M / T / W / T / F / S / SU

Video 1 :- DBMS based and Non DBMS Based.
The MinDBMS

Date

Big Data Management : The "m" in DBMS.

→ DBMS-based and non-DBMS-based Approaches to Big Data.

- Explain the various adv. of using a DBMS over a file system.
- Specify the diff. b/n a parallel and distributed file
- MapReduce-style DBMS.

→ already in SQL book.

Parallel and Distributed DBMS :-

- Parallel database system

→ Improve performance through parallel implementation

- offer allows data replication

→ Data redundancy against table corruption

→ More concurrent queries.

- Distributed database system :

→ Data is stored across several sites, each site managed by a DBMS capable of running ~~it~~ indepenHy !

* DBMS and MapReduce-style systems

- started with a different problem focus

→ DBMSs : efficient storage, transactions and retrieval

- Partitioned data parallelism
- Account for computation and communication cost
- Not node failure

MapReduce-style systems : complex data processing over a cluster of machines.

→ HDFS-based.

- Analytics - data Mining, clustering, ml
- Multi-stage, problem-specific algo

Priority

• To climb steep hills requires slow pace at first •

- Operate on wider variety of data including text

M / T / W / T / F / S / SU/ Date

Shifting Requirements

- Data loading - a new bottleneck

→ Does the app need data sooner than the loading time

- Too much functionality

→ Does the application use only a few data management features?

- Combined Transactional and Analytical capabilities.

No single solution :-

- Mixed soln^s

→ DBMS on HDFS

- Hadoop - DBMS interoperation

→ Relational operations in MapReduce systems like Spark

→ Streaming i/p to DBMS

→ New parallel programming models for analytical computation within DBMS.

Priority

M/T/W/T/F/S/SU/

Video 2 Week 5

Date

→ Explain at least 5 desirable characteristics of a Big data management system

→ Explain the diff. b/n ACID and BASE.

→ Describe what the CAP theorem states.

→ List examples of BDMS and describe some of their similarities and differences.

Describe Desired Characteristics of BDMS :-

→ A flexible, semistructured data model
- "schema first" to "schema never"

→ Support for today's common "Big Data data types"
- Textual, temporal, and spatial data values.

→ A full query language
- Expectedly at least the power of SQL.

→ An efficien parallel query runtime

→ Wide range of query sizes

→ Continuous data ingestion
- Stream ingestion

→ Scale gracefully to manage and query large vol^n s of data
- Use large clusters.

→ Full data management capability
- Ease of operational simplicity

* ACID and BASE :-

→ ACID properties hard to maintain in a BDMS

→ BASE relaxes ACID

BA : Basic Availability

S : Soft State

E : Eventual consistency

Priority

M/T/W/T/F/S/SU/ Date

CAP Theorem :- A distributed computer system cannot simultaneously achieve
→ consistency
→ Availability
→ Partition Tolerance.

Redis :- An Enhanced key-value store
→ In-memory data structure store
• strings, hashes, lists, sets, sorted sets.
→ Look-up Problem
• case 1 : (key : string, value : string)
→ keys may have internal structure and expiry
• case 2 : (key : string, value : list)
→ Ziplists compress lists.
→ Twitter innovation : list of ziplists.
• case 3 : (key : string, value : attribute - value pairs)
→ REDIS Hashes.

