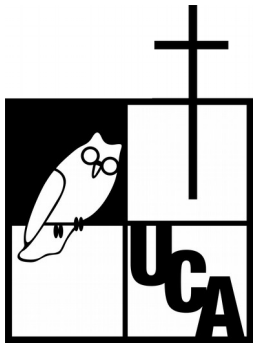
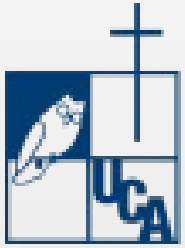


NoSQL

Administración de Bases de Datos



Departamento de
electrónica e informática

Carlos Juan Martín Pérez

Conceptos I

Antes de cambiar hay que valorar bien dónde estás:
¿Cuál es la virtud de los SGBDR? **ACID**

- **Atomicidad:** Si una transacción consiste en una serie de pasos, se ejecutan **todos o ninguno**, es decir, las transacciones son completas.
- **Consistencia:** Cualquier transacción llevará a la base de datos desde un estado válido a otro también válido según las reglas de **integridad** definidas (clave, valor/tipo, referencial y semántica).
- **Aislamiento:** La realización de dos transacciones **concurrentes** sobre la misma información sean **independientes** y no generen ningún tipo de error.
- **Durabilidad:** una vez realizada la transacción, ésta **persistirá** y no se podrá deshacer.

Muy rico y todo pero...

¡necesitamos distribuir/replicar los datos!

Teorema CAP o conjetura de Brewer: en cualquier sistema de almacenamiento distribuido de datos solo es posible cumplir **simultáneamente dos** de las siguientes garantías.

- **Consistencia:** Toda lectura obtiene el último dato o un error (todos los nodos ofrecen el mismo dato en un momento determinado).
- **Disponibilidad:** Toda petición de datos recibe una respuesta no errónea, aunque no se pueda garantizar que es el dato más reciente.
- **Tolerancia a particiones:** Si se produce algún fallo/retraso de conexión entre los nodos el sistema sigue operando sin error. Dos opciones:
  - Cancelación → + consistencia, – disponibilidad.
  - Confirmación → – consistencia, + disponibilidad.

Y los NoSQL, ¿qué virtudes tienen? **BASE**

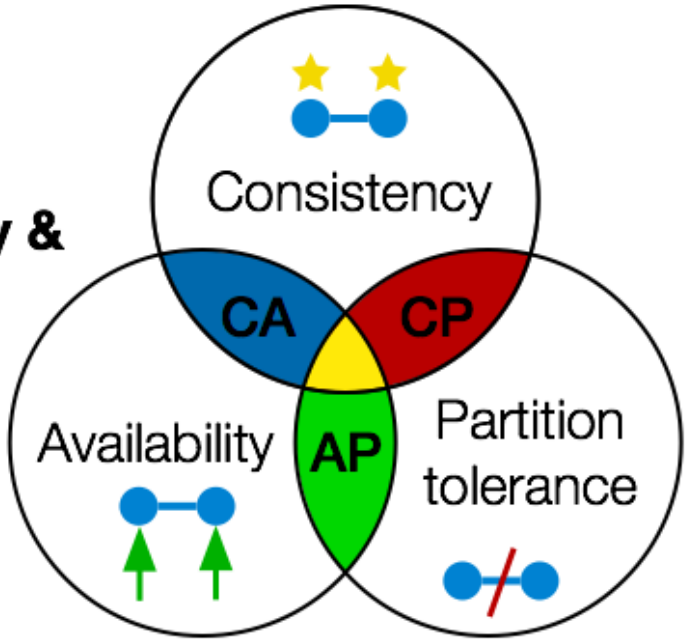
- **Básicamente disponibles:** Garantizan la disponibilidad, es decir, toda petición de datos tendrá respuesta, PERO sin garantía de persistencia en escritura ni de obtención del último dato en una lectura.
- **Estado “suave”:** Al no haber garantía de consistencia, se tiene un estado **probable** de la BD no determinista.
- **Eventualmente consistente:** Tras la realización de escrituras, se garantiza que en algún momento todos los nodos convergerán al mismo estado.

Menos lero lero, ayúdame a vivir: ¿qué elijo?

¿**SGBDR** vs **NoSQL**?

