Course code	: **CSE3009**
Course title	: **No SQL Data Bases**
Module	: **3**
Topic	: **2**

Key –Value Data Stores

Objectives

This session will give the knowledge about

- Properties of keys
- Characteristics of Values

Keys: Properties

keys are used to identify, index, or otherwise reference a value in a key-value database. The one essential property of a key is that it must be unique within a namespace.

How to Construct a Key

Designers could use one counter to generate primary keys for all tables, or they could use a different counter or sequence for each table.

In NoSQL databases, and key-value databases in general, the rules are different. Key-value databases do not have a built-in table structure. With no tables, there are no columns. With no columns, there is no way to know what a value is for except for the key.

Keys: Properties

Consider a shopping cart application using a key-value database with meaningless keys: Cart[12387] = ‘SKU AK8912j4’. Instead create another namespace, such as custName. Then you could save a value such as CustName[12387] = ‘Katherine Smith’

you can construct meaningful names that entail information about entity types, entity identifiers, and entity attributes. For example:

Cust:12387:firstName

could be a key to store the first name of the customer with customerID 12387.

Syntax: Entity Name + ‘:’ + Entity Identifier +’:’ + Entity Attribute
(The delimiter does not have to be a ':' but it is a common practice.)

Using Keys to Locate Values

There has been some mention of the idea that keys are used to look up associated values, but there has been no explanation about how that happens.

If key-value database designers were willing to restrict you to using integers as key values, then they would have an easy job of designing code to fetch or set values based on keys.

They could load a database into memory or store it on disk and assume that the first value stored in a namespace is referenced by key 1, the next value by key 2, and so on.

Using Keys to Locate Values

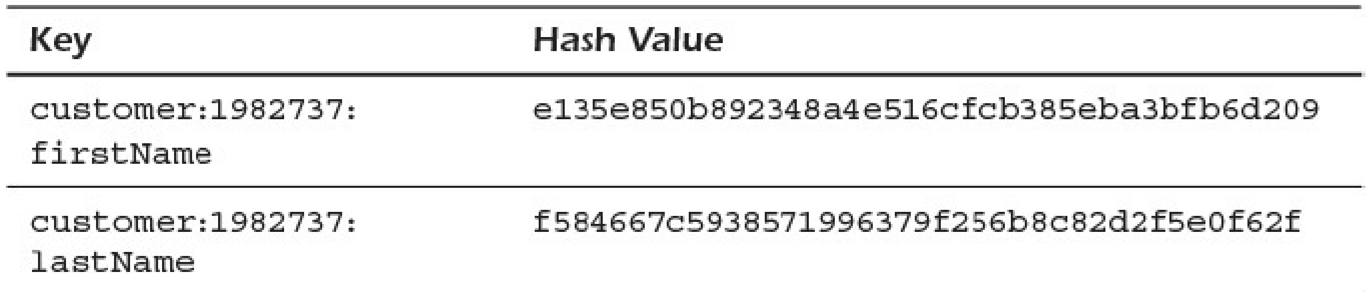
Using numbers to identify locations is a good idea, but it is not flexible enough. You should be able to use integers, character strings, and even lists of objects as keys if you want.

The trick is to use a function that maps from integers, character strings, or lists of objects to a unique string or number. These functions that map from one type of value to a number are known as hash functions.

Not all key-value databases support lists and other complex structures. Some are more restricted in the types and lengths of keys than others.

Hash Functions: From Keys to Locations

A hash function is a function that can take an arbitrary string of characters and produce a (usually) unique, fixed-length string of characters.



Key	Hash Value
customer:1982737: firstName	e135e850b892348a4e516cfcb385eba3bfb6d209
customer:1982737: lastName	f584667c5938571996379f256b8c82d2f5e0f62f

Each hash value is quite different from the others, although they all have the same 'customer:1982737:‘ here, the SHA-1 hash function is used to generate the hash values.

Actually, the value returned by the hash function is not always unique; sometimes two unrelated inputs can generate the same output. This is known as a collision.

Keys Help Avoid Write Problems

Assume you are working with the eight-server cluster. You can take advantage of the fact that the hash function returns a number.

Because the write load should be evenly distributed across all eight servers, you can send one eighth of all writes to each server.

You could send the first write to Server 1, the second to Server 2, the third to Server 3, and so on in a round-robin fashion, but this would not take advantage of the hash value.

Keys Help Avoid Write Problems

One way to take advantage of the hash value is to start by dividing the hash value by the number of servers. Sometimes the hash value will divide evenly by the number of servers.

If the hash function returns the number 32 and that number is divided by 8, then the remainder is 0. If the hash function returns 41 and it is divided by 8, then the remainder is 1. If the hash function returns 67, division by 8 leaves a remainder of 3.

As you can see, any division by 8 will have a remainder between 0 and 7. Each of the eight servers can be assigned a number between 0 and 7.

Values: Characteristics

NoSQL databases are also simple with respect to the way they store data. Key-value databases do not expect you to specify types for the values you store.

