TOPIC 3
NORMALIZATION

WHAT IS NORMALIZATION?

NORMALIZATION IS THE PROCESS OF ORGANIZING DATA IN A DATABASE SO THAT:

- There is no redundant (repeated) data.
- There are no anomalies (insertion, deletion, update issues).
- Data is stored in a clean, consistent, and manageable way.

PURPOSE OF NORMALIZATION:

- To make the database more efficient.
- To prevent wasted storage space.
- To ensure data integrity (data remains accurate and consistent).
- To make queries easier and faster to manage.

Before Normalization: Repeated Data

Emp No	Employee Name	Dept No	Dept Name	Timecard No	Timecard Date
10	Thomas Arquette	20	Marketing	106	11/02/2002
10	Thomas Arquette	20	Marketing	115	11/09/2002
500	Alan Cook	50	Shipping	107	11/02/2002
500	Alan Cook	50	Shipping	107	11/02/2002
700	Ernest Gold	50	Shipping	108	11/02/2002
700	Ernest Gold	50	Shipping	116	11/09/2002

WHY IS NORMALIZATION IMPORTANT?

1. Redundancy – data stored multiple times unnecessarily.
2. Update difficulties – changing data in one place but not in others causes inconsistencies.
3. Slower queries – messy structure makes searching inefficient.
4. Wasted storage space – repeated data wastes memory.

WITH NORMALIZATION

1. Database saves space.
2. Data is consistent.
3. Easier to maintain.

Before Normalization
(Redundant Data)

Emp No	Employee Name	Dept No	Dept Name	Timecard No
10	Thomas Arquette	20	Marketing	106
10	Thomas Arquette	20	Marketing	115
500	Alan Cook	50	Shipping	107
500	Alan Cook	50	Shipping	107
700	Ernest Gold	50	Shipping	108
700	Ernest Gold	50	Shipping	116

After Normalization
(Clean & Efficient)

Employees

Emp No	Employee Name	Dept No
10	Thomas Arquette	20
500	Alan Cook	50
700	Ernest Gold	50

Departments

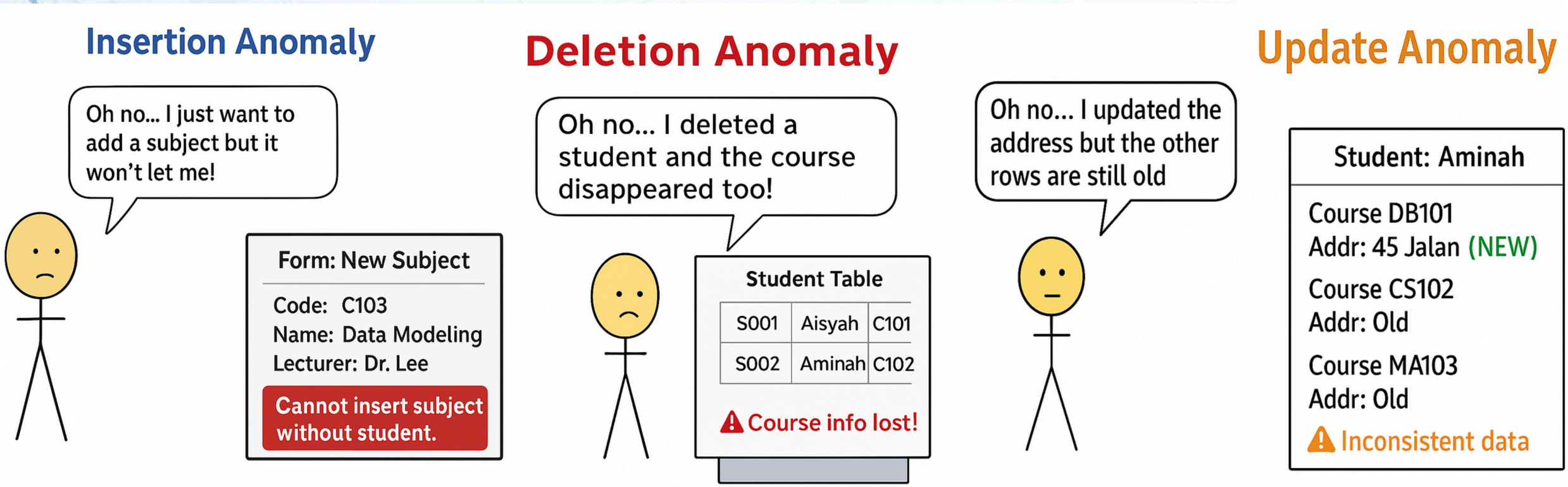
Dept No	Dept Name
20	Marketing
50	Shipping

