⇒ Keys

- Super
- Candidate
- Primary
- Composite
- foreign key

→ ToDos

- Super Keys

DBMS ⇒ R + DBMS

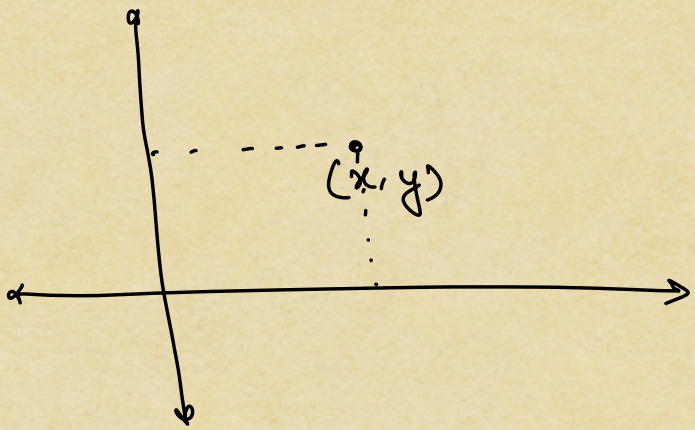
⇓

Relational DBMS

* data ⇒ rows + columns
* tables [rows, columns]
* tables can also be related to each other ⇒ Cardinality
* each row should be unique
* each row should be uniquely identifiable

Student

Name	Phone	Email	PSP	Passout
Nitish	1234	—	90	2019
Ravi	5678	—	60	2020
Vijay	91011	—	70	2021
Sawan	1112	—	69	2017
Ravi	1314	—	99	2000



each row should be uniquely identifiable ⇒ a single or a set of columns

i) name → ✗

ii) phone → ✓

iii) email → ✓

iv) psp → ✗

v) passout → ✗

vi) {name, psp, passout} → ✗

vii) {name, email, psp} → ✓

viii) {email, phone} → ✓

ix) {phone, psp} → ✓

x) {psp, passport} → ✗

xi) {phone, passport} → ✓

xii) {name, phone, email, psp, passport} → ✓

* If a column or set of columns helps us to uniquely identify a row in a table, that comb^{n} (single or multiple cols) is called a super key

* Candidate keys

Super keys

	Super	Candidate
i) phone →	✓	✓
iii) email →	✓	✓
vii) ~~{name~~, email, ~~psp}~~ →	✓	✗
viii) {email, phone} →	✓	✗
ix) {phone, psp} →	✓	✗

(ii) {phone, passport} → ✓ ✗

(iii) {name, phone, email, psp, passport} → ✓ ✗

A super key with no redundant column is called a candidate key

↓

take a super key, remove columns and check if we can identify a row uniquely. keep removing until all redundant columns, left combination will be a candidate key

* If 1 column super key always → Candidate key ✓

* Only 1 column super key is candidate key ✗

Scaler

↓ given to a student when he/she joins the class

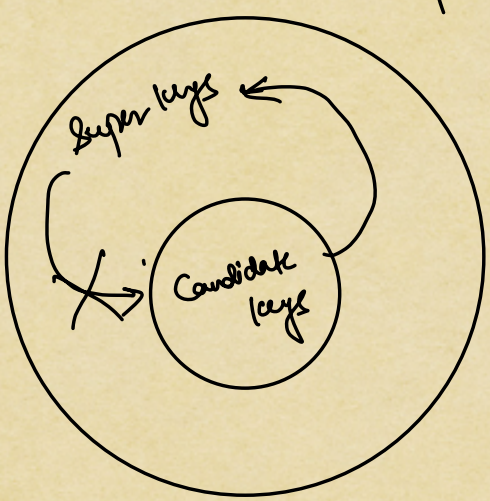
classId	studentId	class duration	%
100	1	120	90
100	2	120	80
100	3	120	65
100	4	120	95
101	1	180	90
101	2	180	90
101	3	180	90

