

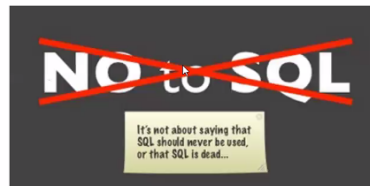
## Background

- Relational databases → mainstay of business
- Web-based applications caused spikes
  - explosion of social media sites (Facebook, Twitter) with large data needs
  - rise of cloud-based solutions such as Amazon S3 (simple storage solution)
- Hooking RDBMS to web-based application becomes trouble



## What is NOSQL?

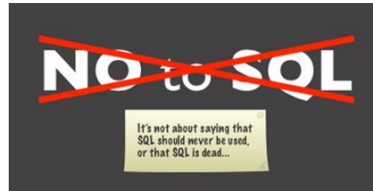
- The Name:
  - Stands for **Not Only SQL**
  - The term NOSQL was introduced by Carl Strozzi in 1998 to name his file-based database
  - It was again re-introduced by Eric Evans when an event was organized to discuss open source distributed databases
  - Eric states that “... *but the whole point of seeking alternatives is that you need to solve a problem that relational databases are a bad fit for.* ...”



## What is NOSQL?

### • The Name:

- Stands for **Not Only SQL** ✓
- The term NOSQL was introduced by Carl Strozzi in 1998 to name his file-based database
- It was again re-introduced by Eric Evans when an event was organized to discuss open source distributed databases
- Eric states that "... but the whole point of seeking alternatives is that you need to solve a problem that relational databases are a bad fit for. ..."



90% write  
10% read

Cash Table  
Account masks  
Data integrity

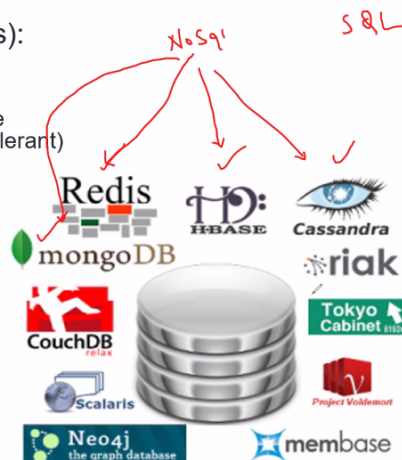
90% read  
10% write

- If more write, less read, and you want to maintain data integrity, use SQL [like in banking]
- If more read, less write, you want speed, and there is no certainty about how many users will connect, you use NOSQL [like in e-commerce]

## What is NOSQL?

### • Key features (advantages):

- non-relational
- don't require schema
- data are replicated to multiple nodes (so, identical & fault-tolerant) and can be partitioned:
  - down nodes easily replaced
  - no single point of failure
- horizontal scalable
- cheap, easy to implement (open-source)
- massive write performance
- fast key-value access



## → Types of NOSQL

- |                   |                                     |
|-------------------|-------------------------------------|
| a. Key-value pair | Amazon(DynamoDB)                    |
| b. Columnar       | Google's BigTable, Cassandra, HBase |
| c. Document       | MongoDB, CouchDB                    |
| d. Graph-based    | Neo4J                               |