Timecards

Timecard No	Emp No
106	10
115	10
107	500
108	700
116	700

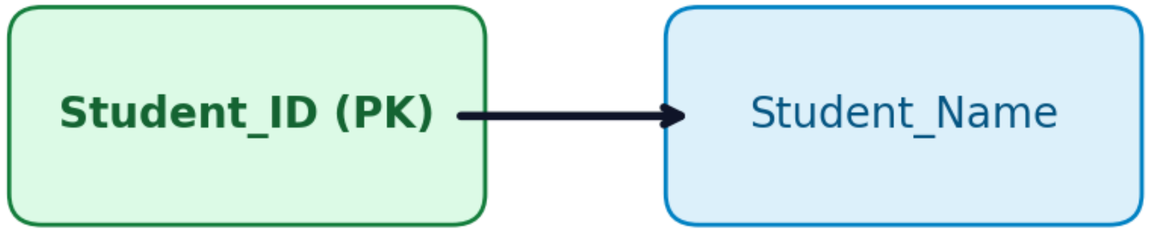
PROBLEMS WITHOUT NORMALIZATION (ANOMALIES)

1. Insertion Anomaly – Cannot insert new data unless other data exists.
2. Deletion Anomaly – Deleting one record removes other important info.
3. Update Anomaly – Updating one record but others stay unchanged.



FUNCTIONAL DEPENDENCY

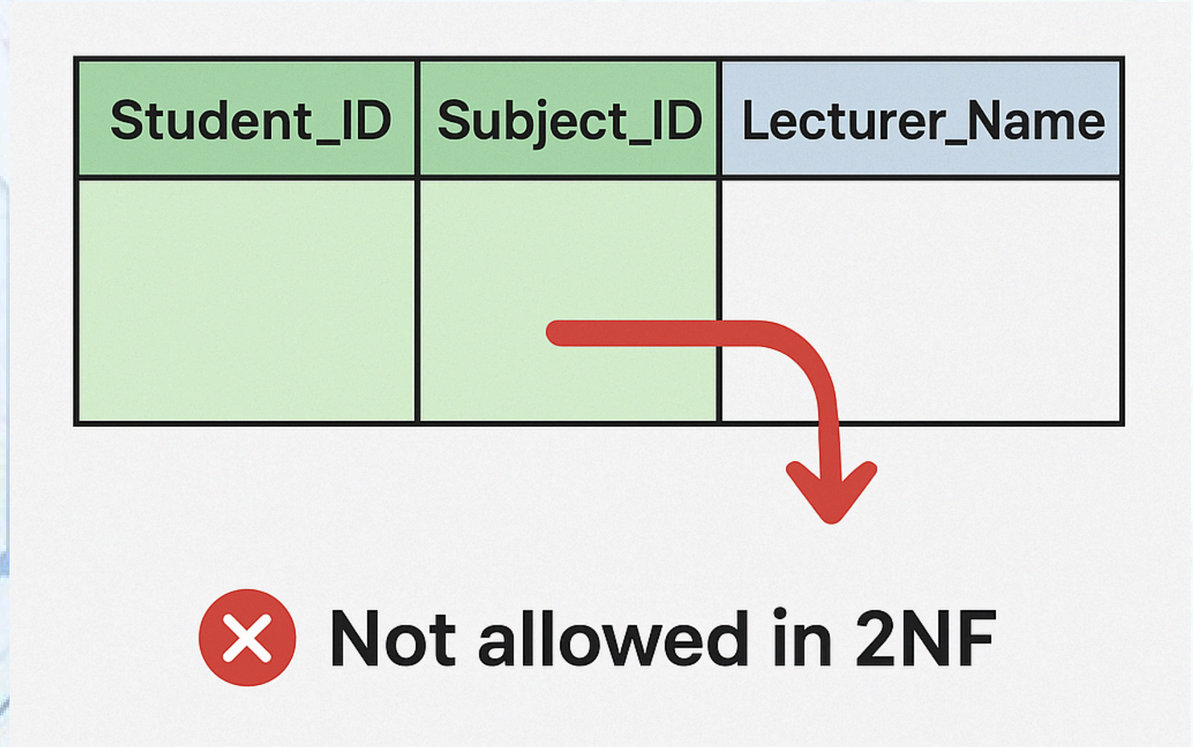
1. One attribute depends on another (usually PK).



