Databases and Information Systems CS303

Key Value Pair Databases
18-10-2023

Key Value Pair Database

- Modification of Array data structure

- Key-Value Pair Database is Permanent

Array

- Ordered list of elements.

- Examples:
  - A = [4, 3, 6, 1, 2]
  - B = [True, False, False, True, True, True, False]
  - C = [‘a’, ‘c’, ‘w’, ‘q’, ’t’, ‘b’]
  - D = [“Hello”, “Namaskara”, “Vanakkam”]

- A[2], B[0]…. denotes data in the corresponding position (index)

Associative Array

- Indices are user-defined.

- Example:
  - A[‘Pi’] = 3.14
  - A[‘CapitalFrance’] = ‘Paris’
  - A[17234] = 34468
  - A[‘Foo’] = ‘Bar’
  - A[‘Start_Value’] = 1

‘Pi’	3.14
‘CapitalFrance’	‘Paris’
17234	34468
‘Foo’	‘Bar’
‘Start_Value’	1

- Keys / Indices can be integers or strings

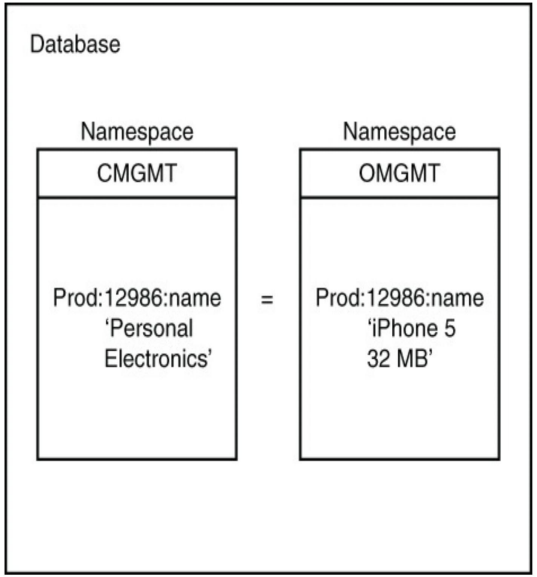
  Also called: Dictionary / HashMap / HashTable ....

Features of Key Value Pair Database

- Simplicity:
  - Bare minimum Data structure
  - No Joins
  - Flexible (Values have no pre-defined data types)
- Speed :
  - Suitable for Data Intensive operations
  - Delivers high speed
- Scalability :
  - Easy to add/remove servers depending on the requirements

Key

- Used to identify / index a key-value pair
- Must be unique within the namespace
- Namespace:
  - Collection of key-value pairs without duplicate keys
  - One namespace can be an entire database



Key

- How to construct keys?
  - CustName[123] = “Harry Potter”
  - CustCart[123] = [“Wand”, “Invisible Cloak”]
- But we have only one ‘Array’
- Use meaningful keys
  - Cust:123:Name = “Harry Potter”
  - Cust:123:Cart = [“Wand”, “Invisible Cloak”]
  - General Structure:
    - Entity_type: Entity_Identifier: Attribute_name = Attribute_value

Uses of Key

- To Locate Values
  - Uses Hashing to access a key location directly

- Values cannot be accessed directly in the query
  - Find the name of Customer with ID 123
    - What is the value for the key Cust:123:Name (allowed)
  - Find the ID of the Customer “Harry Potter”
    - What is the key for the value “Harry Potter” (NOT allowed)

Choice of Key

- Should be meaningful and unambiguous
- Use delimiter (typically :)
- Keep the key short without sacrificing previous properties
- Well designed key saves amount of Code
    - Example: If we use Cust: ID : Attribute_name as Key
    - Then single function with two attributes can retrieve any value

```
GetAttribValue ( ID, Attribute_name)
    Index = “Cust:”+ID+“:”+“Attribute_name”
    Return ( db[Index] )
```

Choice of Key

- Must Depend on the implementation
  - Size of Keys (Foundation DB)
  - Riak treats keys are binary values or strings
  - Redis allows various data types for keys

Values

- Can be anything
  - String, List, JSON
    - “Indian Institute of Technology, Dharwad, 580011”
    - [“Indian Institute of Technology”, “Dharwad” , 580011]
    - { ‘name’ : ‘Indian Institute of Technology’,
      ‘city’ : ‘Dharwad’,
      ‘pincode’ : 580011 }

Modelling the database

- Schemaless: No need to define what are the keys, values, data types etc

Key-Value Database	
Keys	Values
cust : 8983 : firstName	‘Jane’
cust : 8983 : lastName	‘Anderson’
cust : 8983 : fullName	‘Jane Anderson’

Updates and Deletes

- Search / Add/ Delete / Update all based only on Keys
- Does not support the access of the database based on Value

Advantages

- Simple
- Fast
- High availability
- Used in write intensive settings

Limitations

- Access to database only using Keys
- Does not support range queries
- There is no standard Language