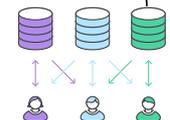


# Bases de Datos Relacionales y SQL: Una Introducción

José María Fernández González

Nodo de Coordinación, INB  
Life Sciences Department, BSC



José M<sup>a</sup> Fernández (jose.m.fernandez@bsc.es)



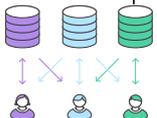
Cofinanciado por  
la Unión Europea





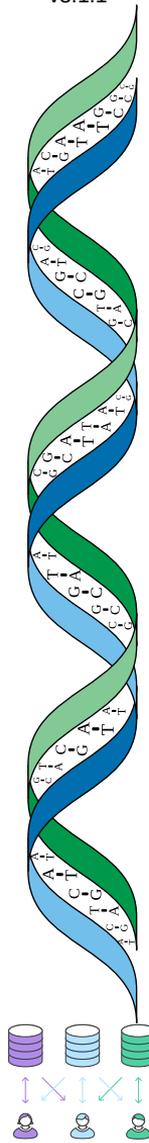
# ¿Qué es una base de datos?

- Una base de datos es una colección de información, normalmente organizada, sobre la que se pueden realizar búsquedas estructuradas (por ejemplo, con el software apropiado).
- Según <https://tudip.com/blog-post/7-database-paradigms/>, existen principalmente 7 paradigmas de bases de datos:
  - ➔ Clave → Valor
  - ➔ “Columna amplia” (una extensión del anterior)
  - ➔ Documentales: documentos estructurados.
  - ➔ Relacionales.
  - ➔ Orientadas a grafos (ejemplos, *linked data*, semánticas, ...).
  - ➔ Indexadas por contenido (estructurado y no estructurado).
  - ➔ Multimodelo.



# ¿Qué es una base de datos relacional?

- Una base de datos relacional es aquella gestionada mediante un **Sistema Gestor de Base de Datos Relacional**.
- Estos sistemas gestores son el software que median en el uso de las bases de datos relacionales, controlando y optimizando el acceso al contenido de las mismas.
  - ➔ El almacenamiento físico de los datos lo gestiona única y exclusivamente el gestor de la base de datos. El usuario sólo debería preocuparse de la estructura lógica de los mismos.
  - ➔ La manipulación de la estructura y contenido de una base de datos relacional se realiza mediante el lenguaje de consultas SQL (*Structured Query Language*).



# Algunos gestores de bases de datos relacionales

<https://hpi.de/naumann/projects/rdbms-genealogy.html> Genealogy of Relational Database Management Systems

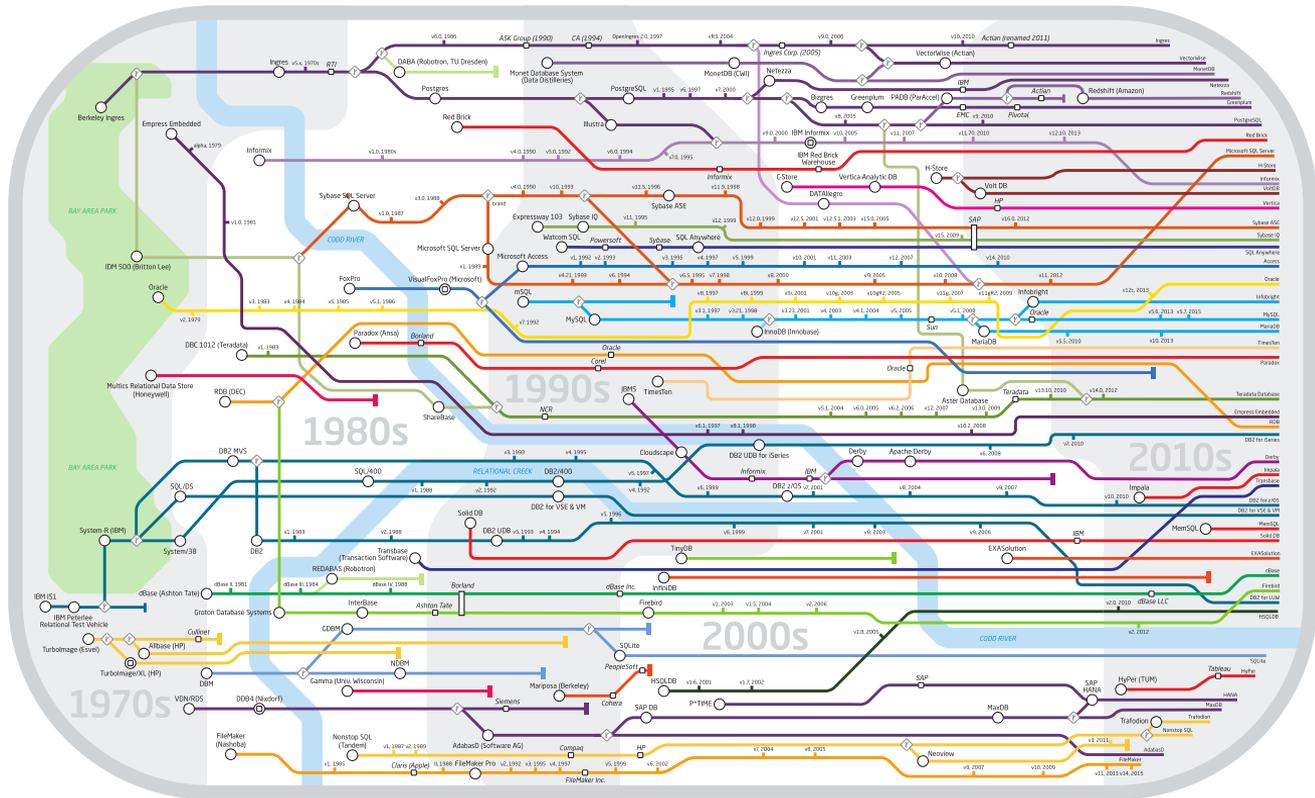
Actualmente existen decenas de sistemas gestores de bases de datos relacionales, habiendo soluciones para todos los escenarios:

- Gestores *open source*:

- PostgreSQL
- MariaDB / MySQL
- SQLite

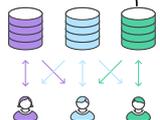
- Gestores de pago:

- Oracle
- SAP Sybase
- IBM Db2

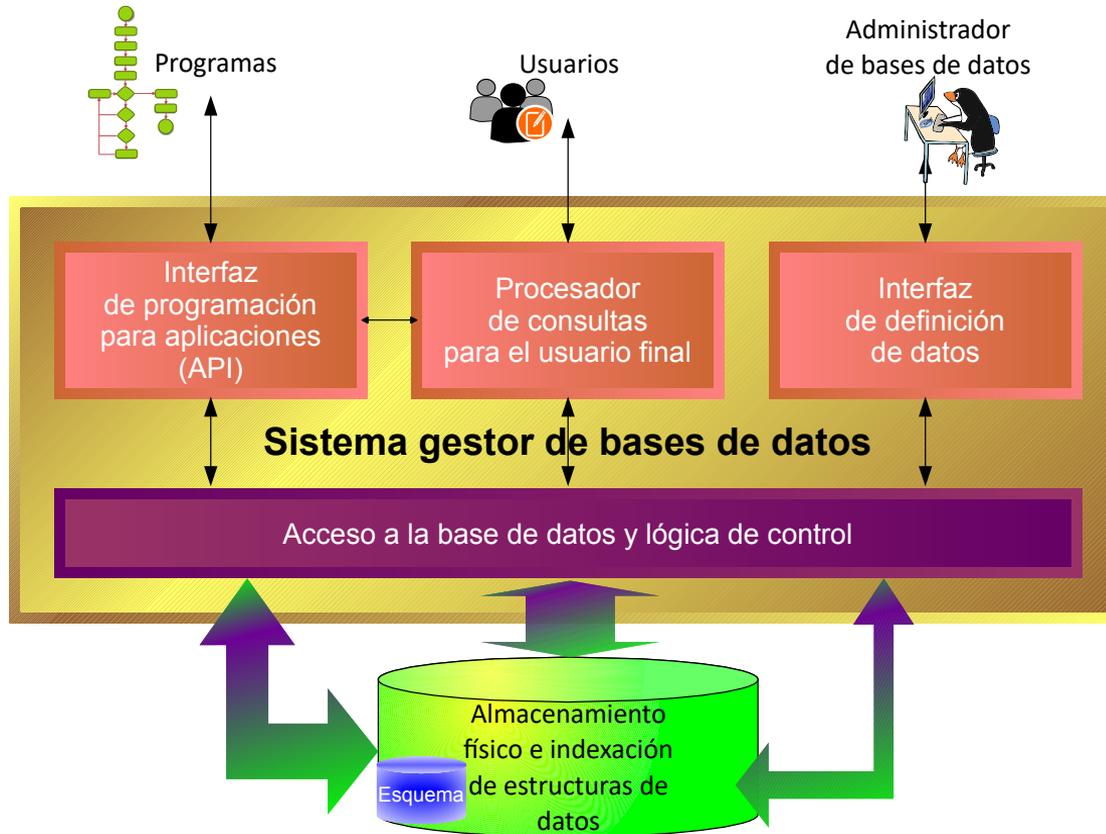


# “Usuarios” de gestores de bases de datos

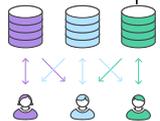
- Los sistemas gestores de bases de datos (principalmente cliente / servidor, pero también algunas embebidas) permiten tener “usuarios” autenticados con distintos privilegios de acceso a los contenidos. 
- Los “usuarios” de una base de datos no están necesariamente ligados con los usuarios del sistema.
  - ➔ Un 'administrador' crea los “usuarios”, y les otorga o deniega privilegios (operaciones que pueden realizar): 
    - crear, modificar o borrar una tabla; consultar, insertar, borrar o modificar los datos de una tabla; consultar o crear una vista; crear usuarios o grupos; otorgar privilegios; etc...



