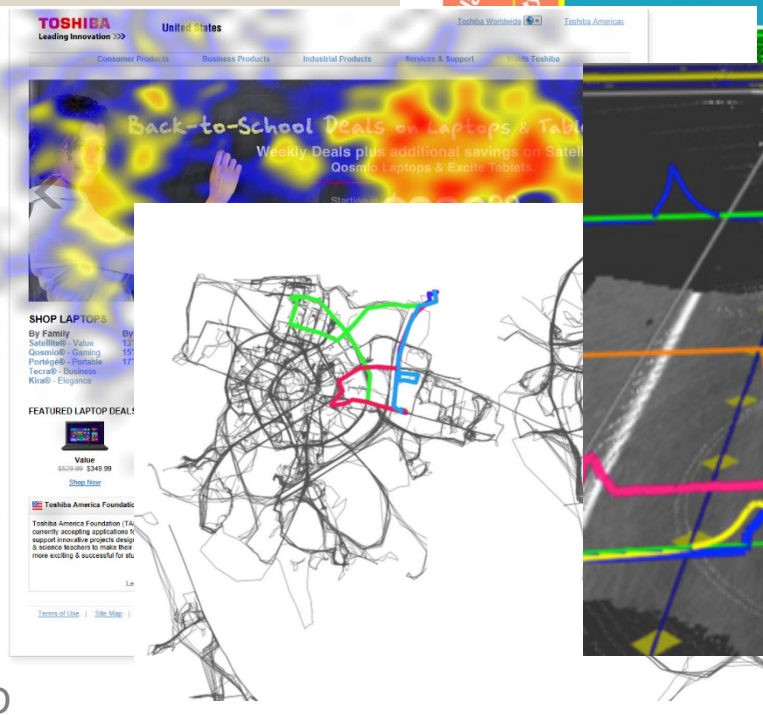
. . .

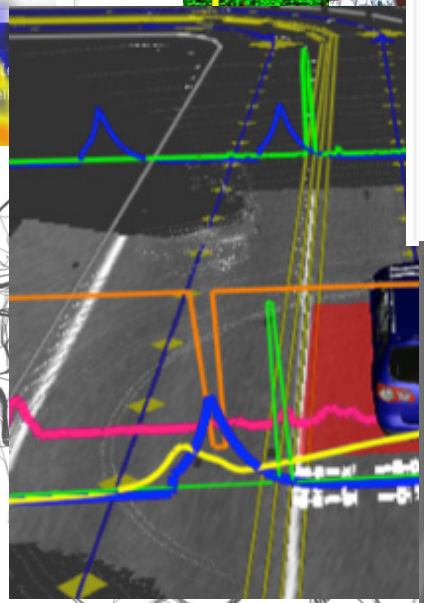
Table 7 Top 10 cancer incidence rates in cancer registration areas in 2009

Rank	Both					
	Site	Incidence rate ($1/10^5$)	%	ASIRC* ($1/10^5$)	Site	
1	Lung (C33-34)	53.57	18.74	25.34	Lung (C33-34	
2	Stomach (C16)	36.21	12.67	17.85	Stomach (C16	
3	Colon, rectum	29.44	10.30	14.21	Liver (C22)	