1) Imprescindible asegurar consistencia y disponibilidad (Si no podemos tolerar particiones): SGBDR.
2) Queremos disponibilidad a toda costa, inclusive sucediendo fallos en los nodos: NoSQL.
3) Volumen de datos/registros:
   - Cientos de miles hasta millones: SGBDR
   - MUCHOS millones: NoSQL
4) Estructura:
   - Restricciones de integridad: SGBDR
   - Flexibilidad: NoSQL

La elección en NoSQL parte de la elección entre consistencia y la disponibilidad ante la partición.



Consistency & Availability
- MySQL
- PostgreSQL

Consistency & Partition tolerance
- HBase
- MongoDB
- Redis
- Memcache

Availability & Partition tolerance
- Riak
- Cassandra
- CouchDB
- DynamoDB

No se vayan todavía, ¡aún hay más!. ¿Cuál NoSQL?

Tipos principales de DBMS NoSQL:

1) Clave-valor
2) Columnas
3) Documental
4) Grafos

Modelo	Rendimiento	Escalabilidad	Flexibilidad	Complejidad
Clave valor	alto	alto	alto	ninguna
Columnas	alto	alto	moderado	bajo
Documental	alto	variable (alto)	alto	bajo
Grafos	variable	variable	alto	alto
Relacional	variable	variable	bajo	moderado

Lo que conocemos hasta ahora:

Name	Birthday	PERSONID
Jos The Boss	11-12-1985	1
Fritz von Braun	27-1-1978	2
Freddy Stark		3
Delphine Thewiseone	16-9-1986	4

Person info table: represents person specific information

PERSONID	HOBBYID
1	2
1	2
2	3
2	4
3	5
3	6
3	1

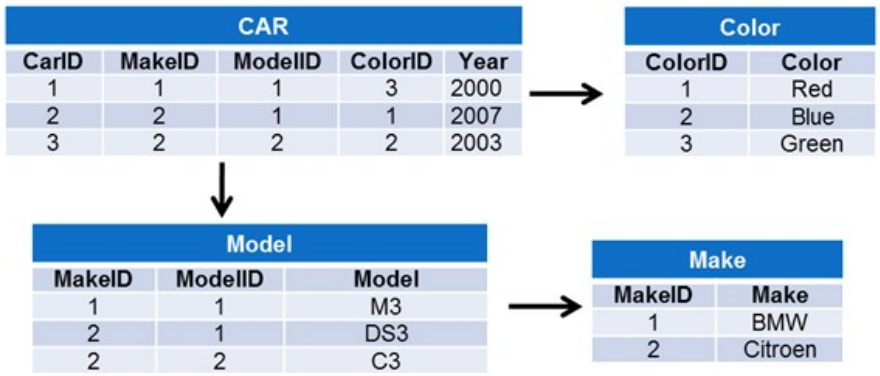
Person-Hobby linking table. This is necessary because of the many to many relationship between hobbies and persons.

HOBBYID	HobbyName	HobbyDescription
1	Archery	Shooting arrows from a bow
2	Conquering the world	looking for trouble with your neighboring countries
3	building things	also known as construction
4	surfing	catching waves on a plank
5	swordplay	fencing with swords
6	lollygagging	hanging around doing nothing

Hobby info table: represents hobby specific information

Relational databases strive towards **normalization** (making sure every piece of data is stored just once). Each table has unique identifiers (primary keys) that are used to model the relation between the entities (tables) hence "relational".

Relational Data Model:



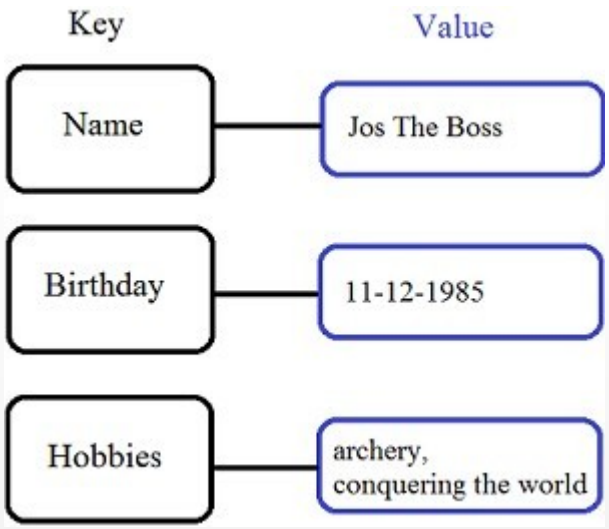
Clave - Valor

Los DBMS clave-valor son probablemente la forma más simple de sistemas gestores de bases de datos. Solo pueden almacenar pares de claves y valores, tal cual, así como recuperar valores cuando se conoce una clave.

