C

# Comp. Sci. Fundamentals

# DBMS - Introduction

Tanmay Kacker

CODE

DATA

DATABASES

-> what?

-> why?

→ Types of DB

- Normalisation
- Index
- SQL
- ACID

} NoSQL - HLD

- ER

---

- MySQL
- Postgres
- MongoDB

} DBMS

- Redis
- Oracle
- Cassandra

## Data

- Information

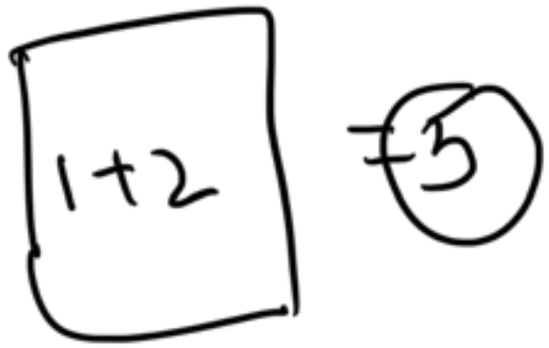
- By How many wickets did India lose to England?

- Temperature in Bangalore

Why?

→ user requirement

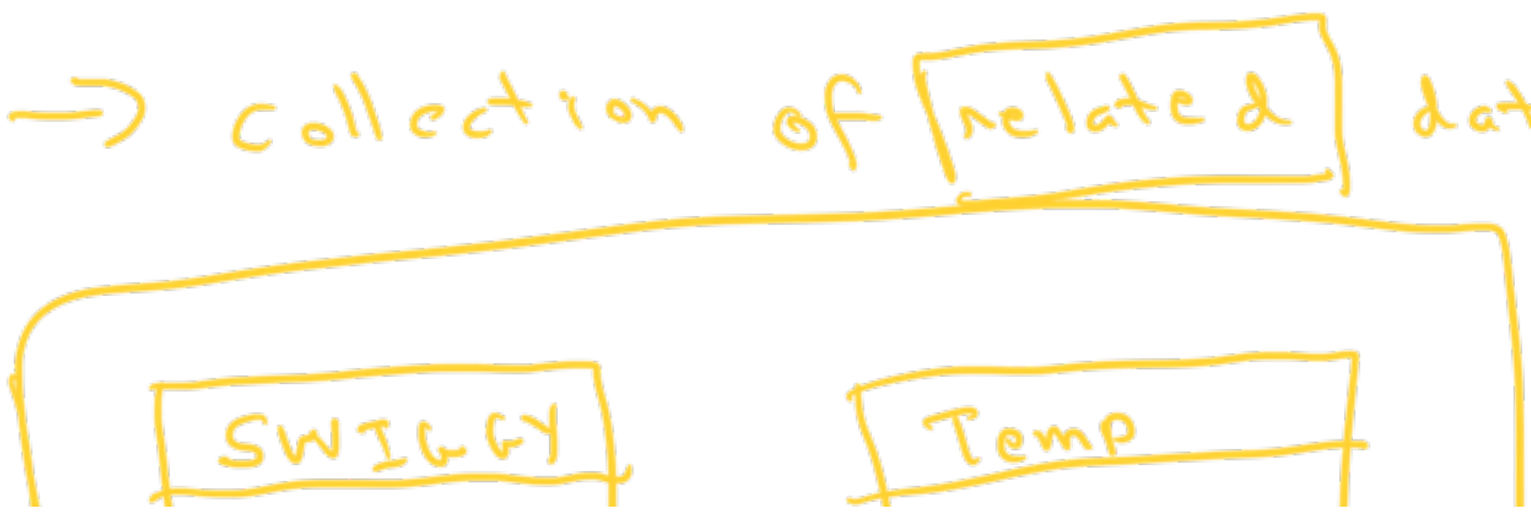
→ analyse




Persistence

Database

→ collection of related data



DBMS

- manage databases

---

Scalen

- Student

- mentors
- batches

Excel sheets

- Files
- CSV files .txt
  comma-separated values

Students, mentors

- name
- email
- phone
- address

- age

Batch

- name
- start date
- type
- mentor

Search for a value

- Iterate over all the rows

O(N)

- Each time we have to get a

value the file has to be read

Google - 15 exabyte - 2015

billion gig.

