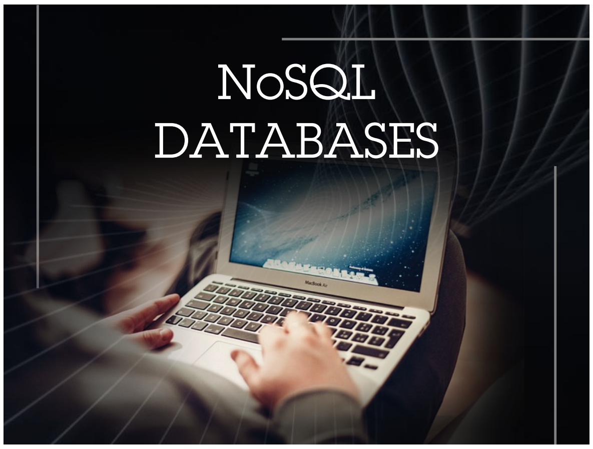
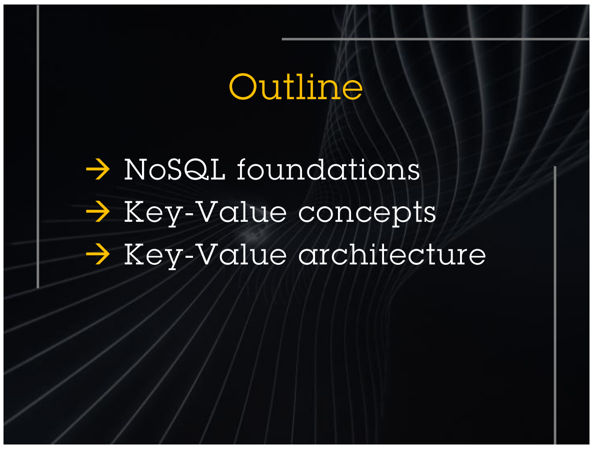
INTRODUZIONE AI BIG DATA

Prof. Flavio Venturini
Lez.05 - NoSQL Databases

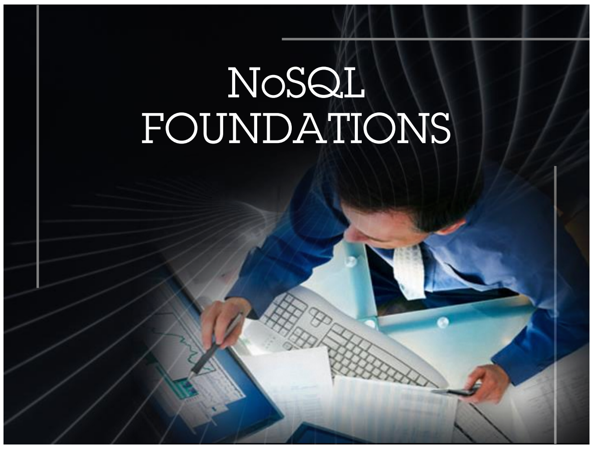


NoSQL DATABASES

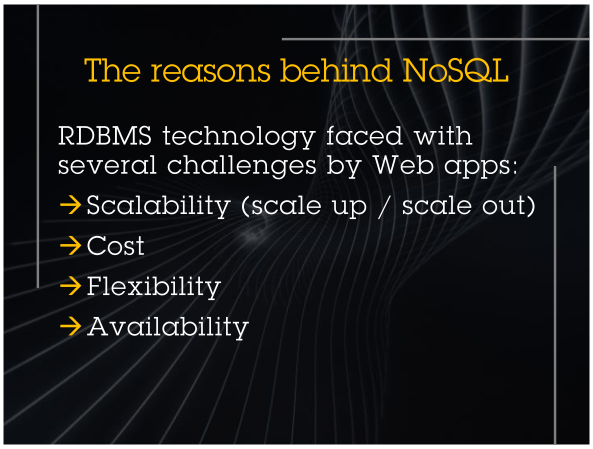


Outline

- → NoSQL foundations
- → Key-Value concepts
- → Key-Value architecture



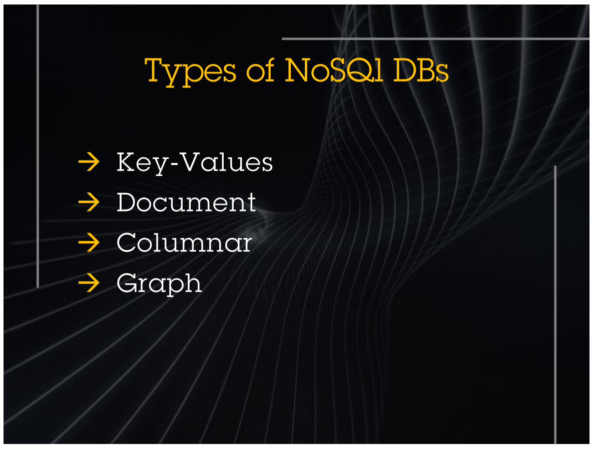
NoSQL FOUNDATIONS



The reasons behind NoSQL

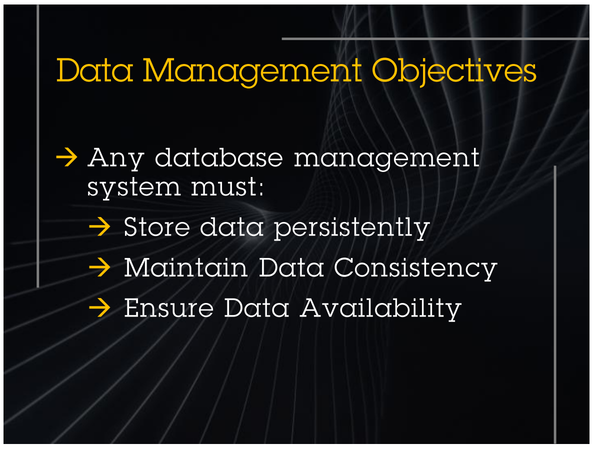
RDBMS technology faced with several challenges by Web apps:

- → Scalability (scale up / scale out)
- → Cost
- → Flexibility
- → Availability



Types of NoSQl DBs

- → Key-Values
- → Document
- → Columnar
- → Graph

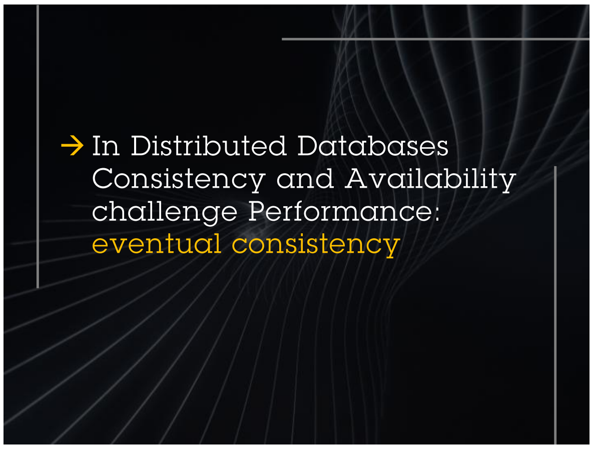


Data Management Objectives

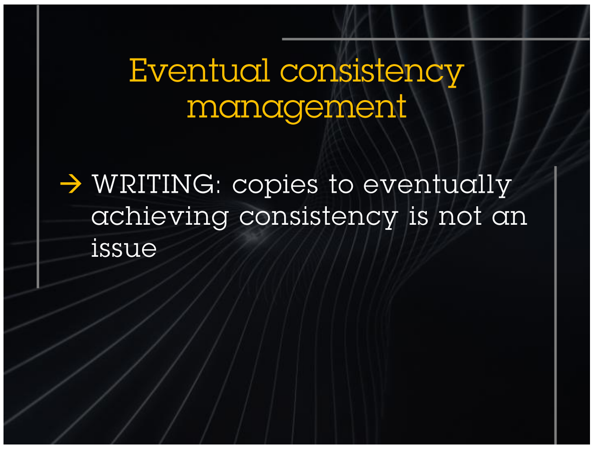
- → Any database management system must:
  - → Store data persistently
  - → Maintain Data Consistency
  - → Ensure Data Availability

UNIVERSITÀ TELEMATICA INTERNAZIONALE UNINETTUNO

Copyright © Università Telematica Internazionale UNINETTUNO

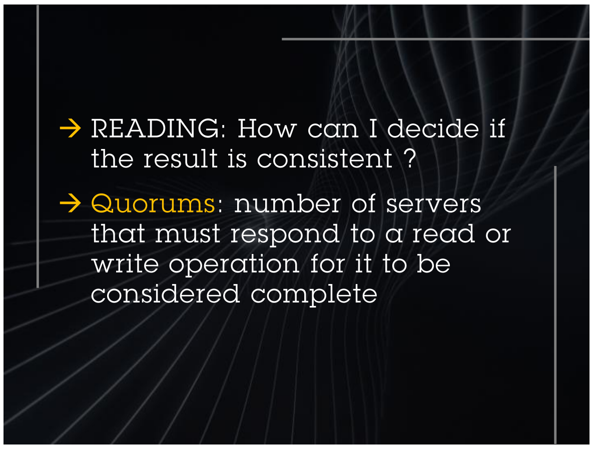


- In Distributed Databases Consistency and Availability challenge Performance: eventual consistency

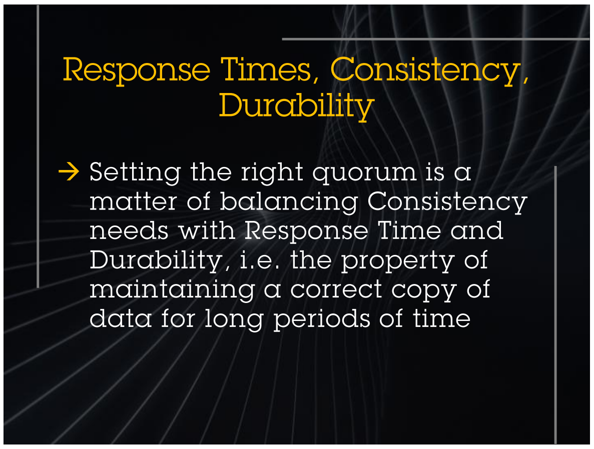


Eventual consistency management

- WRITING: copies to eventually achieving consistency is not an issue

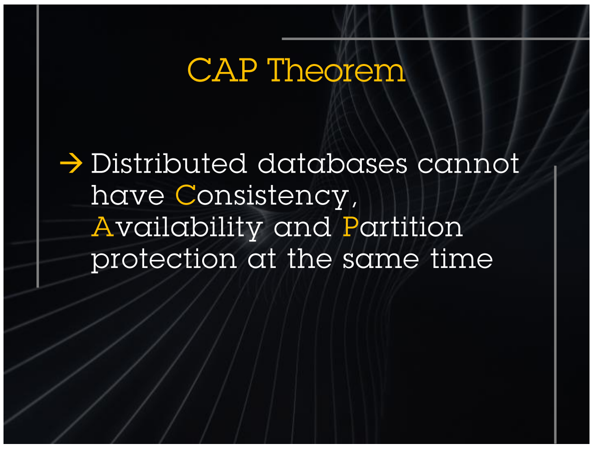


- READING: How can I decide if the result is consistent ?
- Quorums: number of servers that must respond to a read or write operation for it to be considered complete



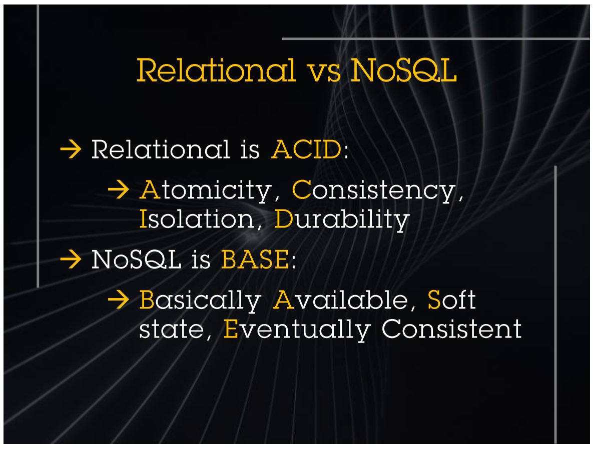
Response Times, Consistency, Durability

- Setting the right quorum is a matter of balancing Consistency needs with Response Time and Durability, i.e. the property of maintaining a correct copy of data for long periods of time