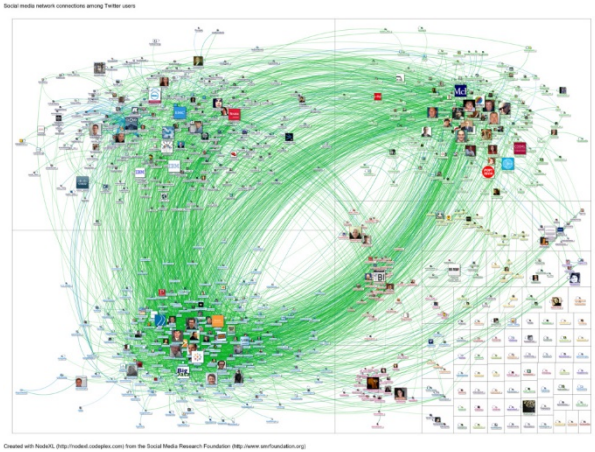




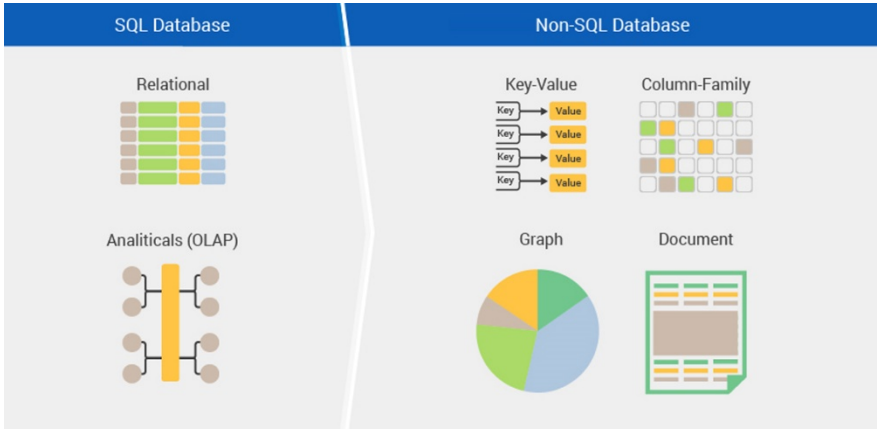


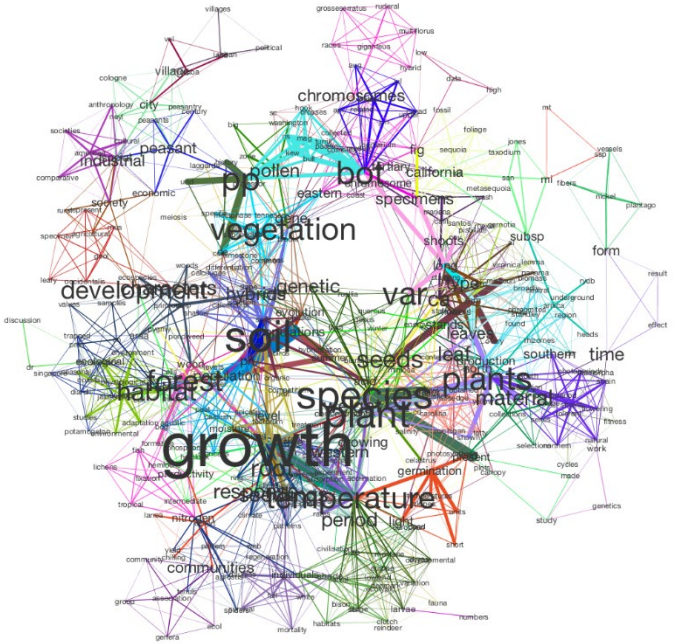
La revolución de los datos

- Big data



- NoSQL





- Web semántica / linked data

Big data

- Big data
  - grandes colecciones de datos complejos
  - procesos complejos:
    - captura, integración, limpieza, anotación
    - almacenamiento, disponibilidad, acceso
      —50-80% trabajo antes de que puedan utilizarse
    - análisis, visualización, modelos de conocimiento
  - ámbitos de aplicación: economía, sociología, salud, industria, administraciones públicas, …
  - alinear big data con objetivos específicos: qué datos capturar, cómo tratarlos, para obtener qué
  - Vs: *volume, velocity, variety —veracity, value*

5 Vs

Velocity + Volume + Variety + Veracity + Viability = Value

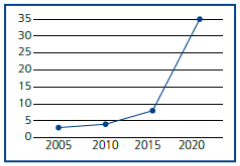
Velocity
Frequency of data generation

300 hours
of video uploaded to YouTube every minute (estimate)

2,400,000 search queries
per minute on Google

4,170,000 posts liked
on Facebook per minute

Volume
The growth of world data



1 terabyte holds the equivalent of roughly 210 single sided DVDs.

Variety
Structured and unstructured data - types of Big Data


Web and social media


Machine to machine


Big transaction data


Biometric


Human-generated

Veracity
Establishing trust in data


1 in 3
business leaders don't trust the information they use


Uncertainty
due to inconsistency, ambiguity, latency and approximation

Viability
Relevance and feasibility

"Can we use mobile phone data to monitor cross-border tourism?"

