DBMS III - Normalisation & CRUD

Agenda

→ Integer datatypes

→ Normalisation
- Anomalies
- Normal forms
- How to fix your schema.

→ SQL
→ CRUD
→ SQL queries.

String

↳ char → fixed

↳ varchar → variable

↳ TEXT → when varchar does not suffice

→ indexing is not straightforward

Numeric datatype

→ integer

→ floating points

Integer

→ int → 4 bytes

→ TINYINT 1B → flags

→ TINYINT

→ SMALLINT 2B → Ase

→ MEDIUMINT 3B

→ INT 4B

→ BIGINT 8B → iAS

age — [100] bytes

flags — did a join a session

{

Lid: int — limited

10^9

}

testing ✓ — [ ] dev

id → 10^9+1

ERROR

int to bigint

↳ Alter table

modify ( )

---

why not use string as id

id → searching

String

int

---

floating points

10/3 → 3.333333

3

3.3

3.33

Decimal (S, P) → precision

length

10.

3.3333

decimal (2,1) ⇒ 3.3

(3,2) 3.33

8 - 65 bytes IEEE 754

→ float - 3.3333

↓

approximation

decimal (length , precision)

↓ how many digits

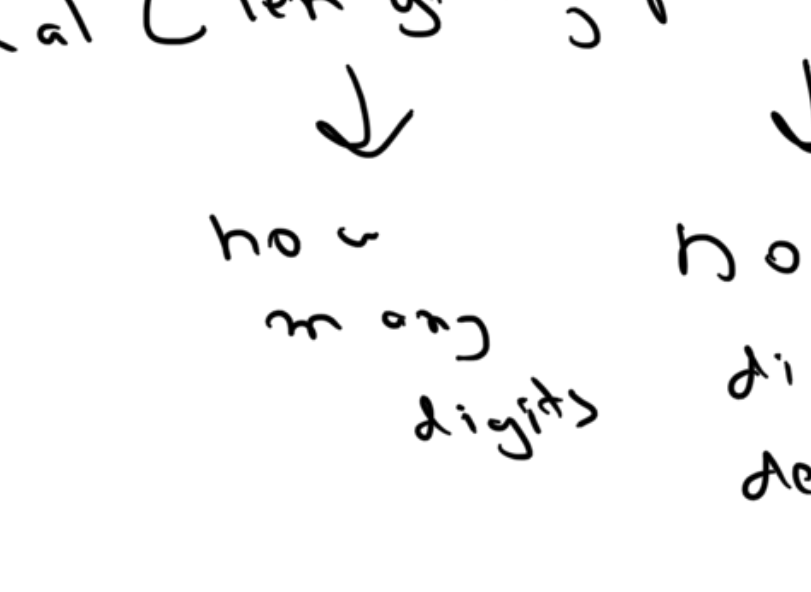
↓ how many digits after decimal

---

float

→ no way to decide how many digits you should have

Approximation



if (result - value > 1)

d oth is

2 - 1.0716

0.0001

0.00004

INTEGER

~~SHORT~~ INTINT 0/1

BIGINT

Floating point

length x Precision

0.948

1.052

2 - 1.0716

Points

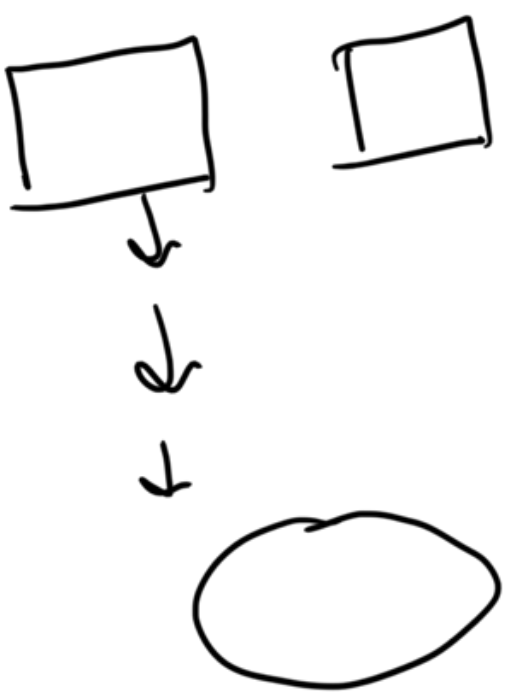
→ comparisons are tricky

Position → small value,

→ decimal

→ float — approximation

Normalisation



Structured approach

intuitive

Normalisation

→ reducing data duplication

Normal forms — 1NF, 2NF

Why is normalisation required?