CAP Theorem

- Distributed databases cannot have Consistency, Availability and Partition protection at the same time

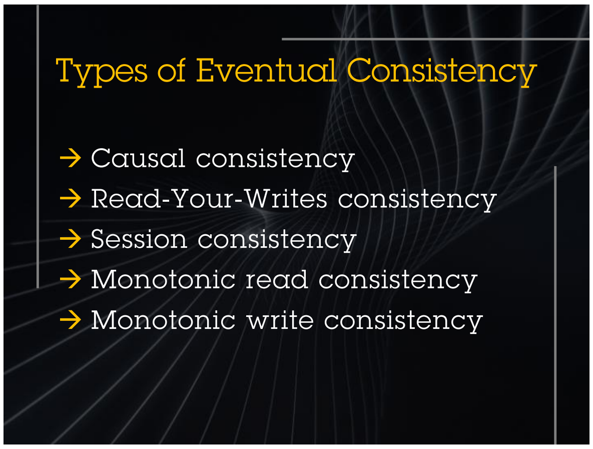


Relational vs NoSQL

- Relational is ACID:
  - Atomicity, Consistency, Isolation, Durability
- NoSQL is BASE:
  - Basically Available, Soft state, Eventually Consistent

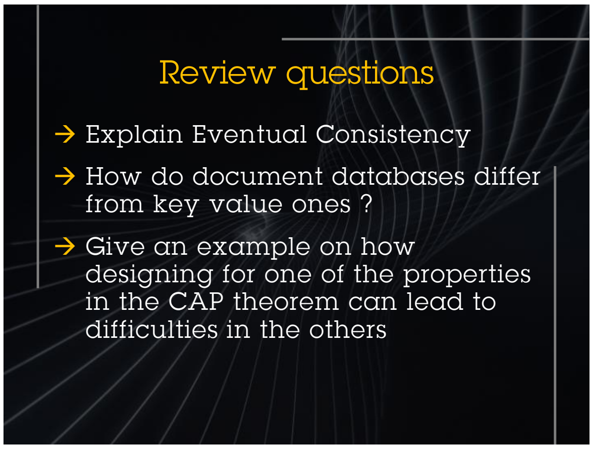
UNIVERSITÀ TELEMATICA INTERNAZIONALE UNINETTUNO

Copyright © Università Telematica Internazionale UNINETTUNO



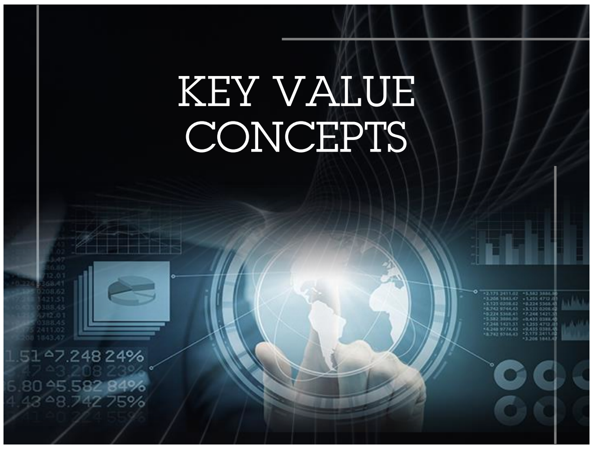
Types of Eventual Consistency

- → Causal consistency
- → Read-Your-Writes consistency
- → Session consistency
- → Monotonic read consistency
- → Monotonic write consistency

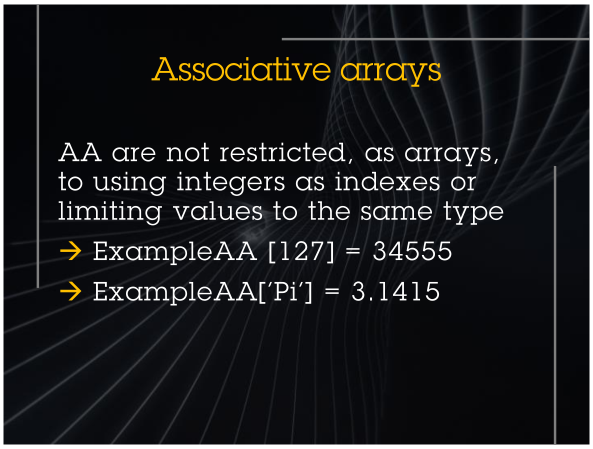


Review questions

- → Explain Eventual Consistency
- → How do document databases differ from key value ones ?
- → Give an example on how designing for one of the properties in the CAP theorem can lead to difficulties in the others