Hypothesis
Validation to determine if the data will have a meaningful impact


Long-term
rewards and better outcomes from hidden relationships in data

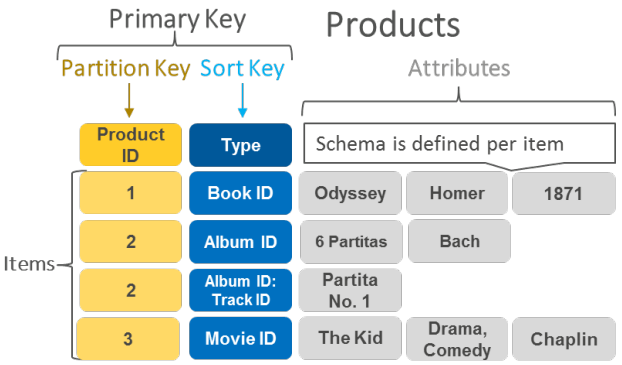
Value
Return on investment


Costs
Risk of simply creating Big Costs without creating the value

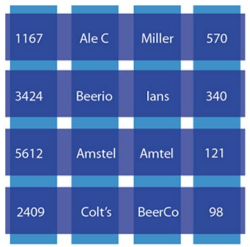

Insights
Sophisticated queries, counterintuitive insights and unique learning

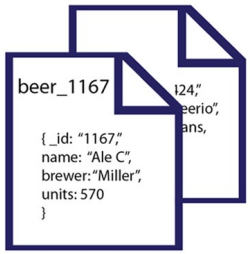
GII/ADBD

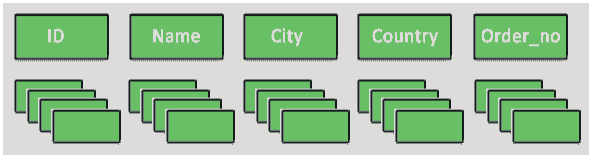
NoSQL

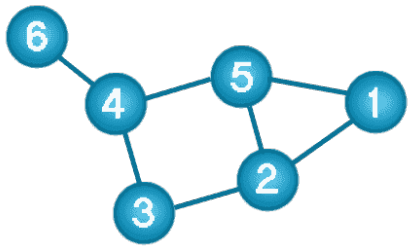


- Tipos
  - *key-value*: datos como colección indexada de registros / campos / datos (diccionario); sencillos modelos y accesos por clave
  - *document*: datos semi-estructurados en formato documento; atributos/cabeceras con grandes volúmenes de datos documentales
  - *column:* tratamiento orientado a columna (= tipo datos) rendimiento en agregaciones, datos derivados, business intelligence
  - *graph:* entidades conectadas aplicaciones con representación de conectividad, interacción, etc.









...

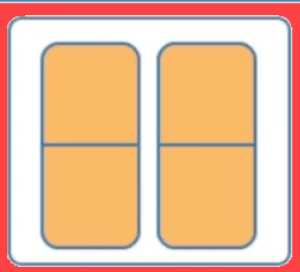
Key Value

Example:
Riak, Tokyo Cabinet, Redis server, Memcached, Scalaris

Document-Based

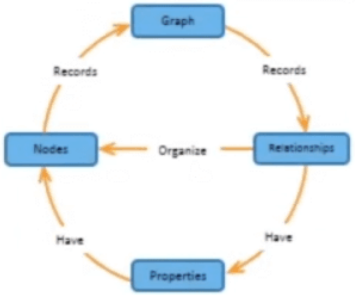
Example:
MongoDB, CouchDB, OrientDB, RavenDB

Column-Based



Example:
BigTable, Cassandra, Hbase, Hypertable

Graph-Based



Example:
Neo4J, InfoGrid, Infinite Graph, Flock DB

Semántica

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE recipe PUBLIC "~//Happy-Monkey//DTD RecipeBook//EN"
"http://www.happy-monkey.net/recipebook/recipebook.dtd">

<recipe>

    <title>Peanut-butter On A Spoon</title>

    <ingredientlist>
        <ingredient>Peanut-butter</ingredient>
    </ingredientlist>

    <preparation>
        Stick a spoon in a jar of peanut-butter,
        scoop and pull out a big glob of peanut-butter.
    </preparation>

</recipe>
```

- Web semántica
  - contenido semántico, procesable por máquinas
  - web de los datos —vs web de los documentos
  - componentes (habituales)
    - XML, sintaxis básica de estructuración de contenidos
    - XML Schema, proporciona y restringe la estructura de contenidos
    - RDF, modelo de datos en forma de <sujeto, propiedad, objeto>; normalmente en sintaxis XML
    - RDF Schema, vocabularios de propiedades
    - OWL, ontologías / conocimiento
    - SPARCQL, lenguaje de consulta
- Linked data
  - reformulación de la web semántica