O(n)

- Concurrency

- Security

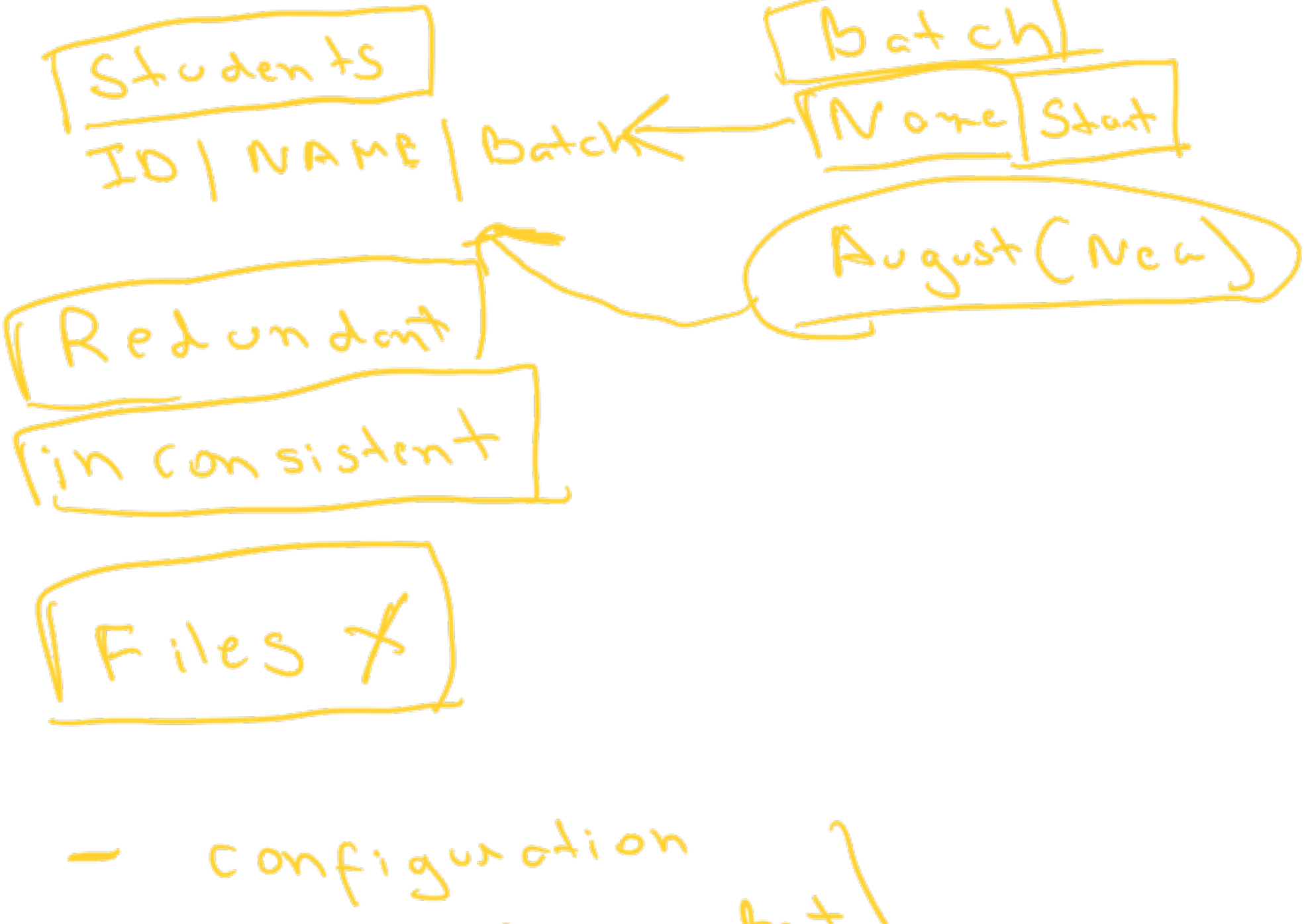


password

TAs

SID | Name |

→ Data integrity




- configuration

Spring

- logs
- .txt

- Static data
  - no Frequent change
  - small amount CSV
  - simple - read, write

# Sequential access

vs

random

access


O(n)

O(1)

(1) Persisted

(2) Random vs sq. access

Set

Arrays

Linked List

---

DBMS

Scalen -

- Concurrency
- Security ✓
  - Encrypted
  - Compressed
- Integrity
- efficient ✓

- RDMS — read-intensive

- Views




- Fault tolerance
  data lost
  DBMS

Integrity

batch | name | type | start | mentor name

Sheldon

Age | phone

mentors

Shame

Concurency - LLD

10 Pm Tony →



---

DRY

Types of DBMS

- relational
- non-relational (NoSQL)