# Estructura interna de un sistema gestor

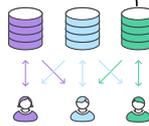
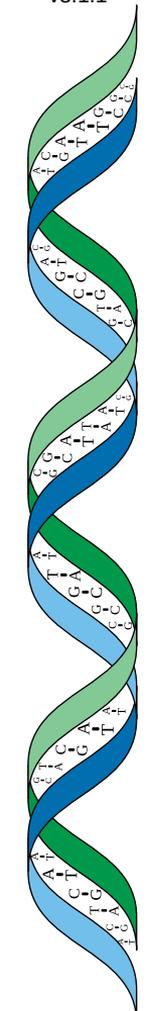


Los programas que quieran consultar una base de datos usarán librerías que implementen todos los detalles de comunicación con el gestor que la administre



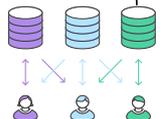
# Structured Query Language

- Es el idioma 'estándar' que se emplea para consultar y modificar bases de datos relacionales.
- A pesar del proceso de estandarización de SQL, los desarrolladores de cada gestor de bases de datos sólo suelen cumplir con aquellas partes más comunes, y además añadir extensiones propias.
- Por ello, nos encontramos con muchos 'dialectos' SQL muy parecidos entre sí a nivel semántico, pero a veces sintácticamente incompatibles.



# Partes del lenguaje SQL

- Definición (*Data Definition Language, DDL*): Es la parte del lenguaje que se ocupa de la gestión de la base de datos: creación y borrado de los usuarios, tablas, vistas, etc...; gestión del control de acceso; manipulación de la estructura de las tablas; optimización del acceso a los datos; tipos de datos...
- Manipulación (*Data Manipulation Language, DML*): Es la parte del lenguaje SQL que se ocupa de las operaciones de inserción, borrado, actualización y consulta de datos.



# ¿Que conceptos definen los estándares SQL y manejan los gestores de bases de datos?

- Tipos de datos.

**INTEGER, VARCHAR, DATE, REAL**



- Datos, organizados en tuplas.

- Tablas+columnas.



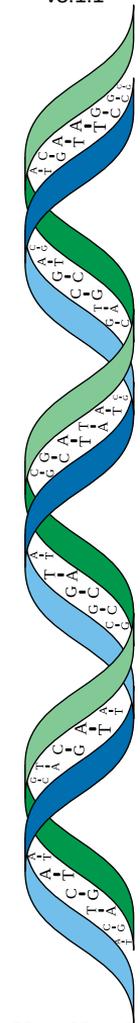
- Usuarios.



- Restricciones.



- Índices y otros elementos.



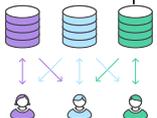
# Tablas: Introducción

A muy grandes rasgos, una tabla SQL es como un *data frame* de R o una hoja de cálculo con restricciones. Cada tabla tiene un nombre, está estructurada en una o más columnas, y puede tener una o más restricciones asociadas a la misma.

AccNumber CHAR(6) PRIMARY KEY	Identificador VARCHAR(256) NOT NULL	Descripción VARCHAR(4096)
A1WWE5	ACP_HALHL	Acyl carrier protein
A8ESU2	RS10_ARCB4	30S ribosomal protein S10
Q9X2A1	ASSY_THEMA	Argininosuccinate synthase
A8FEJ8	TRPB_BACP2	Tryptophan synthase beta chain
A1JJ31	AMPA_YERE8	Probable cytosol aminopeptidase
Q83EL7	KGUA_COXBU	Guanylate kinase
P48307	TFPI2_HUMAN	Tissue factor pathway inhibitor 2
Q1CC21	MALK_YERPA	Maltose/maltodextrin import ATP-binding protein MALK
A9SU70	U4976_PHYPA	UPF0497 membrane protein 6

Tabla Proteína

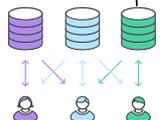
Una base de datos relacional está compuesta de varias tablas relacionadas entre sí



# Tablas: Tuplas y Columnas

AccNumber	Identificador	Descripción
CHAR(6) PK	VARCHAR(25) NOT NULL	VARCHAR(4096)
A1WWE5	ACP_HALHL	Acyl carrier protein

- El contenido de cada tabla SQL está compuesto por 0 o más tuplas (filas) de tantos elementos como columnas tenga la tabla. Cada uno de los valores de la tupla se rige por una columna en concreto de la tabla.
- En cada tupla, una columna puede tener asociado a lo sumo un valor.
- Cada columna tiene nombre, y un tipo de datos.
- Cada columna puede participar en una o varias restricciones.
- Las restricciones básicas de una columna son: de contenido nulo, de restricciones de contenido.
- Se puede asignar a una columna una expresión por omisión. Se emplea cuando se guarda una tupla en la que no se haya dado explícitamente un valor a esa columna.

