Date: DBMS Day: M T W T F S

Ch: 7

~~Data Modelling~~

XML & JSON

used for data interchange

⇒ Extensible Markup language

⇒ Tag : label for a section e.g <header>

⇒ Element : Data Section beg. <tag> end </tag>

⇒ Attributes: Represented by "name = value" inside opening tag of an ~~an~~ element

Attribute vs Subelements

⇒ Subelement : <tag1> <tag2> ... </tag2> </tag1>

⇒ Attr. are part of ~~document~~ markup

Duplication

⇒ Subelement are part of basic document

⇒ Suggestion : Attr → id of elements
sub-ele → contents

⇒ String that used as tags
<![CDATA[<account> ... </account>]]>
Here <account> and </account> are treated as strings

⇒ Document Schema
⇒ DTD (Document Type Definition)
⇒ XML Schema (New)

Date: ___________ Day: M T W T F S

→ **JSON**

⇒ String, number, object, array, ~~true, f~~ bool, null

⇒ Can be used by JSON function.

Ch: 11

NO SQL

Why → Joins are expensive

⇒ Hard to scale

⇒ Impedance mismatch

⇒ Expensive product cost

⇒ Speed

⇒ Partition

⇒ Availability.

⇒ **Characteristics**

Avoids
- ⇒ Avoids overhead Acid transactions
- ⇒ " Complexity of SQL
- ⇒ " Schema design
- ⇒ " DBA presence

Provides
- ⇒ Provides easy and freq. changes to DB
- ⇒ Fast dev
- ⇒ Large data (Big data e.g. Google)
- ⇒ Schemaless

Scanned with CamScanner

Date:
Day: M T W T F S

When/Why?

⇒ RDBMS is restrictive (Flexible schema) needed
⇒ ACID not "really" needed
⇒ Logging data from distributed systems.
⇒ Temporary Data (Carts, sessions, favorites)
⇒ Polyglot Persistance:
Best Data Store depending on data

Schema less

⇒ In RDB
⇒ Can't add record with different datatypes, less attr., or multiple ~~fields~~ data.
⇒ Should consider Primary key, Joins etc
⇒ In No-SQL
⇒ No Schema, no-unneeded cell, no-datatype

Aggr. Data Models

⇒ Key-value (redis) ⇒ Document (mongoDB)
⇒ Col. family (Apache) ⇒ Graph (Neo4j)

⇒ Key-Value

⇒ Easiest, access data using keys (string, hashed)
⇒ Operations: insert, fetch, update, delete

⇒ Col Family

Cassandra ⇒ Facebook Search

Date: ______

Day: M T W T F S

⇒ MySQL > 50GB

⇒ Write : ~300ms, Read: ~350ms

⇒ Cassandra > 50GB

⇒ write : 0.12 ms, read : 15ms

⇒ **Document**

⇒ Pair key with complex data structure

⇒ Indexes with B-trees

⇒ Nested documents

SQL vs NO-SQL (Diff)

⇒ mySQL vs mongo

⇒ ROW & tables vs key-value, JSON, XML

⇒ Schemas : Static vs Dynamic

⇒ scaling : vertical, horizontal vs horizontal

⇒ Data Manipulation : Queries vs OO API's

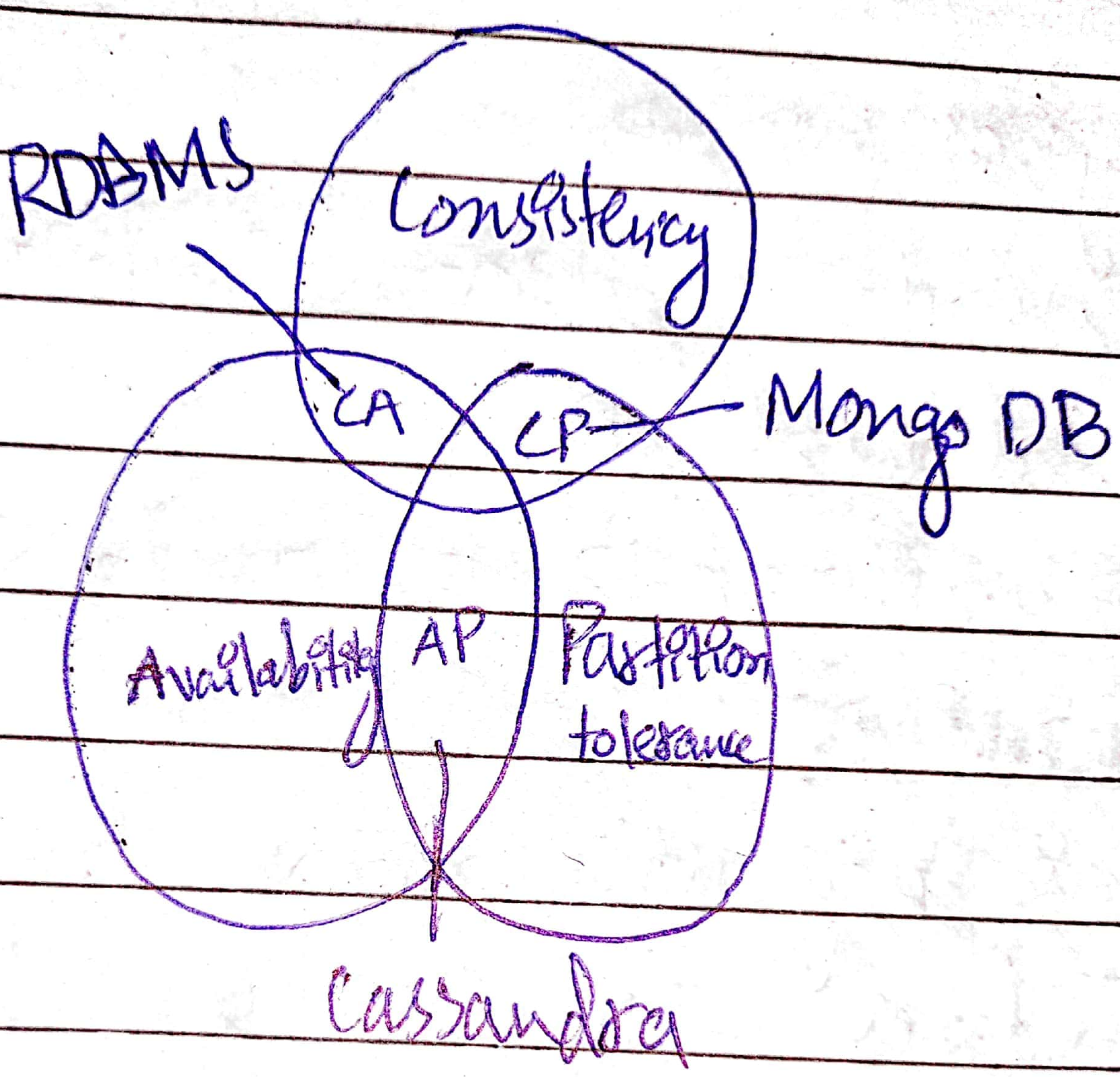
CAP theorem :

⇒ Scalability, Consistency, availability, fault tolerance impossible at a time.

⇒ Impossible for any shared data-system to guarantee simultaneously all three

→ consistency → Availability → Partition tolerance

Scanned with CamScanner



NOSQL : AP

RDBMS : CP

[illegible]