"SQL 2"

Agenda

- Keys
  - Super Keys
  - Candidate Keys
  - ** Primary Keys
  - Composite Keys
  - Foreign Keys (Constraints)

CRUD

Installation guide

⊛ KEYS

col / attribute

"Every Row is Unique"

Students

	id	Name	Phone
	1	Ankit	
→	2	Vivek	12345
→	3	Vivek	12345

Avoid Ambiguity → Confusion

Key :- A col. or a group of cols. that is guaranteed to help identify a row uniquely in a table

⭐ Types of Keys :-

① SUPER KEY

same as def^{n} of KEYs (above)

a col. or a set of cols. to uniquely identify a row

<C1, C2, C3, C4, C5> SK

PK

<email> → SK, CK
<phone> → SK, CK

Students (@ Scaler)

name	psp	email	phone	batch_id	.....

? What all can be SK (uniquely identify a row)

- phoneNo ✓
- email ✓
- <phoneNo, email> ✓
- name ✗

SK

<name, psp, batch_id, email>

↳ <name, email>

<name, email> uniquely

⊛ Candidate Keys

Prob. w/ SK ⟶ Extra cols.

Def^{n} :- (SK) of smallest size

not necessarily single

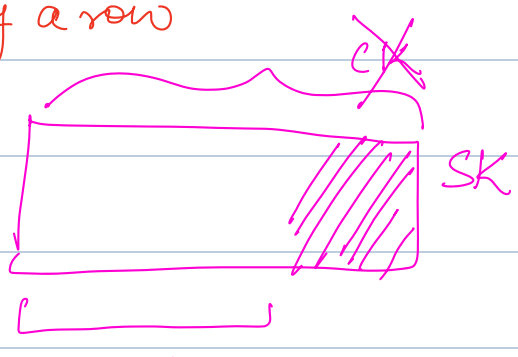
can't remove even single col. from set

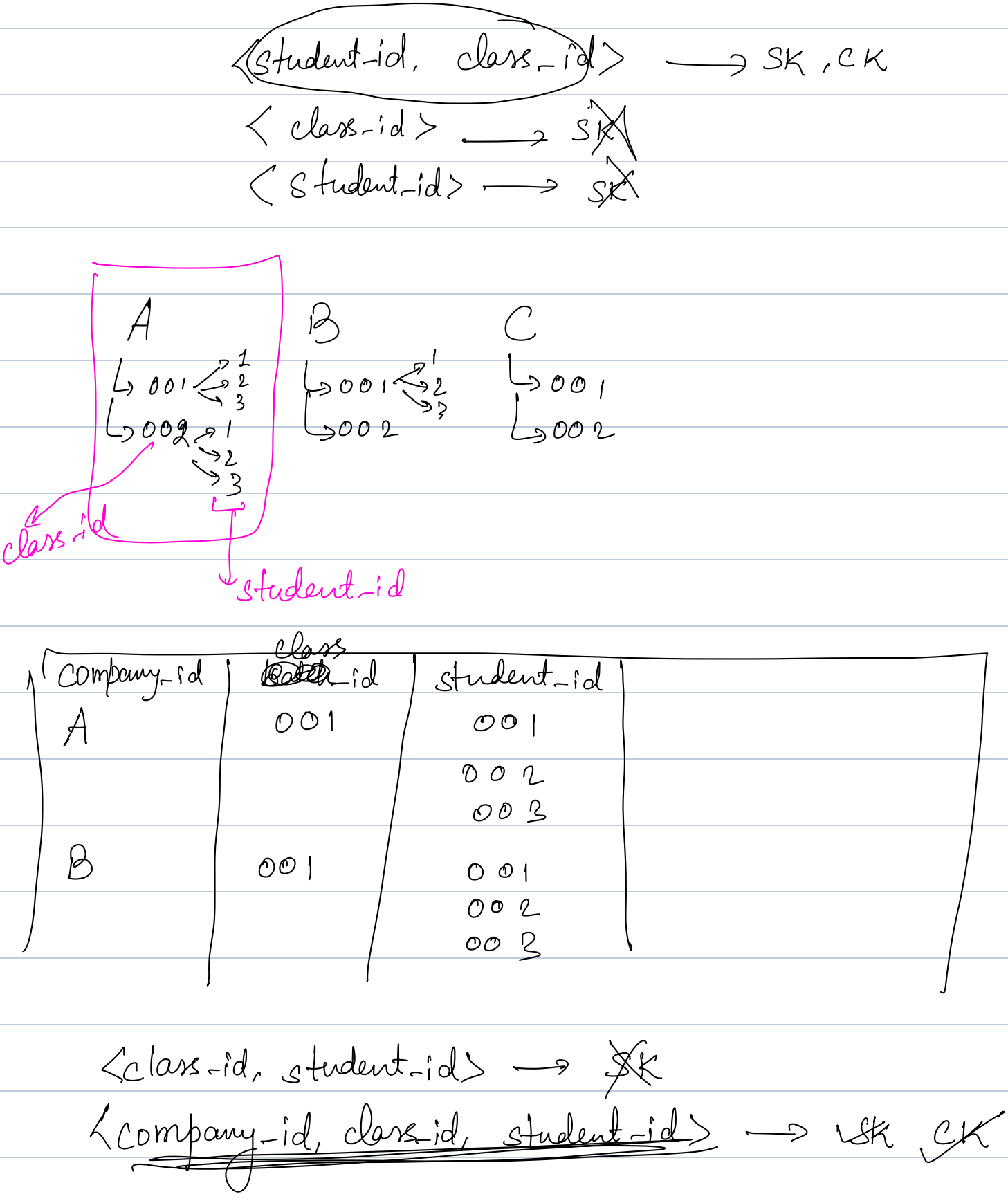
Table: "Attendance"