↑

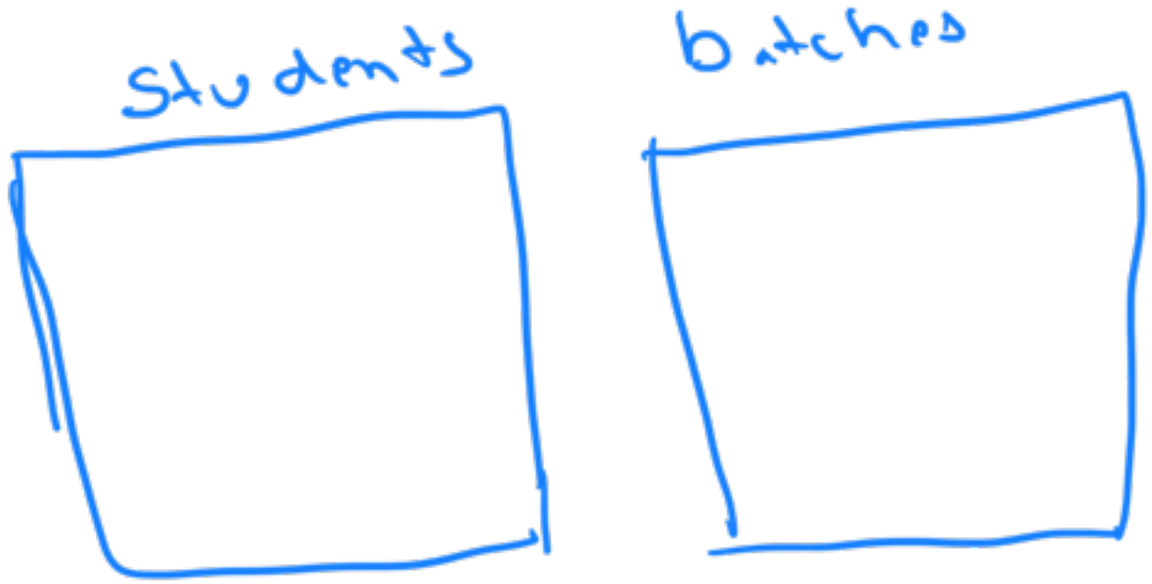
Relational

AWS - Aurora

Relations

- students

→ mentors




↓

Table

RDBMS

- MySql
- Postgres

- Oracle J
- SQLite

Non-relational

- Graph
  - ↳ graphs
  - → social media
  - → Neo4J
- Key Value

Map {

"1" : [illegible]

"2" : "John"

