Modelos de Datos

Gestión y Modelación de Datos

María Constanza Pabón
mcpabon@javerianacali.edu.co

Contenido

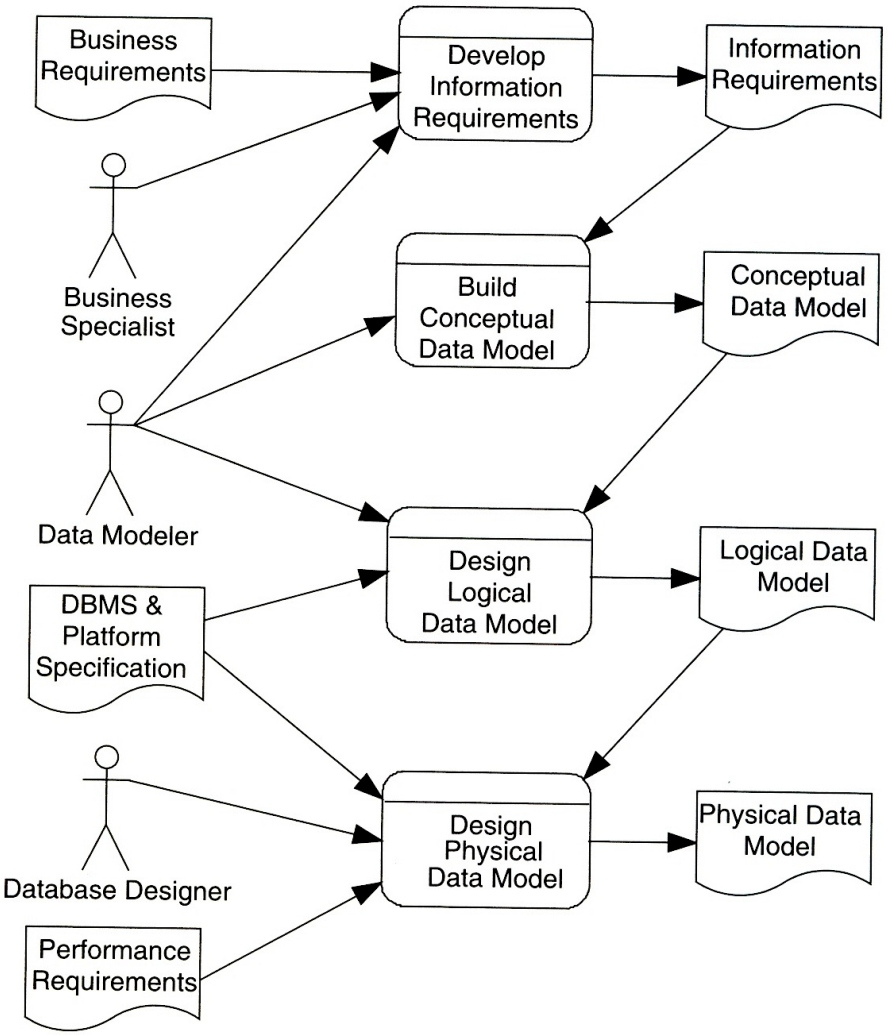
- ¿Qué es un Modelo de Datos?
- Etapas de diseño de BD
- Modelo Relacional
- Modelos Orientados a Objetos
- NoSQL (post-relacionales): XML, Grafos, RDF, otros
- Un buen modelo de datos

¿Qué es un Modelo de Datos?

Un Modelo de Datos es una notación para describir datos o información. Consta de [Codd, 80]:

- Estructuras de datos
- Operaciones de los datos: consultas, modificaciones
- Reglas de integridad: definen los estados consistentes de la BD

