Big Data Management

HBase (BigTable)

HBase: Overview

- **HBase is a distributed column-oriented data store built on HDFS**
- **HBase is Apache open source project for storage for Hadoop (BigTable Clone)**
- **Data is logically organized into tables, rows and columns**

HBase

- A distributed data store that can scale horizontally to 1,000s of commodity servers and petabytes of indexed storage.

- Designed to operate on top of Hadoop distributed file system (HDFS) for scalability, fault tolerance, and high availability.

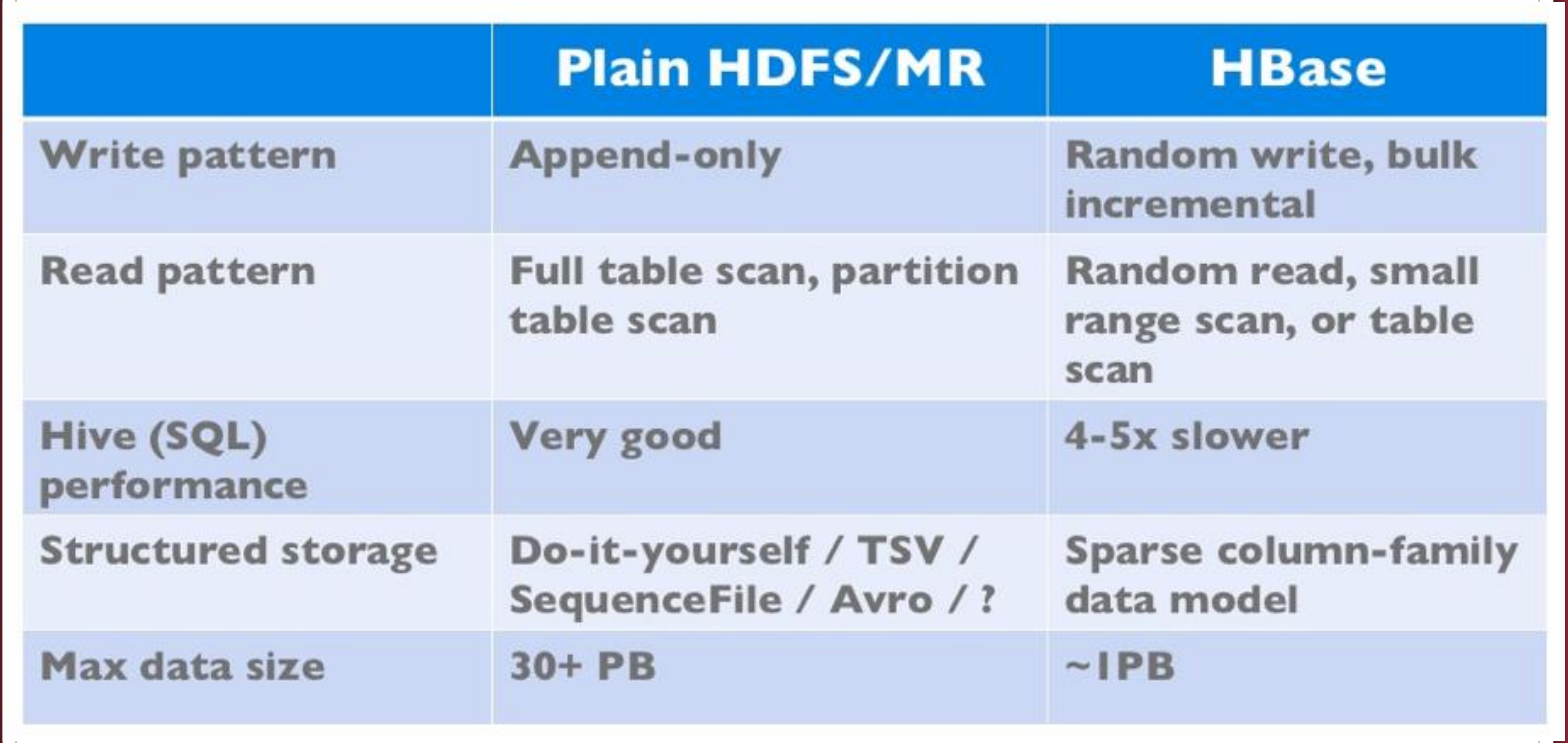
HBase vs. HDFS

- Both distributed systems scale to thousands of nodes
- ***<u>HDFS</u>*** is good for batch processing (scans over big files)
  - Not good for record lookup
  - Not good for incremental addition of small batches or updates
- ***<u>HBase</u>*** is designed to efficiently address the above
  - Fast record lookup
  - Support for record-level insertion & updates (but not in place)
- **HBase updates done by creating new versions of values**

HBase Is Not …

- Tables have one primary index, the *row key*.
- There are three types of lookups (select columns):
  - Fast lookup using row key and optional timestamp.
  - Full table scan
  - Range scan from region start to end.
  - **(No join operators !)**
- Limited atomicity and transaction support:
  - HBase supports multiple batched mutations of single rows only.
- Not accessed or manipulated via SQL:
  - Programmatic access via Java, REST, or Thrift APIs.