student_id	class_id	attendance_percentage
400 ✓	001	~~75~~ 84
401	001	100
402	001	93
400 ✗	002	84
401	002	84
402	002	100

<st_id, class_id> uniquely identify a row

	SK	CK
(1)	student_id ✗	✗
(2)	class_id ✗	✗
(3)	<student_id, class_id> ✓	✓ PK
(4)	<student_id, class_id, attendance_percentage> ✓	✗




⟨student-id, class-id⟩ → SK, CK
⟨class-id⟩ → SK
⟨student-id⟩ → SK
A
001
002
1
2
3
1
2
3
B
001
002
1
2
3
C
001
002
class-id
student-id
company-id
class-id
student-id
A
001
001
002
003
B
001
001
002
003
⟨class-id, student-id⟩ → SK
⟨company-id, class-id, student-id⟩ → SK CK
Remember
Super Key → uniquely identify a row

↳ Candidate Key → uniquely identify a row + No unnecessary col.

⊛ Primary Key

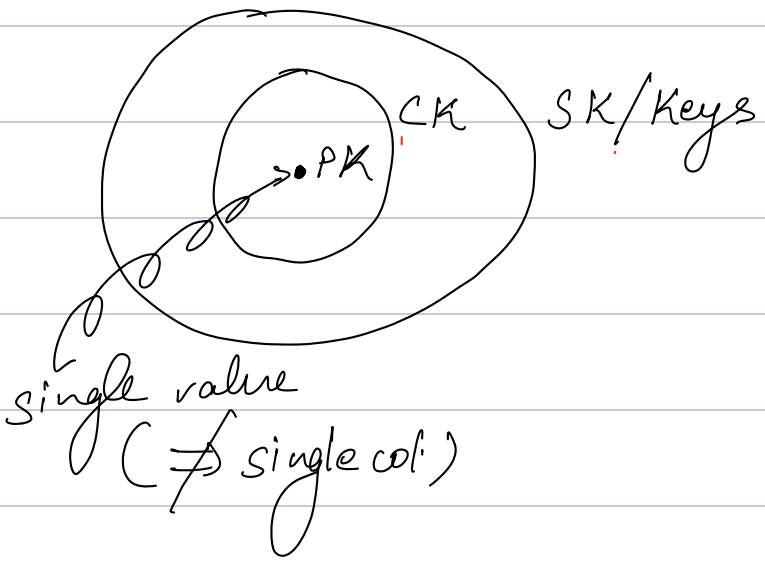
Party Workers → Super Keys

Best → Candidate Keys

Only One CM → Primary Key

DB enforces DB designer/dev. to define ONE PK

Def^{n} : PK is a CK selected to be unique set of col. to uniquely identify a row in a table



Composite

↳ Any key w/ more than one col.

Q Why SQL enforces to select PK?

↳ creating index

↳ DB sorts the data PK

↳ DB outputs data (sorted)

PK is the CK that DB dev. chooses.

Students

id	name	email	phone No	batch_id
1	Aman	xyz@a...		
	Aman			

we end up explicitly adding an id col
↳ autoincremented

~~Rule~~
Best Practices → PK length should be as less as possible

<id> → CK → PK
<name> → ''
<email> → ''

* Composite Key ??