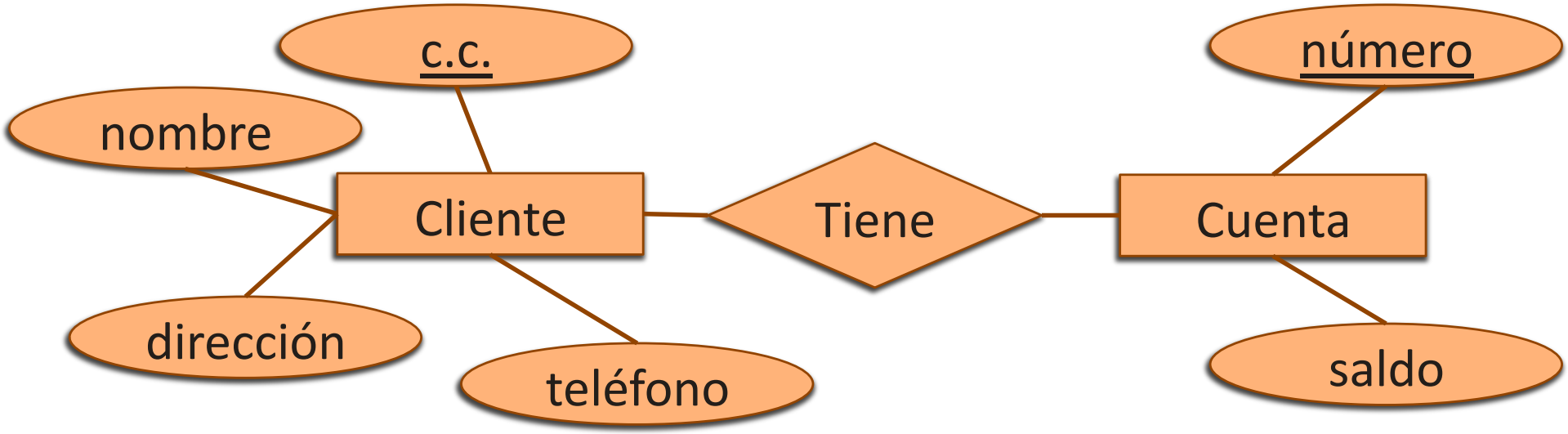
Etapas del diseño de una BD



- Modelo conceptual: independiente de la tecnología de BD. Soporta comunicación con los usuarios
- Modelo Lógico: estructuras que se van a implementar en el SGBD
- Modelo Físico: incluye especificaciones sobre almacenamiento físico (ej. distribución) y mecanismos de acceso. Se busca eficiencia.

Modelo Entidad-Relación

Ejemplo: cuentas de ahorros en un banco



Modelo Relacional

Ejemplo: cuentas de ahorros en un banco

Cliente

c.c.	Nombre	Dirección	Teléfono
1234	María	Cra 5	90882
4567	Juan	Cll 20	28273
7890	Carlos	Cra 12	28272

Relación

Cuenta

Número	Saldo	Cliente
654886	30.000.000	4567
974272	2.000.000	7890
142826	500.000	1234

Modelos Orientados a Objetos

Ejemplo: cuentas de ahorros en un banco

```
interface Cliente {
  attribute str cc;
  attribute str nombre;
  attribute str direccion;
  relationship Set<Cuenta> Cuentas
  inverse Cuenta::Dueño;  }
```

Clase

```
interface Cuenta {
  attribute long Numero,
  attribute double Saldo,
  relationship Cliente Dueño inverse Cliente::Cuentas;
  void Consigna (in double Valor);
  void Retira (in double Valor); }
```

NoSQL (Post-relacionales / No-relacionales)

- Aplicaciones con requerimientos de almacenamiento diferentes:
  - Intercambio de datos (XML)
  - Interconectividad, rutas, sistemas geográficos, sistemas de transporte (Grafos)
  - Representación de conocimiento (Ontologías, RDF, OWL)
  - Google, Amazon, Facebook, Twitter, ...: Aplicaciones web con alto trafico, gran cantidad de datos y contenido generado por usuarios

