Database Systems (ISYS1001/ISYS5008)

Lecture 4

More on Keys, Mapping ER to Relations, Normalization

Updated: 18th August,2021

Discipline of Computing
School of Electrical Engineering, Computing and Mathematical Sciences (EECMS)

CRICOS Provide Code: 00301J

Copyright Warning

COMMONWEALTH OF AUSTRALIA
Copyright Regulation 1969

WARNING

This material has been copied and communicated to you by or on behalf of **Curtin University of Technology** pursuant to Part VB of the *Copyright Act 1968* (**the Act**)

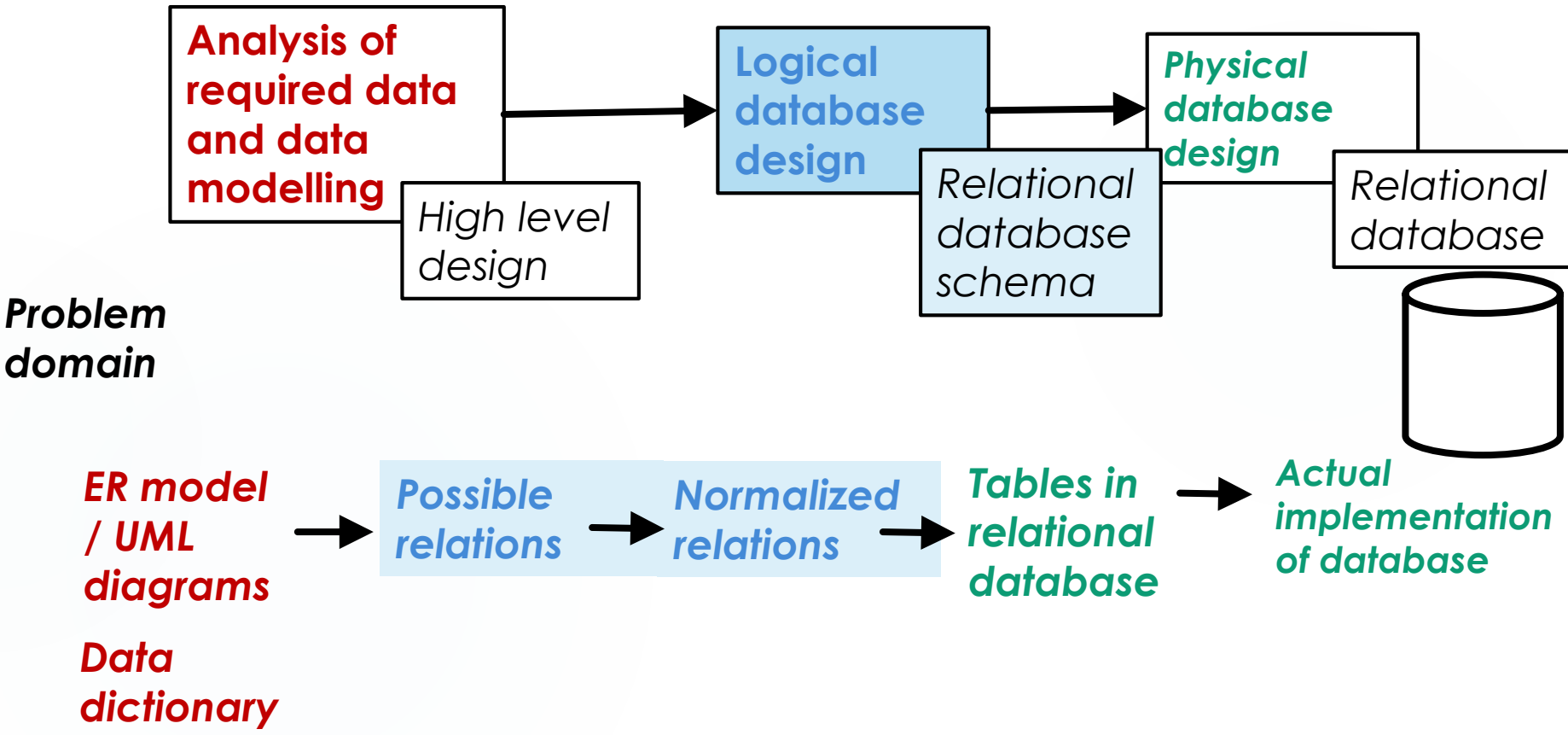
The material in this communication may be subject to copyright under the Act. Any further copying or communication of this material by you may be the subject of copyright protection under the Act.

Do not remove this notice

Learning outcomes

- Converting a designed ER model to a proper relational schema using well established guidelines
- Describe anomalies which can occur in poorly designed relational schema.
- Use normal tests and verify whether relational schema is of a particular normal form (1NF, 2NF, 3NF,BCNF), and if not do the decomposition to convert it to the normal form.
- Explain how functional dependencies are used in the normalization process.
- Discuss the main properties of functional dependencies.

Real world to database



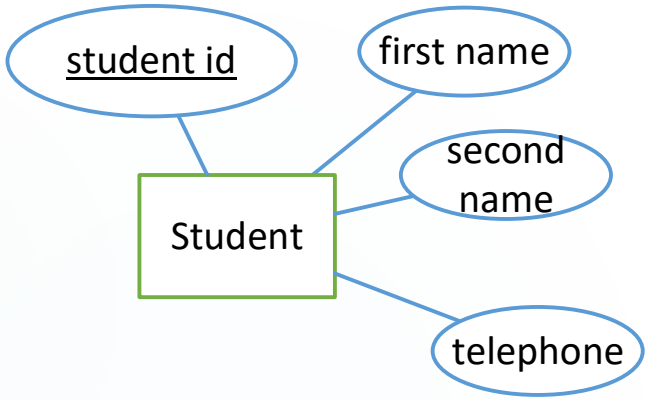
We have looked at the main ideas of ER model last week.
We will be looking at logical database design stage in this lecture.

Mapping ER model to relational schema

- ER model resulted in a diagram with entities, attributes and relationships about the concerned domain.
  - E.g. ER model for a student information system
- Process of converting the ER model to a relation schema is called mapping the ER model to relational schema or relational mapping.
- Entities, attributes, relationships and any constraints identified in ER model become main concerns in relational mapping.
- Goal of relational schema design is to avoid anomalies and redundancy in the resulting database.

Mapping ER model to relational schema

- When mapping ER to relational schema, entities can become relations.
- Then the attributes of entities can become attributes of relations
- Example:




```
Students(studentID, firstname,
secondname,telephone)
```

- Key of an entity is the identifier of the entity.
- One of the candidate keys become the primary key of the relation.