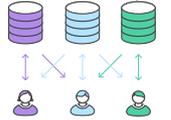
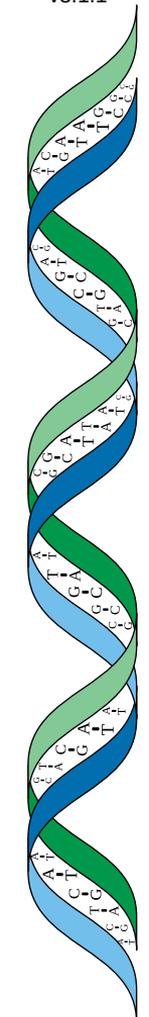
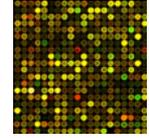


# Tablas: Tipos y Valores SQL

(varía de gestor a gestor de base de datos)

- INTEGER 3, -2
- CHAR 'S', 'KP'
- VARCHAR 'QLF'
- BOOLEAN TRUE, FALSE
- TIMESTAMP Fri Nov 21 18:51:39 CET 2014
- DATE 2013-11-29
- TIME 17:25:38
- NUMERIC 0.0315600
- REAL ≈, 1415
- CLOBs
- BLOBs
- NULL
- etc...

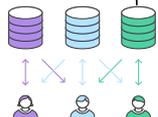
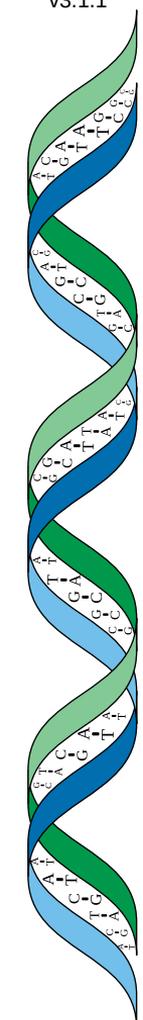
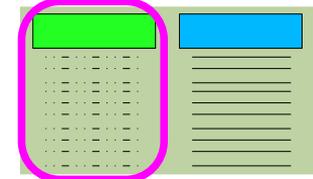
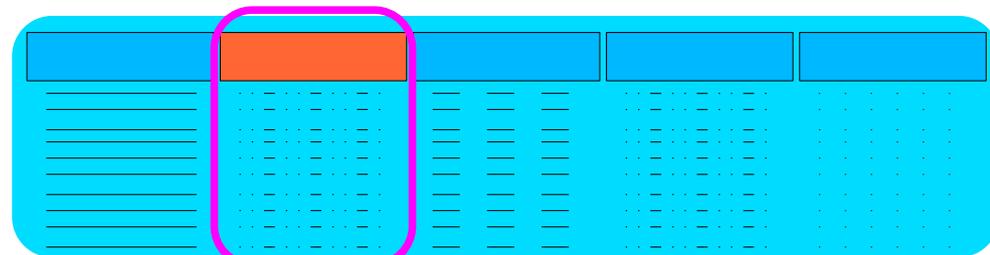
En un lugar de la Mancha de cuyo nombre no quiero acordarme vivía un hidalgo de los de espada y rocín, conocido como



# SQL: Creación de tablas

```
CREATE TABLE SWISSTABLE (
  id VARCHAR(10) NOT NULL,
  accnumber VARCHAR(7) NOT NULL,
  secuencia TEXT NOT NULL,
  molweight NUMERIC(8,2),
  description VARCHAR(255),
  PRIMARY KEY (accnumber),
  UNIQUE(id)
);
```

```
CREATE TABLE REL_SWISS_PDB (
  accnumber_r VARCHAR(7) NOT NULL,
  pdbcode VARCHAR(8) NOT NULL,
  FOREIGN KEY TOSWISS (accnumber_r) REFERENCES
    SWISSTABLE (accnumber)
    ON DELETE CASCADE ON UPDATE CASCADE,
  FOREIGN KEY TOPDB (pdbcode) REFERENCES
    PDB_TABLE (pdb_id)
    ON DELETE RESTRICT ON UPDATE CASCADE
);
```

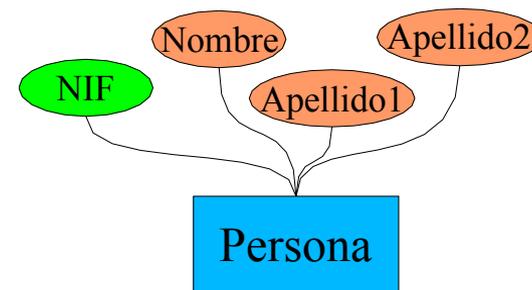


# Tabla: Restricciones

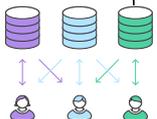
Una restricción es una premisa que siempre se debe cumplir. Por ello, los datos almacenados en una tabla siempre deben cumplir todas las restricciones de dicha tabla.

Existen varios tipos de restricciones

- De columna (explicado anteriormente)
- De clave única
- De clave primaria
- De clave externa
- Otras...



No todos los gestores de bases de datos permiten cambiar las restricciones en tablas ya creadas.



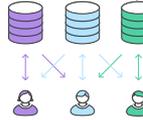
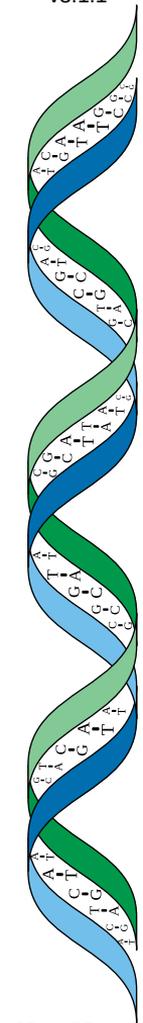
# Restricción de Clave Única

- Esta restricción se construye sobre una o más columnas, y obliga a que los valores asociados a esas columnas sean únicos. Por ejemplo:

(nombre, apellido1, apellido2)

podría definir una clave única, de forma que no pudiese haber dos personas con el mismo nombre y apellidos.