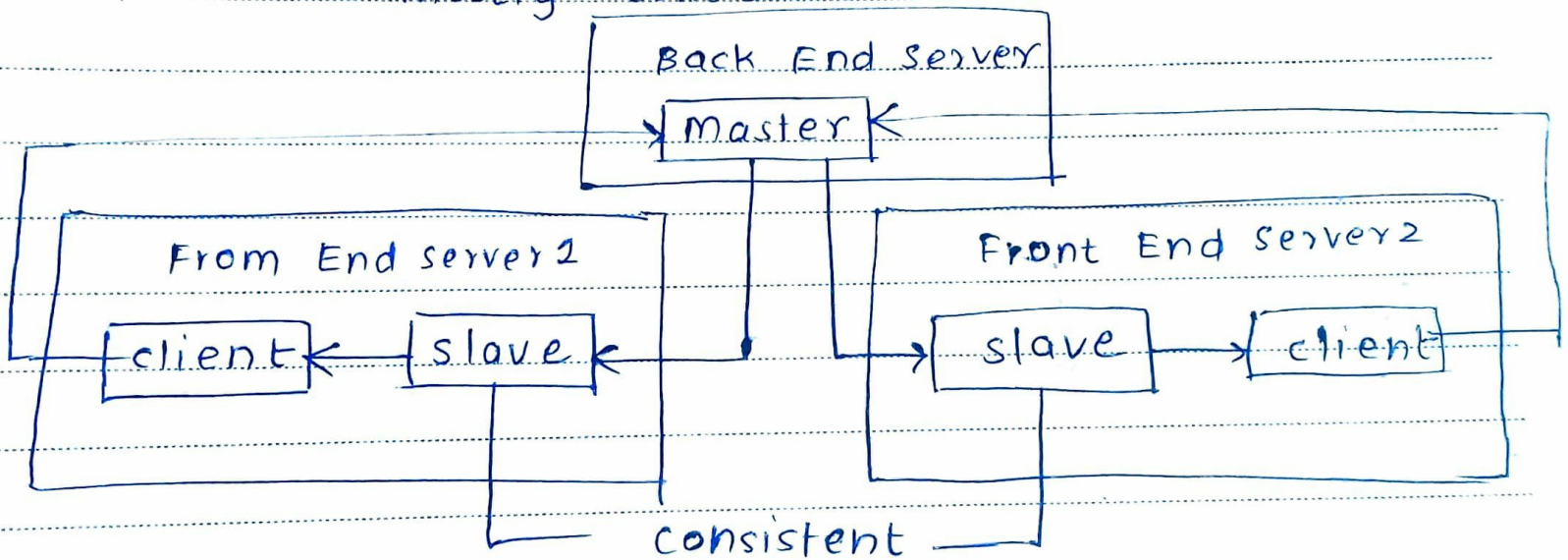
# Redis and Scalability
• Partitioning and Replication
① Range Partitioning :-
Example :- User record number 1-10000 get to machine 1
— " — 10001 - 20000 get to machine 2.
② Hash partitioning
→ pick a key of a record eg. "abcde"
→ Using a hash function, turn it into a no.
→ 152 mod 10 is 2, so the record goes to machine 2.

Priority

- Partitioning and Replication:

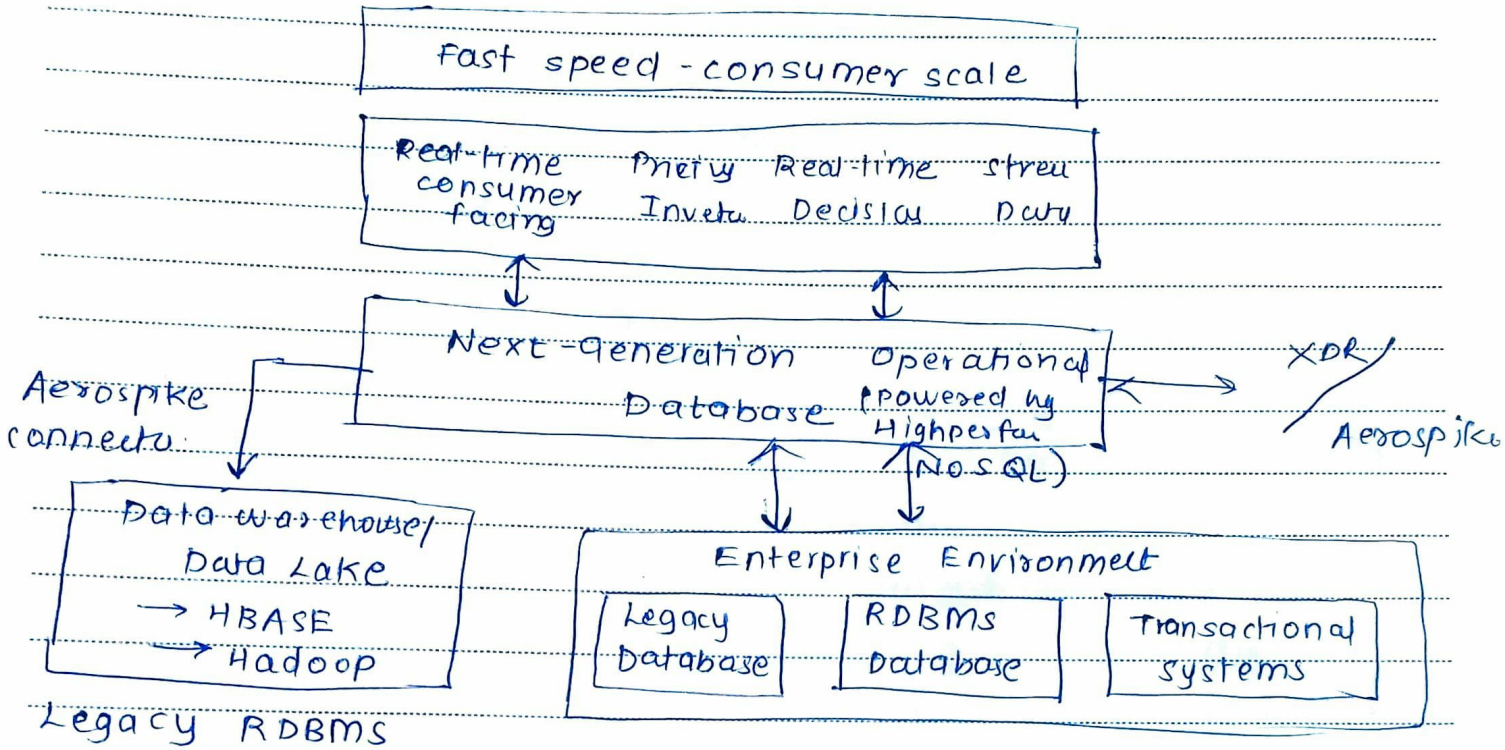
→ Master-slave mode replication

- client write to master, mast replicates to slaves.
- Clients read from slaves to scale up read performance.
- slaves are mostly consistent.



M/T/W/T/F/S/SU/ Date

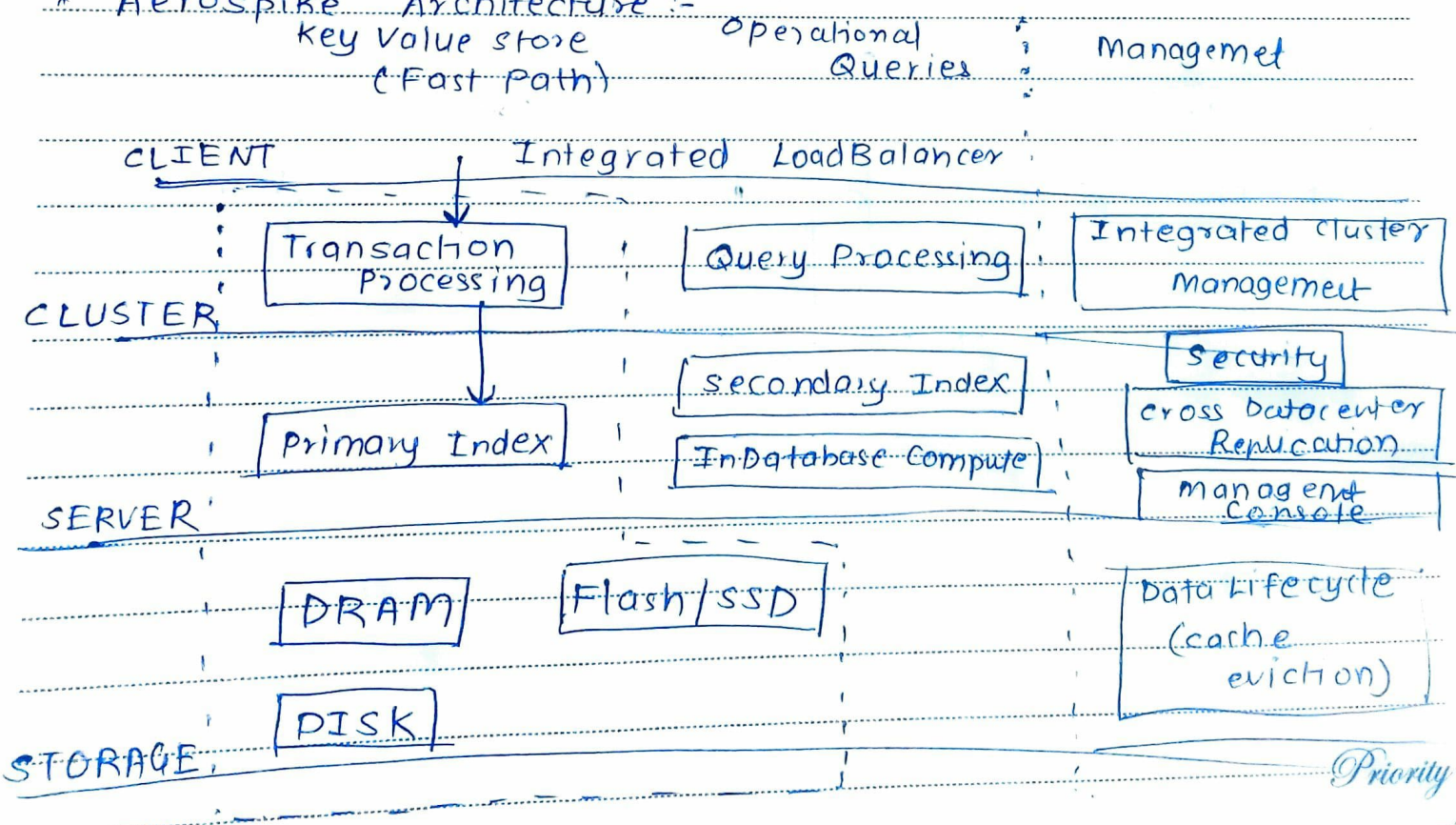
Aerospike :- a New Generation KV Store.