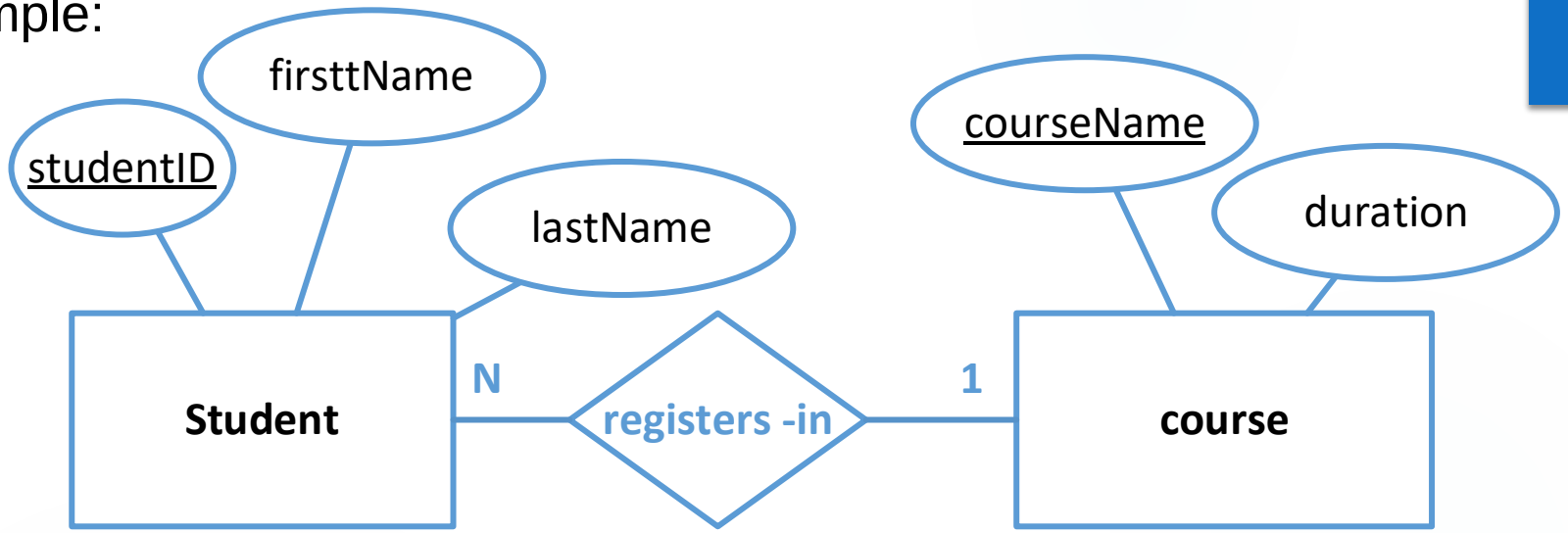
Mapping ER model to relational schema

- Relationships between entities are important to maintain the correct information in the database.
- Recall Practical -2 , **`Units, Students, Enrolments`** tables
  - If student name is changed in **`Student`** table, does it affects the **`Enrolment`** table?
  - If a unit is deleted in **`Units`** table while a student is enrolled to that unit (**`Enrolment`** table) what will happen to that enrolment?
  - These changes are not properly updated in the **`Enrolment`** table as it is not connected with the other tables.
  - *When one part of your database expects data to be in another place, but that data has been deleted or changed, problems can arise.*
- When relations are derived from entities, a key of one entity can be used in a relating entity as an attribute for maintaining the relationship between the two entities.
- They are called foreign keys (FK) of the relating entity.

Foreign keys and referential integrity constraints

- We place restrictions on data to help the DBMS keep the data valid.
- Referential integrity constraint is specified between two relations and is used to maintain the consistency among tuples in the two relations.
- Informally, the referential integrity constraint states that *a tuple in one relation that refers to another relation must refer to an existing tuple in that relation.*
- Foreign keys enforce referential integrity constraints
- Foreign key attributes in R1 referring to R2 have the following rules:
  - The FK attributes in R1 have the same domain(s) as the primary key attributes if R2
  - The value of foreign key in tuple t1 in R1 must referrer to an existing primary key value in tuple t2 of R2
- SQL allows us to specify the referential integrity constraints on tables.

Example:




```
Student(studentID,firstName,secondName,courseName)
Course(courseName,duration)
```

Primary key (PK)

Foreign key (FK)

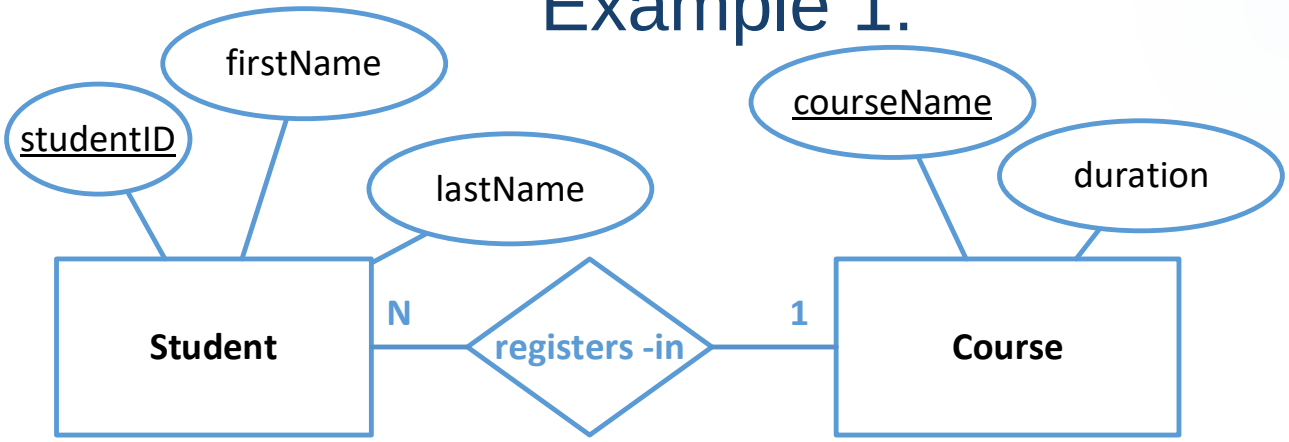
Steps to map ER model to relational schema

- Step 1: mapping entities
  - Every entity-type become a relation.
  - All the attributes of the entity become the attribute of the relation

- Step 2: mapping 1: N relationships
  - Each 1: N relationship type is mapped into the associated N-side entity type as follows:

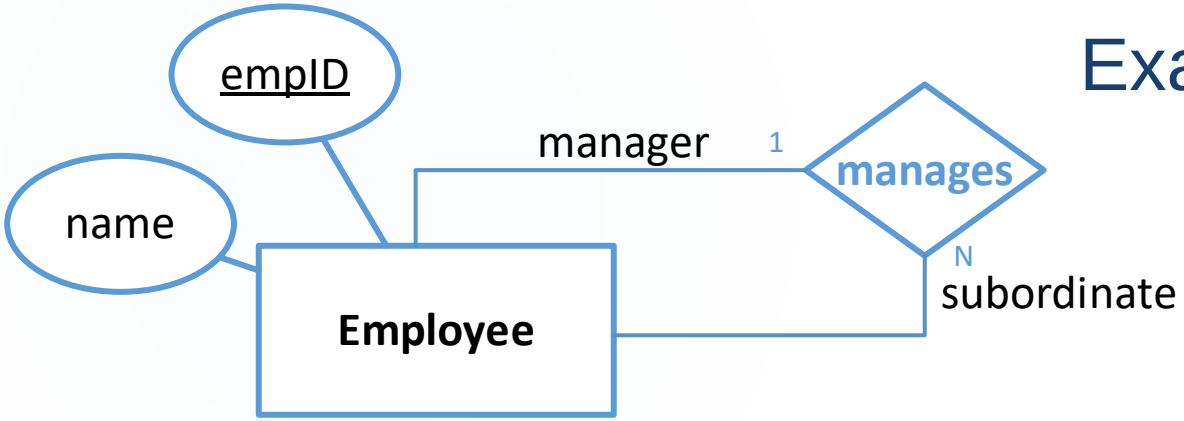
1) For each N-side entity type,
   1. Add the primary key of the 1-side entity type. This added PK becomes a foreign key.
   2. If there are any descriptive attributes in the relationship, add them also.
2) If the 1:N relationship is a recursive relationship,
   1) The primary key of the 1-side entity type is annotated with its role name.

Example 1:




```
Student(studentID,firstName,secondName,courseName)
Course(courseName,duration)
```

Example 2:

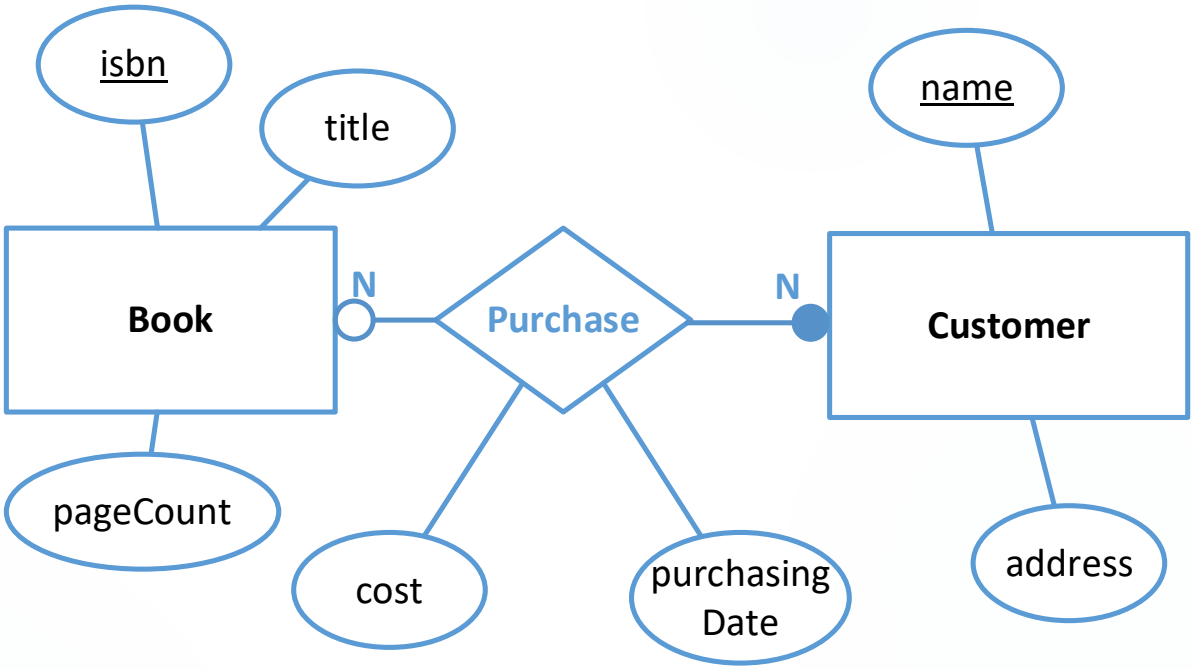


```
Employee(empID,name,manager_empID)
```

Steps to map ER model to relational schema

- Step 3 : Each M:N relationship type become a separate relation.
  1. Add primary key of the participating entity types to the relation. (They become foreign keys)
  2. Add all the descriptive attributes of the relationship type to the relation.
  3. The primary key of the relation consist of the primary keys of the two participating entities.
  4. When this rule is applied to M:N recursive relationship types,
     1. The two roles are annotated with their role names.