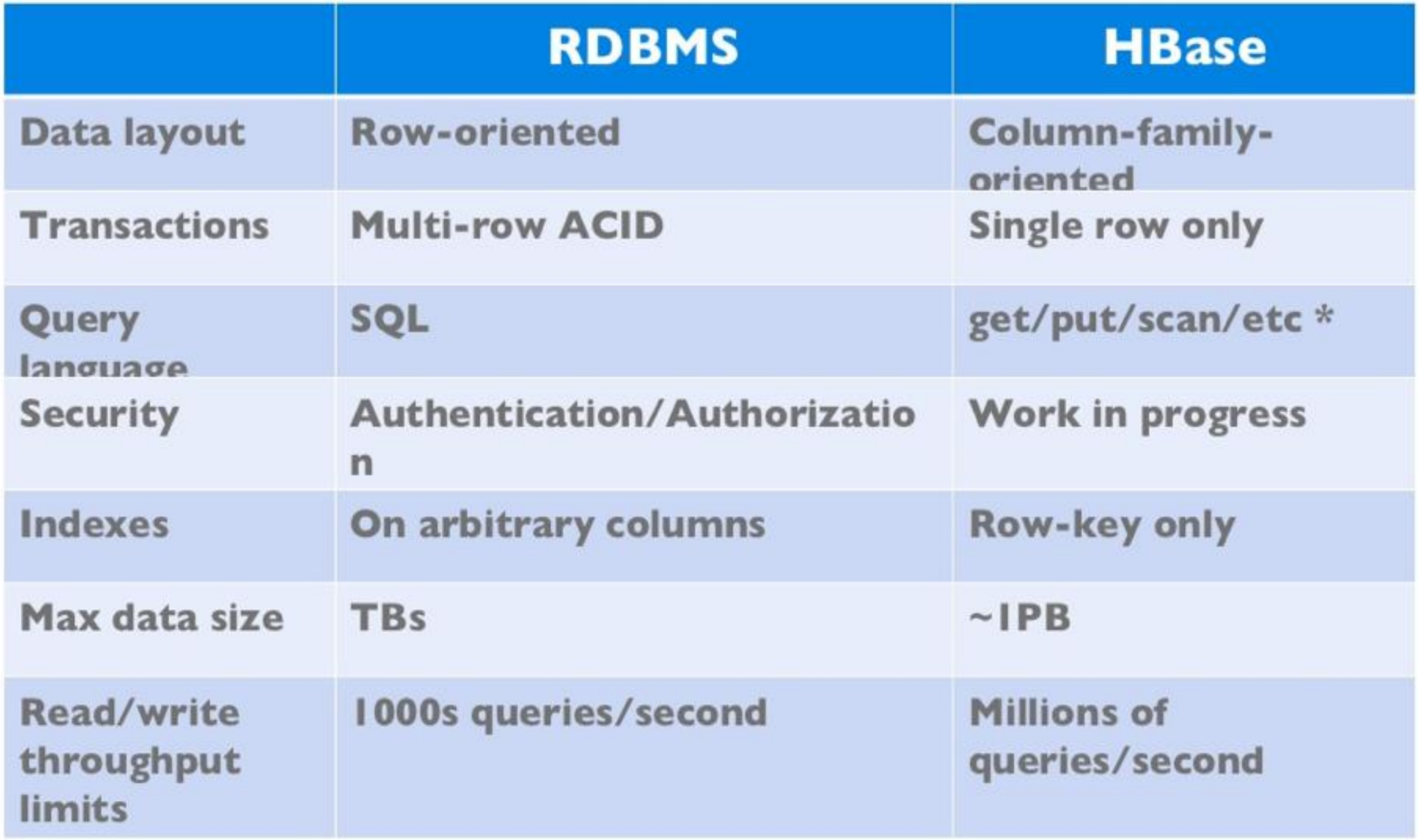
HBase vs. HDFS



	Plain HDFS/MR	HBase
Write pattern	Append-only	Random write, bulk incremental
Read pattern	Full table scan, partition table scan	Random read, small range scan, or table scan
Hive (SQL) performance	Very good	4-5x slower
Structured storage	Do-it-yourself / TSV / SequenceFile / Avro / ?	Sparse column-family data model
Max data size	30+ PB	~1PB

If application has neither random reads/writes ➔ Stick to HDFS

HBase vs. RDBMS



	RDBMS	HBase
Data layout	Row-oriented	Column-family-oriented
Transactions	Multi-row ACID	Single row only
Query language	SQL	get/put/scan/etc *
Security	Authentication/Authorizatio n	Work in progress
Indexes	On arbitrary columns	Row-key only
Max data size	TBs	~1PB
Read/write throughput limits	1000s queries/second	Millions of queries/second

HBase Data Model (Google's BigTable Model)

Tables are sorted by Row

Table schema define its *column families* .