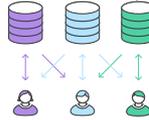
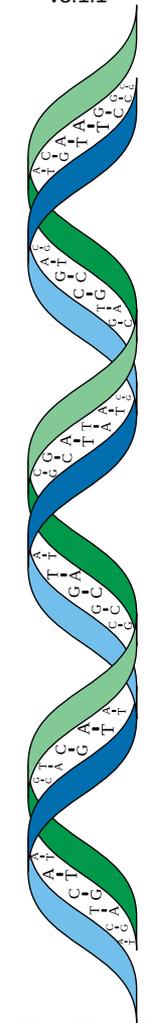
- Una tabla puede tener más de una restricción de clave única. Por ejemplo, una clave única sobre el NIF y otra sobre el número de pasaporte.



# Restricción de Clave Primaria

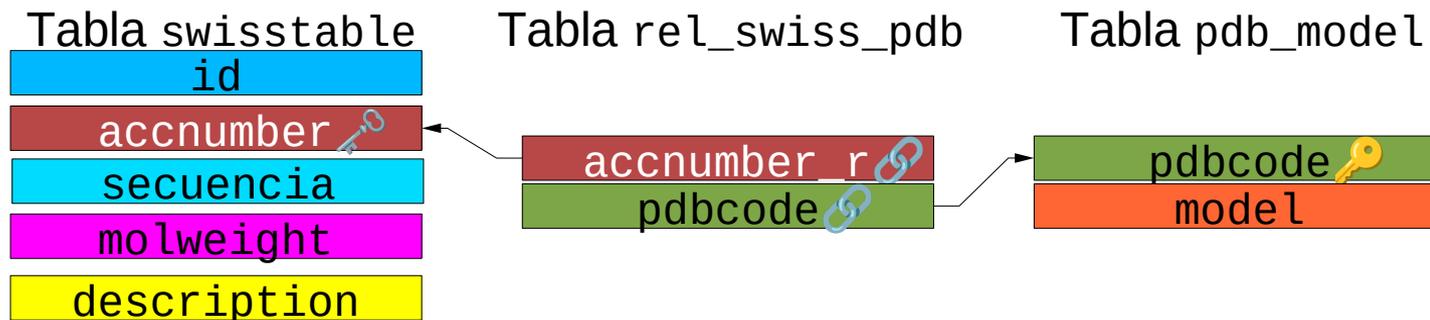
- Este tipo de restricciones es similar en concepto a las de clave única. Adicionalmente, los valores que toman las columnas de la clave primaria en cada tupla se emplean para identificar dicha tupla de forma lógica.
- Sólo se puede definir una clave primaria por tabla. En caso de existir varios candidatos a clave primaria, lo más conveniente es elegir el más representativo para el contexto de uso.

Por ejemplo, para un coche, tanto la matrícula como el nº de bastidor se podrían emplear como clave primaria.

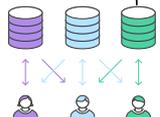


# Restricción de Clave Externa

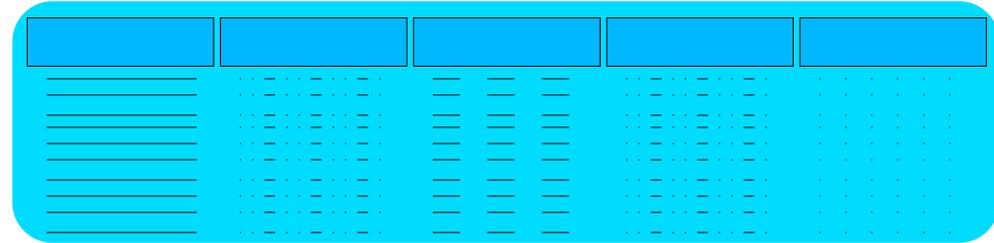
- Las restricciones de clave externa (o de clave foránea) sirven para mantener la coherencia entre los datos almacenados en distintas tablas. Se establecen desde campos de una tabla a los campos de clave primaria de otra.



- Por ejemplo, una base de datos con la tabla `swisstable` y la tabla `pdb_model`, que relacionan sus contenidos a través de la tabla `rel_swiss_pdb`. Para mantener la coherencia, los cambios en el `accnumber` de alguna entrada de `swisstable` o bien estarán prohibidos, o bien provocarán un cambio automático en las entradas de `rel_swiss_pdb` con el mismo `accnumber_r`.