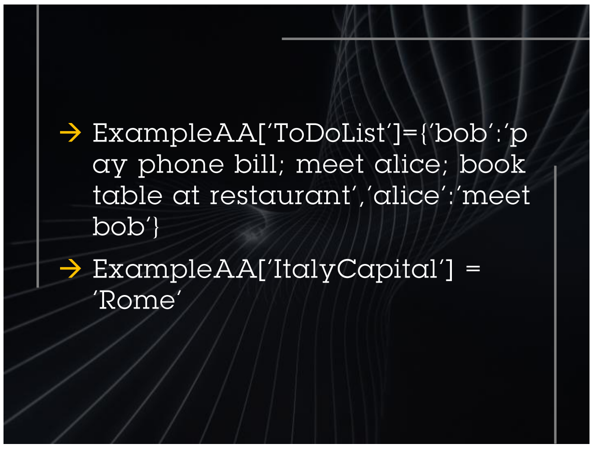
KEY VALUE CONCEPTS



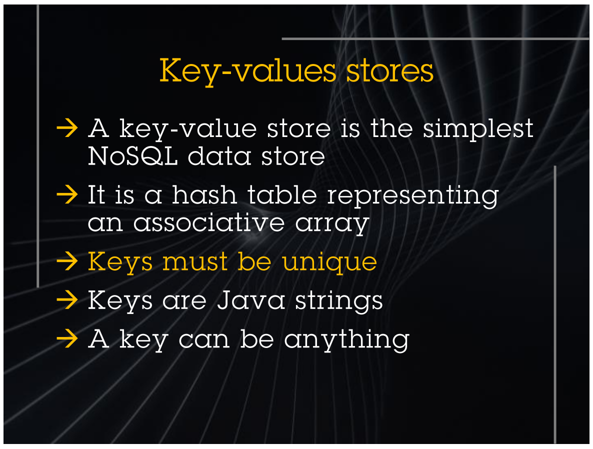
Associative arrays

AA are not restricted, as arrays, to using integers as indexes or limiting values to the same type

- → ExampleAA [127] = 34555
- → ExampleAA['Pi'] = 3.1415



- → ExampleAA['ToDoList']={'bob':'pay phone bill; meet alice; book table at restaurant','alice':'meet bob'}
- → ExampleAA['ItalyCapital'] = 'Rome'

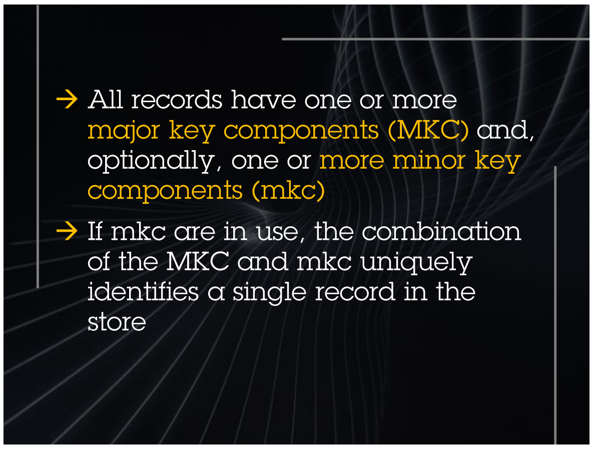


Key-values stores

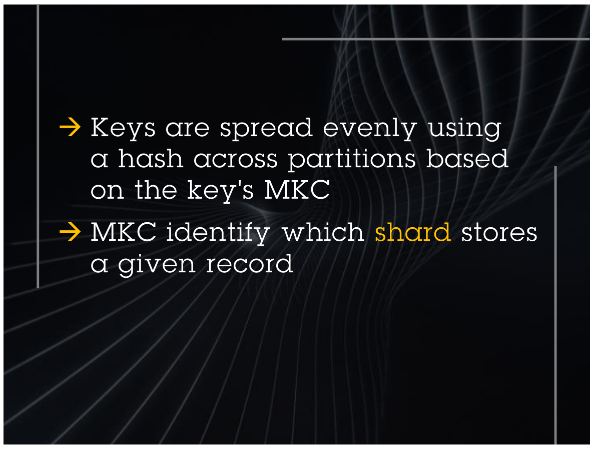
- → A key-value store is the simplest NoSQL data store
- → It is a hash table representing an associative array
- → Keys must be unique
- → Keys are Java strings
- → A key can be anything

UNIVERSITÀ TELEMATICA INTERNAZIONALE UNINETTUNO

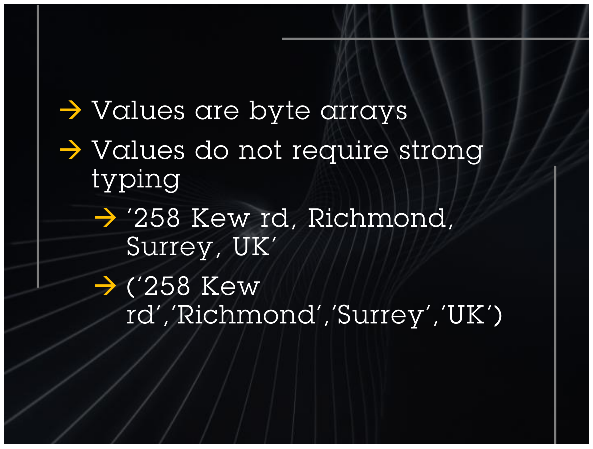
Copyright © Università Telematica Internazionale UNINETTUNO



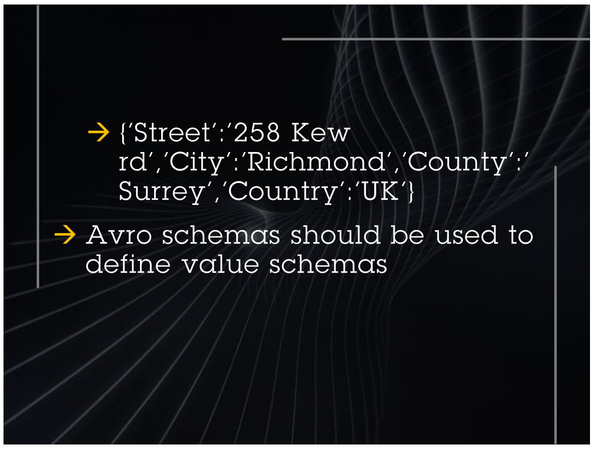
- All records have one or more major key components (MKC) and, optionally, one or more minor key components (mkc)
- If mkc are in use, the combination of the MKC and mkc uniquely identifies a single record in the store



