NoSQL Databases

These are databases that are NOT organized around tables and not around objects as primary data structures.

They do not use SQL as the method to access data.

Recently several of those have become popular.

Why?

The two main problems with relational databases are:

1) They are often inefficient when many big joins have to be performed. And due to normalization, joins ALWAYS have to performed.

(Remember this was one of the reasons for the OO model too!)

2) Much of modern data goes beyond the simple data values that are stored in tables. People want to store images, videos, sound files, and whole documents.

This lecture is based on:

2) Numerous Wikipedia pages.

3) Home pages of the different systems.

This website lists many more:

http://nosql.findthebest.com/

Four major kinds:

1) Key-Value Store

Like a Hashtable in Java.
But more complicated, of course.

You send in the key, you get back the value.

Examples: Redis, Riak.
2) Columnar Databases

On disk, all data values of one column are stored together.

(In a normal relational database data rows are stored together.)

Examples: HBase, Cassandra

3) Document Databases

An extension of the Key-Value model. Very flexible.

Examples: MongoDB, CouchDB

4) Graph Databases

Designed for storing "node and link" structures.

Example: Neo4J

What's a Key-Value store.

When you insert data, you provide pairs of data items. When you query you provide the first element of a pair and expect to get the second element back.

{"firstname": "John", "lastname": "Smith", "city": "Newark"}

Quick Introduction to Six NoSQL Databases

MongoDB

-------

MongoDB is a document database. It's name comes from huMONGOus DataBase.

(The name "document" is misleading though.)

Mongo is a database of JSON documents.

What is JSON?

Javascript Object Notation
JSON is a language-independent data format. It is not connected to JavaScript anymore.

JSON example from WikiPedia:

```
{
    "firstName": "John",
    "lastName": "Smith",
    "age": 25,
    "address": {
        "streetAddress": "21 2nd Street",
        "city": "New York",
        "state": "NY",
        "postalCode": 10021
    },
    "phoneNumbers": [
        {
            "type": "home",
            "number": "212 555-1234"
        },
        {
            "type": "fax",
            "number": "646 555-4567"
        }
    ]
}
```

http://www.w3schools.com/json/default.asp

We have studied XML. But know now that JSON is considered as a more efficient alternative to XML.

A Mongo document is like a relational table row, without a schema. The values can be nested to any depth.

Just by looking at the example above we see that a curly bracket { starts a nested structure for one key. So that is a "value" that itself consists of keys and values. For example, "address" is nested.

A square bracket [ starts a list of values for one key.

MongoDB is free and open source. MongoDB has been adopted as backend software by a number of major websites and services, including Craigslist, eBay, Foursquare, SourceForge, and The New York Times.

http://docs.mongodb.org/manual/tutorial/getting-started/
Based on early work of Amazon. Written in the programming language Erlang. Erlang was designed by Ericsson (the phone company). It supports "hot swapping" which means the program can be changed without stopping it and restarting it.

Riak is a Key-Value store that is fault-tolerant by being replicated on several (typically 3) "nodes" (computers).

Riak databases are accessed over the web, with a URL. The main operations are:

- POST (that means create)
- PUT (update)
- GET (read back)
- DELETE (delete)

(People call these operations generically CRUD... create, update, read, delete)
So the above shows you "how you say CRUD in Riak."

Access is possible from the languages Ruby, Java, Erlang, Python, PHP, and C/C++

Side Comment:

Riak supports Mapreduce. (Or MapReduce).

What is Mapreduce?

"Mapreduce" is a framework for processing parallelizable problems across huge datasets using a large number of computers (nodes), collectively referred to as a cluster.

"Map" step: The master node takes the input, divides it into smaller sub-problems, and distributes them to worker nodes. The worker node processes the smaller problem, and passes the answer back to its master node.