Example :

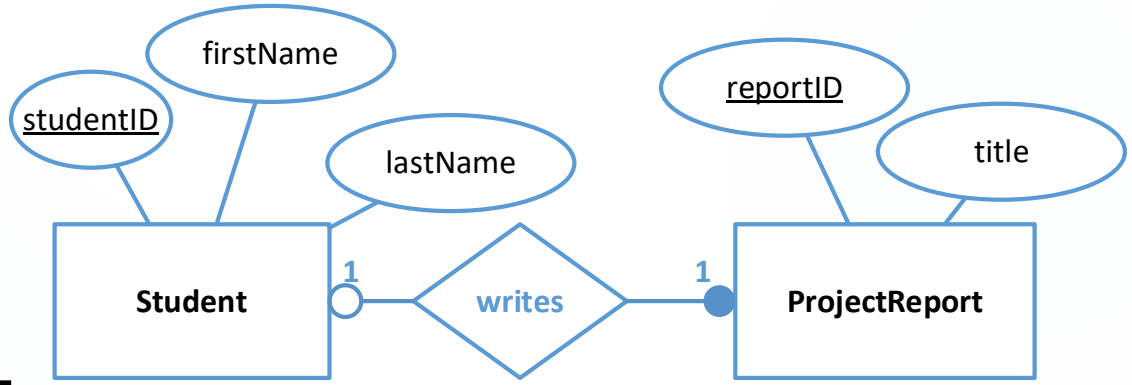



```
Book(isbn,title,pageCount)

Customer(customerName,address)

Purcahse(isbn,customerName,cost,purchasingDate)
```

Steps to map ER model to relational schema

- Step 4 : Each 1:1 relationship can be combined with either side of the entity type or can be treated like 1: N relationship type.
- However, if both sides of the relationship are with partial participation, two entities become two relations.

Example:



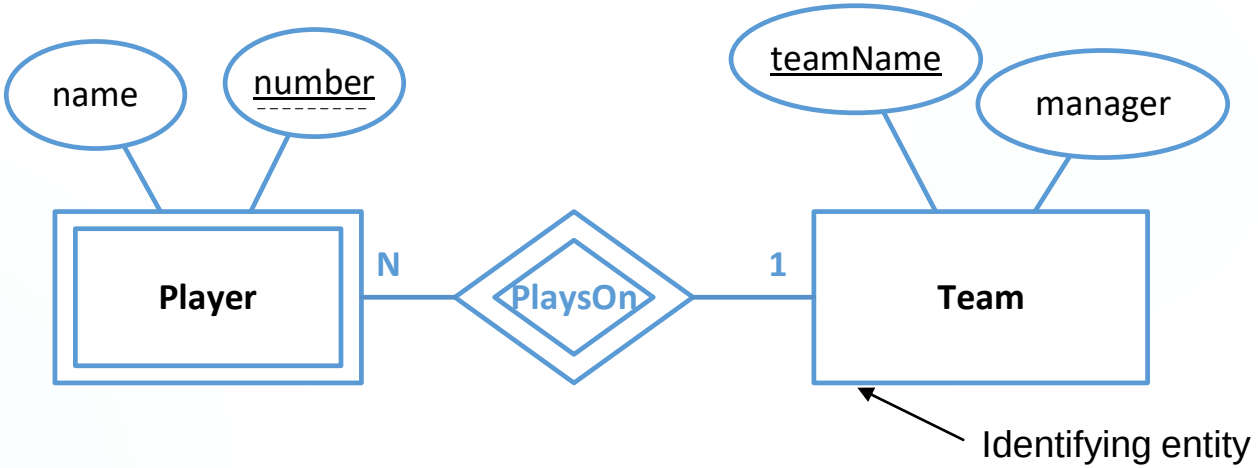

```
Student(studentID,firstName,lastName, reportId,title)

or

ProjectReport(reportId,title,studentID,firstName,lastName)
```

Note: `Student(studentID,firstName,lastName, reportId,title)` is the suitable choice due to optionality(partial participation) of `Student` entity

Steps to map ER model to relational schema

- Step 5 : A weak entity type becomes a relation that includes the primary key of the identifying owner entity type.
  1. The primary key of the relation consists of the foreign key and the discriminator.

Example:

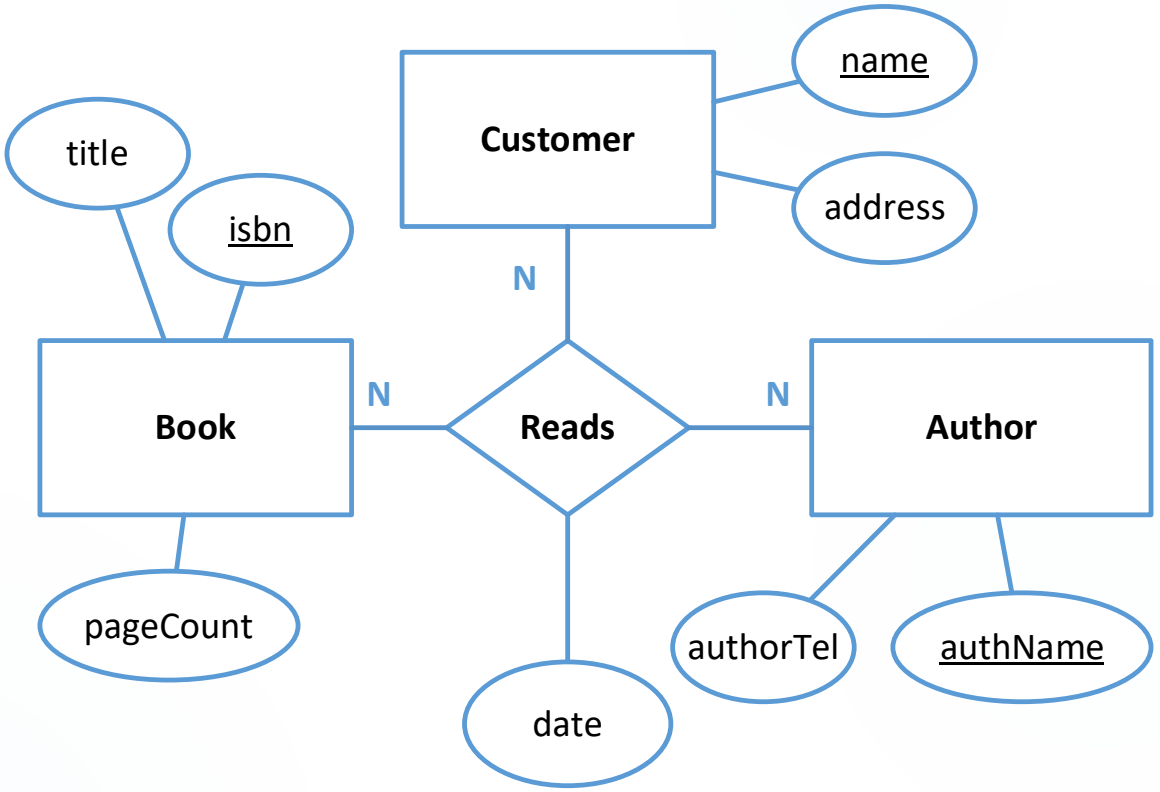



```
Player(teamName,number,name)

Team(teamName,manager)
```

Steps to map ER model to relational schema

- Step 6 : Each ternary relationship type becomes a separate relation (ex : 3-ary relationships)
  1. Add primary keys of the participating entity types to the relation. They become foreign keys.
  2. Add all the descriptive attributes of the relationship type to the relation.
  3. The primary key of the relation consists of the three participating entity types when the cardinality is many-to-may. When the cardinality is not many-to-many, the primary key of the relation consists of at least two foreign keys, where all the foreign keys coming from many side entity types much be included in the primary key of the relation.