Estos sistemas simples normalmente no son adecuados para aplicaciones complejas. Por otro lado, es exactamente esta simplicidad la que hace que tales sistemas sean atractivos en ciertas circunstancias. Por ejemplo, los almacenes de valor-clave de uso eficiente de los recursos a menudo se aplican en sistemas integrados o como bases de datos en proceso de alto rendimiento.

¿Como quién?: redis, memcached

Clave – Valor: cómo se come eso



Artist Info

Key	Value
artist:1:name	AC/DC
artist:1:genre	Hard Rock
artist:2:name	Slim Dusty
artist:2:genre	Country

```
{"internal data":[{"entities":[
    {"customer":[
        {"id":1,"name":"Freddy"},
        {"id":2,"name":"Fritz"}
    ]},
    {"legal entities":[
        {"id":1,"company":"Maiton"}
    ]}]
},{"Products":[
    {"furniture":[
        {"id":1,"name":"Octopus Table","stock":1}
    ]
}]}]}
```

Documentos

Se caracterizan por su organización de datos sin esquemas. Eso significa:

1) Los registros no necesitan tener una estructura uniforme, es decir, diferentes registros pueden tener diferentes columnas.
2) Los tipos de los valores de las columnas individuales pueden ser diferentes para cada registro.
3) Las columnas pueden tener más de un valor (matrices).
4) Los registros pueden tener una estructura anidada.

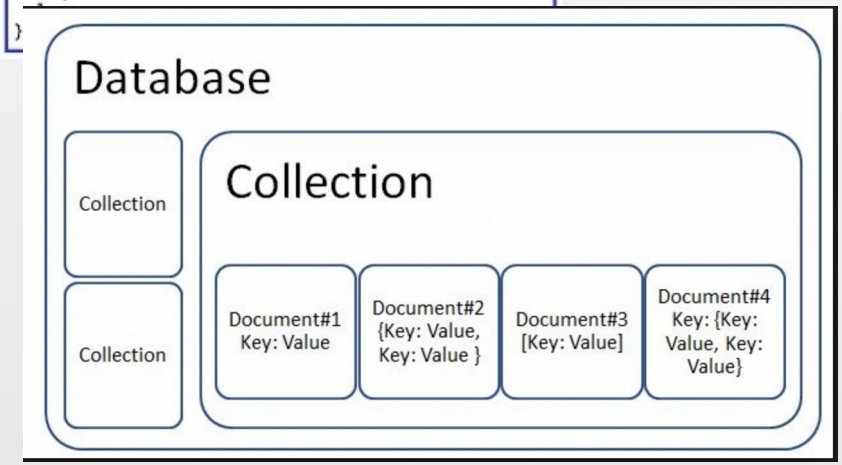
Los almacenes de documentos a menudo usan notaciones internas para ser procesados directamente en las aplicaciones (XML,JSON, etc.)

¿Como quién?: MongoDB, CouchBase, CouchDB

Documentos: cómo se come eso


```
{
  "articles": [
    {
      "title": "title of the article",
      "articleID": 1,
      "body": "body of the article",
      "author": "Isaac Asimov",
      "comments": [
        {
            "username": "Fritz",
            "join date": "1/4/2014",
            "commentid": 1,
            "body": "this is a great article",
            "replies": [
                {
                    "username": "Freddy",
                    "join date": "11/12/2013",
                    "commentid": 2,
                    "body": "seriously? it's rubbish"
                }
            ]
        },
        {
            "username": "Stark",
            "join date": "19/06/2011",
            "commentid": 3,
            "body": "I don't agree with the conclusion"
        }
      ]
    }
}
```

Whereas relational databases chop up data, Document stores save documents as a single entity