NoSQL: XML un modelo de datos semiestructurado

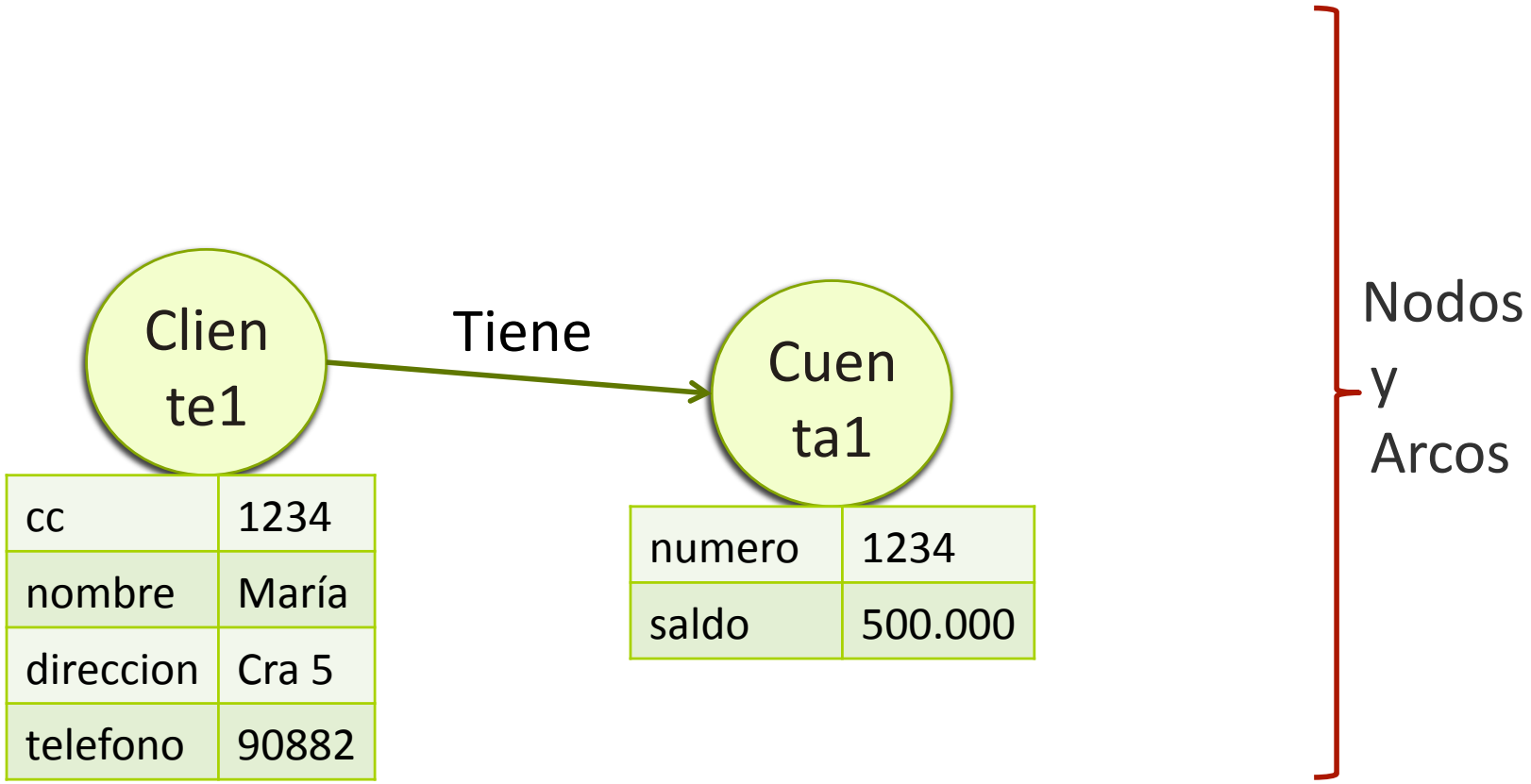
Ejemplo: cuentas de ahorros en un banco

```
<cliente>
    <cc>1234</cc>
    <name>María</name>
    <direccion>Cra 5</direccion>
    <telefono>90882</telefono>
    <cuenta>
        <numero>142826</numero>
        <saldo>500.000</saldo>
    </cuenta>
</cliente>
```