Legacy RDBMS

HDFS Based

* Aerospike Architecture :-



Querying Aerospike :-

9.00 → Data types :- standard scalar, lists, maps, geospatial, large objects.

10.00 → KV store operations :- Geospatial queries like point-in-polygon.

11.00 → AQL : an SQL-like language.

```
SELECT name, age
12.00 FROM users.profiles.
```

Transactions in Aerospike :-

1.00 Aerospike ensures ACID

- consistency :- all copies of a data item are in sync
2.00 → uses synchronous write to replicas
→ Mechanisms to relax immediate consistency

3.00 Durability :- → Flash storage
→ Replication management.

4.00 Network partitioning reduced
→ Tighter cluster control

5.00

AsterixDB :- a DBMS for semistructured data

#6.00 Options for Querying in AsterixDB

→ AQL is a natively-supported query language.

```
7.00  for $user in dataset TwitterUsers
      order by $user.followers_count desc,
          $user.lang asc return $user.
```

NOTES Hive queries

```
SELECT a.val, b.val
FROM a LEFT OUTER JOIN b ON (a.key = b.key)
```

→ Xquery
→ Hadoop MR jobs
→ SQL ++ (coming up)

* Operating over a cluster.

9.00 • Hyracks :- Query execution engine for partitioned parallel execution of queries.

10.00 Ex :- CUSTOMER (C_CUSTKEY, C-MKTSEGMENT, ...)
ORDERS (O_ORDERKEY, O-CUSTKEY, ...)

11.00

Accessing External Data :-

12.00 → Real-time data from files in a directory path.
→ Real-time data from an external API.

1.00

Solr :- Managing Text

2.00 → Basic challenges with text
• Defining a match

3.00 Analyze = Analyse
Dr = Doctor.

4.00 Inverted Index
→ Vocabulary :- All terms in a collection of documents
→ multi-word terms, synonym sets.

5.00 → Occurence :- For each term in the collection
• List of docID
• List of docID, [position of occurence]

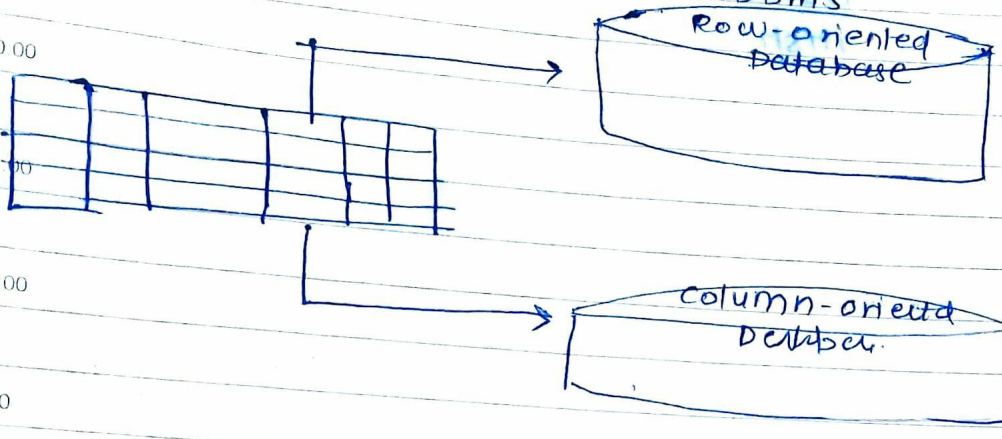
6.00

7.00 * Solr Functionality :-
→ Enterprise search platform
→ Inverted index
• For every field in a structured text documed.
→ Indexes text, numbers.
→ Faceted search
→ Term highlighting.

NOTES

Solr Functionality :- • Index-time Analyzers – Tokenizers – Filter.