Example: Student_ID → Student_Name

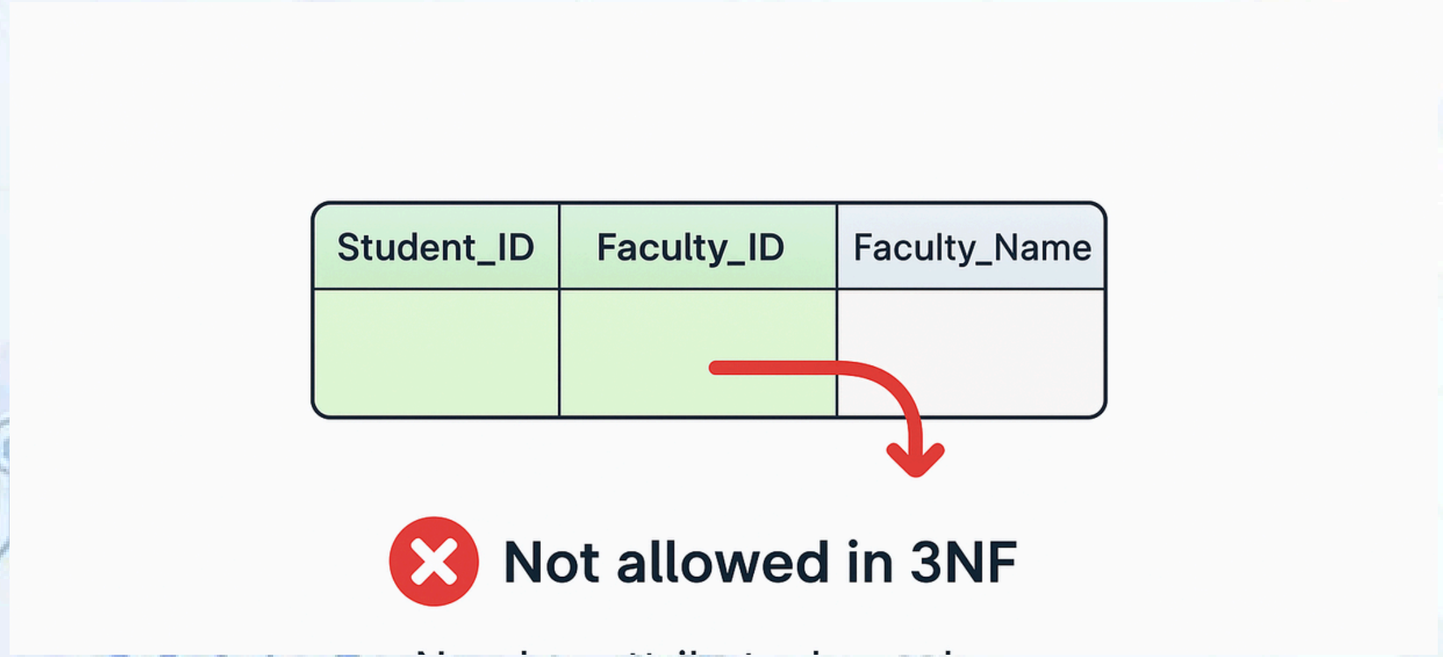
PARTIAL DEPENDENCY

1. Non-key attribute depends only on part of a composite key.
2. Example: (Student_ID, Subject_ID) → Lecturer_Name.
3. ❌ Not allowed in 2NF.



TRANSITIVE DEPENDENCY

1. Non-key attribute depends on another non-key attribute.
2. Example: Student_ID → Faculty_ID → Faculty_Name.
3. ❌ Not allowed in 3NF.



NORMAL FORMS (NF)

1NF

- No repeating groups, all values atomic.

2NF

- Must be in 1NF.
- No partial dependency.