Arbol

NoSQL: Grafo atribuido

Ejemplo: cuentas de ahorros en un banco



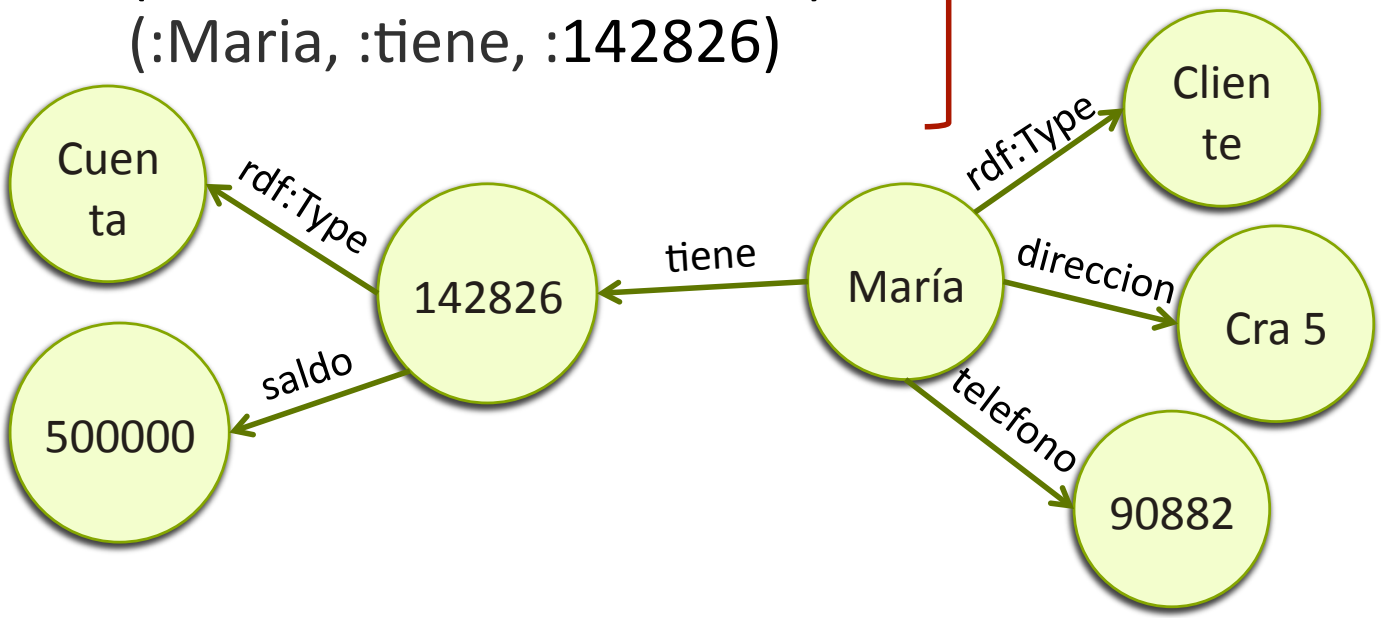
NoSQL: RDF (Resource Description Framework) – RDFS (RDF Schema)

Ejemplo: cuentas de ahorros en un banco

(:Maria, rdf:Type, :Cliente)
(:Maria, :direccion, :Cra 5)
(:Maria, :telefono, 90882)
(:142826, rdf:Type, :Cuenta)
(:142826, :saldo, 500.000)
(:Maria, :tiene, :142826)

Tripletas

(sujeto, propiedad, objeto)



NoSQL y la WEB

- La Web introdujo aplicaciones con nuevas escalas en términos de:
  - Usuarios concurrentes (millones de requerimientos por segundo)
  - Datos (peta-bytes generados diariamente)
  - Procesamiento (de todos esos datos)
  - Crecimiento exponencial (picos impredecibles en la demanda)

NoSQL y la WEB

- Sin embargo, las aplicaciones Web son, generalmente, de uso libre (gratis), por lo tanto pueden sacrificar integridad de datos / consistencia.
- Nadie los puede demandar por no tener la información actualizada de:
  - Estado de los amigos (Facebook)
  - Resultados de una búsqueda (Google)
  - Items en el carrito (Amazon)

NoSQL y la WEB

- En 2000, Eric Brewer formuló la conjetura CAP: en un sistema distribuido solamente es posible optimizar 2 de las siguientes características:
  - **Consistencia (Consistency)**
  - **Disponibilidad (Availability)**
  - **Tolerancia a Fallos (Partition tolerance)**
- En 2002 el teorema fue demostrado