You could, for example, store a string along with a key for a customer's address:

‘1232 NE River Ave, St. Louis, MO’

or you could store a list of the form:

(‘1232 NE River Ave’, ‘St. Louis’, ‘MO’)

or you could store a more structured format using JavaScript Object Notation, such as

{ ‘Street:’ : ‘1232 NE River Ave’, ‘City’ : ‘St. Louis’,: ‘State’ : ‘MO’ }

Values: Characteristics

Key-value databases make minimal assumptions about the structure of data stored in the database.

While in theory, key-value databases allow for arbitrary types of values, in practice database designers have to make implementation choices that lead to some restrictions.

Different implementations of key-value databases have different restrictions on values.

Values: Characteristics

It is important to consider the design characteristics of the key-value database you choose to use.

Consult the documentation for limitations on keys and values. Part of the process in choosing a key-value database is considering the trade-off of various features.

One key-value database might offer ACID transactions but limit you to small keys and values. Another key-value data store might allow for large values but limit keys to numbers or strings.

Your application requirements should be considered when weighing the advantages and disadvantages of different database systems.

Limitations on Searching for Values

Keep in mind that in key-value databases, operations on values are all based on keys. You can retrieve a value by key, you can set a value by key, and you can delete values by key.

Key-value databases do not support query languages for searching over values. There are two ways to address this limitation.

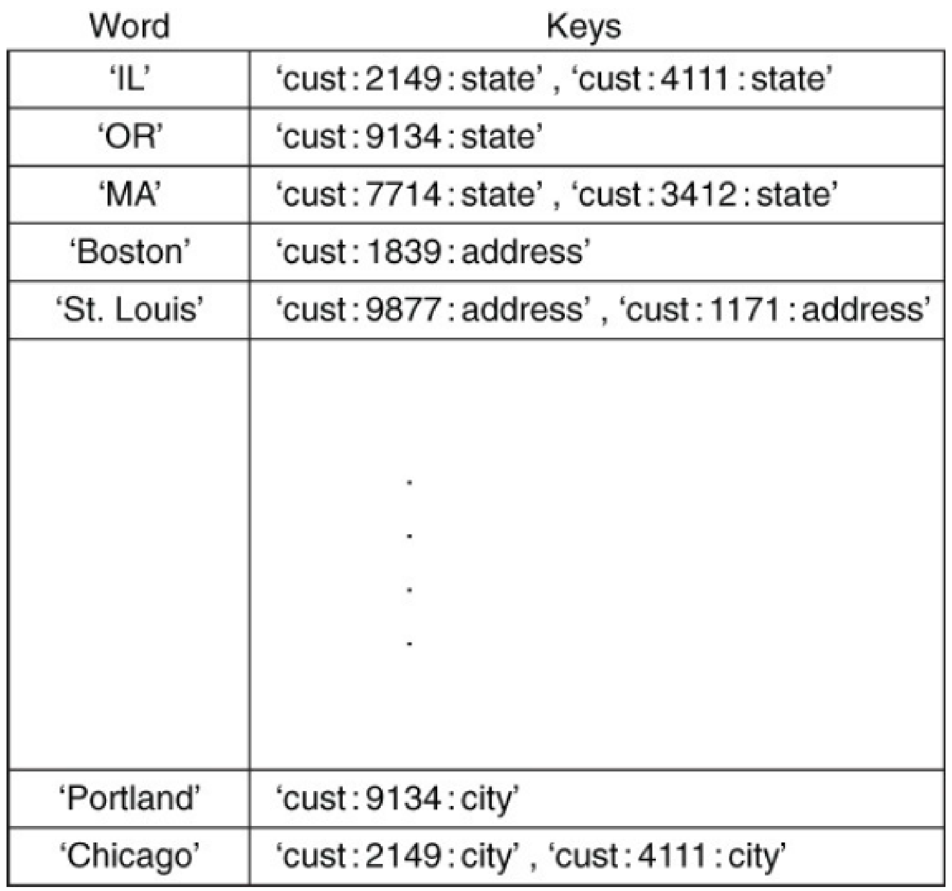
Some key-value databases incorporate search functionality directly into the database. This is an additional service not typically found in key-value databases but can significantly add to the usefulness of the database.

Limitations on Searching for Values

A built-in search system would index the string values stored in the database and create an index for rapid retrieval.

Rather than search all values for a string, the search system keeps a list of words with the keys of each key-value pair in which that word appears.

A search index helps efficiently retrieve data when selecting by criteria based on values.



Word	Keys
'IL'	'cust:2149:state', 'cust:4111:state'
'OR'	'cust:9134:state'
'MA'	'cust:7714:state', 'cust:3412:state'
'Boston'	'cust:1839:address'
'St. Louis'	'cust:9877:address', 'cust:1171:address'
	. . . .
'Portland'	'cust:9134:city'
'Chicago'	'cust:2149:city', 'cust:4111:city'

Summary

This session will give the knowledge about

- Properties of keys
- Characteristics of Values
- Reference: NoSQL for Mere Mortals by Dan Sullivan