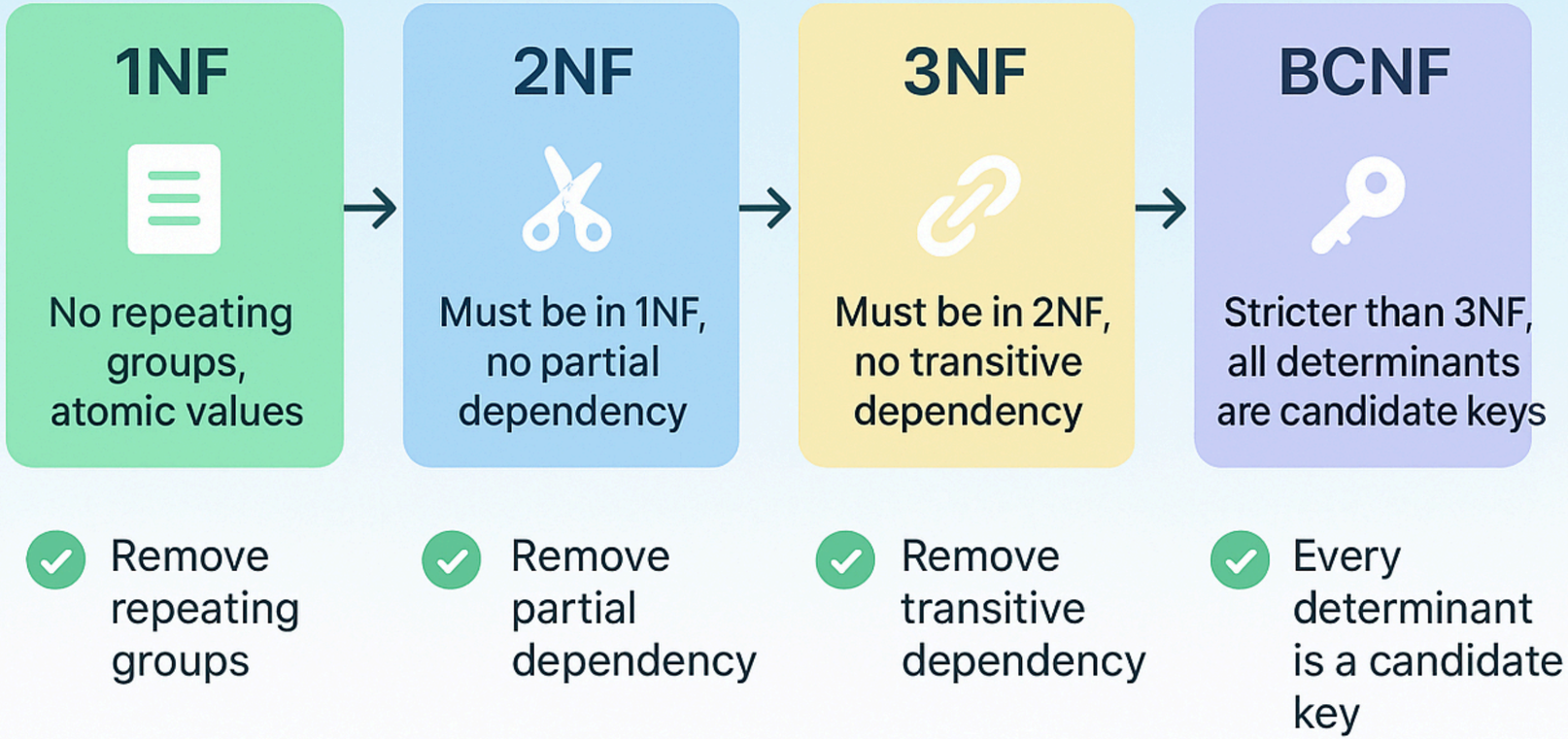
3NF

- Must be in 2NF.
- No transitive dependency.

BCNF

- Stricter than 3NF.
- All determinants are candidate keys.

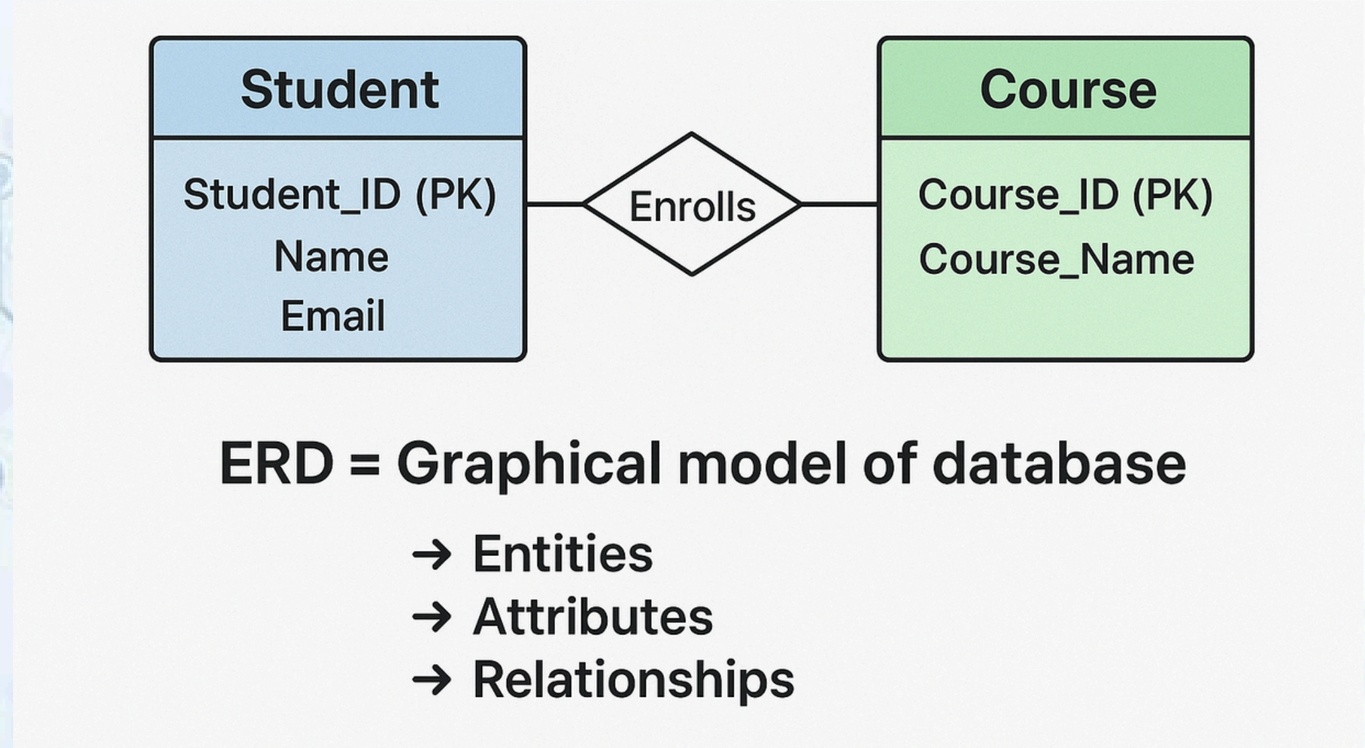
📌 SUMMARY:



Normalization is step-by-step refinement.

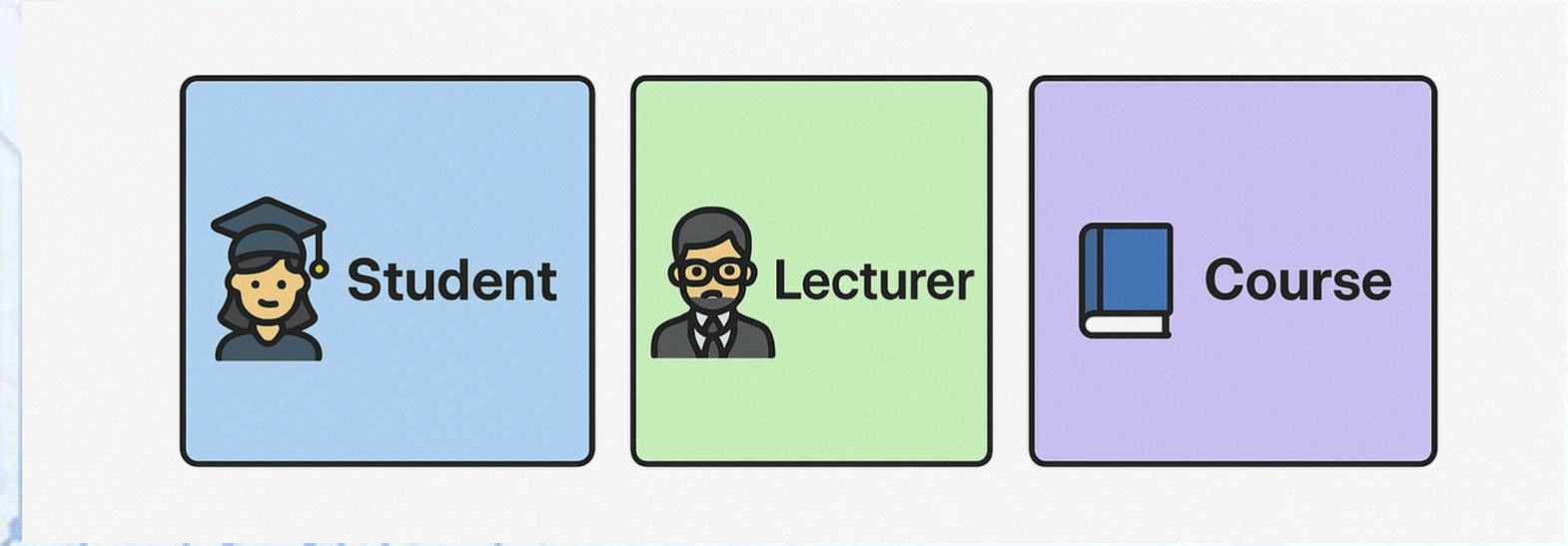
WHAT IS ERD?

- ERD = graphical model of database.
- Shows: Entities, Attributes, Relationships.



