BDA Expt 7

HBASE

Hadoop Database

Google File System

To solve distributed **storage**

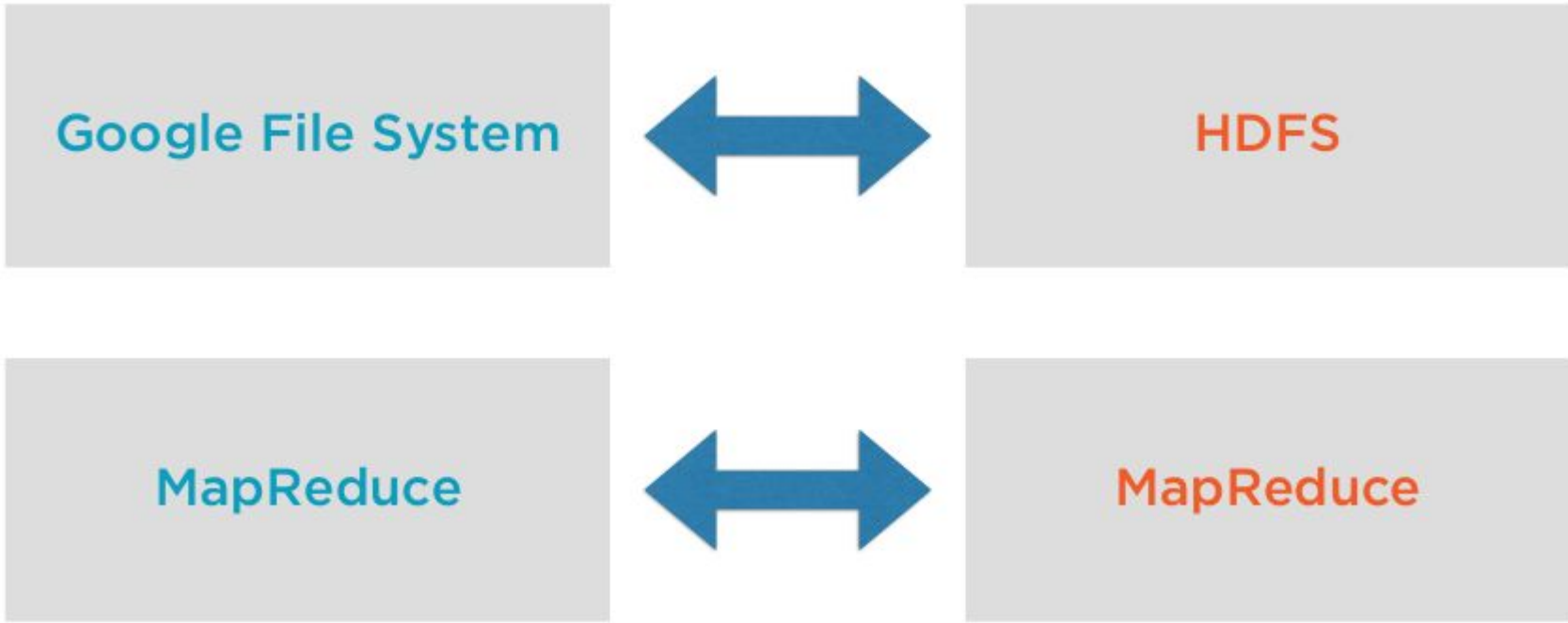
MapReduce

To solve distributed **computing**

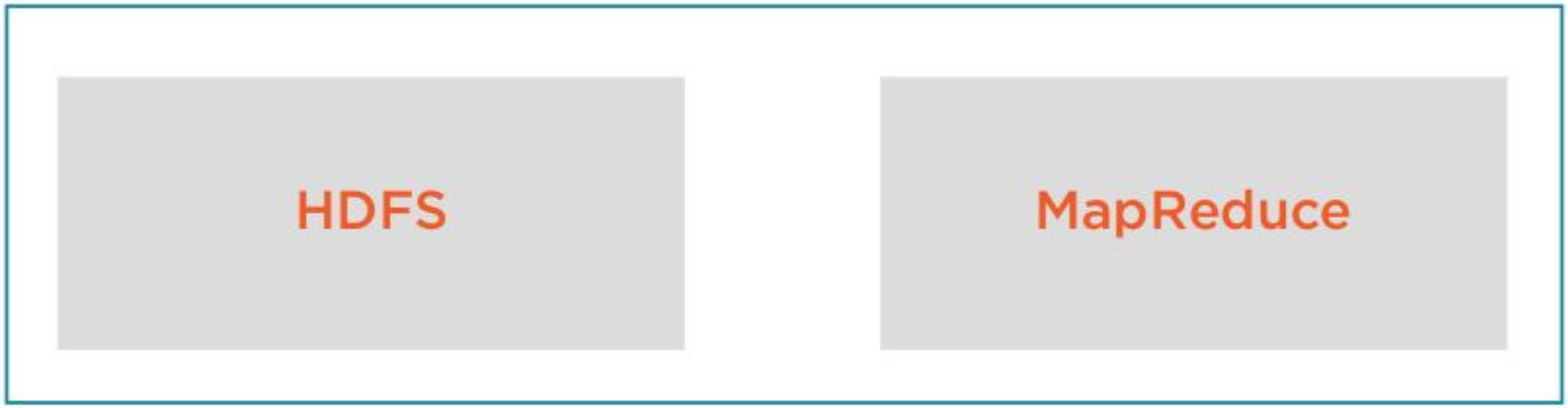
Google File System

MapReduce

Apache developed open source versions of these technologies


Google File System
HDFS
MapReduce
MapReduce

Hadoop



A file system to manage the storage of data

A framework to process data across multiple servers

Hadoop is a big data processing framework

Hadoop is **not** a database!

The Importance of Databases

What Kind of Data Do Organizations Store?

Order Management

An e-commerce site stores order information

Payroll

A company stores employee payroll details

Accounts

A bank stores account and transaction information

Requirements of a Database

Structured: Rows and columns

Random access: Update one row at a time

Low latency: Very fast read/write/ update operations

ACID compliant: Ensure data integrity

What Are ACID Properties?

Atomicity

Transactions on a database should be all-or-nothing

Atomicity

Transferring Money

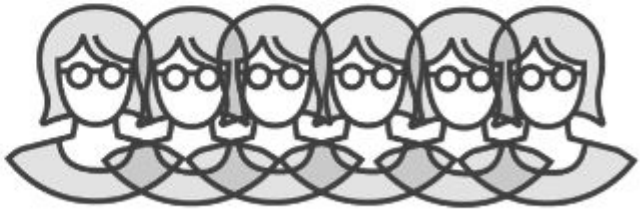
Both withdrawal and deposit should occur or none at all!

Consistency

Database updates should not violate any constraints

Consistency

Enrolling Students



Every student should have a unique student id

Isolation

Concurrent operations on the database should appear as though they were applied in some sequence

Isolation

Granularity of Updates

Can employee address be updated at the same time as employee salary?