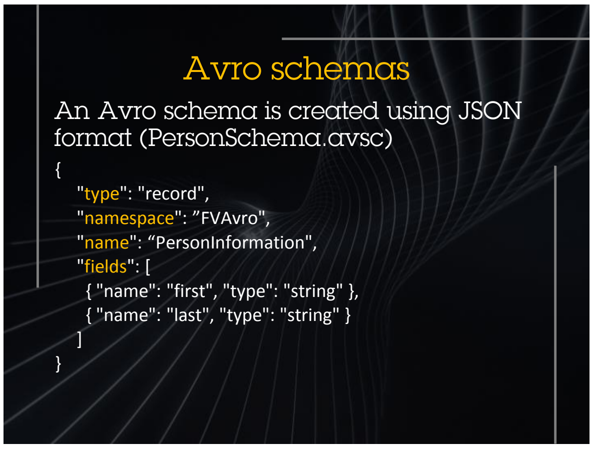
- Keys are spread evenly using a hash across partitions based on the key's MKC
- MKC identify which shard stores a given record



- Values are byte arrays
- Values do not require strong typing
  - '258 Kew rd, Richmond, Surrey, UK'
  - ('258 Kew rd','Richmond','Surrey','UK')



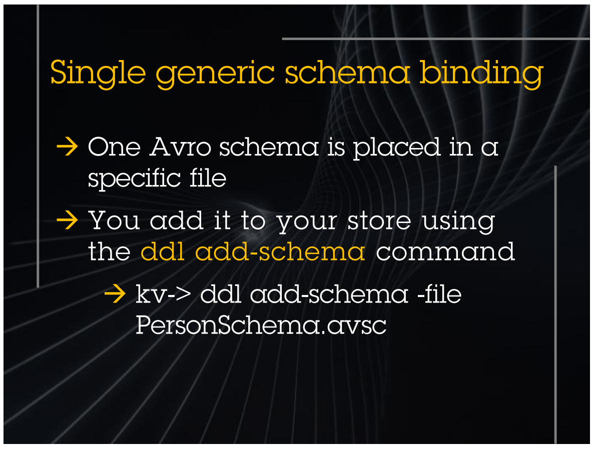
- {'Street':'258 Kew rd','City':'Richmond','County':' Surrey','Country':'UK'}
- Avro schemas should be used to define value schemas



Avro schemas

An Avro schema is created using JSON format (PersonSchema.avsc)

```
{
   "type": "record",
   "namespace": ”FVAvro",
   "name": “PersonInformation",
   "fields": [
     { "name": "first", "type": "string" },
     { "name": "last", "type": "string" }
   ]
}
```

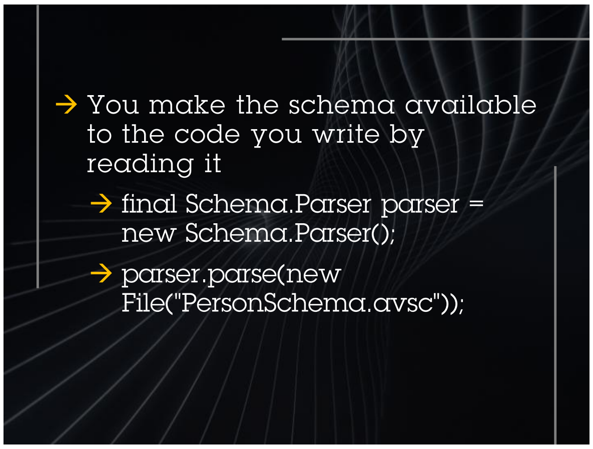


Single generic schema binding

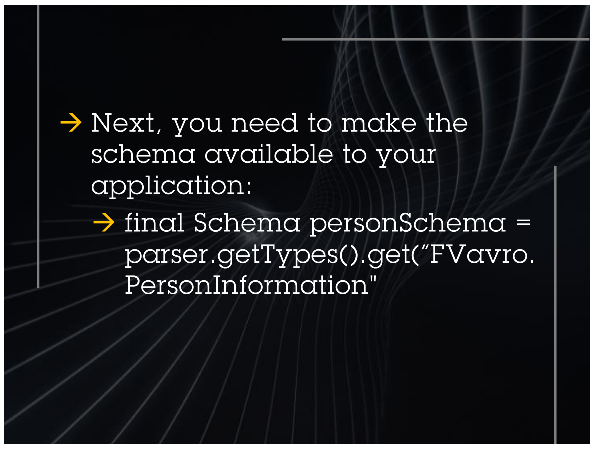
- One Avro schema is placed in a specific file
- You add it to your store using the ddl add-schema command
  - kv-> ddl add-schema -file PersonSchema.avsc

UNIVERSITÀ TELEMATICA INTERNAZIONALE UNINETTUNO

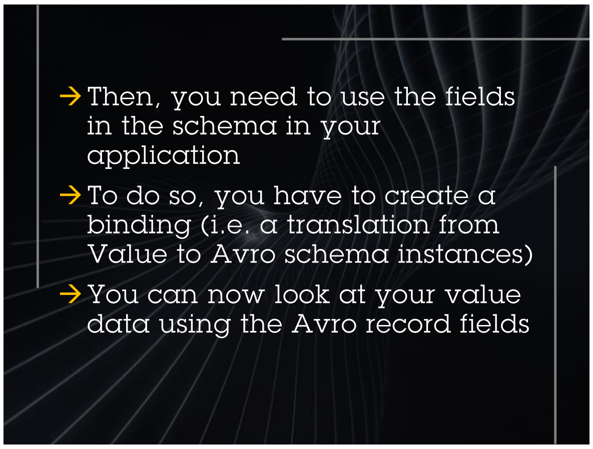
Copyright © Università Telematica Internazionale UNINETTUNO



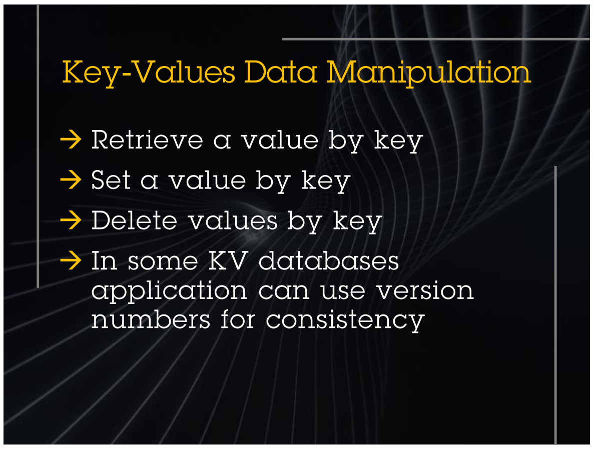
- You make the schema available to the code you write by reading it
  - final Schema.Parser parser = new Schema.Parser();
  - parser.parse(new File("PersonSchema.avsc"));