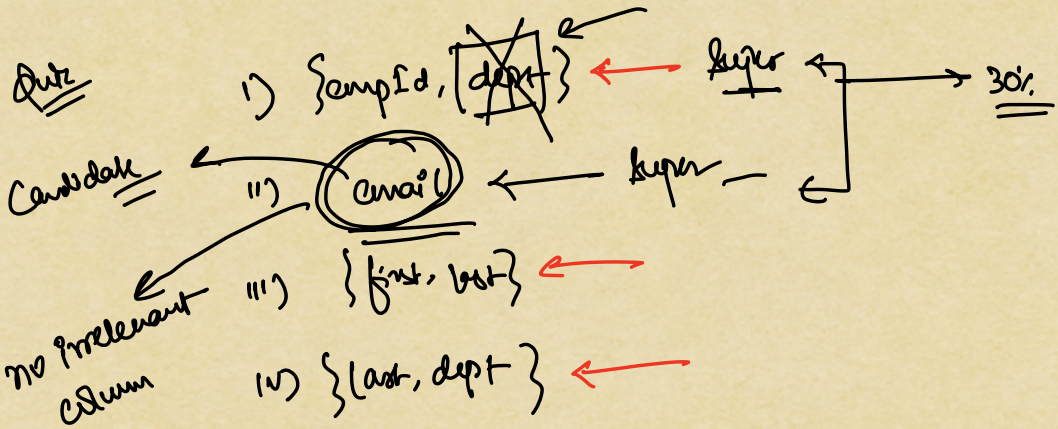
101	4	180	90

{classId, studentId} → Superkey as well as a candidate key

Name	Country code	Phone No.
—	+91	1234
—	+91	5678
—	+91	91011
—	+91	1112
—	+1	1234
—	+1	5678
—	+1	91011
—	+1	1112

Super key + Candidate keys





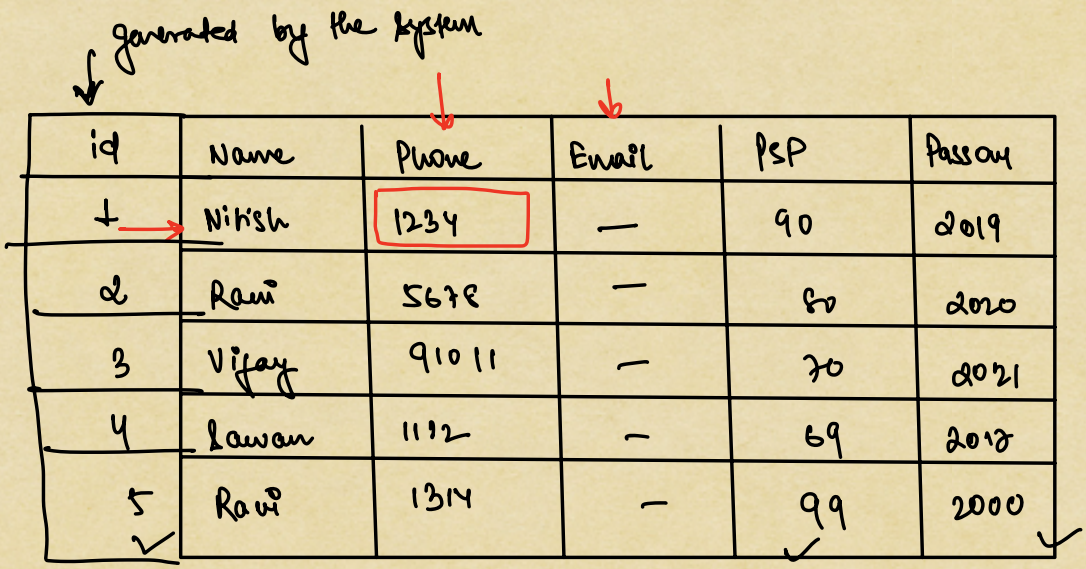
⇒ Primary key :-

the best candidate key is the primary key

⇒ good primary key properties :-

i) faster to compare
ii) faster to sort
iii) smaller and defined size
iv) should ideally never change
v) should ideally not be updatable by end user.