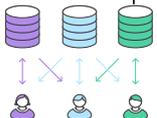
# Borrado de tablas. Permisos



```
DROP TABLE REL_SWISS_PDB;
```

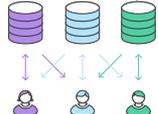
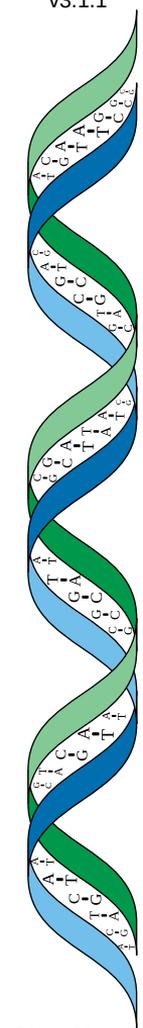
```
GRANT SELECT, INSERT, UPDATE, DELETE  
ON SWISSTABLE TO pepe;
```

```
REVOKE SELECT ON SWISSTABLE TO PUBLIC;
```



# Manipulación de datos

- Una vez definida la estructura de las tablas, podremos insertar, actualizar, borrar y consultar datos en ellas. 
- Las consultas serán las operaciones más realizadas, tanto para recuperar información previamente almacenada, como para calcular estadísticas o extraer conclusiones de los datos almacenados. 
- Un conjunto de operaciones de manipulación de datos se puede realizar en transacción, para garantizar la coherencia de las mismas y de la base de datos.



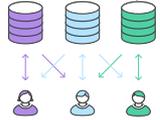
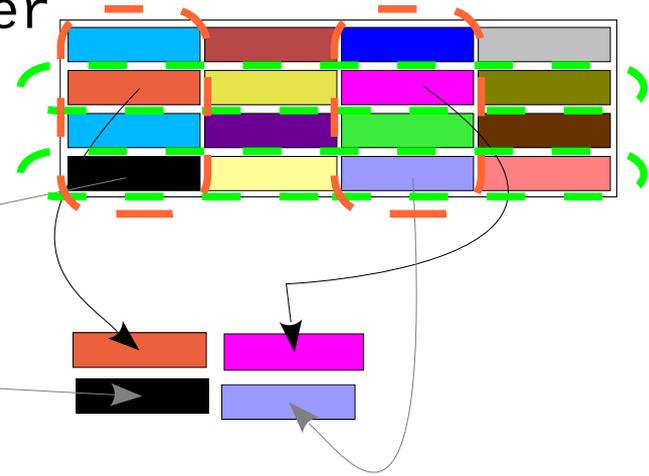
# SQL: Recuperando datos (I)

## Consulta normal

```

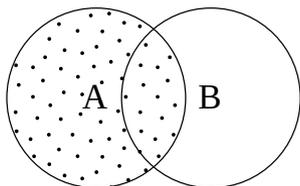
SELECT p.pdbcode, s.id AS "Swissprot ID"
FROM rel_swiss_pdb p, swisstable s
WHERE p.accnumber_r = s.accnumber
AND description LIKE '%3D%';

```

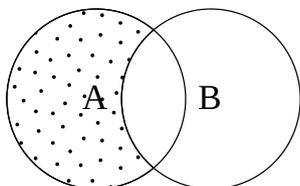


# SQL: Recuperando datos (II). Joins

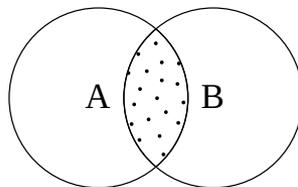
## SQL Joins



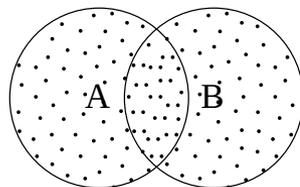
```
SELECT <select_list>
FROM TablaA A
LEFT JOIN TablaB B
ON A.claveA = B.claveB
```



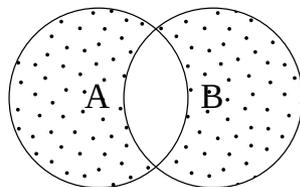
```
SELECT <select_list>
FROM TablaA A
LEFT JOIN TablaB B
ON A.claveA = B.claveB
WHERE B.claveB IS NULL
```



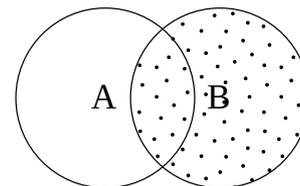
```
SELECT <select_list>
FROM TablaA A
INNER JOIN TablaB B
ON A.claveA = B.claveB
```



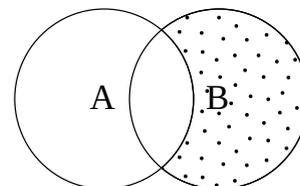
```
SELECT <select_list>
FROM TablaA A
FULL OUTER JOIN TablaB B
ON A.claveA = B.claveB
```



```
SELECT <select_list>
FROM TablaA A
FULL OUTER JOIN TablaB B
ON A.claveA = B.claveB
WHERE A.claveA IS NULL
OR B.claveB IS NULL
```



```
SELECT <select_list>
FROM TablaA A
RIGHT JOIN TablaB B
ON A.claveA = B.claveB
```

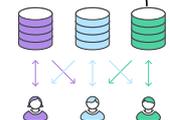


```
SELECT <select_list>
FROM TablaA A
RIGHT JOIN TablaB B
ON A.claveA = B.claveB
WHERE A.claveA IS NULL
```

Los tipos de consulta más comunes son los *inner joins*, donde se correlacionan entradas de varias tablas por la igualdad de los valores de columnas en una y otra tabla para las entradas.