A key w/ more than one col.

composite SK

<student_id, class_id> → CK, PK, SK

composite "CK" composite PK

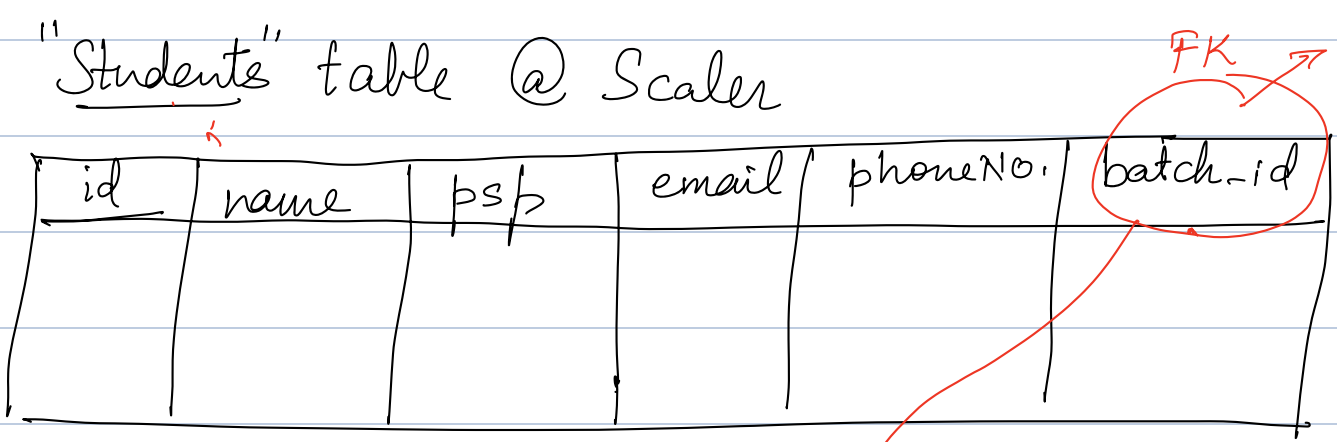
[ ] CK (SK)

"10:50pm"

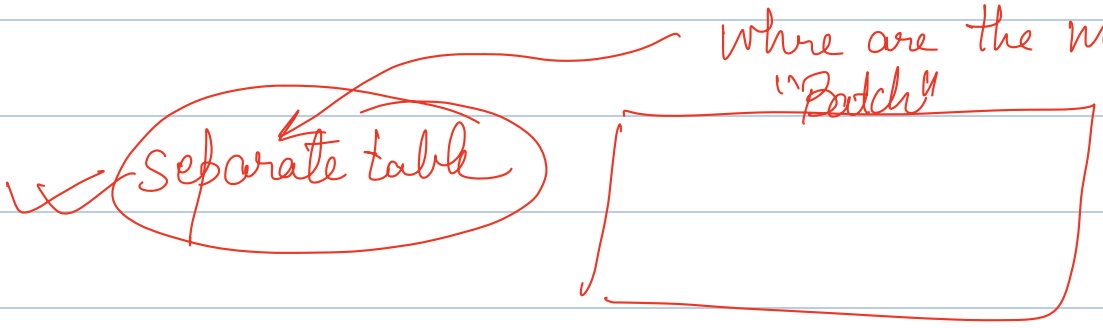
⊛ Foreign Keys (Constraint)

Has nothing to do → PK, CK, SK

"Students" table @ Scaler



Where are the more details?



"Batch_id" ⟶ helping to uniquely identify a row in another table (name "Batch")

Def^n (FK) :- Its is a set of col/s. that helps to uniquely identify a row in another table

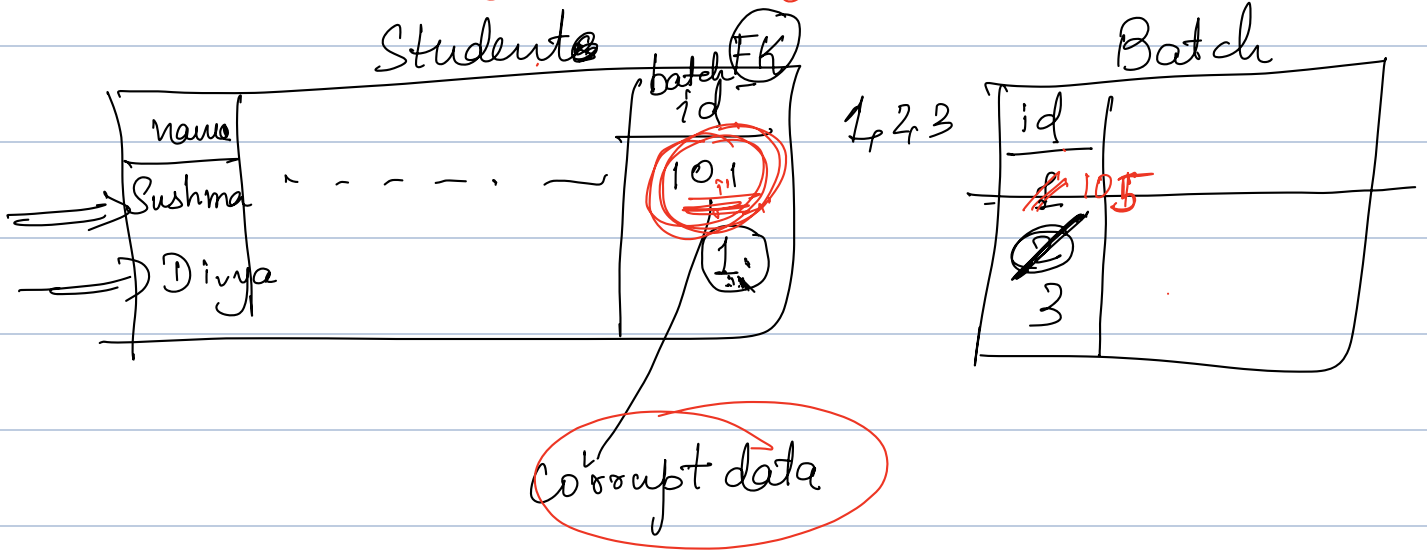
Q Why its called a Key
↳ Help to uniquely identify