Example:



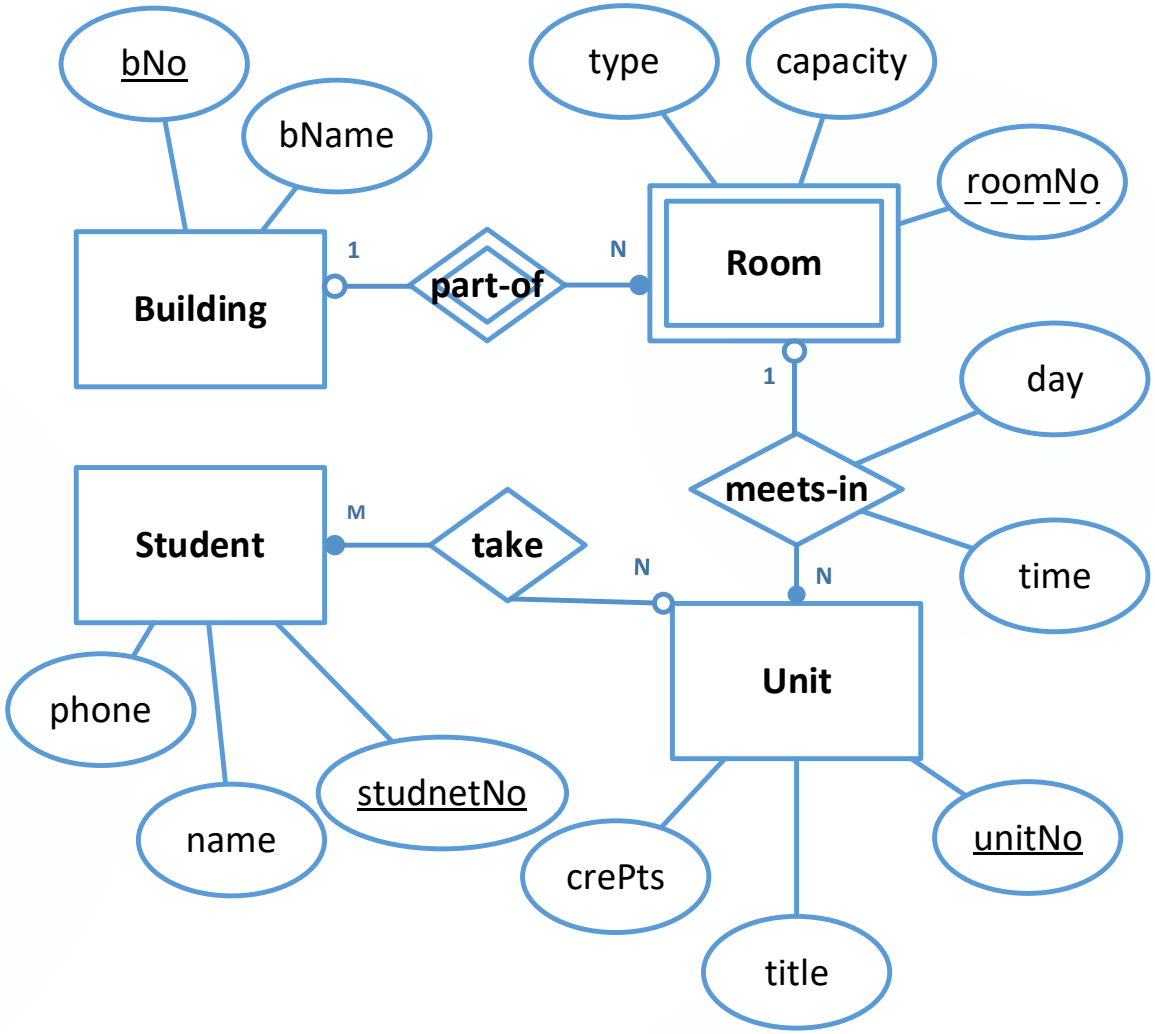

```
Book(isbn,title,pageCount)
Customer(name,address)
Author(authNme, authorTel)
Reads(isbn,name,authName,date)
```

Relational mapping : more tips

- In the examples above, we have mapped tables considering one type of relationship at a time. However, when translating the ER to relational schema, ER diagram as a whole needs to be considered.
- Multi-values attributes:
  - Relational model does not support multi-values attributes in the concerned relation.
  - It can be handled either by:
    a) creating a separate relation to keep the multi-valued attributes and link it to the other relation via a foreign key.
    b) keep values of the multi-valued attribute in multiple tuples of the relation.
    c) add fixed number of additional attributes (if only the number of values known) to the relation to keep multiple values.
  - We will look at this in another example later (this lecture)
- Composite attributes:
  - Each composite attribute become attributes of the relation.
- Generalization and mapping of is-A relationships to relational schema is covered in extended ER models(EER). (We may do this as an additional practical in a later practical class)
- Constraints such as primary key, foreign key, participation constraints, cardinality will be further incorporated to the schema when it is converted to SQL statements.

Exercise:

- Convert the ER diagram from the previous class example to relational schema.



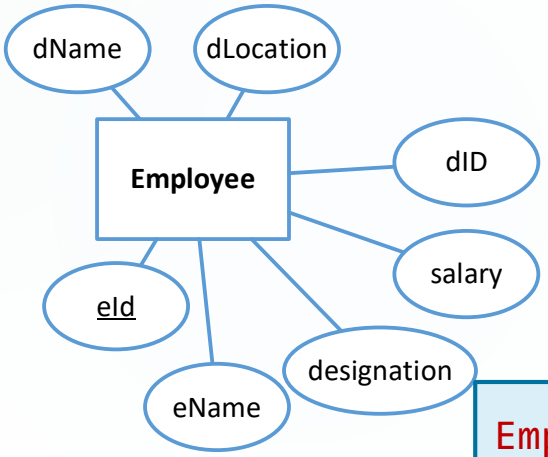
Revising the relational schema

- The above mapping process resulted in a set of relations, which are the candidate relations to be used in actual database implementation.
- When entities, attributes and relationships among them are identified to capture the concerned scenario and ER model is created **properly**, and the ER to relational mapping is performed using **standard process**, resulting relational schema would be in a standard form which reduces duplicate data and avoid anomalies.
- This standard form is called “normalized” database model.
- The relations in normalized form aims at making sure redundant data are avoided and maintaining integrity of data when update/ delete data (i.e.anomalies are reduced) in related relations.
- Generally, an ER model will lead to a normalized database model, however there can be issues which we have not seen in the ER modelling or mapping stages.

Why schema refinement is required?

- In complex scenarios, identifying the attributes and keys to be kept in an entity might be not done properly.
- Sometimes, without following the ER design to relational schema approach,relations would be defined.
- Therefore, after identifying candidate relations ,(a) we may have to check whether the relations are in a proper form, and (b) if any weaknesses are identified, a schema refinement would be required.

Example:



ER and the resulting schema:

Employee(eId,eName,designation,salary,dName,dLoc,dId)

- Sample table with data for the relation Employee(eId,eName,designation,salary,dName,dLoc,dId)

eId	eName	designation	salary	dID	dName	dLocation
100	John	Accountant	….	10	Administration	B101
101	Peter	Receptionist	…	10	Administration	B101
102	Alan	Designer	…	12	Design	B102
103	Lin	DBA	…	15	Development	B200
104	Elena	Programmer		15	Development	B200
105	Peter	Programmer		12	Design	B102
106	Paul	Programmer		12	Design	B102

Anomalies in relational schema

eId	eName	designation	salary	dID	dName	dLocation
100	John	Accountant	….	10	Administration	B101
101	Peter	Receptionist	…	10	Administration	B101

- Insertion Anomaly
  - *E.g. Inserting a new employee to the Employee table*
    - Department information is repeated
  - *E.g. Inserting a department with no employees*
    - Impossible since eID cannot be null (key)
- Deletion Anomaly
  - *E.g. Deleting the last employee from the department*
    - This will lead to loosing information about the department
- Update Anomaly
  - *E.g. Updating the department's location*
    - needs to be done for all employees working for that department

Functional dependencies

- In order to reduce above issues, `Employee` relation can be decomposed.
- However, random decomposition may still resulted in issues.
- A mathematical form exists for identifying keys and then use them to decompose tables(refining the schema to reduce anomalies and redundancies) to convert a relation to a normalized relation.
- This systematic process of normalization can be used to verify whether the relations we have derived are normalized or not as well.
- There are several stages of normalization applicable to databases.
- Therefore, we can use series of checks to test whether a schema meets some well defined criteria or not, to verify whether our relations are in a proper form.
- Concepts of Functional dependencies and use of them to identifying keys helps us in this regard.