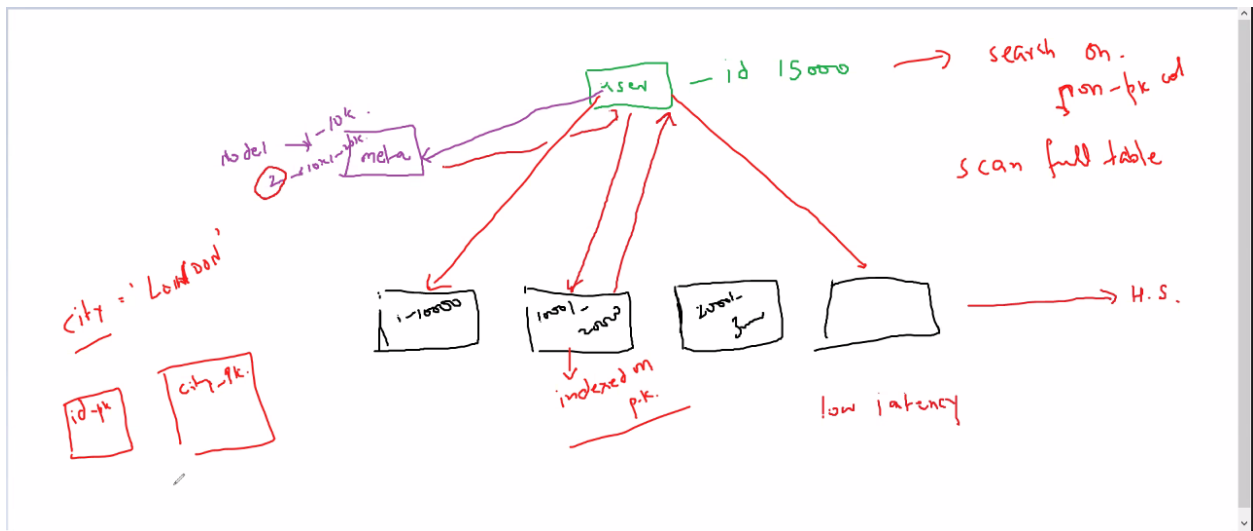
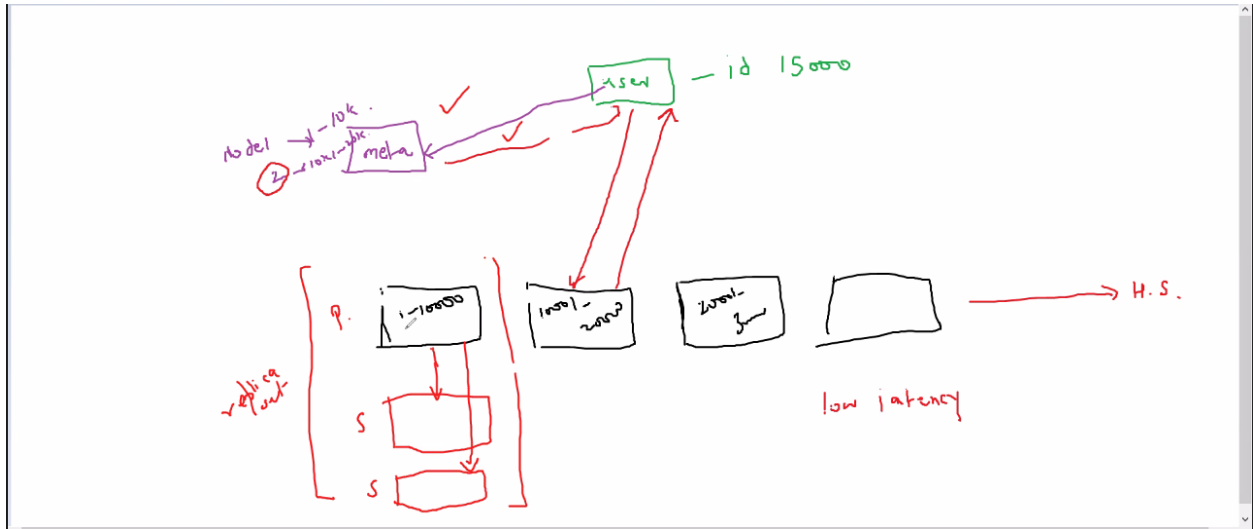
→

- a. In MongoDB, table is called collection, row is called document
- b. mongoDB follows node replication

```
Untitled - Notepad
File Edit Format View Help

    amt : 2000
  }
  {
    order_id : 105
    order_dt : 02.01.23
    amt : 2050
  }
  {
    order_id : 110
    order_dt : 01.02.23
    amt : 3000
  }
]

_id : 102
fname : Alan
lname : Smith
orders : [
  {
    order_id : 100
    order_dt : 01.01.23
    amt : 2000
  }
]
```