Durability

Once changes have been made to the data they are permanent

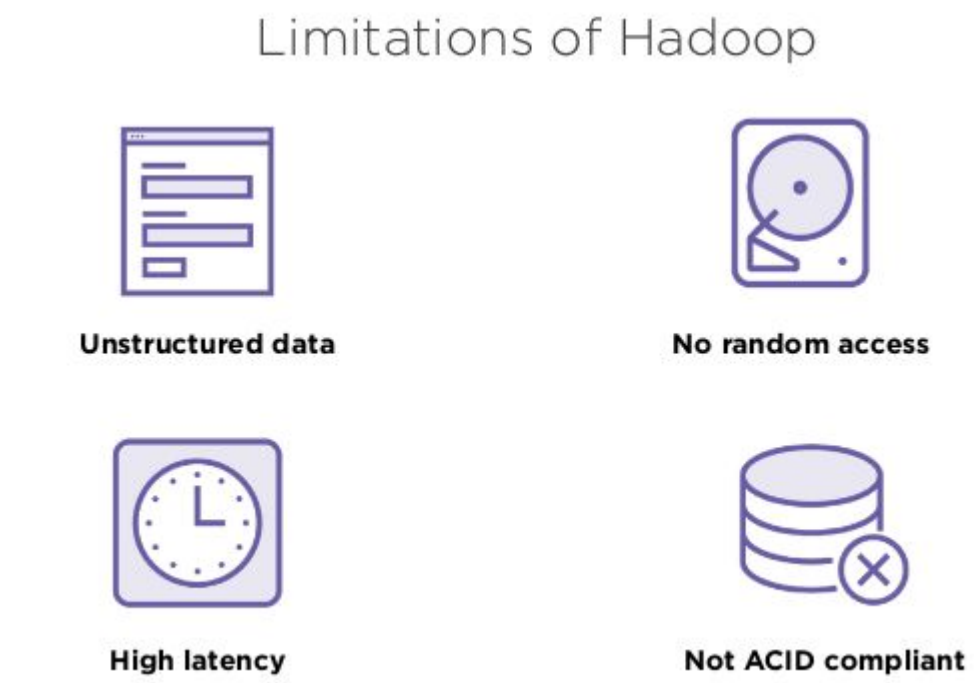
Durability

Safety of Data

In case of power loss, crashes, errors

Limitations of Hadoop

Unfortunately, Hadoop makes a very poor database



Limitations of Hadoop

Basic structure exists for some file types

- CSV files
- XML files
- JSON files

Hadoop enforces no constraints on these

Unstructured data

Limitations of Hadoop

Cannot create, access and modify individual records in a file

MapReduce parses **entire** files to extract

No random access

Limitations of Hadoop

Not suited for real-time processing where a user waits for data to be retrieved

Batch processing with long running jobs

High latency

Limitations of Hadoop

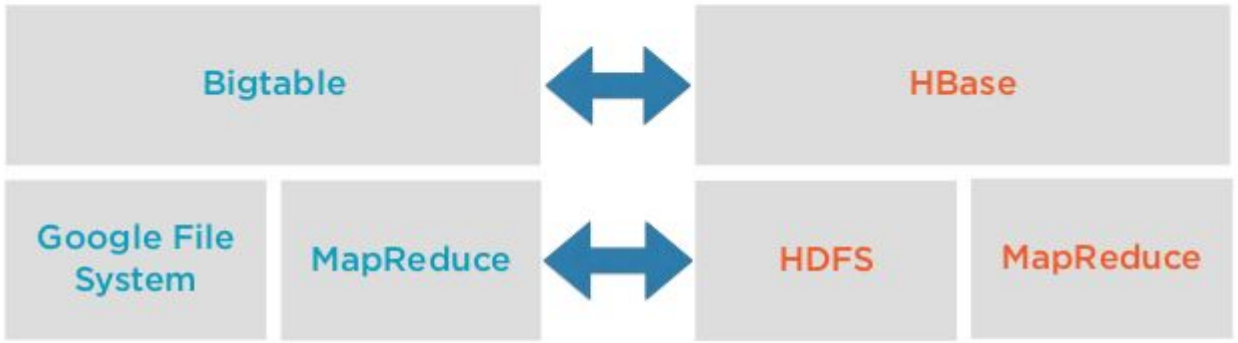
HDFS is a file storage system and provides no guarantees for data integrity

Not ACID compliant

How Did Google Solve This for Search?

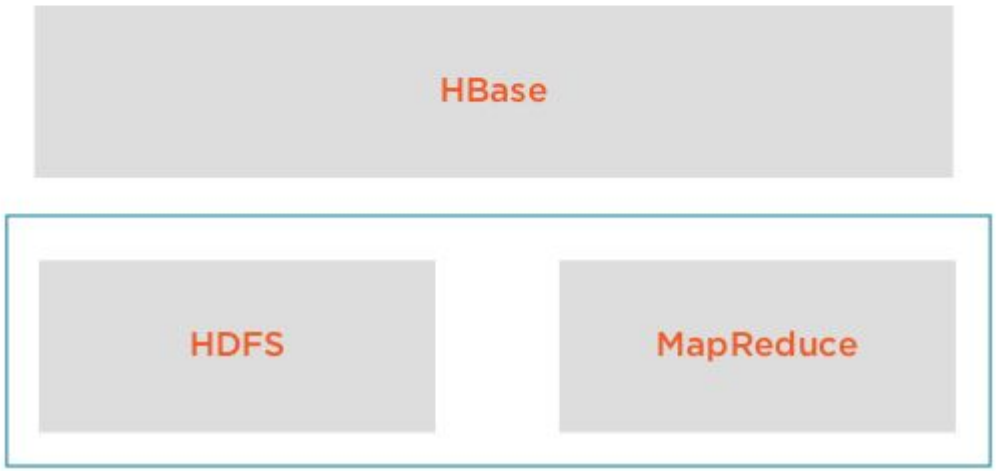
Google published a paper on Bigtable a distributed storage system for structured data