Functional dependencies

- Functional dependencies are concerning about the dependency between attributes of a relation.
- In real world situations, knowing about the domain (business rules) is important to know what sort of dependencies are between attributes.
- Functional dependency is a relationship that exists when one attribute uniquely determine another attribute.
- Functional dependencies are a mathematical concept

Attribute B is functionally dependent on attribute A if it always holds true that each value of attribute A has exactly one value of attribute B associated with it.

Functional dependencies

Example :

- Consider the relation `Student(studentID, studentName)`
- we know that studentID is unique for each student
- Therefore,

`studentName` is functionally dependent on `studentID`

(for each studentID there is only one studentName)

- For each `studentName`, there can be more than one `studentID`.
- Therefore,

`studentID` is not functionally dependent of `studentName`

The above functional dependency can be shown as :

`studentID` → `studentName`

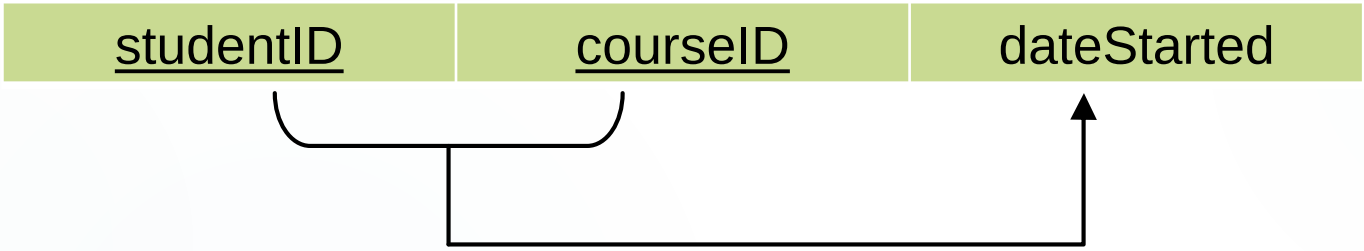
- there are other notations as well

`studentID` identifies `studentName`

`studentID` determines `studentName`

Functional dependencies

- Dependency can occur between one attribute and another attribute or between combination of attributes.
- When a key is a composite key, this situation arise.
- Example:



(studentID,courseID)→ dateStarted

It is possible to have FDs like:

AB→ C

A→ DE , meaning A→ D and A→ E

Keys and Functional dependencies

- A key constraint is a special case of a FD X → Y where the attributes in the key play the role of X and the set of all attributes play the role of Y.
- Normalization process analyzes schemas based on keys and on the functional dependencies among their attributes.
- Thus, to start normalization it is essential to find keys of a given relation.
- Attribute closure of an attribute set X, denoted by X^+ can be defined as a set of all attributes which can be functionally determined from it.
- If X^+ = all attributes, then X is a key.

Keys and Functional dependencies

- Attribute closure could be computed using Armstrong Axioms.

Let X, Y, Z are sets of attributes:

- *Reflexivity: If $X \subseteq Y$, then $Y \rightarrow X$*
- *Augmentation: If $X \rightarrow Y$, then $XZ \rightarrow YZ$ for any Z*
- *Transitivity: If $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$*

- Couple of additional rules (that follow from Armstrong Axioms) are also useful:
  - *Union*: If $X \rightarrow Y$ and $X \rightarrow Z$, then $X \rightarrow YZ$
  - *Decomposition*: If $X \rightarrow YZ$, then $X \rightarrow Y$ and $X \rightarrow Z$

- *If you wish to know more about FDs with a mathematical background, look at the lecture-2 slides from last year (available in Blackboard)*

Example:

- Consider a relation R (A, B, C, D), with the following set of functional dependencies over R:

  A → B, B → C, B → D

- Find what attributes would become a key?

Solution:

A → A (reflexivity rule)

A → B (given)

A → B and B → C then A → C (transitivity)

A → B and B → D then A → D (transitivity)

A → ABCD (Union rule)

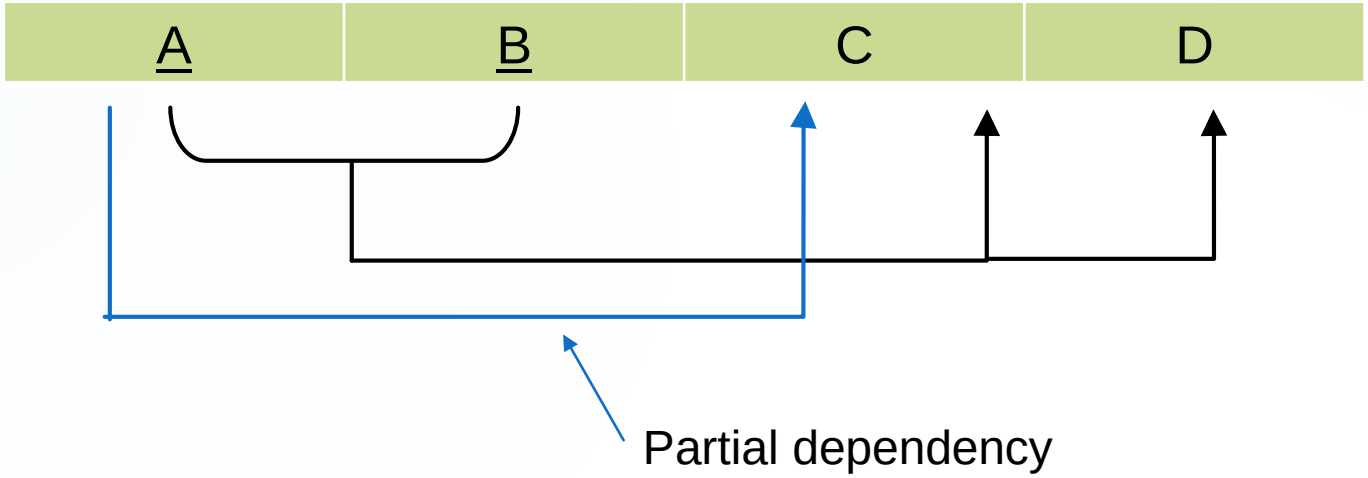
A+={ABCD}

We can find that, B+= {BCD} , C+ = {C} , D+={D}

Therefore A is the key.

Full functional dependency

- **A functional dependency X →Y is a full functional dependency if removal of any attribute A from X means that the dependency does not hold any more.**
- For example, in a relation R (A,B,C,D) where AB → {ABCD}, if A → C , A → C is a partial dependency (not fully functional dependent).



Definitions useful in normalization

- **Superkey:** super set of a key; set of attributes S in relation R such that no two distinct tuples t_1 and t_2 will have $t_1[S] = t_2[S]$

  R(A,B,C,D) then {A,B}, {A,B,C} {A,B,C,D} are superkeys of R
- **Key:** A key is a superkey with the additional property that removal of any attributes from the key will not satisfy the key condition.
- **Candidate Key:** Each key of a relation is called a candidate key.
- **Primary Key:** A candidate key is chosen to be the primary key.
- **Prime Attribute or key Attribute:** an attribute which is a member of a candidate key.
- **Nonprime Attribute or non-key attribute:** An attribute which is not prime.

Normal forms

- A relational schema is said to be in a normal form if it adhere to certain criteria.
- There are several normal forms:
  - **1st normal form (1NF)**
  - **2nd normal form (2NF)**
  - **3rd normal form (3NF)**
  - **Boyce Codd normal form (BCNF)**
  - 4th normal from
  - …
- Test for each normal form is performed in a top-down fashion (1NF first and then 2NF etc.).
- When a relation is said to be in a normal form, it refers to the highest normal form condition it meets.

1NF

- A relation R is in first normal form (1NF) if domains of all attributes in the relation are atomic (simple & indivisible) and the value of any attribute in a tuple is a single value from its domain.
- In other words, if the relation is in 1NF each attribute is non decomposable and is functionally dependent on the key.
- 1NF is considered to be part of the formal definition of a relation in the relational model since the relational model allows only atomic values and disallows multivalued attributes and composite attributes.
- All normal forms other than 1NF are additions to the relational model to create better, stable data representations avoiding redundancies.