* as we can't always satisfy all these properties by any candidate key. the best practice is to create your own id column → pk



* Properties of pk in table :-

1) pk cant be null
2) table is sorted by pk by default
3) there is an index on pk by default
4) By default, searching via pk is the fastest
5) we need to define a pk while creating a table [never]

```
CREATE TABLE students (
    id INT AUTO_INCREMENT,
    firstName VARCHAR(50) NOT NULL,
    lastName VARCHAR(50) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    dateOfBirth DATE NOT NULL,
```

```
    enrollmentDate TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    psp DECIMAL(3, 2) CHECK (psp BETWEEN 0.00 AND 100.00),
    batchId INT,
    isActive BOOLEAN DEFAULT TRUE,
    PRIMARY KEY (id)
);
```

* Composite key

Any superkey or candidate key or primary key, which more than 1 column is a composite key

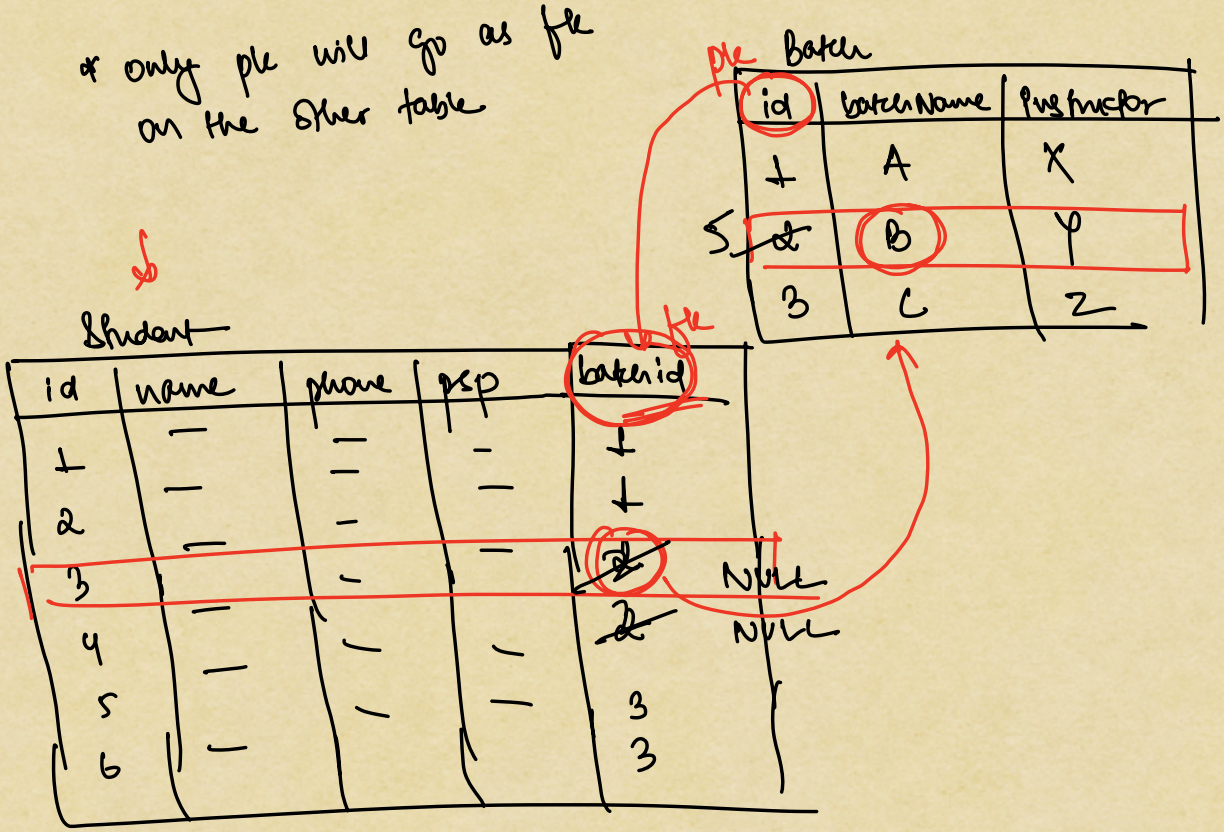
Superkey, Cand key, Pk, Comp key

⇒ Foreign Keys

Super key, Cand key, Pk, Comp key ⇒ identify row in the same table

fk ⇒ identify row in a separate table (reference)

* only pk will go as fk on the other table



maintain fk

1) CASCADE → do the same action on fk table

2) NULL → orphange (make the fk null which was deleted/updated)

3) BLOCK → dont do the action if any row is referencing else allow

H.W

1) Install MySQL &

2) " MySQL workbench

3) try to access "Sakila DB" → MySQL
↓
inbuilt database