How Did Google Solve This for Search?



HBase is a distributed database management system which runs on top of Hadoop

HBase



HBase

Distributed: Stores data in HDFS

Scalable: Capacity directly proportional to number of nodes in the cluster

Fault tolerant: Piggybacks on Hadoop

Structured: A loose data structure

Low latency: Real-time access using row based indices called row keys

Random access: Row keys allow access updates to one record

Somewhat ACID compliant: Some transactions will have ACID properties

Properties of HBase

Columnar store

Denormalized storage

Only CRUD operations

ACID at the row level

Columnar Store

Columnar store

Columnar Store

Id	To	Type	Content
1	mike	offer	Offer on mobiles
2	john	sale	Redmi sale
3	jill	order	Order delivered
4	megan	sale	Clothes sale

Id	Column	Value
1	To	mike
1	Type	offer
1	Content	Offer on mobiles
2	To	john
2	Type	sale
2	Content	Redmi sale
3	To	jill
3	Type	order
3	Content	Order delivered
4	To	megan
4	Type	sale
4	Content	Clothes sale

Advantages of a Columnar Store

Sparse tables: No wastage of space when storing sparse data

Dynamic attributes: Update attributes dynamically without changing storage structure

Sparse Tables

Id	To	Type	Content	Expiry	Order Status
1	mike	offer	Offer on mobiles	2345689070	
2	john	sale	Redmi sale		
3	jill	order	Order delivered		Delivered
4	megan	sale	Clothes sale	2456123989	

And empty cells when data is not applicable to certain rows

These cells still occupy space.

Columnar store

Id	Column	Value
1	To	mike
1	Type	offer
1	Content	Offer on
1	Expiry	2345689070
2	To	john
2	Type	sale
2	Content	Redmi sale
3	To	jill
3	Type	order
3	Content	Order delivered
4	To	megan
4	Type	sale
4	Content	Clothes sale
4	Expiry	2456123989

Dynamically add new attributes as rows in this table

No wastage of space with empty cells!

Properties of HBase

Columnar store

Denormalized storage

Only CRUD operations

ACID at the row level

Denormalized Storage

Id	Name	Function	Grade
1	Emily	Finance	6

Id	Subordinate Id
1	2
1	3

Id	City	Zip Code
1	Palo Alto	94305
2	Seattle	98101

Normalization

- Normalization optimizes storage,
- But storage is cheap in distributed system.
- We have to optimize no. Of disk seeks.

Denormalized Storage

Id	Name	Function	Grade
1	Emily	Finance	6
2	John	Finance	3
3	Ben	Finance	4

Id	Subordinate Id
1	2
1	3

Id	Name	Function	Grade	Subordinates
1	Emily	Finance	6	<ARRAY>
2	John	Finance	3	
3	Ben	Finance	4	

Denormalized Storage

Id	Name	Function	Grade
1	Emily	Finance	6
2	John	Finance	3
3	Ben	Finance	4

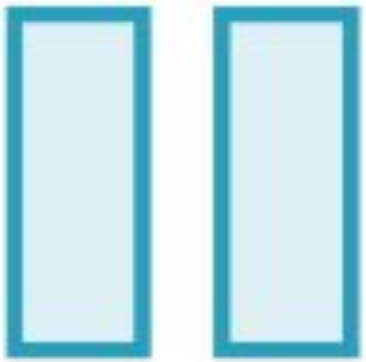
Id	City	Zip Code
1	Palo Alto	94305
2	Seattle	98101