Introduction to NoSQL databases

RDBMS	NoSQL
tables are related	No relation between tables
joins are required	No joins
Data is normalized	Data is de-normalised
query can take time	query is fast
tables are designed as per storage	tables are designed as per the query
cannot handle big data	can handle big data
client-server	distributed data storage
are expensive	are all open-source
multiple records for same id	one record per id
schema is same	<u>schema-less</u>

use cases which are using NoSQL

- 1) Google search - Big Table
- 2) Amazon web app - DynamoDB
- 3) Aadhar data -- MongoDB/HBase - each aadhar id ---one record
- 4) ebay, trivago, makemytrip

common points

- 1) they are having big data
- 2) response time -- very very fast
- 3) more of query less of insert/update

writing is cheap, reading is expensive

*Handwritten notes:*

- very useful.
- mini table (Row 1, Row 2)
- col N, value → 10 cols
- 100 cols

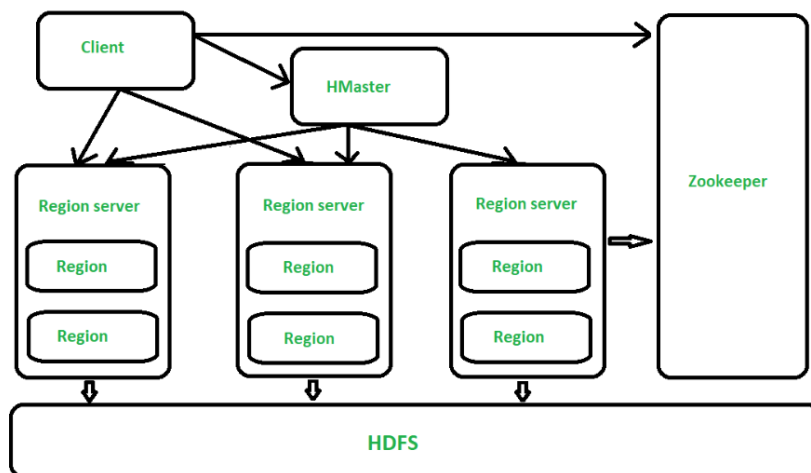
*Handwritten table:*

ID	Name	Age	City
1	John	15	NYC
2	Ryan	18	LA
3	Mary	21	CHI

→ writing is cheap because user can wait for writing, but reading is expensive because user might leave as he has to wait while app loads due to slow reading of data from server

## → HBase

- a. Fault tolerant way of storing sparse data sets
- b. Provides consistent Reads-Writes, and is linearly scalable
- c. Is based on the idea of Google's Big Table
- d. Built on top of HDFS, which stores HTable [MongoDB & Cassandra have independent frameworks, not dependent on HDFS]
- e. Has columnar structure
- f. HTable: table represented by a class in Hbase is called HTable
- g. HTable is divided into regions, regions are maintained by Region servers
- h. Regions are logical splits, but blocks are physical splits
- i. HMaster: stores all meta-data, and monitors Region Servers
- j. Zookeeper: coordinator in Hbase, clients communicate with region servers using Zookeeper
- k. Has only one dataType "ByteArray" for both Key & value, but TimeStamp is of Long type
- l. Data can grow bigger, while it has more rows
- m. Has Four column structure:
  - i. RowKey
  - ii. ColumnName / ColumnFamily
  - iii. TimeStamp
  - iv. Value
- n. Indexed on RowKey (if row key is same, then indexed on col name)

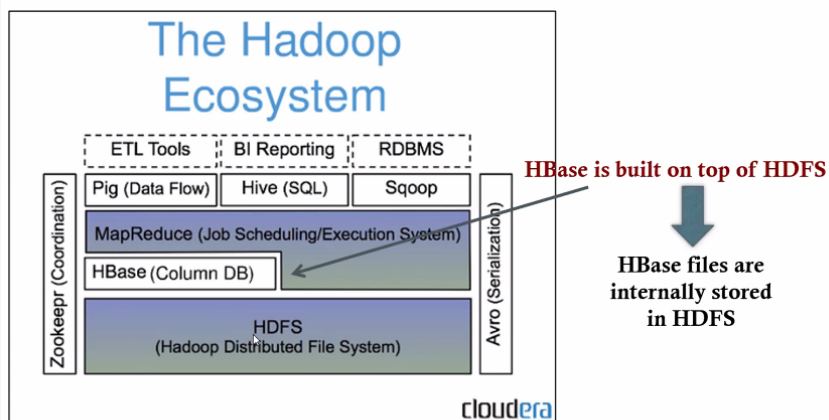
[illegible][illegible]