- Next, you need to make the schema available to your application:
  - final Schema personSchema = parser.getTypes().get("FVavro. PersonInformation"

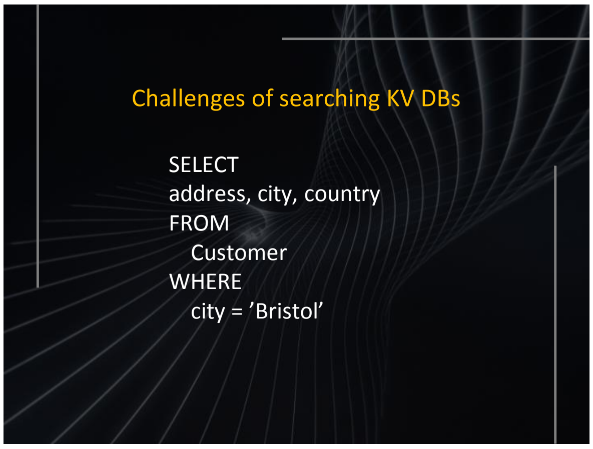


- Then, you need to use the fields in the schema in your application
- To do so, you have to create a binding (i.e. a translation from Value to Avro schema instances)
- You can now look at your value data using the Avro record fields



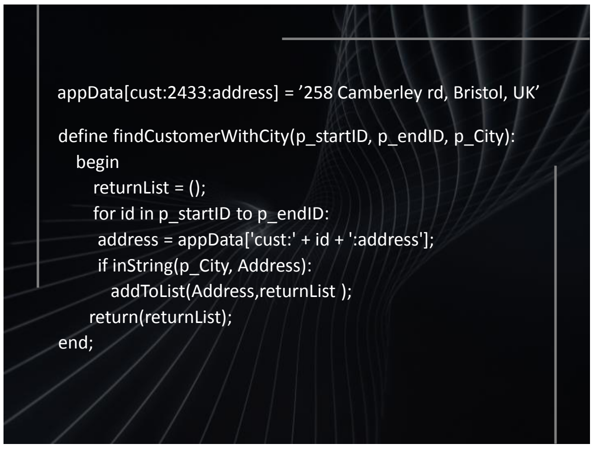
Key-Values Data Manipulation

- Retrieve a value by key
- Set a value by key
- Delete values by key
- In some KV databases application can use version numbers for consistency



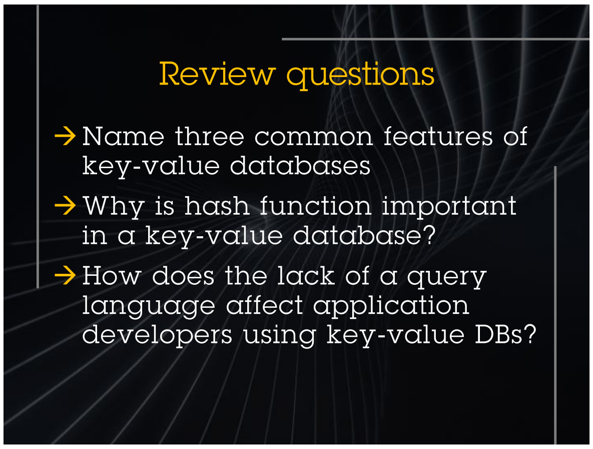
Challenges of searching KV DBs

```
SELECT
address, city, country
FROM
   Customer
WHERE
   city = 'Bristol'
```



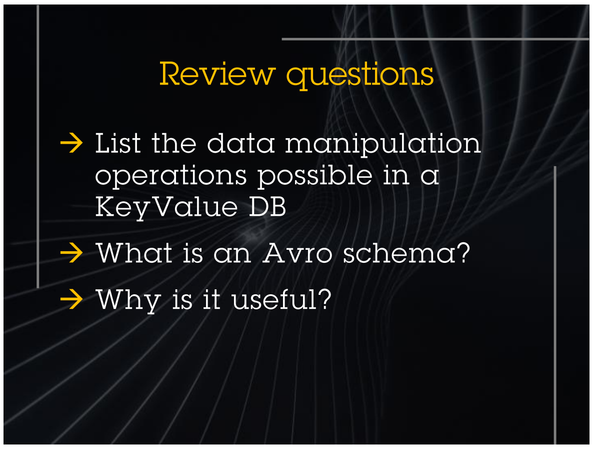
```
appData[cust:2433:address] = '258 Camberley rd, Bristol, UK'

define findCustomerWithCity(p_startID, p_endID, p_City):
   begin
      returnList = ();
      for id in p_startID to p_endID:
       address = appData['cust:' + id + ':address'];
       if inString(p_City, Address):
          addToList(Address,returnList );
     return(returnList);
end;
```



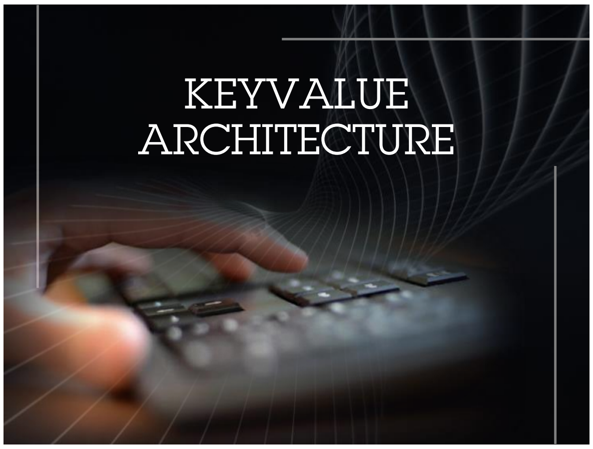
Review questions

- Name three common features of key-value databases
- Why is hash function important in a key-value database?
- How does the lack of a query language affect application developers using key-value DBs?