Example : 1NF

studentID	studentName	facId	facName	studentTel
1001	David Smith	10	Engineering	0417700088, 0412345678
1002	Elena May	15	Humanities	0810567793
1003	John White	20	Medical	0414577781

Above relation is not in 1NF as telephone number is having multiple values
We can convert the above to 1NF in different ways

We can decompose the relation:

studenetID	studentName	facId	factName
1001	David Smith	10	Engineering
1002	Elena May	15	Humanities
1003	John White	20	Medical

studnetID	studentTel
1001	0417700088
1001	0412345678
1002	0810567793
1003	0414577781

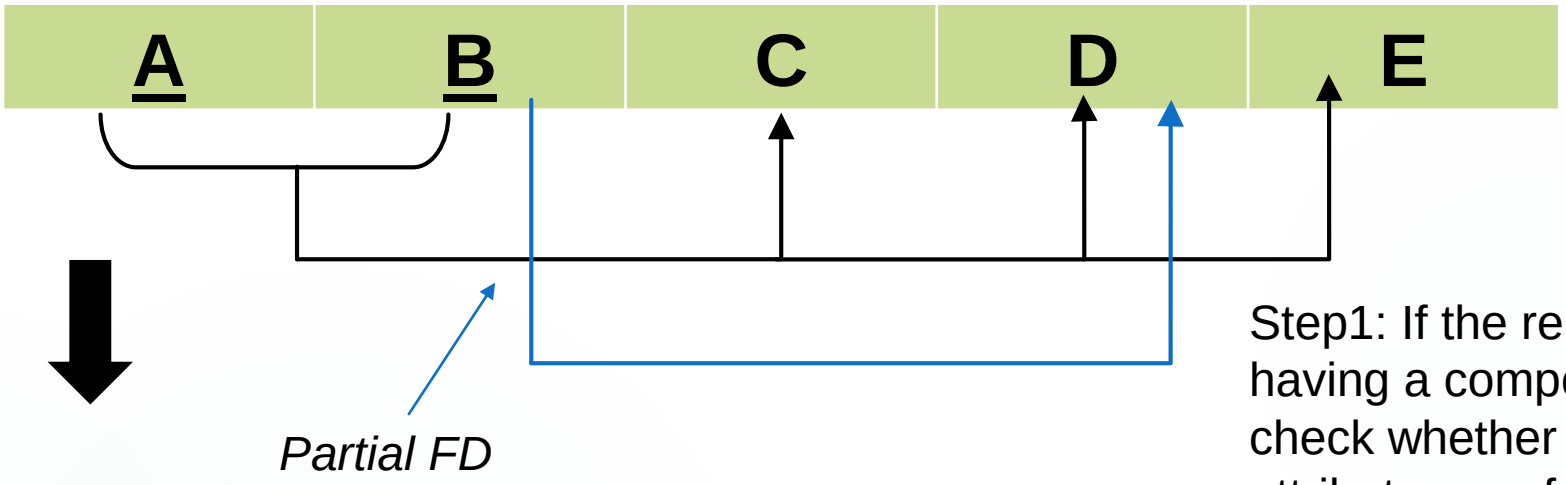
- Alternatively, we can add separate tuple for each value of the multi-valued attribute and change the key to a composite key
- However, this introduces redundant data

studentID	studentName	facId	facName	studentTel
1001	David Smith	10	Engineering	0417700088
1001	David Smith	10	Engineering	0412345678
1002	Elena May	15	Humanities	0810567793
1003	John White	20	Medical	0414577781

2NF

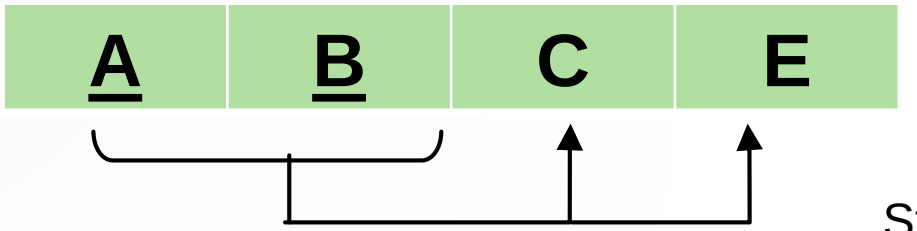
- Second normal form (2NF) is based on the concept of full functional dependency.
- A relation R is in second normal form (2NF) if every nonprime attribute A in R is not partially dependent on any key of R.
- Nonprime attribute : an attribute that does not occur in any candidate key
- A relation in 2NF is already in the 1NF and fulfilling additional requirements.
- We have to check only composite keys to verify 2NF.
- **Relations without composite primary key are already in 2NF**.
- To normalize the relation to 2NF decomposition is performed as follows:

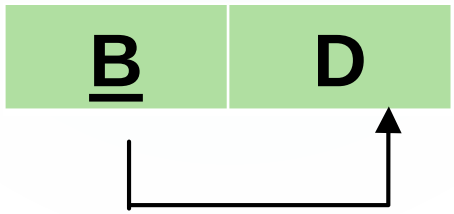
Example: 2NF



Step1: If the relation is having a composite key, check whether non-key attributes are full FDs or partial FDs of any key

Step2: If the result of step 1 is ‘yes’, do decomposition



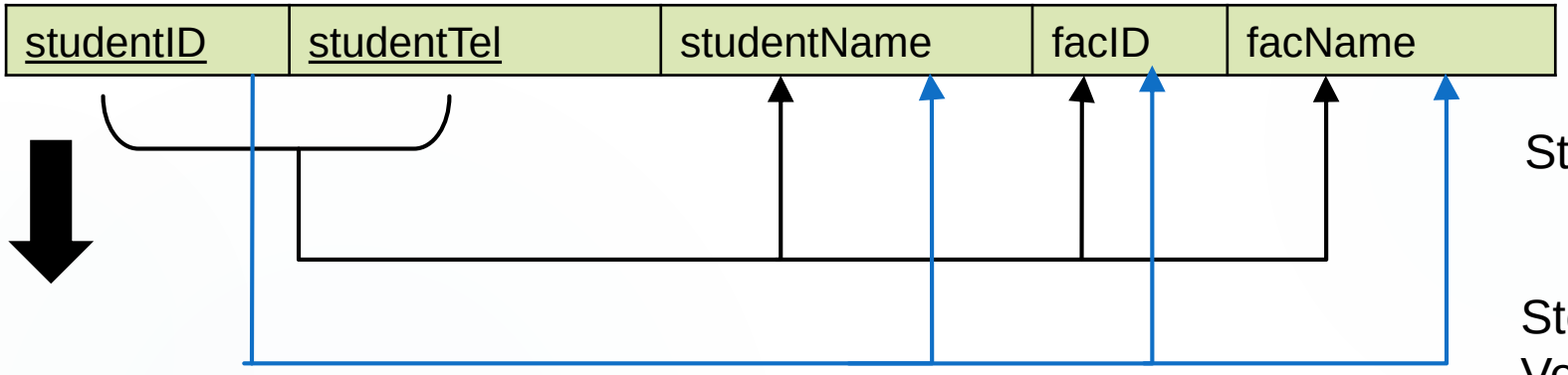


Step3: Verify decomposed relations’ non-key attributes are full FDs or partial FDs of any key of the relation.

Example: 2NF

studentID	studentTel	studentName	facId	facName
1001	**0417700088**	David Smith	10	Engineering
1001	**0412345678**	David Smith	10	Engineering
1002	0810567793	Elena May	15	Humanities
1003	0414577781	John White	20	Medical

1NF



Step1:

Step3: Verification

Step 2: Decomposition

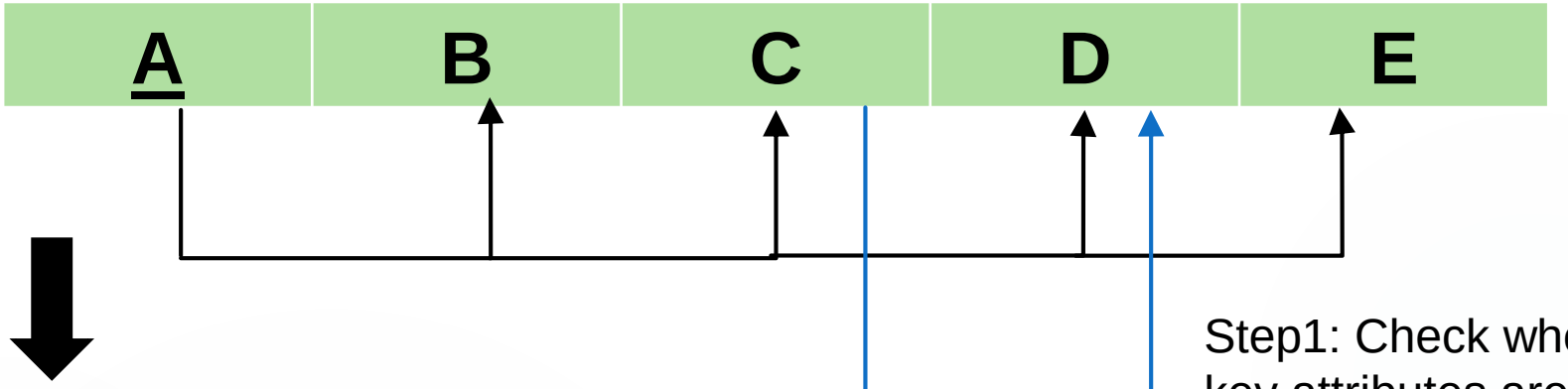
studentID	studentTel
1001	0417700088
1001	0412345678
1002	0810567793
1003	0414577781

studentID	studentName	facId	facName
1001	David Smith	10	Engineering
1001	David Smith	10	Engineering
1002	Elena May	15	Humanities
1003	John White	20	Medical