Columnas

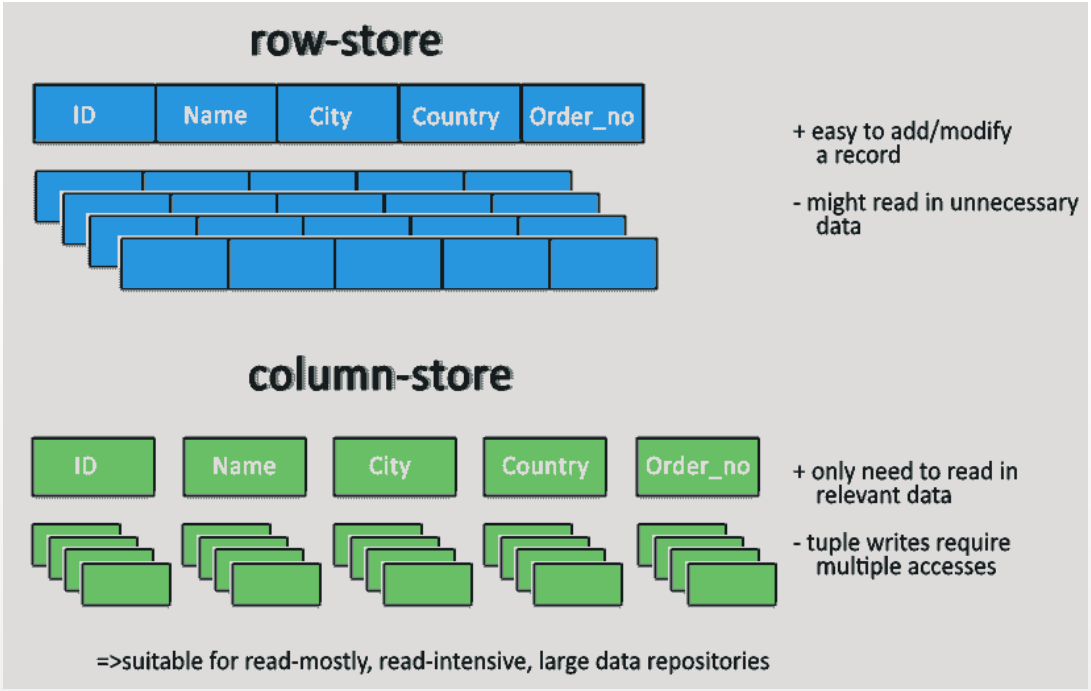
También denominados “Wide Columns”, los datos se almacenan en registros con la capacidad de contener un gran número de columnas dinámicas. Dado que los nombres de columna y las claves no son fijos, y dado que un registro puede tener miles de millones de columnas, los DBMS de columnas se pueden ver como DBMS bidimensionales de clave-valor.

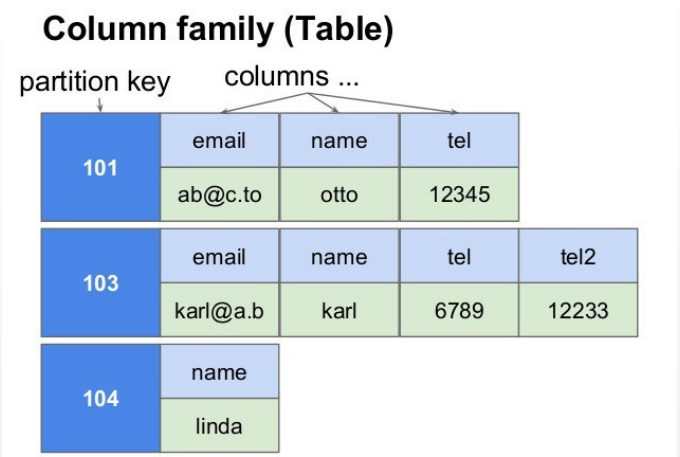
Comparten con los DBMS de Documentos la característica de ser flexibles en cuanto la estructura PERO se debe conocer a priori cómo se consultarán los datos.

Son excelentes agrupando valores y contando entradas (lecturas). Es típico usar una RDBMS para lo transaccional y otra de columnas para mejorar análisis/reporte de datos.

¿Como quién?: Cassandra, HBase

Columnas: cómo se come eso





Name	ROWID
Jos The Boss	1
Fritz Schneider	2
Freddy Stark	3
Delphine Thewiseone	4

Birthday	ROWID
11-12-1985	1
27-1-1978	2
16-9-1986	4

Hobbies	ROWID
archery	1, 3
conquering the world	1
building things	2
surfing	2
swordplay	3
lollygagging	3

A column-oriented database stores each column separately

Columnas: cómo se come eso