# Vertica :- a columnar DBMS




- column store
  → store data column-wise
  → A query only uses the columns needed.
  → Usually must faster for queries even for large data.

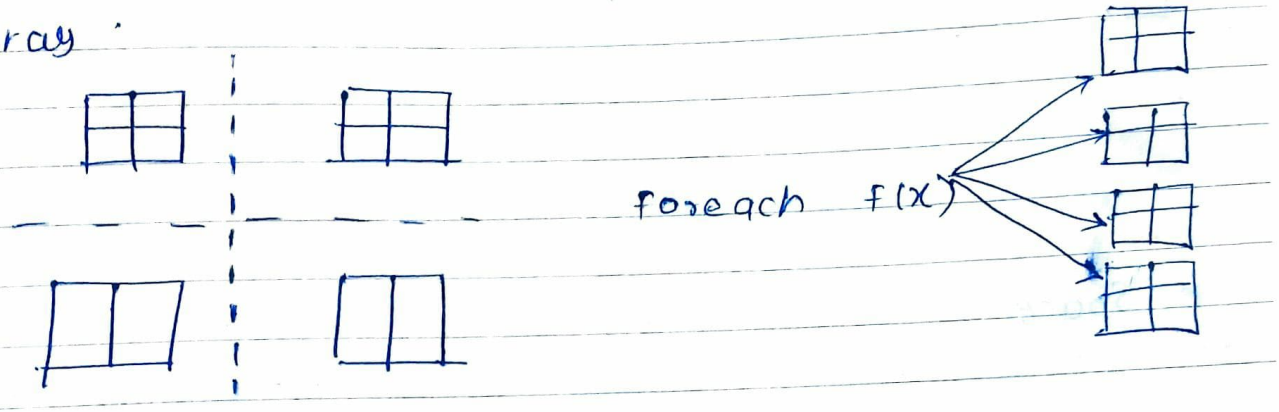
* Space Efficiency
  → Column stores keep column in sorted order.
  → Values in columns can be compressed.
    - Run-length encoding
    - Frame-of-references encoding
  → Compression saves storage space.

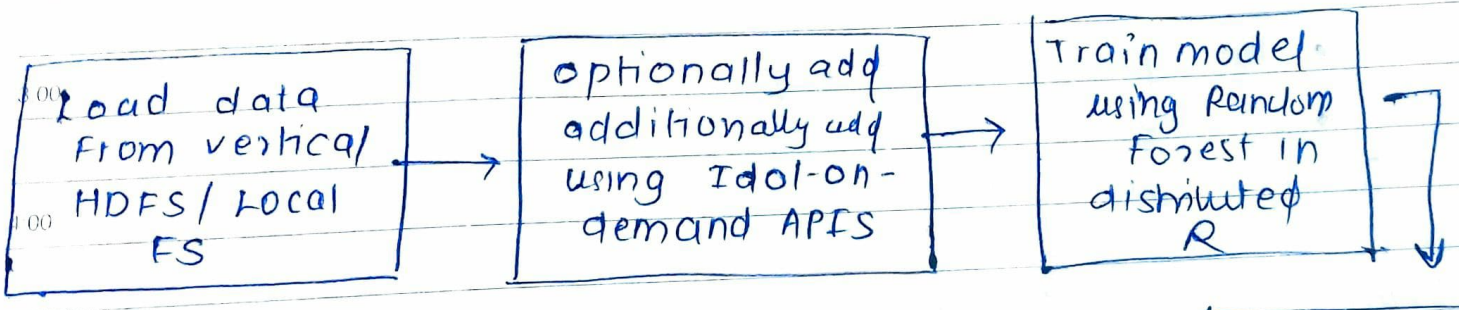
Working with Vertica :-

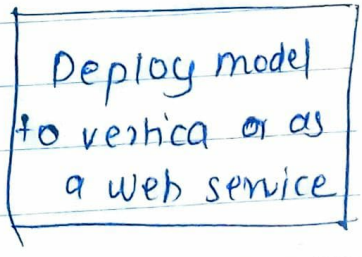
- Column-Groups
  → Frequently co-accessed columns behave as mini-row-stores within the column store.
- Update performance slower.
  → Internal conversion from row-representation to column-representation.
- Enhanced suite of analytical operations
  → Windows statistics.

## Vertica and Distributed R :-

- High-performance statistical analysis.
- master node : schedules tasks and sends code
- Worker node : maintain data partitions and compute.
- Uses a data structure called dArray or distributed array.




NOTES