3NF

- Third normal form is based on the concept of transitive dependency.
- A relation R is in 3rd normal form (3NF) if R is in 2NF, and no nonprime attribute is transitively dependent on any key.
- *Note: A functional dependency X → Y in a relational schema R is a transitive dependency if there is a set of non-key attributes Z where both X → Z and Z →Y hold.*
- To normalize the relation to 2NF decomposition is performed as follows :

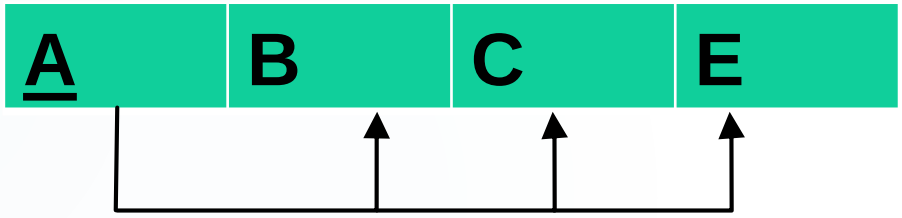
Example: 3NF



Step1: Check whether non-key attributes are transitively dependent on any key.

transitive dependency

Step2: If step 1 resulted in yes, do decomposition



Step3: Verify that decomposed relations' non-key attributes are transitively dependent on any key.

Example: 3NF

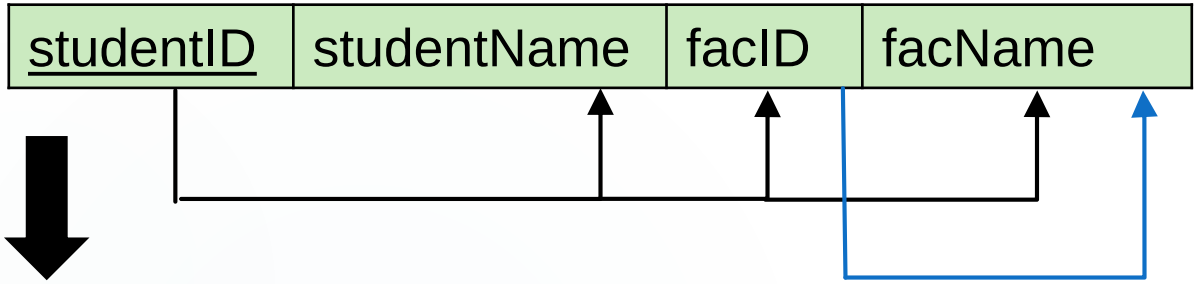
- Check whether the relation resulted from previous stage (shown below) is in 3NF or not
- If it is not in 3NF, convert it to 3NF

studentID	studentName	facId	facName
1001	David Smith	10	Engineering
1001	David Smith	10	Engineering
1002	Elena May	15	Humanities
1003	John White	20	Medical

Example: 3NF

studentID	studentName	facId	facName
1001	David Smith	10	Engineering
1002	Elena May	15	Humanities
1003	John White	20	Medical

2NF



Step1: Check whether non-key attributes are transitively dependent on any key.

Step2: If the result of step 1 is 'yes', do decomposition

studentID	studentName	facId
1001	David Smith	10
1002	Elena May	15
1003	John White	20

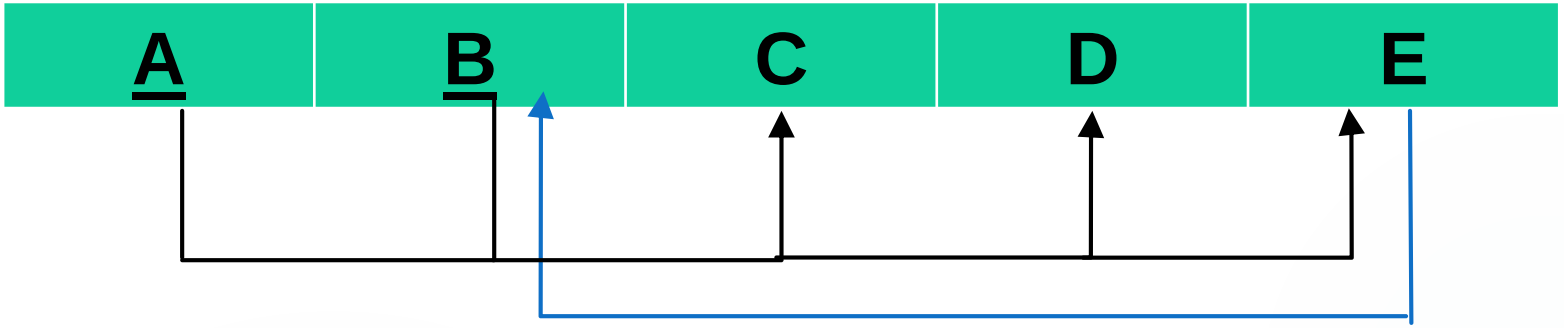
facId	facName
10	Engineering
15	Humanities
20	Medical

Step3: verify decomposed relations

BCNF

- A relation schema R is in Boyce-Codd Normal Form If and only if for every nontrivial functional dependency X→A hold in R, X is a superkey of R.
- In other words, A relation schema R is in Boyce-Codd Normal Form If and only if every non-trivial FD satisfied by R is determined by some super-key of R.
  - Remember: *nontrivial* means *A* is not a member of set *X*.
  - Remember, a *superkey* is any superset of a key
- Mostly occur with overlapping candidate keys
- Decomposition into BCNF:
- Consider relation R with FDs F. If X → Y violates BCNF, decompose R

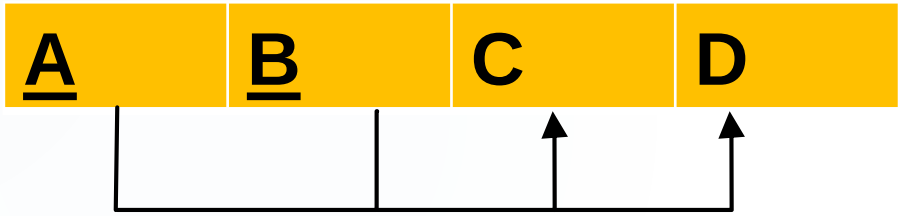
Example: BCNF

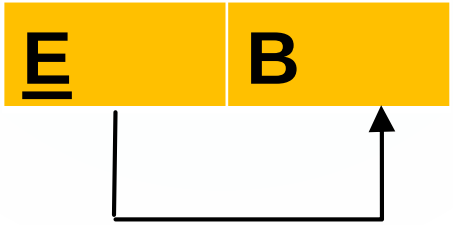


Step2: If step 1 resulted in yes, do decomposition

FD is non-trivial,
E is not a super key

Step1: Check whether there are non-trivial FDs not *'arrows out of super keys' type*





Step3: Verify that decomposed relations' non-trivial FDs are from a superkey.

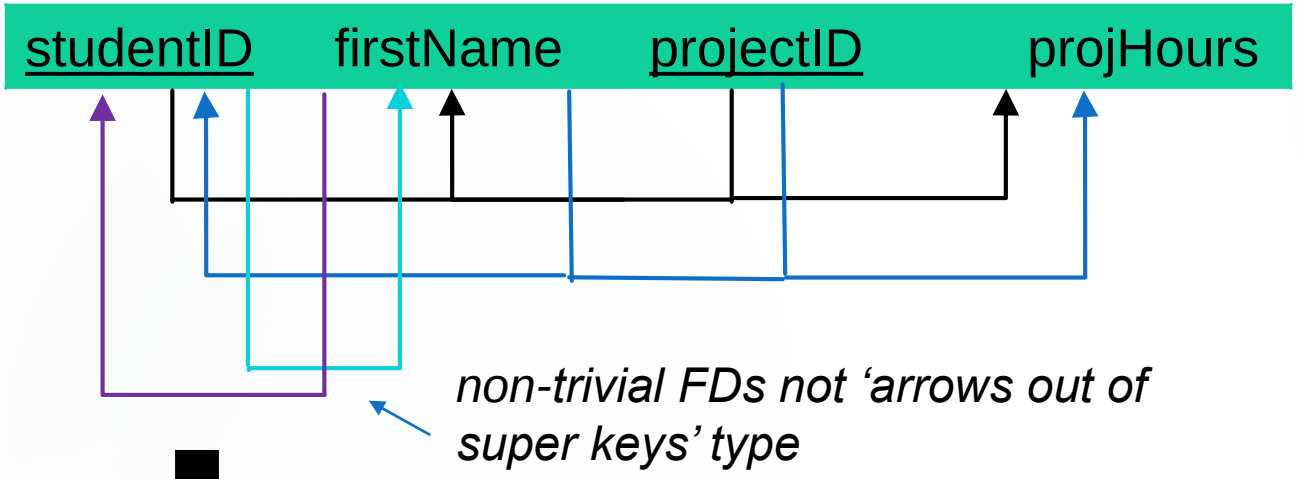
Example: BCNF

studentID	firstName	projectID	projHours
1001	David	A10	10
1001	David	A15	20
1002	Elena	B1015	6
1003	John	A20	24