Id	Name	Function	Grade	Subordinates	Address
1	Emily	Finance	6	<ARRAY>	<STRUCT>
2	John	Finance	3		
3	Ben	Finance	4		

Store everything related to an employee in the same table

Read a single record to get all details about an employee in one read operation

Properties of HBase



Columnar store

Denormalized storage

Only CRUD operations

ACID at the row level

Traditional Databases and SQL

Only CRUD operations

Joins: Combining information across tables using keys

Group By: Grouping and aggregating data for the groups

Order By: Sorting rows by a certain column

Only CRUD operations

HBase does not support SQL

NoSQL

Only CRUD operations

Only a limited set of operations are allowed in HBase

Create

Read

Update

Delete

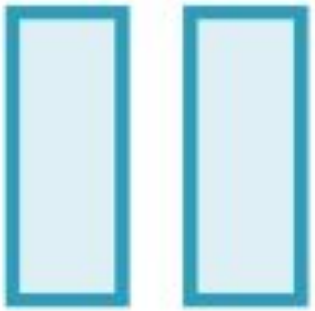
CRUD

Only CRUD operations

Id	Name	Function	Grade	Subordinates	Address

This is why all details need to be self contained in one row

Properties of HBase



Columnar store

Denormalized storage

Only CRUD operations

ACID at the row level

ACID at the row level

Updates to a single row are atomic

All columns in a row are updated or none are

ACID at the row level

Updates to multiple rows are not atomic

Even if the update is on the same column in multiple rows

Traditional RDBMS vs. HBase

Traditional RDBMS	HBase
Data arranged in rows and columns	Data arranged in a column-wise manner
Supports SQL	NoSQL database
Complex queries such as grouping, aggregates, joins etc	Only basic operations such as create, read, update and delete
Normalized storage to minimize redundancy and optimize space	Denormalized storage to minimize disk seeks
ACID compliant	ACID compliant at the row level

How Is Data Laid out in HBase?

Traditional database

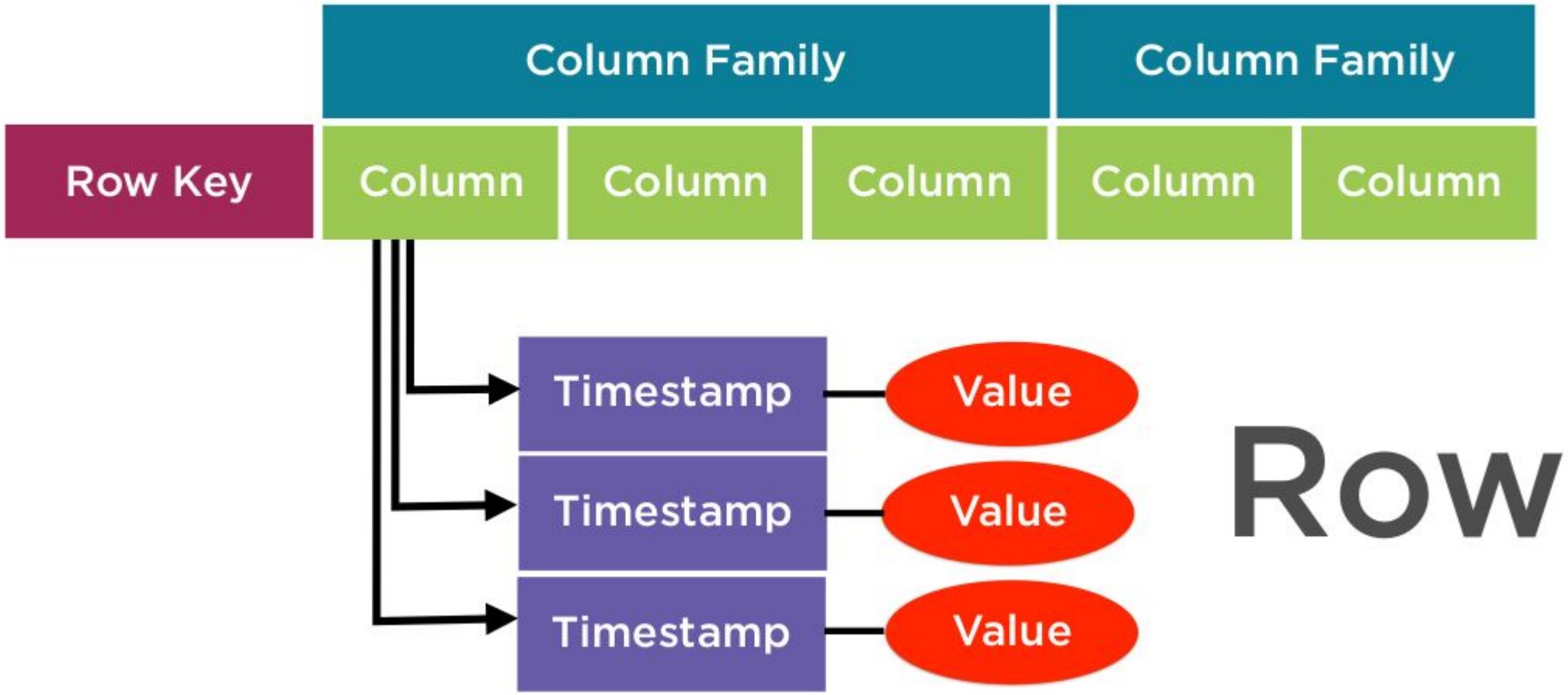
Id	To	Type	Content
1	mike	offer	Offer on mobiles
2	john	sale	Redmi sale
3	jill	order	Order delivered
4	megan	sale	Clothes sale