}

- Redis
- Document-based
  Mongo

{ "1" : { name : "Tantia"
  age : " "
  adres : " " } } metadata

Graphql

Postgre

→ 1, Tantia
→ 2,

T | ID
1, 2, 3
Tantia, John

- Columnar

- Cassandra
- Time scaled - Influx,

10:00:00 —
10:00:10 —

---

6:04 - 6:10

- 10:40

RDBMS

↳ relational

Relational

— row by row

Col - col by col

| 1 | Sherlock | ... |
|---|---|---|
| 2 | John | ... |

↓ T

| 1 | 2 | ... |
|---|---|---|
| Sherlock | John | .... |

Data store

Db

transient

persistent

OS

Data store

Not only Sql

No to Sql

---

Neo4J

Db

C C++

Go, Erlang


Lock

Semaphores

DB + OS + DSA

+ LLD

---

Relational Model

- 1960s
- Edgar Codd
- Turing Award

Relational Algebra

First order logic

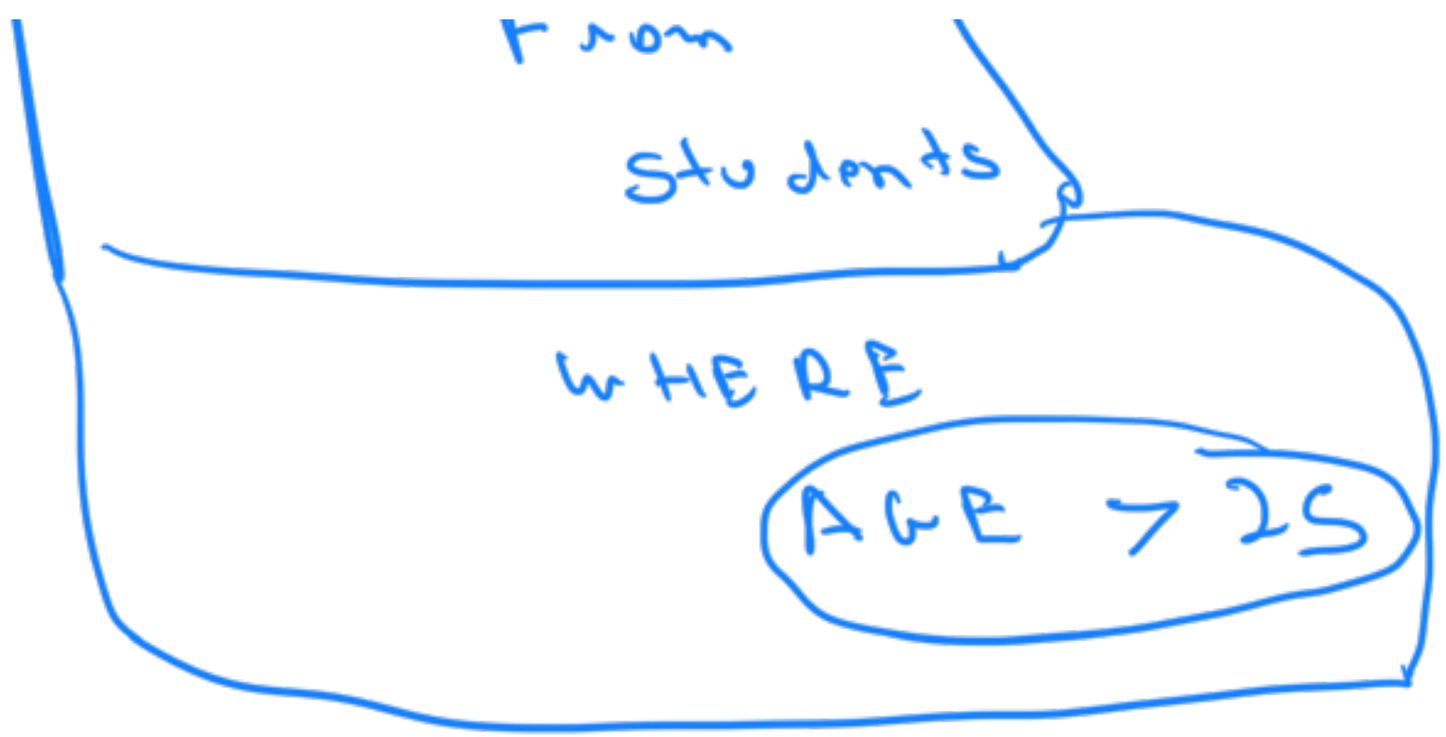
Sets

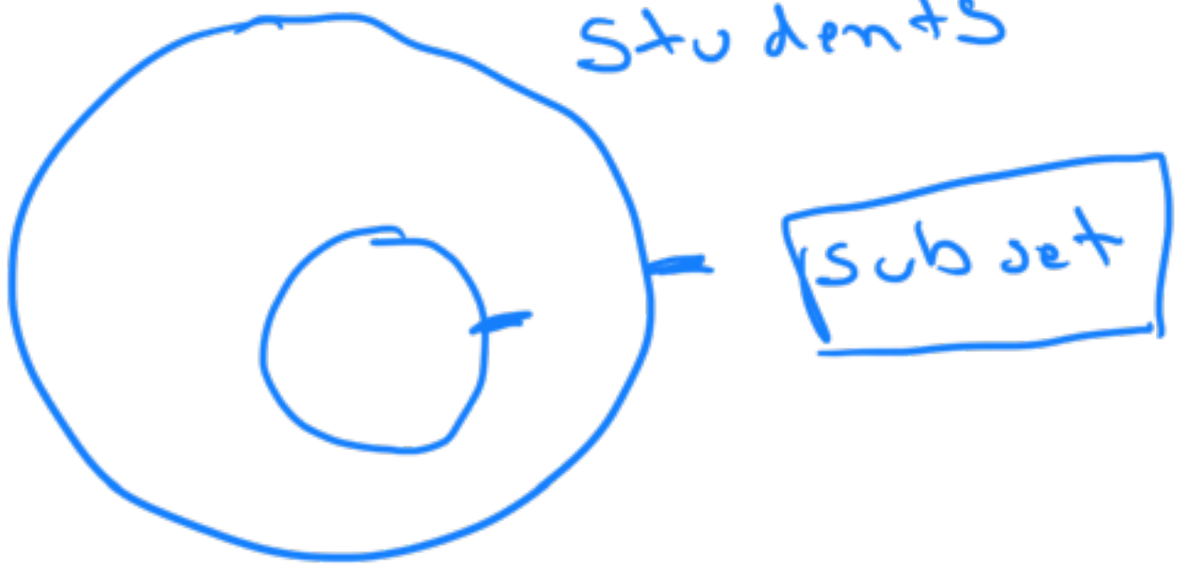
- union
- intersection
- subset
-

Set → unique

Select *

DBMS - language




Select * FROM STUDENTS

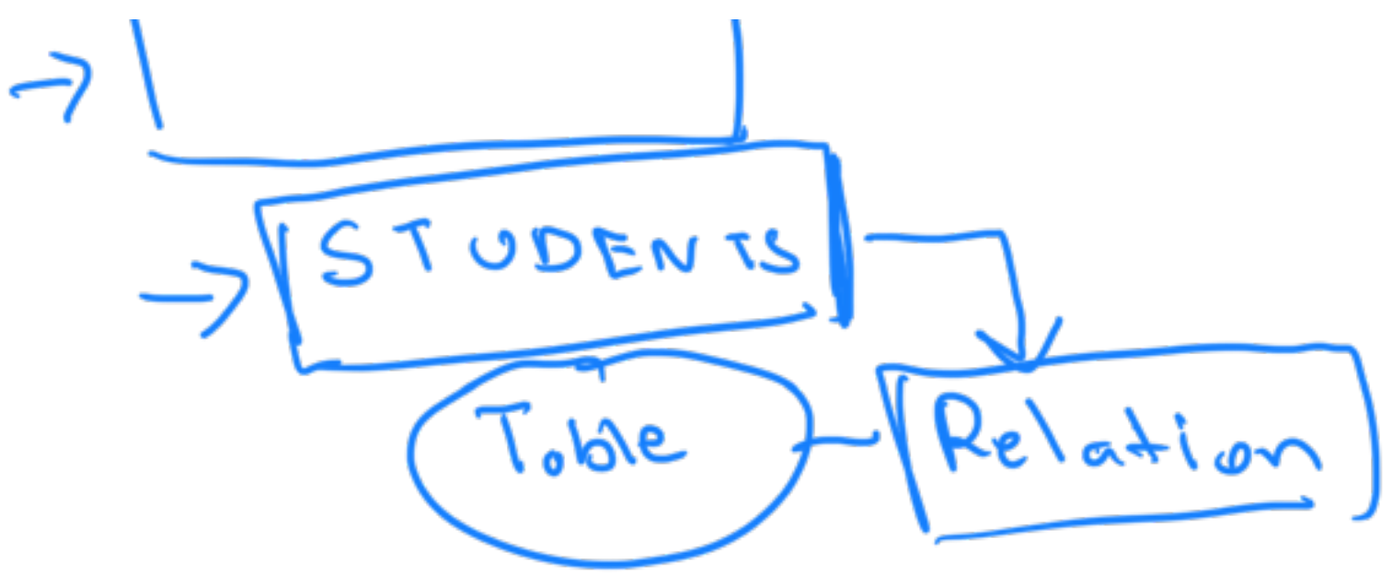
WHERE

AGE > 25 & AGE < 30


Intersection

# Relational Model

(1) Set




## Relational

* Relation - Table
* Attributes - Columns
* Tuples - Rows

## Degree

degree (set)

$\{A, B, C\}$

- no of columns

STUDEN - | ID | NAME | PHONE |

→ 1

→ 2

→ 10

{Students}$^0$ = 3

## Cardinality

→ no. of rows

→ 10

Address

1 - Sherlock - 221 B

2 - Tantia = NULL

---

CREATE

---

Properties

① Unique - Set

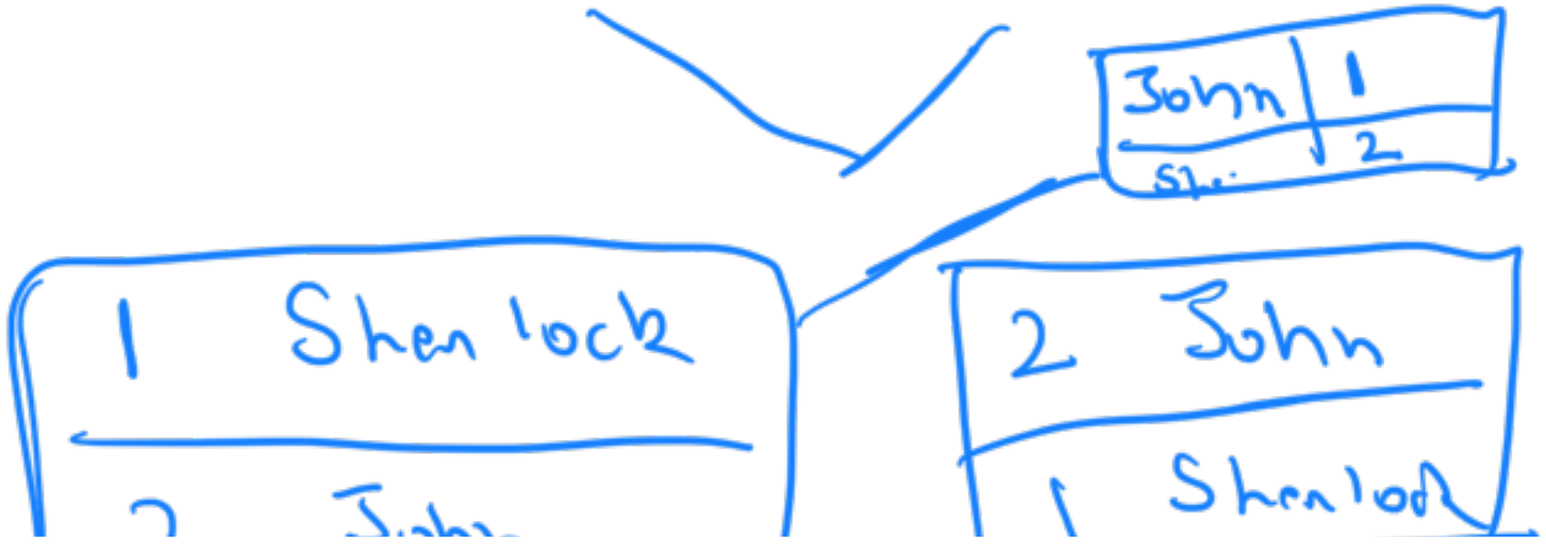
- tuples should be unique
- attributes must be unique