Ejemplo CAP

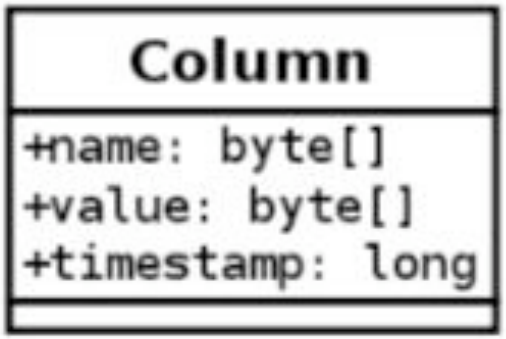
- Una forma de ser mas tolerante a fallos es tener replicas de los datos y programas en varias máquinas
- Por lo tanto cuando se escribe un registro, se debe también actualizar la réplica
- Se tiene que elegir entre:
  - Bloquear las réplicas durante la actualización → sacrifica **disponiblidad**
  - No bloquear las réplicas → sacrifica **consistencia**

NoSQL y la WEB

- Estas empresas han desarrollado una familia de bases de datos que responden a las necesidades de las aplicaciones en la Web:
  - **BigTable (**desarrollado por **Google)**
  - **Hbase (**desarrollado por **Yahoo!)**
  - **Dynamo (**desarrollado por **Amazon)**
  - **Cassandra (**desarrollado por **FaceBook)**
  - **Voldemort (**desarrollado por **LinkedIn)**
  - & a few more:
    - **Riak, Redis, CouchDB, MongoDB, Hypertable**

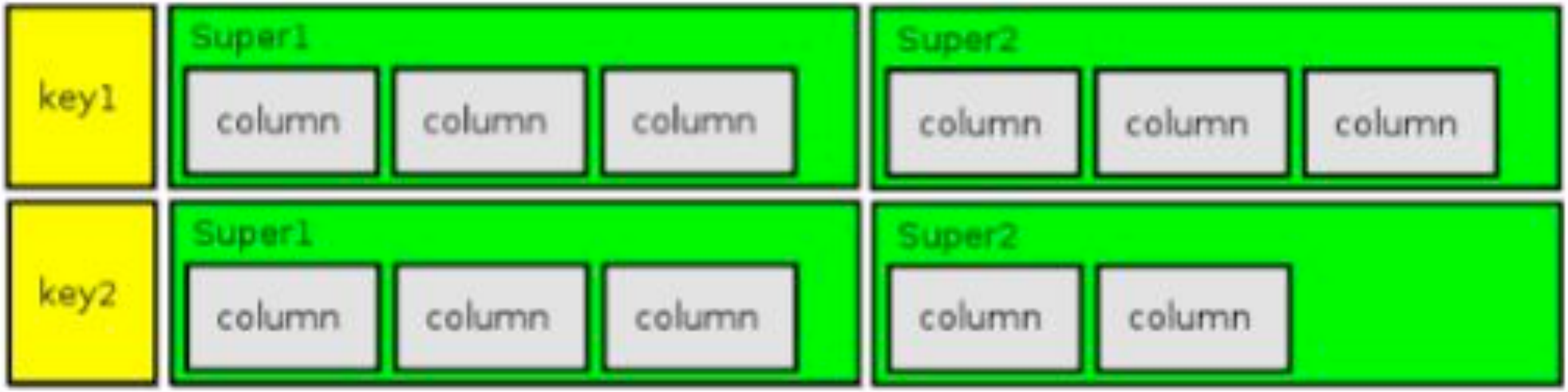
BD Orientadas a Columnas

- Familias de Columnas (una tabla espacida)