Los *left joins* son muy útiles para intentar saber qué entradas no tienen correlación con otras en la otra tabla.

(inspirado en [https://commons.wikimedia.org/wiki/File:SQL\\_Joins.svg](https://commons.wikimedia.org/wiki/File:SQL_Joins.svg))



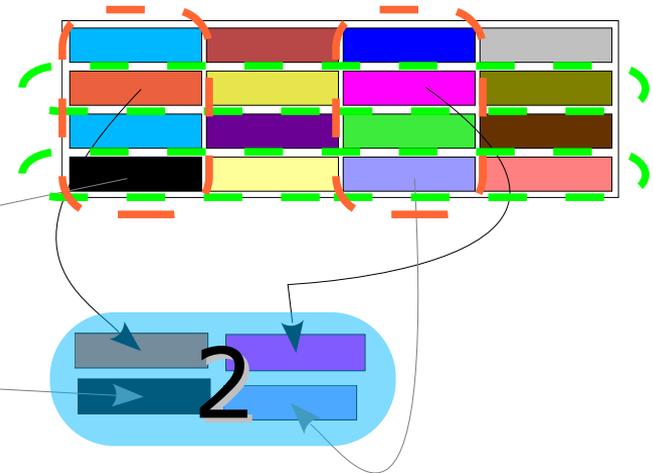
# SQL: Recuperando datos (III). Agregaciones

Consulta de agregación simple

```
SELECT COUNT(*)
FROM SWISSTABLE
WHERE LENGTH(secuencia) > 100;
```

Consulta agregación con *join*

```
SELECT COUNT(*)
FROM rel_swiss_pdb p, swisstable s
WHERE p.accnumber_r = s.accnumber
AND description LIKE '%3D%';
```



# SQL: Recuperando datos (IV)

Consulta de agregación ampliada

```
SELECT (s.id, COUNT(p.pdbcode))
FROM rel_swiss_pdb p, swisstable s
WHERE (p.accnumber_r = s.accnumber)
AND (description LIKE '%3D%')
GROUP BY 1;
```

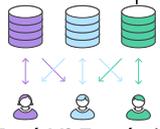
Id	COUNT
Id_A	2
Id_B	1
Id_C	3

Id	Accnumber	Secuencia	Molweight	Description
Id_A	A	QWEF	38	Blah 3D
Id_B	B	ADSFQ	174	3D Bleh
Id_C	C	SGF	23	BI 3D ih
Id_D	D	WEWH	229	Bloh
Id_E	E	NMEGY	151	Bluh
Id_F	F	PEUUh	79	Jaja 3D 2

Tabla swisstable

Accnumber_r	Pdbcode
A	P
C	Q
E	R
B	S
C	T
C	U
A	V

Tabla rel\_swiss\_pdb



# SQL: Manipulación de datos

## Inserción

```
INSERT INTO SWISSTABLE VALUES  
  ('P98765', NULL, 'LSQSDARESM', 18.15, 'ID_RAT');
```

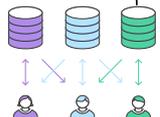
```
INSERT INTO SWISSTABLE (accnumber, id, molweight, secuencia)  
  VALUES ('P98765', 'ID_RAT', 18.15, 'LSQSDARESM');
```

## Borrado

```
DELETE FROM SWISSTABLE  
  WHERE accnumber LIKE 'P98%';
```

## Actualización

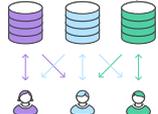
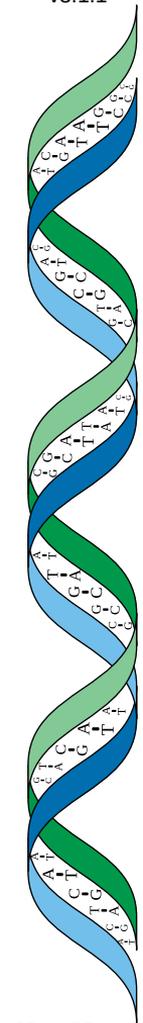
```
UPDATE SWISSTABLE SET  
  molweight = molweight + 1.0  
  WHERE description IS NULL;
```



# ¿Cuál es el mejor sistema gestor?

Si buscáis el mejor sistema gestor, la respuesta es otra pregunta:  
**¿qué es lo que realmente necesitáis hacer?**

- En Bioinformática, son muy populares tanto [PostgreSQL](#) como [MariaDB/MySQL](#) (desarrollados en torno al paradigma cliente / servidor). Ello es debido a que pueden ser instalados prácticamente en cualquier plataforma hardware o cloud sin ningún coste adicional (la licencia de uso es gratuita), salvo el esfuerzo de instalarlo.
  - El principal éxito de PostgreSQL: la simplicidad y muchas de las características, potencia y escalabilidad de los gestores de pago de bases de datos.
  - El principal éxito de MySQL/MariaDB: su gran simplicidad y su velocidad de acceso para consultas sencillas.
- [SQLite](#) es un gestor embebido, no depende de un servidor dedicado. Por su rendimiento y bajos requisitos de memoria, se usa cada vez más, de forma general, en todo.
  - No es la implementación más correcta. Distintos niveles de corrección son activables mediante *PRAGMAS* , *STRICT* tables y otras declaraciones.