This is a 2-dimensional data model

HBase has a **4-dimensional** data model

4 Dimensions

4-dimensional Data Model



Census Data Layout in HBase

	Personal			Professional	
Some ID	name	gender	marital_status	employed	field

Notification Data

Id	To	Type	Content
1	mike	offer	Offer on mobiles
2	john	sale	Redmi sale
3	jill	order	Order delivered
4	megan	sale	Clothes sale

Id	Column	Value
1	To	mike
1	Type	offer
1	Content	Offer on mobiles
2	To	john
2	Type	sale
2	Content	Redmi sale
3	To	jill
3	Type	order
3	Content	Order delivered
4	To	megan
4	Type	sale
4	Content	Clothes sale

Row Key

Row Key

- Uniquely identifies a row
- Can be primitives, structures, arrays
- Represented internally as a byte array
- Sorted in ascending order

Id	To	Type	Content
1	mike	offer	Offer on mobiles
2	john	sale	Redmi sale
3	jill	order	Order delivered
4	megan	sale	Clothes sale

Id	Column	Value
1	To	mike
1	Type	offer
1	Content	Offer on mobiles
2	To	john
2	Type	sale
2	Content	Redmi sale
3	To	jill
3	Type	order
3	Content	Order delivered
4	To	megan
4	Type	sale
4	Content	Clothes sale

Column Family

Column Family

- All rows have the same set of column families
- Each column family is stored in a separate data file
- Set up at schema definition time
- Can have different columns for each row