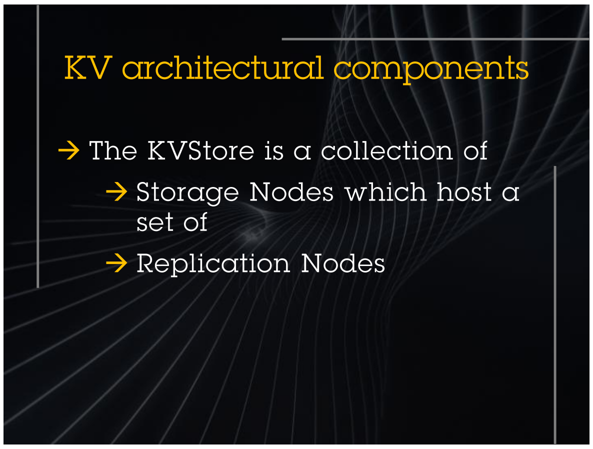


Review questions

- List the data manipulation operations possible in a KeyValue DB
- What is an Avro schema?
- Why is it useful?

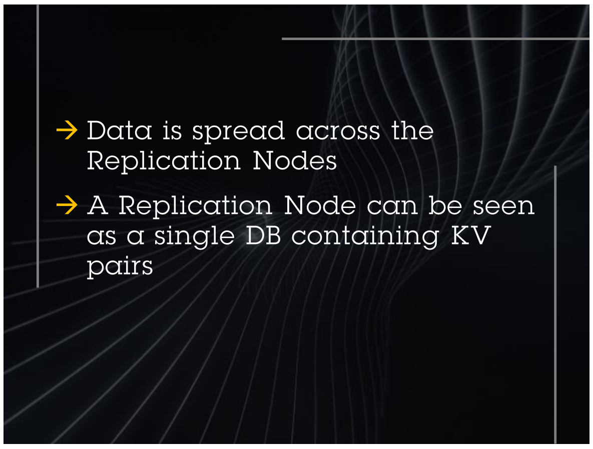


KEYVALUE ARCHITECTURE

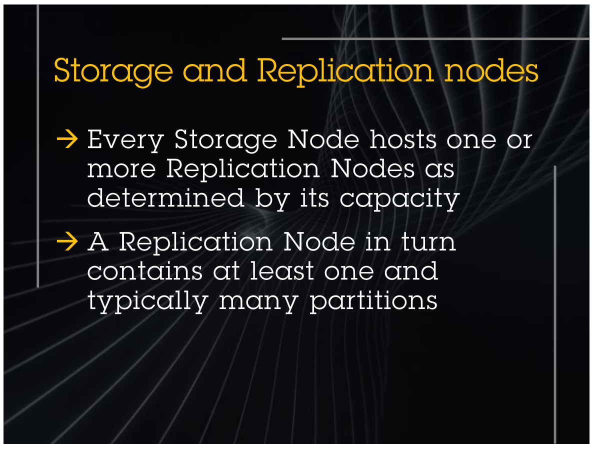


KV architectural components

- The KVStore is a collection of
  - Storage Nodes which host a set of
  - Replication Nodes



- Data is spread across the Replication Nodes
- A Replication Node can be seen as a single DB containing KV pairs

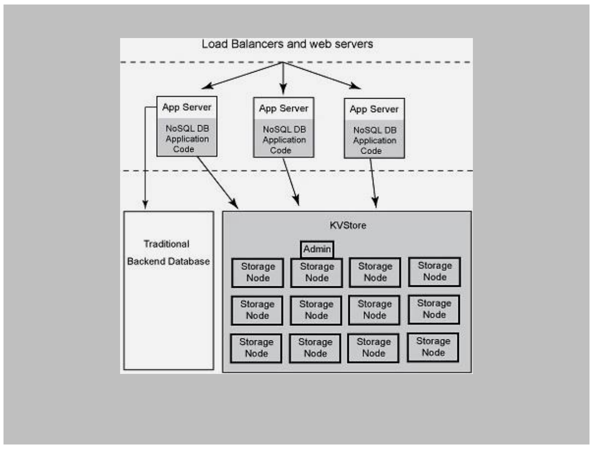


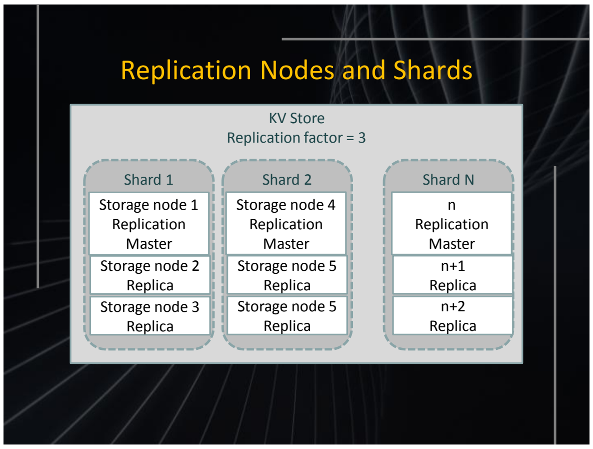
Storage and Replication nodes

- Every Storage Node hosts one or more Replication Nodes as determined by its capacity
- A Replication Node in turn contains at least one and typically many partitions

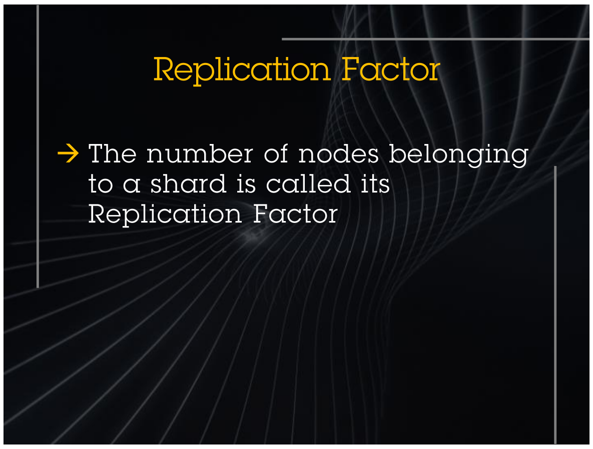
UNIVERSITÀ TELEMATICA INTERNAZIONALE UNINETTUNO