→ memory issues ✗

→ consistency

Anomalies → problem → issue ~~Batch~~

1	A	email	1	B1
2	B	email2	1	B1

Insertion anomalies

ID	NAME	BATCHID	BNAME
1	NULL	1	A

We cannot insert a new row | Batch because this tied to student

Deletion

ID	NAME	BID	BNAME
1	A	1	B1
2	B	1	B1
3	C	1	B1

1, 2

Deletion anomaly

→ Bcoz of tight coupling ,
when we delete all entities of a certain type , we lose information about the other

Updation Anomaly

→ manual errors

Batch 1

ID	NAME	BID
1	A	1
2	B	1
3	C	1

BNAME

Batch

BAtch 1

b1

b2

```
UPDATE
    Set NAME = 'batch 1'
WHERE
    NAME = 'B1'
```

id = 1 or id = 2

Student

id

NAME	ID	BID
	1	1
	2	1

Batch

1 B1

Update B1 ⇒ Batch 1

Anomalies

① Insertion ← no way to have a batch without student

② Deletion – when I delete a row I also delete information about both.

③ Updation – manual error.

– if I miss updating a certain row, then I have two versions of information

SID	BID
1	1
2	1
3	1

BNA

~~B1~~ Batch 1

~~B1~~ Batch 1

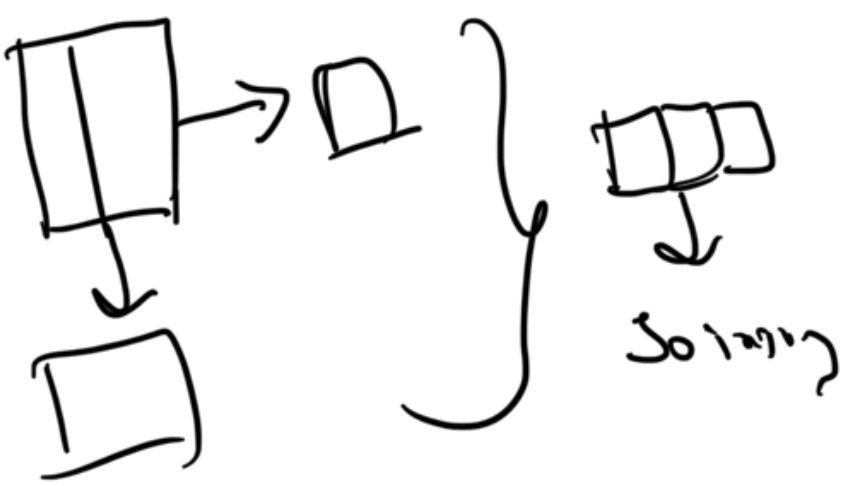
B1

→ Normalisation → Duplication → new table

Normalisation

↳ less or no duplicate information

Con ↳



RDBMS → normalisation

No SQL → Denormalization

user

batch

HLD → redundant info

→ anomalies

→ Mongo → denorm

→ redundant

It doesn't → CAP

Consistency

eventually consistent

Normalisation → Anti tech.

1 NF — loose 1NF
2 NF 2NF
3 NF 3NF

↓ terms of strictness

BCNF — 3.5
4 NF
5 NF — strictest

→ English tech.

1NF - Relational
↓ automatically
1NF

Rule → atomic data type.
↓
Single valued
∄ not collection
[] {}

Student

id	email	[phone number]
1	A	1
2	B	2
3	C	3

Student → 1 phone num.
→ 1 email

id	email	phone_numbr.
→ 1	A	[+91 +90]

1NF

↓
phone_number2

id	name	phone_numbe	
1	A	+91 --	+90 --

Sparse tables → 5 columns

→ Sparse table

→ wasting space

→ Search

→ index → location time increases

+91- ---

MC

Multiple rows

Anomalies

ID (PK)	NAME	PHONE_NUMBER
1	Tandra	+91- —
1	Tandra	+90- —

Composite key

└ (id , phone number)

1 NF

- PK - (id , phone)

→ anomalies

- └ I
- → D
- → U

Create a new table

Student

id	name	email

Phone_numbers

id (Student_id , phone)
1 +91
1 +90
2 +93
2 +94

1 NF

→ loose - loosest (most)

→ Only atomic datatypes

→ no collections

→ Multiple rows

→ Sparse tables.