Cassandra

- “Supercolumnas”: una collección de columnas



- Cassandra es usado en: Facebook, Digg y Twitter

Cassandra

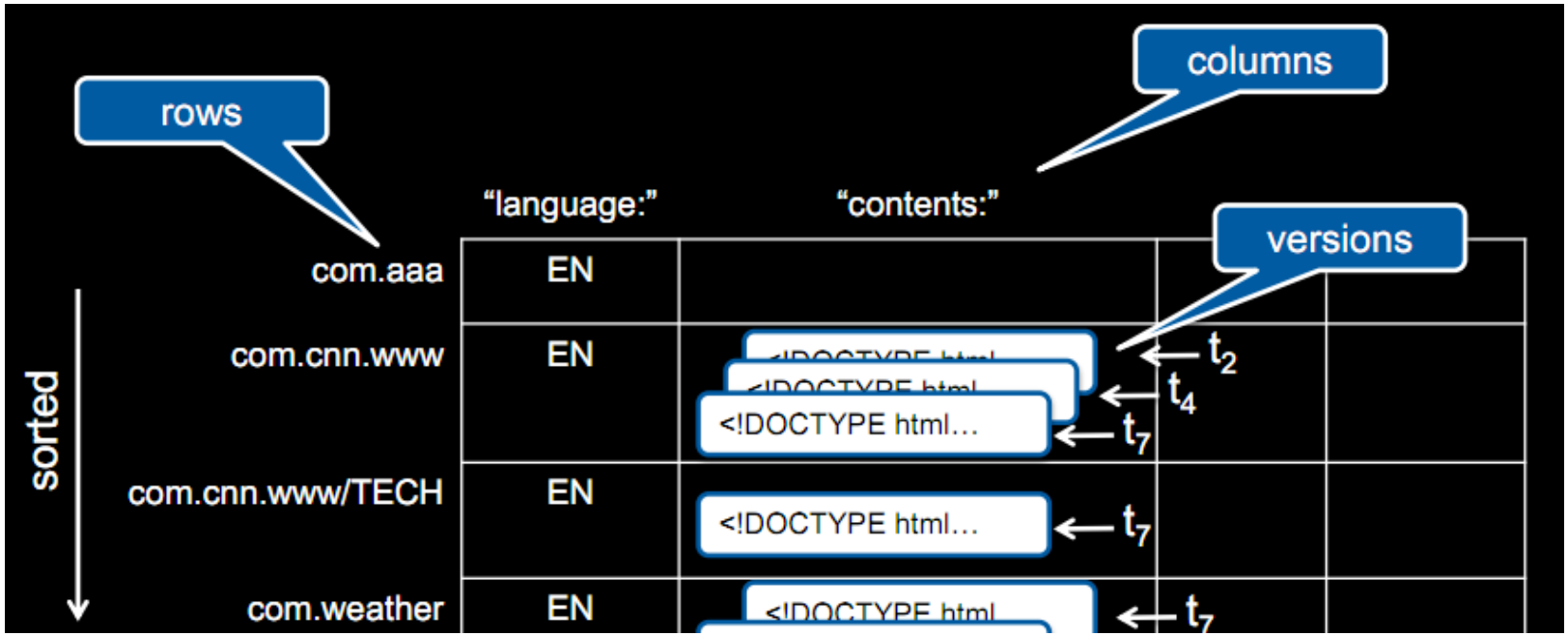
- Cassandra vs. MySQL (50GB)
  - MySQL:
    - 300ms escritura
    - 350ms lectura
  - Cassandra:
    - 0.12ms escritura
    - 15ms lectura
- La escritura:
  - No involucra lecturas ni búsquedas
  - Se escribe en cualquier nodo

NoSQL: BigTable

BigTable (Google):

Tablas multidimensionales, cada dato se indexa por los nombres de fila y columna y timestamp

(row:string, column:string, time:int64) -> string



NoSQL: Key-Value Stores, Column Family

Key-value stores (Dynamo–Amazon)

user1923_color	Red
user1923_age	18
user3371_color	Blue
user1923_height	6' 0”

NoSQL: Document Stores

Document Stores (couchDB, mongoDB)

```
{
  FirstName:"Jonathan",
  Address:"15 Wanamassa Point Road",
   Children: [ {Name:"Michael",Age:10},
               {Name:"Jennifer", Age:8},
               {Name:"Samantha", Age:5},
               {Name:"Elena", Age:2}   ]  }
```