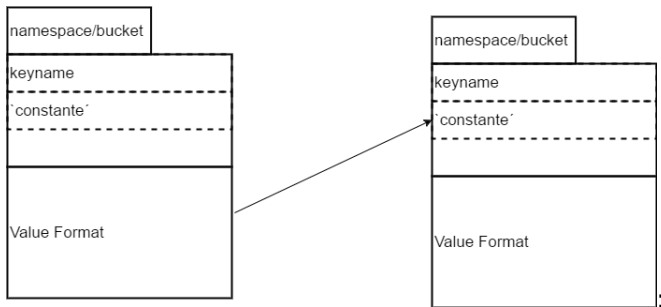


Key-Value

Notación Key-Value

- Espacio de nombres optativo
- Partes de la clave
- Valor en formatos JSON, Lista, Hash, string, etc.
- Relación entre un valor y una clave de otro dato.



Notación Key-Value Ejemplo

```
ConcertApp[ticketLog:9888] = {"conDate":15-Mar-2015, "locDescr": "Springfield Civic Center",
    "assgnSeat": "J38"}
```

ConcertApp	ConcertApp	ConcertApp
'ticketlog'	'ticketlog'	'ticketlog'
nroticket: int	nroticket: int	nroticket: int
JSON: 'assgnSeat': {"type": "string"}, 'conDate': {"type": "string", "format": "date - time "}, 'locDescr': {"type": "string"}	HASH assignSeat: string conDate: date locDescr: string	JSON: Schema ConcertApp

Notación Key-Value Ejemplo referencias

Users	
UserID	Info
7734	City:San Francisco, email:alef@gmail.com
4667	City:New York, email:samuel@yahoo.com
6578	City:Seattle, email:knovoselic@gmail.com
1263	City:San Francisco, email:eray@yahoo.com

Cities	
City	UserIDs
San Francisco	7734, 1263, ...
New York	4667, ...
Seattle	6578, ...

'users'

userid: int

HASH
ciudad: string
email: date
nombre: string
apellido: string

'cities'

nombre: string

LISTA
[userid:int,...]

Ejercicio

Indices

Diseñar la base de datos para un twitter

- Se pueden usar como tipos de datos: Json, Hash, Sets
 - **HSET** usuarios nombre Jhon apellido Doe
 - **HGET** usuarios nombre \Rightarrow Jhon
- Se puede usar una operación: **INCR key**. (Clave tipo INCR en el diagrama)
 - **INCR** prox_id \Rightarrow 10

Familia de Columnas

Diseño Aspectos Generales

Cómo diseñar

Establecer los patrones de escritura y lectura.

- Desnormalizar!
- Usar columnas sin valor
- Usar nombres y valores de columna para almacenar datos
- Modelar una entidad como una fila simple
- Mantener un número adecuado de versiones en los valores de las columnas.
 - HBase permite especificar el mínimo y máximo número de versiones.
- Evitar estructuras complejas en los valores de las columnas.

Reglas de Mapeo

Basados en los DMP (Data Modeling Principles), las reglas de mapeo ayudan a realizar la transición desde el modelo conceptual al modelo lógico.

- 1 **MR1** (*Entities and Relationships*): Los tipos de entidades y relaciones mapean a tablas mientras que los datos se asignan a filas. Los atributos de las entidades y las relaciones se mapean a columnas
- 2 **MR2** (*Equality Search Attributes*): Si se utilizan en una consulta por igualdad de atributos, entonces, éstos forman la clave primaria y tales columnas se incluirán en las particiones por clave
- 3 **MR3** (*Inequality Search Attributes*): Si se utilizan en consultas por desigualdad, estos atributos mapean como columnas dinámicas, conteniendo los valores
- 4 **MR4** (*Ordering Attributes*): Mapea a columna dinámica con orden ascendente o descendente según se especifique en la consulta
- 5 **MR5** (*Key Attributes*): Mapea a clave primaria. Una tabla que almacena datos de entidades o relaciones como filas debe incluir atributos claves que identifique estos datos univocamente

Cassandra - Notación/Método Chebotko

table_name	
column_name_1	CQL Type K ← Partition key column
column_name_2	CQL Type C↑ ← Clustering key column (ASC)
column_name_3	CQL Type C↓ ← Clustering key column (DESC)
column_name_4	CQL Type S ← Static column
column_name_5	CQL Type IDX ← Secondary index column
column_name_6	CQL Type ++ ← Counter column
[column_name_7]	CQL Type ← Collection column (list)
{column_name_8}	CQL Type ← Collection column (set)
<column_name_9>	CQL Type ← Collection column (map)
column_name_10	UDT Name ← UDT column
(column_name_11)	CQL Type ← Tuple column
column_name_12	CQL Type ← Regular column