→ index , searching

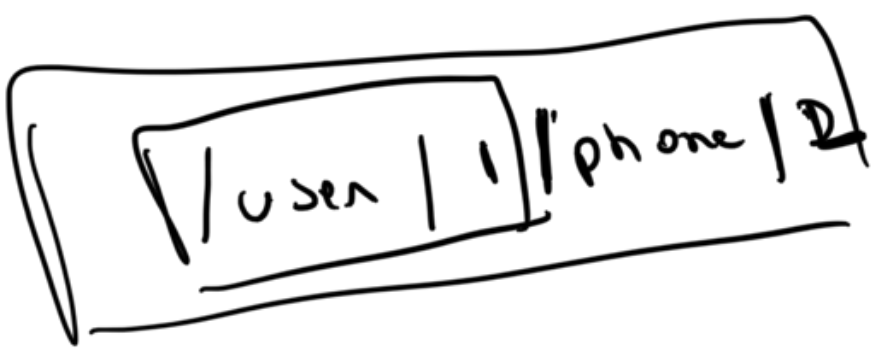
→ Multiple rows

→ anomalies.

→ ...ing table.

→ Separate

user id	Phone number	
1	1	+91
2	1	+90



API design

6:14 6:16

10:46

---

2 NF

① 1 NF → Collection

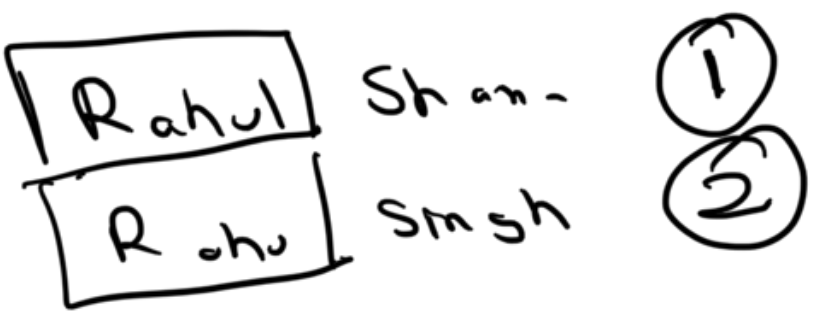
②

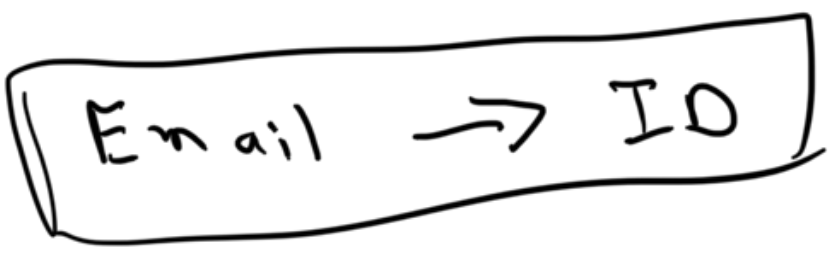
ID	NAME	BATCHID	Batch name	PSP
1	Tantia	1	A	100

Functional dependencies

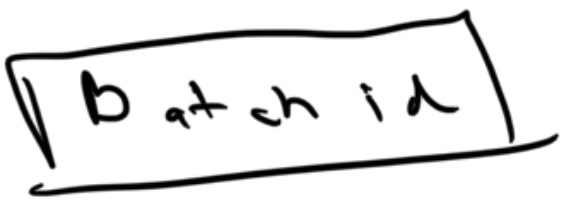
ID →

NAME → ID





Batch id → Batchname



---

2 NF

- 1 NF
- → no partial dependencies
  - → if an attribute is not a part of the PK, it should be derived by the whole PK and not a part of it.

ID | Name | batchID | Batchname | PSP

ID

batch name

ID → batchID → batch name

Mentor ID | Session ID | student ID | Student_name

(Mentor ID | SID | student ID)

→ Student name

Student id → Student name

non PK

attribute

Partial dependency

① INF

② not none partial dependencies

---

class rating

(Class Id | student Id | rating | batchId

1	1	5	1
2	1	4	1
2	2	3	1
1	2	0	1

(CI, SI) → grade

PK → [CI, SI]

[CI, SI] → grade

SI → batch_id

partial dependency

X 2 NF compliant

2NF

PK1 → Non-PK1 — separate

X DD

mentor_id	session_id	student_id	rating

Student id → student_name

XPD

---

2 NF

→ Functional dependency

→ A → B

id → name

mentor_id → mentor_name

→ 1 NF

(A, B) → C

B → D

dependencies on PK

→ By creating a new table

→ PD

student id → student name.

---

2 NF

3 NF → 1 NF

→ 2 NF

→ transitive dependencies

$a \rightarrow b \quad b \rightarrow c$

$a \rightarrow c$

$a = b \quad b = c$

$a = c$

ID	Batch_id	Name	Batch_name

ID → BID

→ NAME

→ Batch Name.

BID → BNAME

ID → BID → BNAME

Transitive dependency

---

* 3NF

- SQL
- CRUD

Stand up | MNC <Anyon