1	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
2	Employee table (logical view)																						
3	Rowkey	name	age	pincode																			
4	EMP001	John	30	123456																			
5																							
6	Rowkey	fname	lname	age	city																		
7	EMP002	Alan	Smith	25	NYC																		
8																							
9	Data model of HBase - HTable																						
10	Rowkey	col name	TS	value																			
11	EMP001	age	1234567890123	30																			
12	EMP001	name	1234567890123	John																			
13	EMP001	pincode	1234567890123	123456																			
14	EMP002	age	1234567890123	25																			
15	EMP002	city	1234567890123	NYC																			
16	EMP002	fname	1234567890123	Alan																			
17	EMP002	lname	1234567890123	Smith																			
18																							
19																							
20																							
21																							
22																							
23																							
24																							
25																							
26																							

Handwritten notes on the spreadsheet:

- 10,000 cols (written in red)
- Region - 1GB → HDFS (as blocks) 8 Blocks (written in red)
- Region 1GB (written in red)
- 1GB (written in red)
- HBase → HDFS (written in red)
- MongoDB + Camd → ind (written in red)
- HMarker (boxed in red)
- R.S (boxed in red)
- emp100 (written in red)

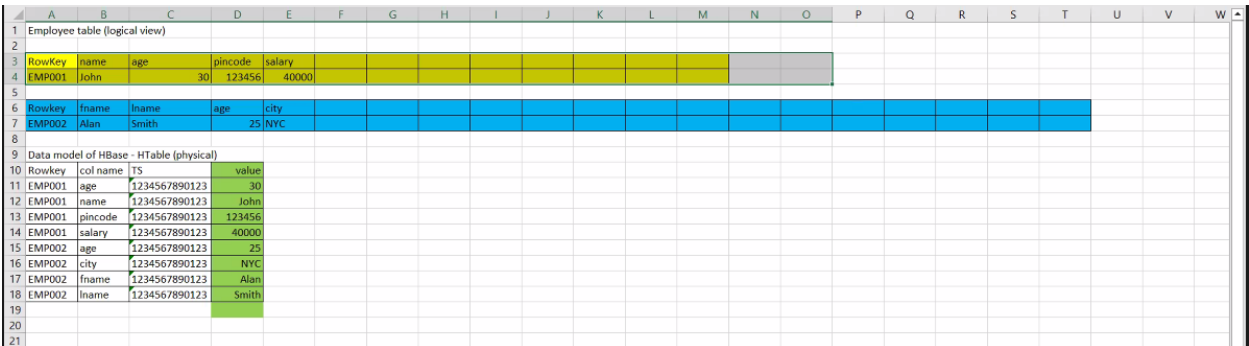
## HBase: Part of Hadoop's Ecosystem



# HBase vs. HDFS

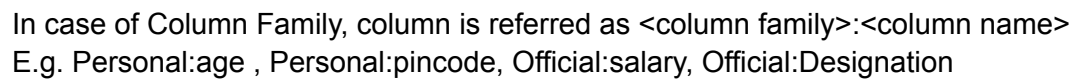
- Both are distributed systems that scale to hundreds or thousands of nodes
- **HDFS** is good for batch processing (scans over big files)
  - Not good for record lookup
  - Not good for incremental addition of small batches
  - Not good for updates

- # HBase Data Model
- HBase is based on Google's Bigtable model
    - Key-Value pairs
- 
- The diagram illustrates the HBase Data Model structure. A **Row key** (e.g., "com.cnn.www") points to a **Column Family** (e.g., "contents:"). Within the column family, there are **Key-Value pairs** (e.g., "CNN", "CNN.com") associated with **Timestamps** (e.g., t<sub>1</sub>, t<sub>2</sub>, t<sub>3</sub>). The **value** of the key-value pair is also shown (e.g., "CNN.com").





→ indexing is on Row Key, but if Row Key is also same, indexing is on Column Name, but if Column Name is also same, indexing will take place on descending order of TimeStamp



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	Employee table (logical view)																				
2	Personal					Official															
3	RowKey	name	age	pincode		desig	salary														
4	EMP001	John		30	456121	Manager	40000														
5																					
6	RowKey	fname	lname	age	city																
7	EMP002	Alan	Smith	25	NYC																
8																					
9	Data model of HBase - HTable (physical)																				
10	Rowkey	col name	TS	value		Rowkey	col name	TS	value												
11	EMP001	Personal:age	1234567890123	30		EMP001	Official:desig	1234567890123	Manager												
12	EMP001	Personal:name	1234567890123	John		EMP001	Official:salary	1234567890123	40000												
13	EMP001	pincode	1234567890999	456321																	
14	EMP001	pincode	1234567890123	123456																	
15																					
16	EMP002	age	1234567890123	25																	
17	EMP002	city	1234567890123	NYC																	
18	EMP002	lname	1234567890123	Alan																	
19	EMP002	fname	1234567890123	Smith																	
20																					
21																					
22	indexed on Rowkey, col_name, desc order of TS																				
23																					
24																					
25																					
26																					
27																					