Copyright © Università Telematica Internazionale UNINETTUNO



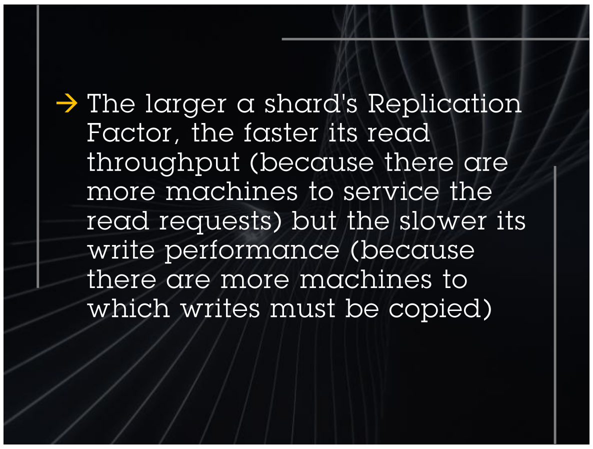


Replication Nodes and Shards

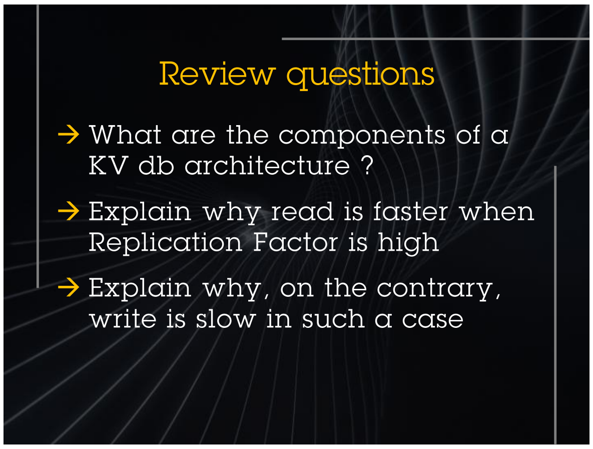


Replication Factor

→ The number of nodes belonging to a shard is called its Replication Factor



→ The larger a shard's Replication Factor, the faster its read throughput (because there are more machines to service the read requests) but the slower its write performance (because there are more machines to which writes must be copied)



Review questions

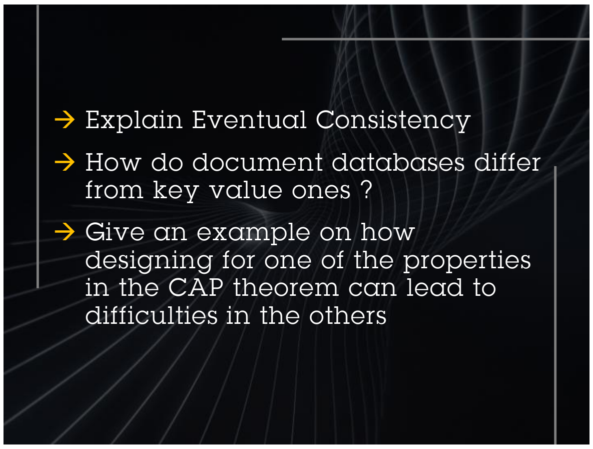
→ What are the components of a KV db architecture ?

→ Explain why read is faster when Replication Factor is high

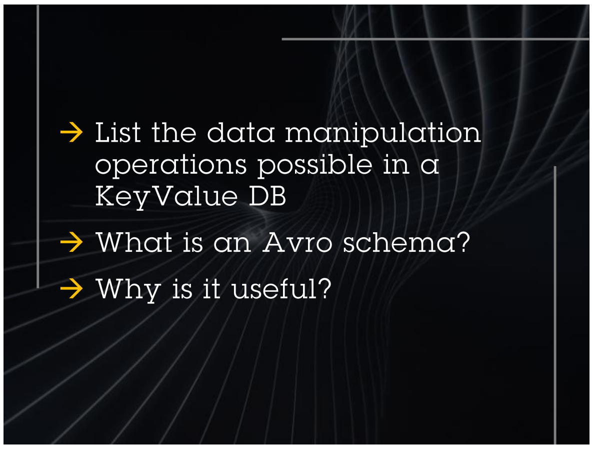
→ Explain why, on the contrary, write is slow in such a case

SUMMARY QUESTION

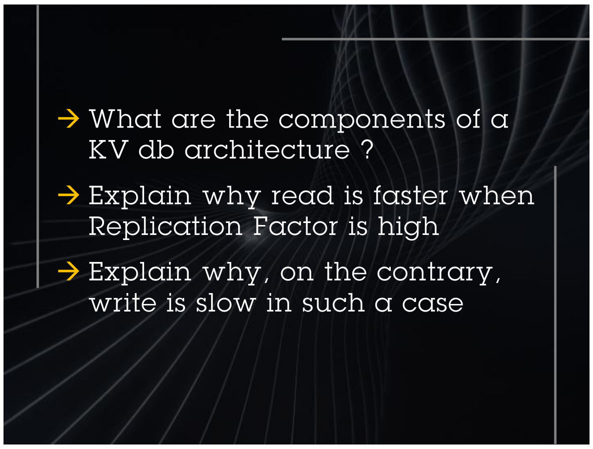
Copyright © Università Telematica Internazionale UNINETTUNO



- Explain Eventual Consistency
- How do document databases differ from key value ones ?
- Give an example on how designing for one of the properties in the CAP theorem can lead to difficulties in the others



- List the data manipulation operations possible in a KeyValue DB
- What is an Avro schema?
- Why is it useful?



- What are the components of a KV db architecture ?
- Explain why read is faster when Replication Factor is high
- Explain why, on the contrary, write is slow in such a case

Copyright © Università Telematica Internazionale UNINETTUNO