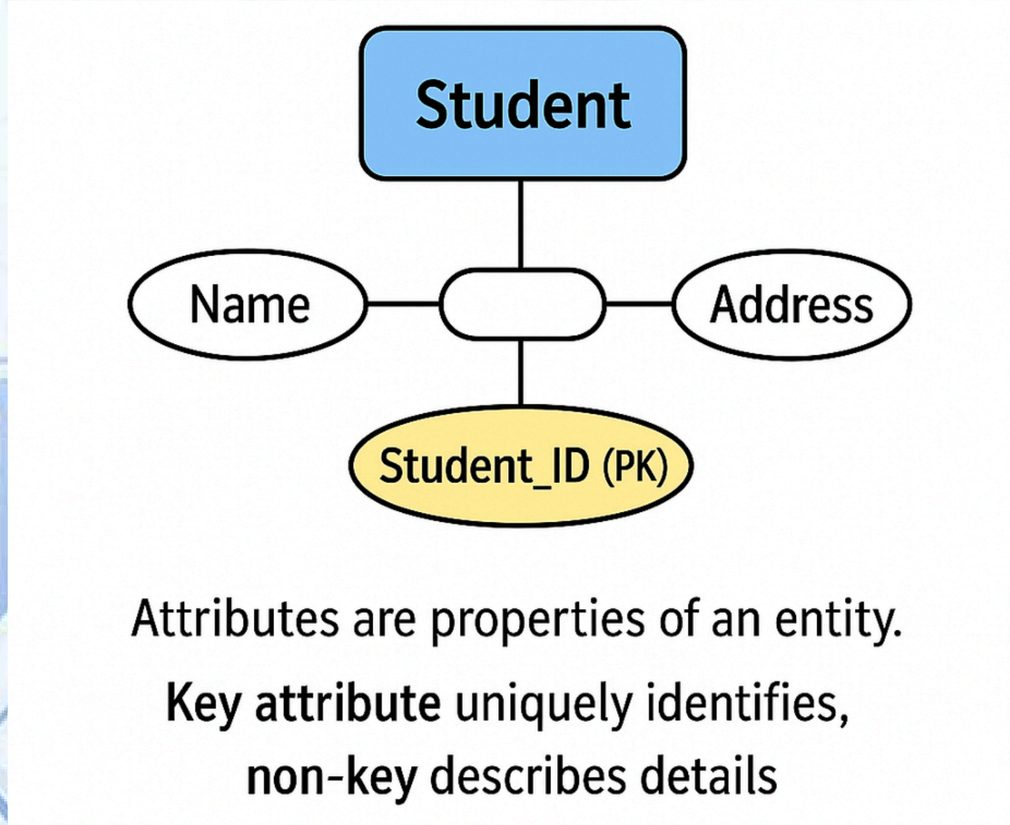
ENTITIES

- Entity = object we store data about.
- Example: Student, Lecturer, Course.