"Reduce" step: The master node then collects the answers to all the sub-problems and combines them in some way to form the output the answer to the problem it was originally trying to solve.
Example: Find the largest number of a million numbers. You have 11 nodes (processors). One Master Node and 10 worker nodes.

Map: Send 100,000 numbers to each node. So every number sits on one of the 10 worker nodes.

Each worker node now finds the largest number of its 100,000 numbers and send it back to the master node.

Reduce: The master node now has 10 numbers and finds what the largest of them is.

HBase
-----
A columnar database. It stores whole columns together. Written in Java. Distributed by the Apache Software Foundation.

<table>
<thead>
<tr>
<th>ID</th>
<th>Last</th>
<th>First</th>
<th>Bonus</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Doe</td>
<td>John</td>
<td>8000</td>
</tr>
<tr>
<td>2</td>
<td>Smith</td>
<td>Jane</td>
<td>4000</td>
</tr>
<tr>
<td>3</td>
<td>Beck</td>
<td>Sam</td>
<td>1000</td>
</tr>
</tbody>
</table>

In a row-oriented database management system, the data would be stored like this:

1, Doe, John, 8000; 2, Smith, Jane, 4000; 3, Beck, Sam, 1000;

In a column-oriented database management system, the data would be stored like this:

1, 2, 3; Doe, Smith, Beck; John, Jane, Sam; 8000, 4000, 1000;

Basically, HBASE is a two level system of key-value pairs.

An HBase table consists of rows, keys, column families columns and values.

A key identifies a row. Within each column family there are several columns. A column identifies a value within a column family.

See a figures at these web sites:

http://chase-seibert.github.io/blog/2013/04/26/hbase-schema-design.html

This is really good for systems with lots of NULL values!
HBASE is based on HADOOP.

Now that is a lecture by itself.
And it is a hot topic.

Here is the Wikipedia definition of Hadoop:

............... 
Apache Hadoop is an open-source software framework for distributed storage and distributed processing of Big Data on clusters of commodity hardware.

Its Hadoop Distributed File System (HDFS) splits files into large blocks (default 64MB or 128MB) and distributes the blocks amongst the nodes in the cluster.

For processing the data, the Hadoop Map/Reduce ships code (specifically Jar files) to the nodes that have the required data, and the nodes then process the data in parallel.

This approach takes advantage of data locality, in contrast to conventional HPC architecture which usually relies on a parallel file system (compute and data separated, but connected with high-speed networking).

............... 
CouchDB 
-------

Also written in Erlang. Can run on any equipment from an Android phone to a data center.

Name stands for Cluster Of Unreliable Commodity Hardware.

Like MongoDB stores JSON objects.

Very fault tolerant.

Also created by Apache.

Also allows MapReduce.

Queried from JavaScript.

Another term you will hear a lot:

REST = REpresentational State Transfer.

REST is a simple stateless architecture that generally runs over HTTP. Stateless means the server does not remember each client.

Simple HTTP is used to make calls between machines.

It us usually introduced as a simpler alternative to
SOAP (Simple Object Access Protocol).

Neo4J
-----

Neo4j is an open-source graph database, implemented in Java. Neo4j is a "disk-based, Java persistence engine that stores data structured in graphs rather than in tables". Neo4j is the most popular graph database.

Redis
-----

Redis is an open-source, in-memory, key-value data store. It is written in ANSI C.

Redis is accessible through almost any programming language. The official way of saying this is: Many languages have Redis bindings.

OK, you asked for it: C, C++, C#, Clojure, Common Lisp, Dart, Erlang, Go, Haskell, Haxe, Io, Java, JavaScript (Node.js), Lua, Objective-C, Perl, PHP, Pure Data, Python, R, Ruby, Scala, Smalltalk and Tcl

Redis supports
Lists of strings
Sets of strings (non-repeating and unordered)
Sorted sets of strings ordered by a score number
Key-value pairs (called hashes).

Redis typically holds the whole dataset in memory. But there are two persistence mechanisms.