Revised Student table with projectID and projHour attributes

Candidate keys :
1.{studentID,projID}
2.{firstName,projID}
selected primary key :
{studentID,projID}

Two candidate keys are overlapping as projID is common for both



non-trivial FDs not 'arrows out of super keys' type

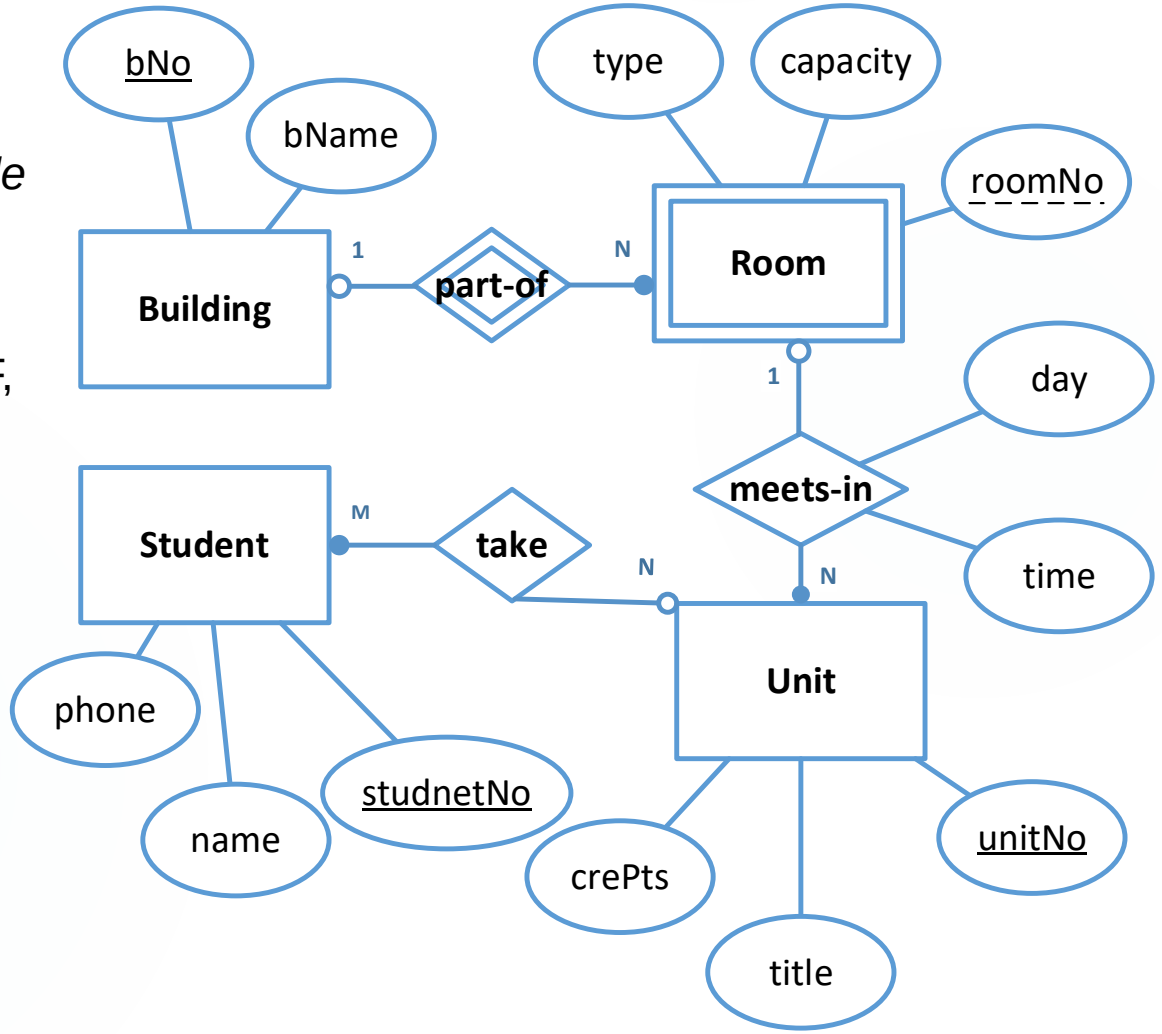
decomposition

studentID	firstName
1001	David
1002	Elena
1003	John

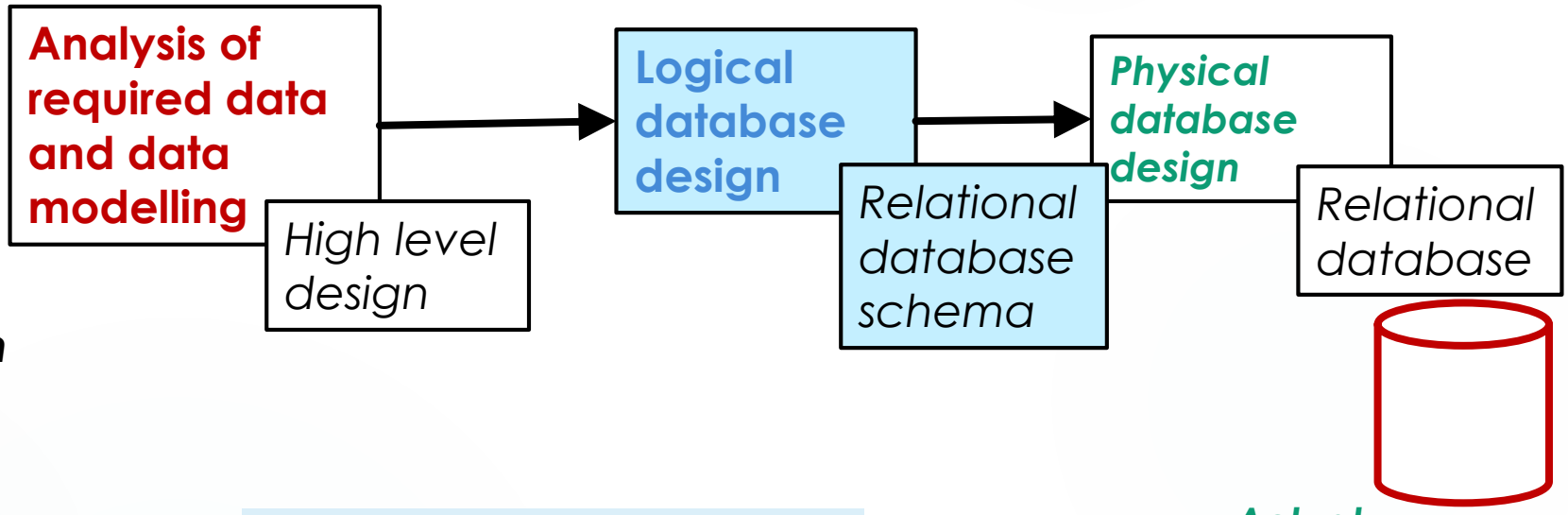
studentID	projectID	projHours
1001	A10	10
1001	A15	20
1002	B1015	6
1003	A20	24

Exercise:

- *Previous exercise: Convert the ER diagram from the previous class example to relational schema.*
- Check whether the relational schema you have derived is in 1NF, 2NF, 3NF or BCNF.



Real world to database



- We have looked at the logical database design stage in this lecture.
- The relational schema derived in this stage can be implemented using SQL statements, as per the DBMS used.

Summary

- ER model can be translated to relational schema following well established guidelines to create a schema avoiding redundancies and anomalies.
- Relational schema avoiding anomalies and redundancies is called a normalized relational schema.
- However, as our designs are not perfect, database would affected by update, delete anomalies and duplicate of data.
- Refinement of the relational schema assists in addressing these issues.
- Functional dependencies are useful in identifying keys of a relation and thereby to check whether relational schema is adhering to different normal forms.
- There are several normal forms, 1NF,2NF, 3NF, BCNF …
- Any schema adhering to relational model basic concepts would be in 1NF

Happy Database systems

Next week : schema implementation, working with multiple tables

Practical worksheet – 5 (Relational mapping and normalization)