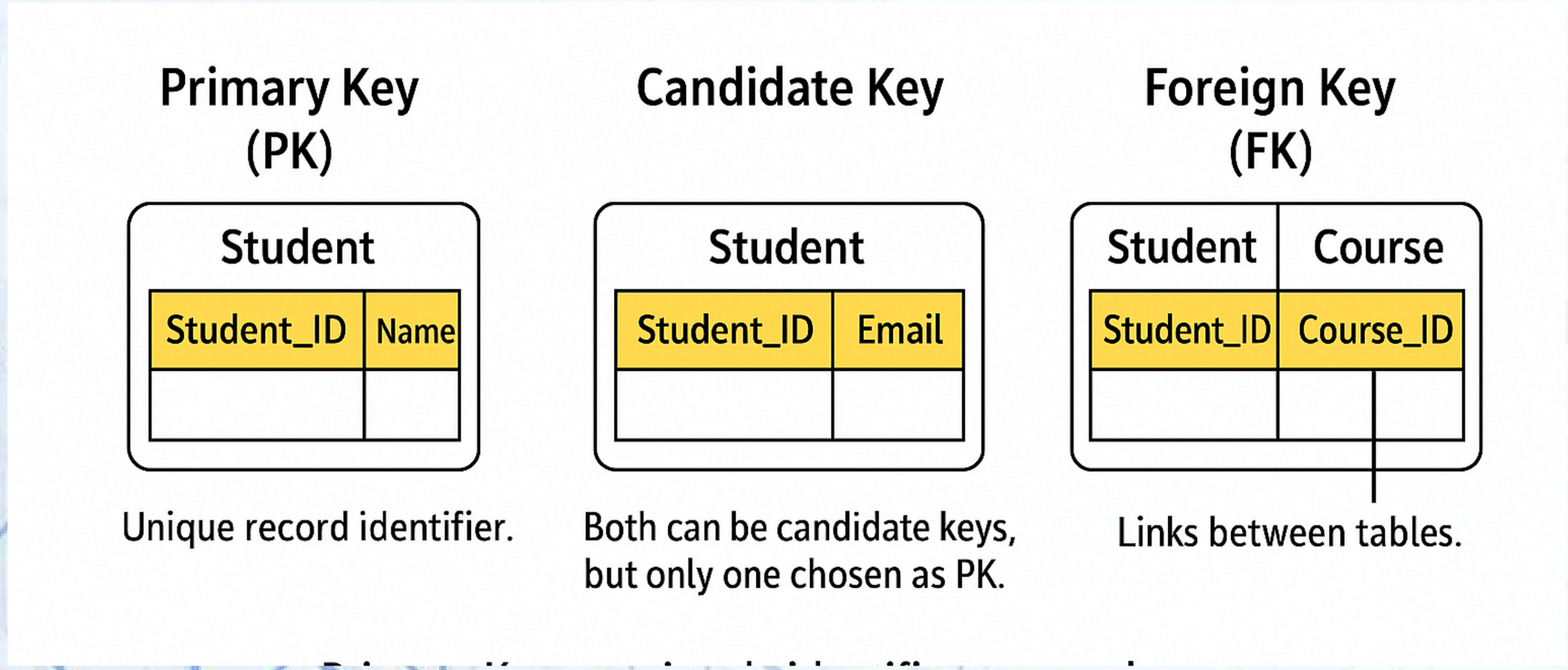
ATTRIBUTES

- Properties of entity.
- Example: Student → Name, Address.
- Types: Key attributes & Non-key attributes.