- Each family consists of any number of columns
- Each column consists of any number of versions
- Columns only exist when inserted, NULLs are free.
- Columns within family are sorted & stored together

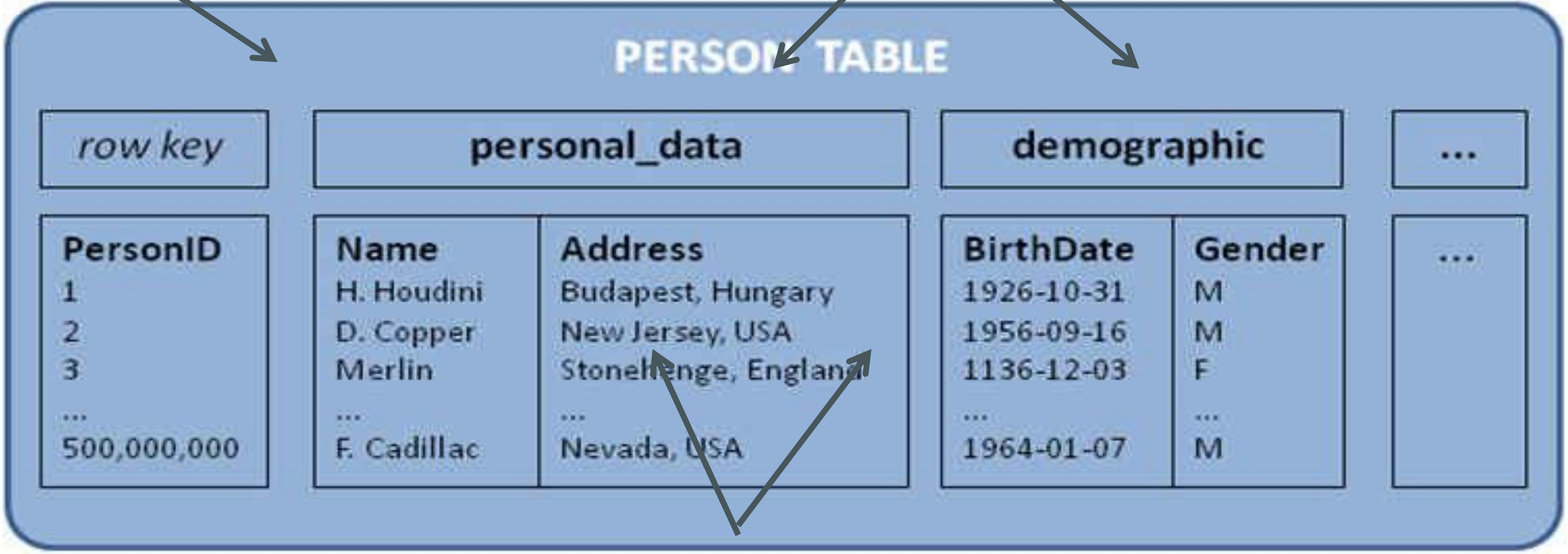
Everything except table names are byte[]

(Row, Family: Column, Timestamp) → Value

HBase: Keys & Column Families

Each row has a ***Key***

Each record is divided into ***Column Families***



PERSON TABLE

row key	personal_data		demographic		...
PersonID	Name	Address	BirthDate	Gender	...
1	H. Houdini	Budapest, Hungary	1926-10-31	M	
2	D. Copper	New Jersey, USA	1956-09-16	M	
3	Merlin	Stonehenge, England	1136-12-03	F	
...	...	...	...	...	
500,000,000	F. Cadillac	Nevada, USA	1964-01-07	M	

Figure 2 - Census Data in Column Families

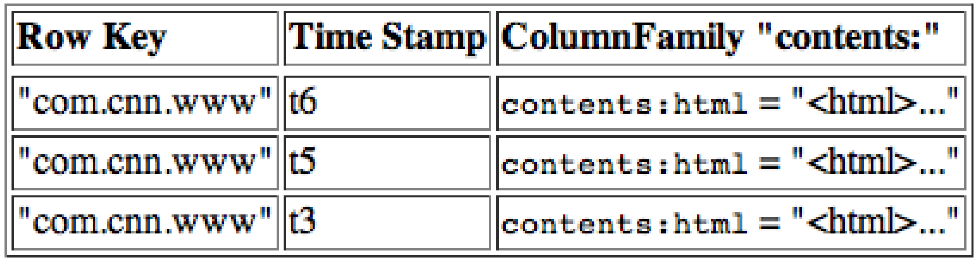
Each column family consists of one or more ***Columns***