Ejemplo: DER

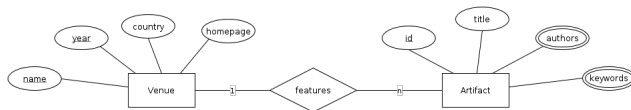
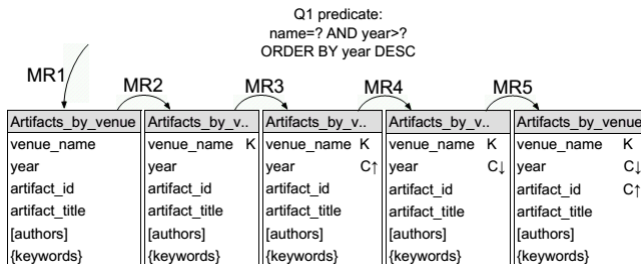


Diagrama de Entidad-Relación

- Q1: Obtener los artefactos publicados en *venue* con un nombre dado después de un año dado. El resultado debe estar ordenado en forma descendente



Map-Reduce

- 1 Entrada: un conjunto de pares clave-valor. De esta manera se logra permitir la composición de procesos
- 2 El desarrollador especifica dos funciones:
 - $Map(k, v) \rightarrow \langle k', v' \rangle^*$
 - Toma un par *key-value* y devuelve un conjunto de pares *key-value*
 - Hay un llamada a Map por cada par (k,v)
 - $Reduce(k', \langle v' \rangle^*) \rightarrow \langle k'', v'' \rangle^*$
 - Todos los valores de v' para la misma k' se reducen y procesan en conjunto

Ejemplo - Contar Palabras

```
map(String input_key, String input_value):  
  // input_key: document name  
  // input_value: document contents  
  for each word w in input_value:  
    EmitIntermediate(w, 1);
```

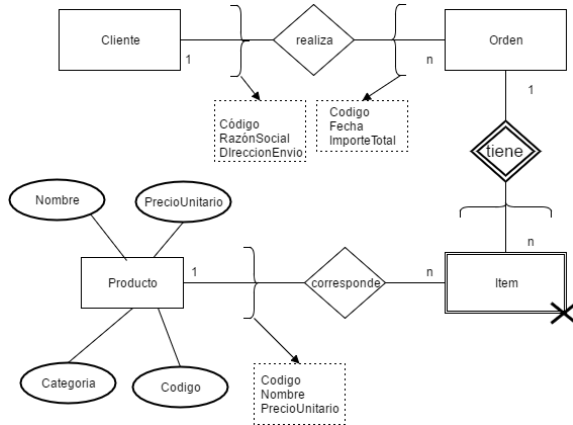
Map

```
reduce(String output_key, Iterator intermediate_values):  
  // output_key: word  
  // output_values: ????  
  int result = 0;  
  for each v in intermediate_values:  
    result += v;  
  Emit(result);
```

Reduce

Document Database

Desnormalización parcial

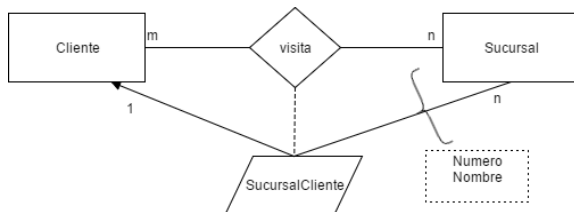


Desnormalización

JSON Schema para Documento Orden

```
"Orden": { "type": "object",
  "properties": {
    "OrdenID": { "type": "integer" },
    "Fecha": { "type": "string", "format": "date-time" },
    "ImporteTotal": { "type": "integer" },
    "Cliente": {
      "type": "object",
      "properties": {
        "Codigo": { "type": "integer" },
        "RazonSocial": { "type": "string" },
        "DireccionEnvio": { "type": "string" }
      }
    },
    "ItemsOrden": {
      "type": "Array",
      "items": {
        "type": "object",
        "properties": {
          "Cantidad": { "type": "integer" },
          "Codigo": { "type": "integer" },
          "Nombre": { "type": "string" },
          "PrecioUnitario": { "type": "string" }
        }
      }
    }
  }
}
```


Uso de Documentos Auxiliares



```
"SucursalCliente": {"type": "object",  
  "properties": {  
    "ClienteID": {"type": "integer" },  
    "Sucursales": {  
      "type": "Array",  
      "items": {"type": "object",  
        "properties": {  
          "Codigo": {"type": "integer" },  
          "Nombre": {"type": "string" },  
        }  
      }  
    }  
  }  
}
```