→ Has only one dataType "ByteArray" for both Key & value, but TimeStamp is of Long type

- **Key**

- Byte array
- Serves as the primary key for the table
- Indexed for fast lookup

- **Column Family**

- Has a name (string)
- Contains one or more related columns

- **Column**

- Belongs to one column family
- Included inside the row
  - familyName:columnName

*Personal: age*

Row key	Time Stamp	Column "content s."	Column "anchor."	
"com.apache.www"	t12	"<html>..."		
	t11	"<html>..."		
	t10		"anchor.apache.com"	"APACHE"
"com.cnn.www"	t15		"anchor.cnn.com"	"CNN"
	t13		"anchor.mylook.ca"	"CNN.com"
	t6	"<html>..."		
	t5	"<html>..."		
	t3	"<html>..."		

11

Version number for each row

- **Version Number**

- Unique within each key
- By default → System's timestamp
- Data type is Long

- **Value (Cell)**

- Byte array

Row key	Time Stamp	Column "content s."	Column "anchor."	
"com.apache.www"	t12	"<html>..."		
	t11	"<html>..."		
	t10		"anchor.apache.com"	"APACHE"
"com.cnn.www"	t15		"anchor.cnn.com"	"CNN"
	t13		"anchor.mylook.ca"	"CNN.com"
	t6	"<html>..."		
	t5	"<html>..."		
	t3	"<html>..."		

12

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	Employee table (logical view)																				
2	Personal										Official										
3	Rowkey	name	age	pincode			desig	salary													
4	EMP001	John	30	456321			Manager	40000													
5																					
6	Rowkey	lname	lname	age	city																
7	EMP002	Alan	Smith	25	NYC																
8																					
9	Data model of HBase - HTable (physical)																				
10	Rowkey	col name	TS				Rowkey	col name	TS												
11	EMP001	Personal:age	1234567890123	30			EMP001	Official:desig	1234567890123	Manager											
12	EMP001	Personal:lname	1234567890123	John			EMP001	Official:salary	1234567890123	40000											
13	EMP001	pincode	1234567890999	456321																	
14	EMP001	pincode	1234567890123	123456																	
15	EMP002	age	1234567890123	25																	
16	EMP002	city	1234567890123	NYC																	
17	EMP002	lname	1234567890123	Alan																	
18	EMP002	lname	1234567890123	Smith																	
19																					
20																					
21	indexed on Rowkey, col_name, desc order of TS																				
22																					
23																					
24																					
25																					
26																					
27																					

## → Use Cases of HBase

### a. Civic administration:

- Has departments viz. Electric, PWD, Sanitation, Accounts
- Suppose, we have some roads in a city associated with different departments, and if one department is going to dig up some road to do some maintenance, it can make entry in table to communicate with other departments, so that road is not digged multiple times, and it could save some budget and time

A23							Rowkey														
5																					
6	Rowkey	lname	lname	age	city																
7	EMP002	Alan	Smith	25	NYC																
8																					
9	Data model of HBase - HTable (physical)																				
10	Rowkey	col name	TS																		
11	EMP001	Personal:age	1234567890123	30																	
12	EMP001	Personal:lname	1234567890123	John																	
13	EMP001	pincode	1234567890999	456321																	
14	EMP001	pincode	1234567890123	123456																	
15	EMP002	age	1234567890123	25																	
16	EMP002	city	1234567890123	NYC																	
17	EMP002	lname	1234567890123	Alan																	
18	EMP002	lname	1234567890123	Smith																	
19																					
20																					
21	indexed on Rowkey, col_name, desc order of TS																				
22																					
23	Rowkey																				
24	RD1																				
25																					
26																					
27																					

Untitled - Notepad

File Edit Format View Help

civic admin - depts

1) electric

2) pwd

3) sanitation

4) accounts and finance

Use - cases.

Centralized