STUDENT

| ID | Name |
|---|---|
| 1 | Sherlock |
| 2 | Sherlock |

(2) unordered

{1, 2}  {2, 1}



tuples - ordered

≠

attr.

Set → Relational

→ unique

- unordered

{1, 2}  {2}

Uniform data type

| | | Age |
|---|---|---|
| 1 | Sherlock | 21 |
| 2 | Tantra | Twenty |

Across all

Tuple , the same type

should be used for an

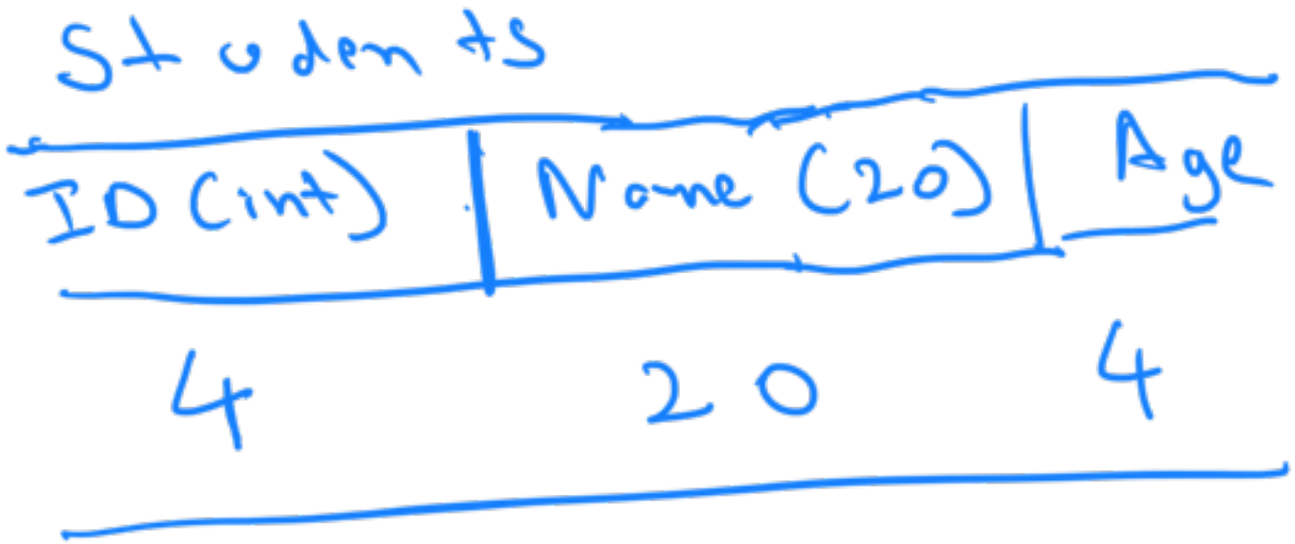
attribute

→ **Atomicity**

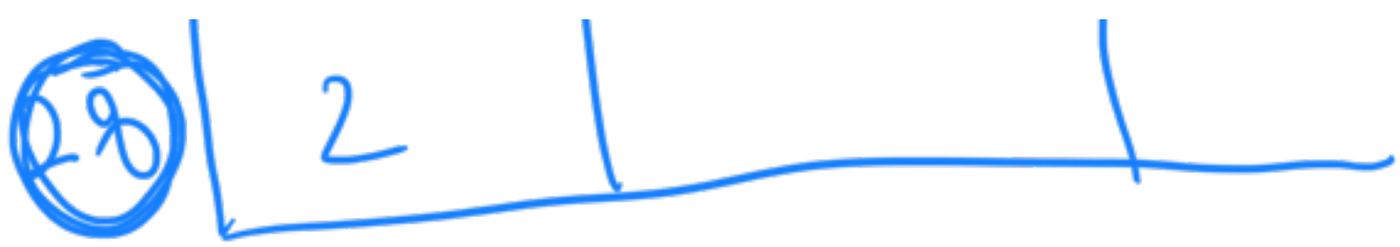
- [1, 2, 3]