HBase Physical Model

- Each column family is stored in a separate file (called ***HTables***)
- Key & Version numbers are replicated with each column family
- Empty cells are not stored

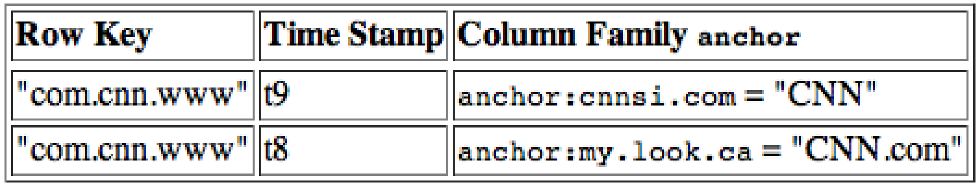
HBase maintains a multi-level index on values:
<key, column family, column name, timestamp>

Table 5.3. ColumnFamily `contents`



Row Key	Time Stamp	ColumnFamily "contents:"
"com.cnn.www"	t6	`contents:html` = "<html>..."
"com.cnn.www"	t5	`contents:html` = "<html>..."
"com.cnn.www"	t3	`contents:html` = "<html>..."

Table 5.2. ColumnFamily `anchor`



Row Key	Time Stamp	Column Family `anchor`
"com.cnn.www"	t9	`anchor:cnnsi.com` = "CNN"
"com.cnn.www"	t8	`anchor:my.look.ca` = "CNN.com"

Column Families

- Different sets of columns may have different properties and access patterns
- Configurable by column family:
  - Compression (none, gzip, LZO)
  - Version retention policies
  - Cache priority
- CFs stored separately on disk: access one without wasting IO on the other.

HBase Regions

- Each HTable (column family) is partitioned horizontally into ***regions***
  - Regions are counterpart to HDFS blocks

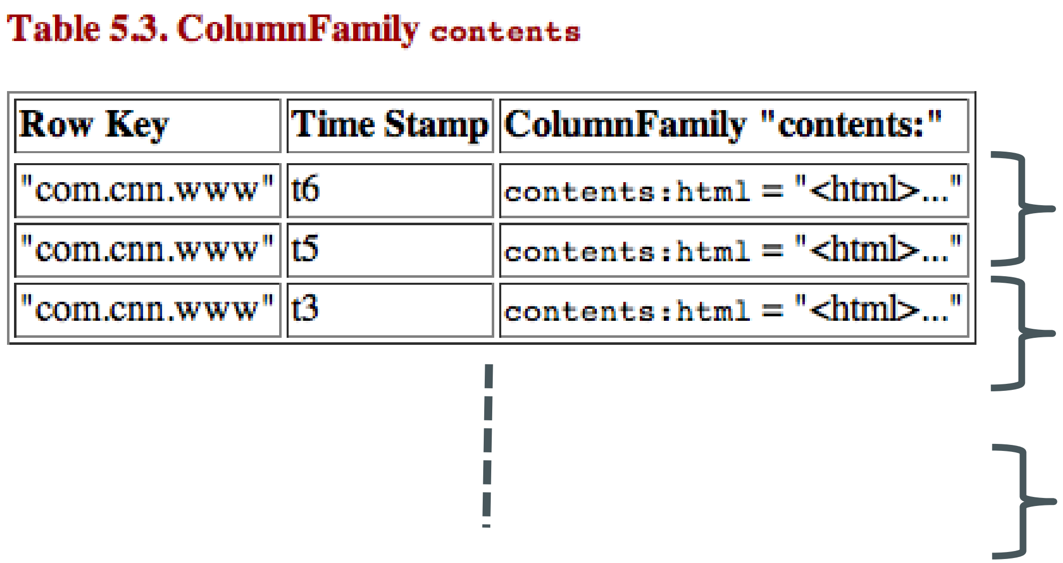


Table 5.3. ColumnFamily `contents`

Row Key	Time Stamp	ColumnFamily "contents:"
"com.cnn.www"	t6	`contents:html` = "<html>..."
"com.cnn.www"	t5	`contents:html` = "<html>..."
"com.cnn.www"	t3	`contents:html` = "<html>..."

Each will be one region

When to use Hbase?

- You need random write, random read, or both (*but not neither*)
- You need to do many thousands of operations per second on multiple TB of data
- Your access patterns are well-known and simple