Note :-

↳ Something that uniquely identifies you/yourself
  ↳ PK, CK, SK
  ↳ " , " " someone else
  ↳ FK

Q Why need FK ?

Ans Specifying FK brings "Data Integrity"



(*) Solve FK prob. ?

(1.) Insert a student (with invalid batch_id)

(2.) Delete a row from "Batch" Table
  ↳ delete all students (referencing that batch)
  ↳ Make it as NULL (in Student table)

Orphaning

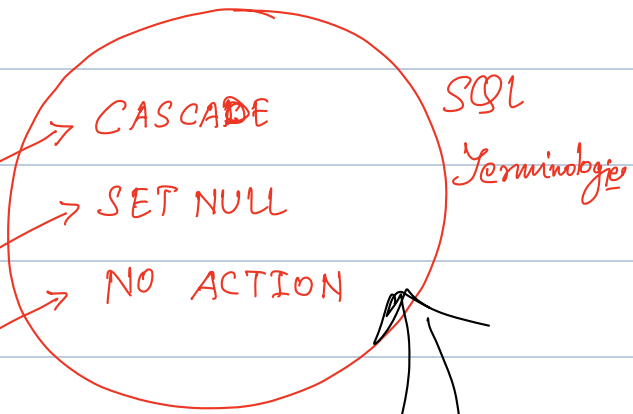
↳ Disallow deletion

(3.) Update a batch (with diff. batch_id)

Inconsistency ↳ all 3 alternatives as above

(*) Sol^n

- do what you did
- NULL (replace)
- Restrict / Don't Allow



(*) Note :-

In SQL, when you create a FK, you can specify what to do in case of DELETE or UPDATE on another table

Q Which is better?

Ans: Depends (on Prob. Statement)

(*) FK need not be PK in another table

(*) Composite FK

Conclusion

Keys

- SK
- CK
- PK
- FK

Composite Key & Non-composite Key

Student

student_id	class_id	psp
01	1	
02	1	
03	1	
01	2	
02	2	
03	2	

SQL

SQL

<student_id, class_id>

CK, PK

Leaderboard

student_id	class_id	marks	time_taken

<student_id, class_id>

FK (composite)

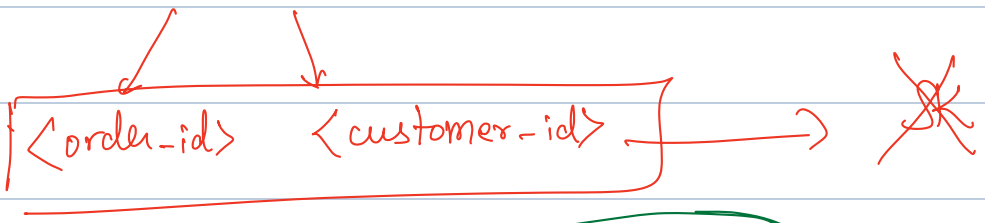
HW :- (Install)

- MySQL 8
- MySQL Workbench (IDE)
- Sakila DB (sample data set)

(*) Composite SK ⟶ SK w/ more than one col.
Composite CK ⟶ CK " " " " "
Composite PK ⟶ PK " " " " "
Composite FK ⟶ FK " " " " "

can be non-composite also.

<order_id, customer_id> ⟶ SK



~~a.)~~ Composite Key
~~b.)~~ Candidate " ⟶ C CK
~~c.)~~ SK ⟶ C SK
~~d.)~~ PK ⟶ C PK