- Sets

- maps




28





10th no w

28 * offset

2 20


→ length * type

10 * int = 40

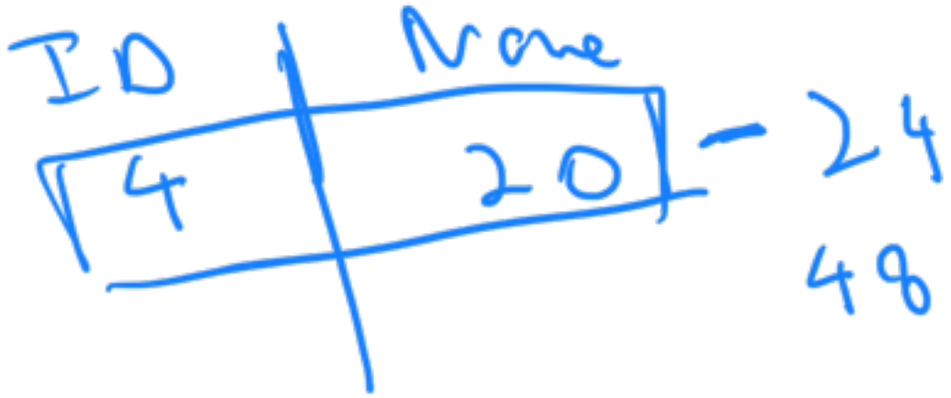
Transaction

Atomicity

Collection


Properties of RDBMS

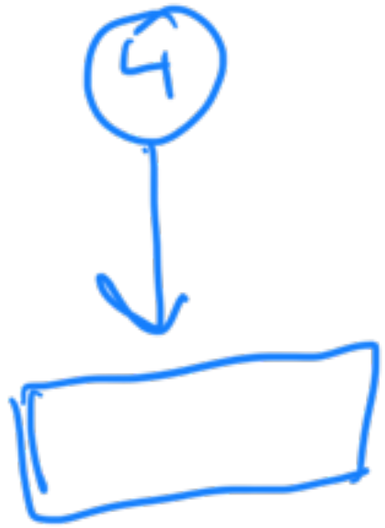
- Unique
- Unordered
- All values in an Attribute must be of the same type
- Atomicity




int + None + list

4 20

Key




Key

→ Uniqueness

hash

| Id | Student | Age | email | Phone |
|---|---|---|---|---|
| 1 | Sherlock | 21 | @.com | |
| 2 | John | 20 | 3@.com | |