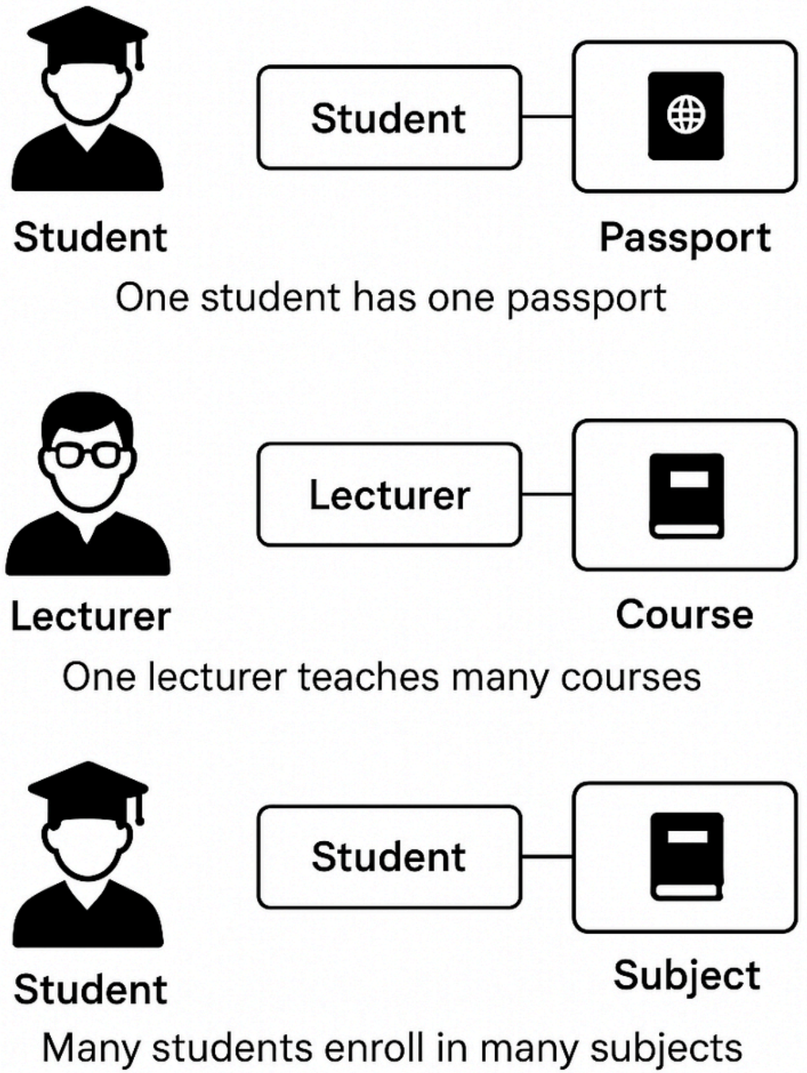
KEYS

1. Primary Key – unique record.
2. Candidate Key – possible PK.
3. Foreign Key – links between tables.



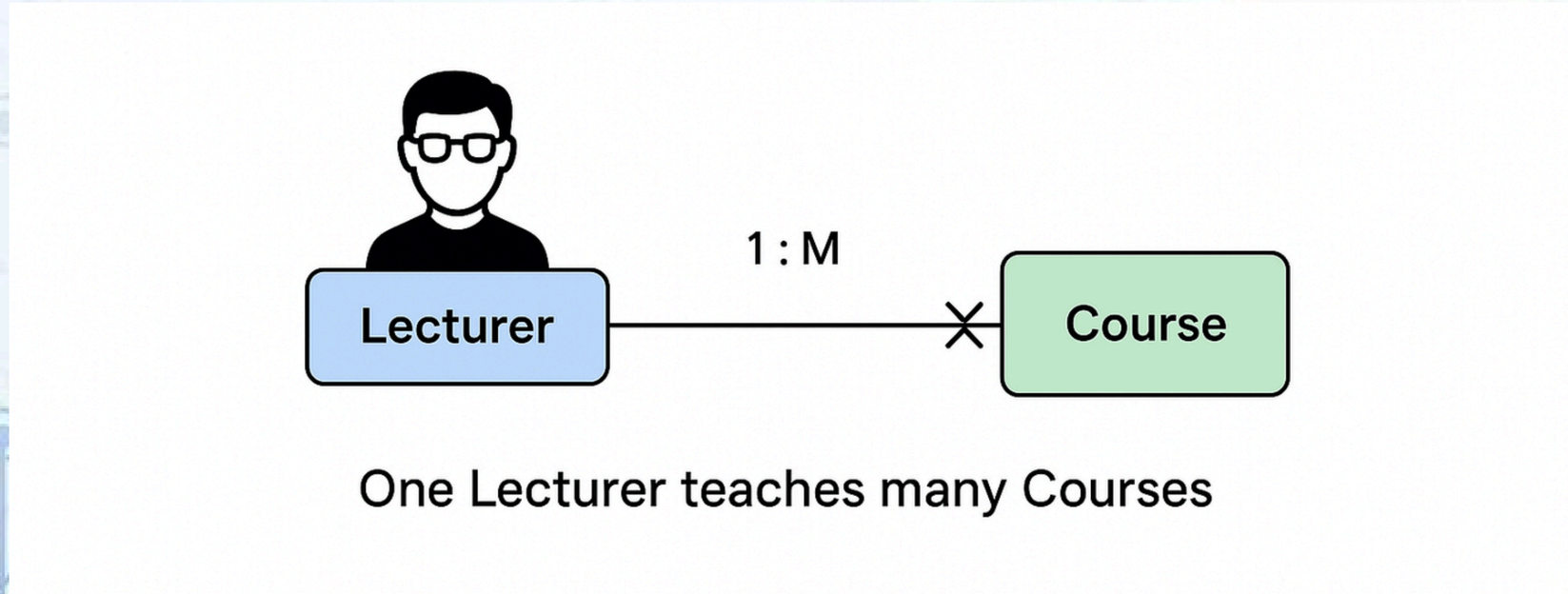
RELATIONSHIPS

- 1:1 → Student – Passport.
- 1:M → Lecturer – Course.
- M:N → Student – Subject.



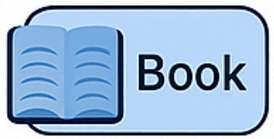
CARDINALITY

- Shows how many instances of entity connect to another.



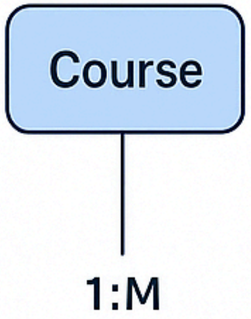
STEPS TO BUILD ERD

1) Identify entities



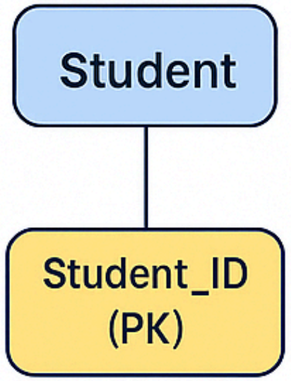
List the main objects (nouns) in the domain

2 Define relationships



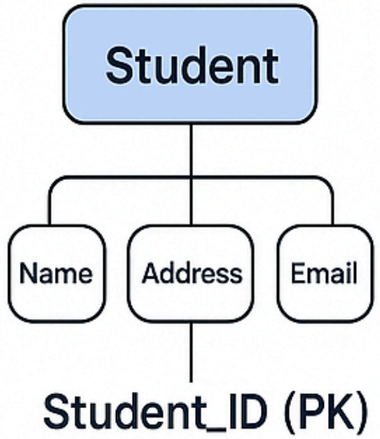
Set 1:1, M, or M:N

3 Assign keys



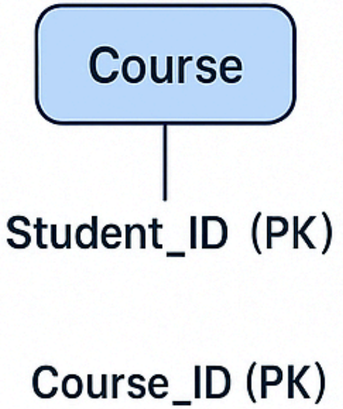
Choose a unique identifier

4 Add attributes



List key & non-key attributes

5 Draw with correct notatio



Use standard ERD notation (Chen / Crow's Foot)

📌 CONCLUSION

Normalization

clean, consistent, no redundancy

Normal Forms (1NF → BCNF)

stepwise refinement

ERD

visual modeling of entities, attributes, relationships

Database design becomes efficient, consistent, and easier to manage.