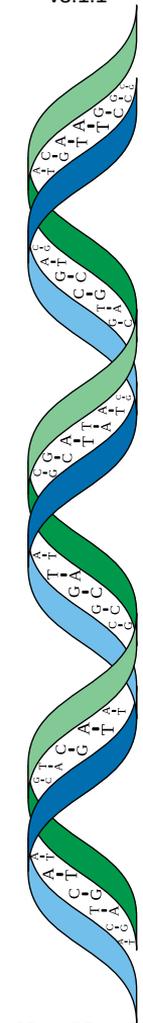




# Diseño de una base de datos

## ¿Arte o Ciencia?

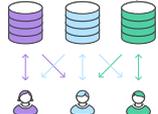
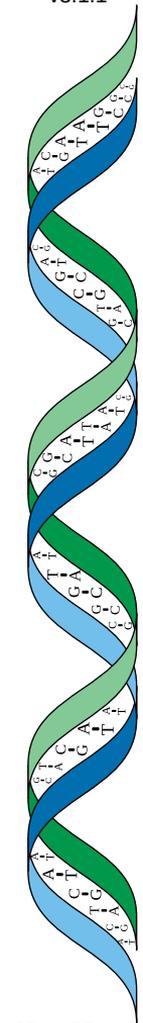
- El diseño de la base de datos influye tanto en qué se puede almacenar y consultar, como en los métodos de consulta.
- Dependiendo del tipo de base de datos, habrá o mucho trabajo manual o multitud de herramientas para realizar el diseño a bajo y alto nivel de una base de datos. Adicionalmente, existen varias metodologías de diseño, que proporcionan las directrices básicas.
- Quienes diseñen la base de datos deben conocer tanto el dominio del problema, como el dominio de uso de la futura base de datos.





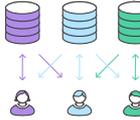
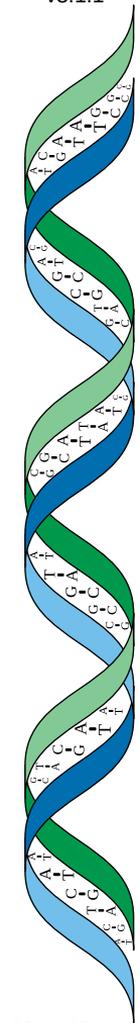
# Pasos a la hora de diseñar una base de datos relacional

- 1) Enumerar los conceptos que se quieren almacenar en la base de datos.
- 2) Clasificar esos conceptos en tablas o atributos de tabla.
- 3) Identificar las tablas que están relacionadas entre sí, la direccionalidad y cardinalidad.
- 4) Asignar cada concepto etiquetado como atributo a las tablas con las que estén relacionados.
- 5) Identificar qué atributos pueden actuar como claves únicas o primaria, y elegir la clave primaria.
- 6) Crear los atributos correspondientes a las claves externas en las tablas que están siendo apuntadas por otras. Esos atributos vienen de copiar los atributos etiquetados como clave primaria.
- 7) Ver si el esquema resultante hace lo que debe hacer. Si no, ir al paso 1.



# Referencias

- Prácticas: <https://bbddmasterisciii.github.io/>
- Manual de SQL de SQLite: <https://www.sqlite.org/lang.html>
- Manual de PostgreSQL 17.2:  
<https://www.postgresql.org/docs/17/index.html>
- MariaDB “Introduction to relational databases”:  
<https://mariadb.com/kb/en/mariadb/introduction-to-relational-databases/>
- “Structured Query Language” wikibook:  
[https://en.wikibooks.org/wiki/Structured\\_Query\\_Language](https://en.wikibooks.org/wiki/Structured_Query_Language)
- Carpentry “Databases and SQL”:  
<https://swcarpentry.github.io/sql-novice-survey/>
- “Mastering SQL”, Martin Gruber, Ed. Sybex
- “SQL for Smarties”, Joe Celko, Ed. Morgan Kaufmann



# Bonus track: algunos hechos sobre SQLite

- SQLite es el gestor más desplegado en todo el planeta. Hay alrededor de  $10^{12}$  de instancias en funcionamiento.
- Su código fuente es mantenido por tres personas.
- SQLite “nació” en los 2000 en un destructor de la armada de los EEUU. D. Richard Hipp estaba escribiendo software para el USS Oscar Austin, el software existente dejaría de funcionar si el servidor SQL dejara de funcionar (inaceptable para un navío de guerra).
- Es de código abierto, pero no de contribución abierta.
- Por cada línea de código hay 600 líneas de test:  
*The test suite is extremely diverse, including fuzz tests, boundary value tests, regression tests, and tests that simulate operating system crashes, power losses, I/O errors, and out-of-memory errors.*
- SQLite es tan rápido que compite con la llamada `fopen` de la librería `stdio` de C. En algunos escenarios SQLite puede ser hasta un 35% más rápido que `fopen` y otras llamadas.

<https://avi.im/blag/2024/sqlite-facts/>