Columns are units within a column family

New columns can be added on the fly

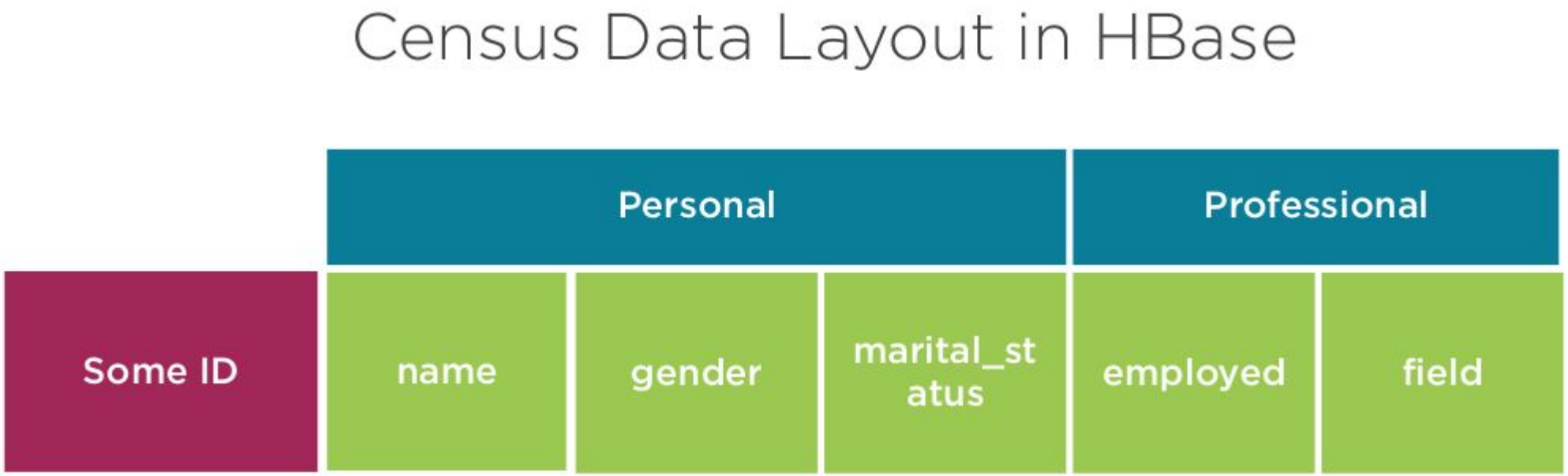
ColumnFamily: ColumnName = Work:Department

Timestamp

Used as the version number for the values stored in a column

The value for any version can be accessed

Hbase shell commands



```
put 'census', 1, 'personal:name', 'Mike Jones'
put 'census', 1, 'personal:marital_status',
'unmarried'
put 'census', 1, 'personal:gender', 'male'
put 'census', 1, 'professional:employed', 'yes'
put 'census', 1, 'professional:education_level',
'high school'
put 'census', 1, 'professional:field', 'construction'

put 'census', 3, 'personal:name', 'Jill Tang'
put 'census', 3, 'personal:marital_status', 'married'
put 'census', 3, 'personal:spouse', 'Jim Tang'
put 'census', 3, 'professional:education_level',
'post-grad'
put 'census', 3, 'personal:gender', 'female'
```

```
hbase(main):003:0> list
TABLE
Employees
Sales
notifications
3 row(s) in 0.0480 seconds
```

```
hbase(main):004:0> create 'census', 'personal', 'professional'
```

```
hbase(main):008:0> count 'census'
0 row(s) in 0.0510 seconds
```

```
hbase(main):007:0> describe 'census'
Table census is ENABLED
census
COLUMN FAMILIES DESCRIPTION
{NAME => 'personal', BLOOMFILTER => 'ROW', VERSIONS => '1', IN_MEMORY => 'false', KEEP_DELET
'FALSE', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', COMPRESSION => 'NONE', MIN_VERSION
OCKCACHE => 'true', BLOCKSIZE => '65536', REPLICATION_SCOPE => '0'}
{NAME => 'professional', BLOOMFILTER => 'ROW', VERSIONS => '1', IN_MEMORY => 'false', KEEP_D
 => 'FALSE', DATA_BLOCK_ENCODING => 'NONE', TTL => 'FOREVER', COMPRESSION => 'NONE', MIN_VER
, BLOCKCACHE => 'true', BLOCKSIZE => '65536', REPLICATION_SCOPE => '0'}
2 row(s) in 0.1170 seconds
```

Insert and update data using the HBase shell

- Add rows, each row represents data specific to one person
- Edit cells in a row

```
hbase(main):001:0> put 'census', 1, 'personal:name', 'Mike Jones'
```

The row key can be any data structure

```
hbase(main):001:0> put 'census', 1, 'personal:name', 'Mike Jones'
```

Insert data one cell at a time

The column family prefix for every column qualifier

```
put 'census', 1, 'personal:marital_status', 'unmarried'
```

Another column inserted for the same row

```
hbase(main):003:0> scan 'census'
ROW                           COLUMN+CELL
 1                            column=personal:marital_status, timestamp=1479390670583, value=unmarried
 1                            column=personal:name, timestamp=1479390660627, value=Mike Jones
1 row(s) in 0.0520 seconds
```

The timestamp serves as a version number for the value in the cell

- Syntax of put is same for insertion and updation.