Sherlock

→ email, phone
→ roll number ID

Student

| ID | NAME | Batch Name |
|---|---|---|

Batch ID

→ describe relationship between

Relations

---

**Super Key** - any set of attributes that make a tuple unique

**Candidate Key**

Student

| ID | NAME | PHONE | AGE |
|---|---|---|---|

{NAME + PHONE + AGE}

SK

{ NAME , PHONE }

SK

{ Age }

{ All the attributes }

{ ID, Phone , Email }

**Super Keys**

Candidate

- minimal

→ C.K

SK { ID, ~~NAME~~, EMAIL }

SK' { ID, ~~EMAIL~~ } ⊂ CK

{ ID } ⊂ CK

{ phone }

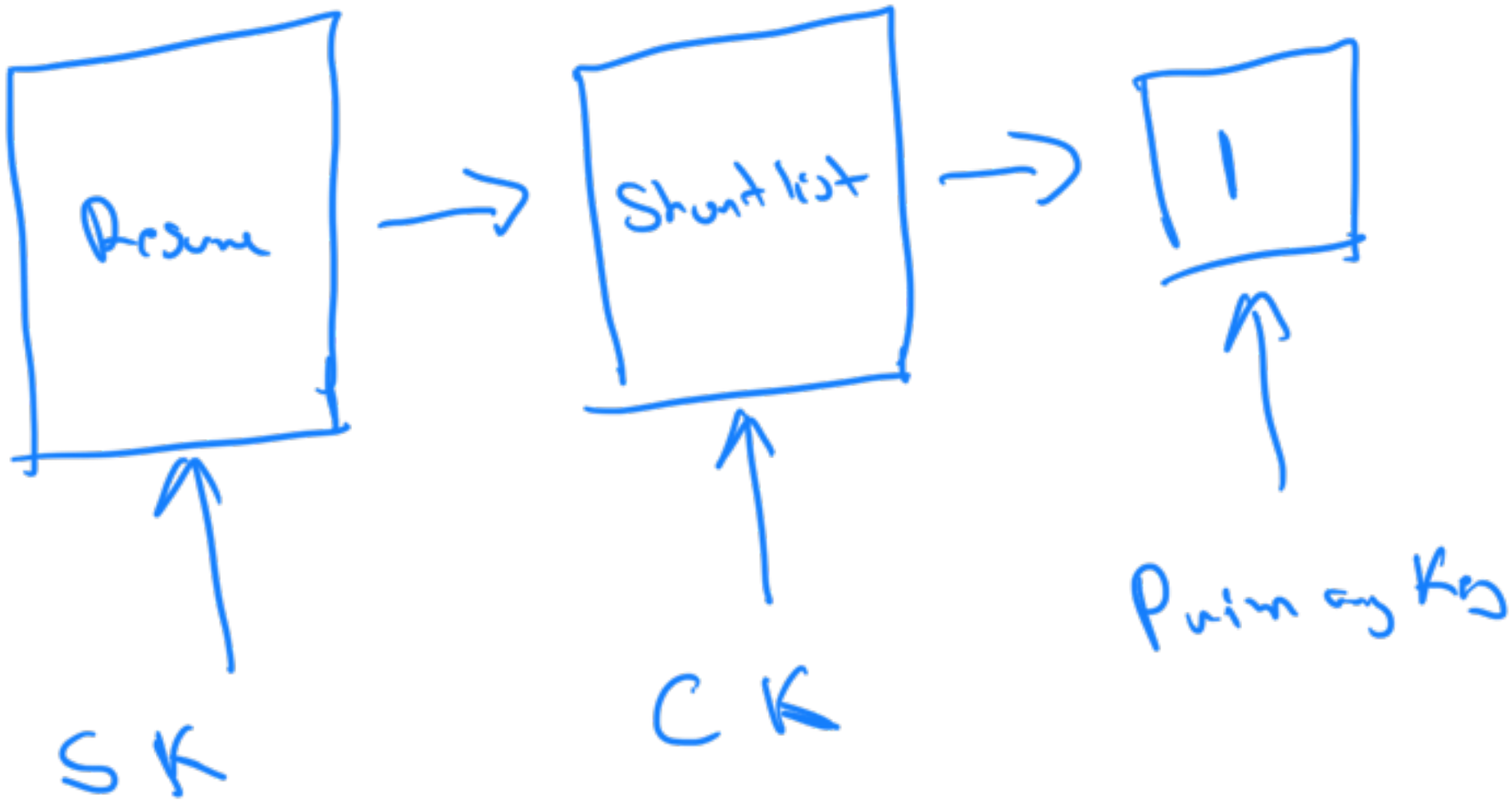
SK ✓

CK ✗

{ ~~age~~ email } SK ✓

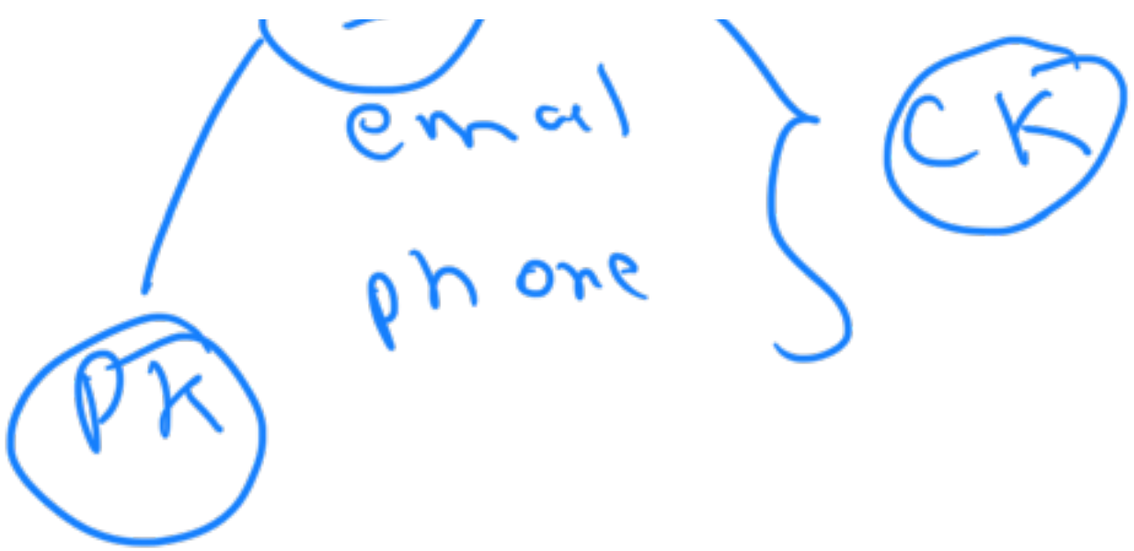
CK ✓

Primary Key




one of the Candidate Keys

ID ∧ SK




Composite


Composite

Prim ary

Composte

Keys

- SK - 2 all the sets
- CK - 2 minimal sets
- PK - 21y

relation ship

Batch  Mentor ID

Student  Batch ID