



Prifysgol Cymru
Y Drindod Dewi Sant
University of Wales
Trinity Saint David



MAPPING ERD TO TABLE AND FUNCTIONAL DEPENDENCY

17 October 2023

Dr.Suresh Shanmugasundaram

So far...

1. Relational Model
2. Entity Relationship Model
 - ERD
 - Representation of elements (entities, attributes, relationships)

Learning Objectives



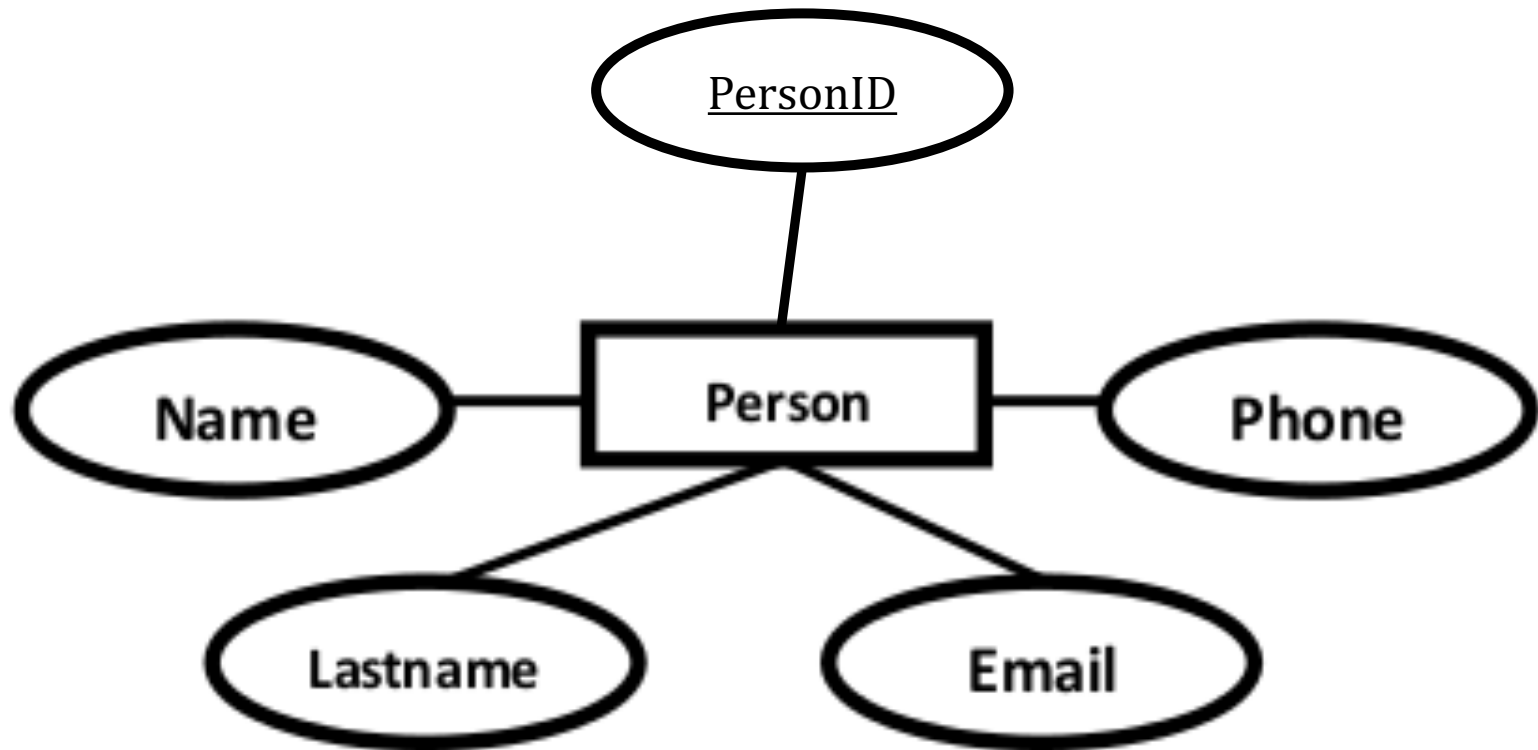
- ❑ ERD to Relational Table – Mapping Algorithm
- ❑ Functional Dependency
- ❑ Normalisation

Recap Quiz



<https://forms.office.com/e/DfFEqu59B4>

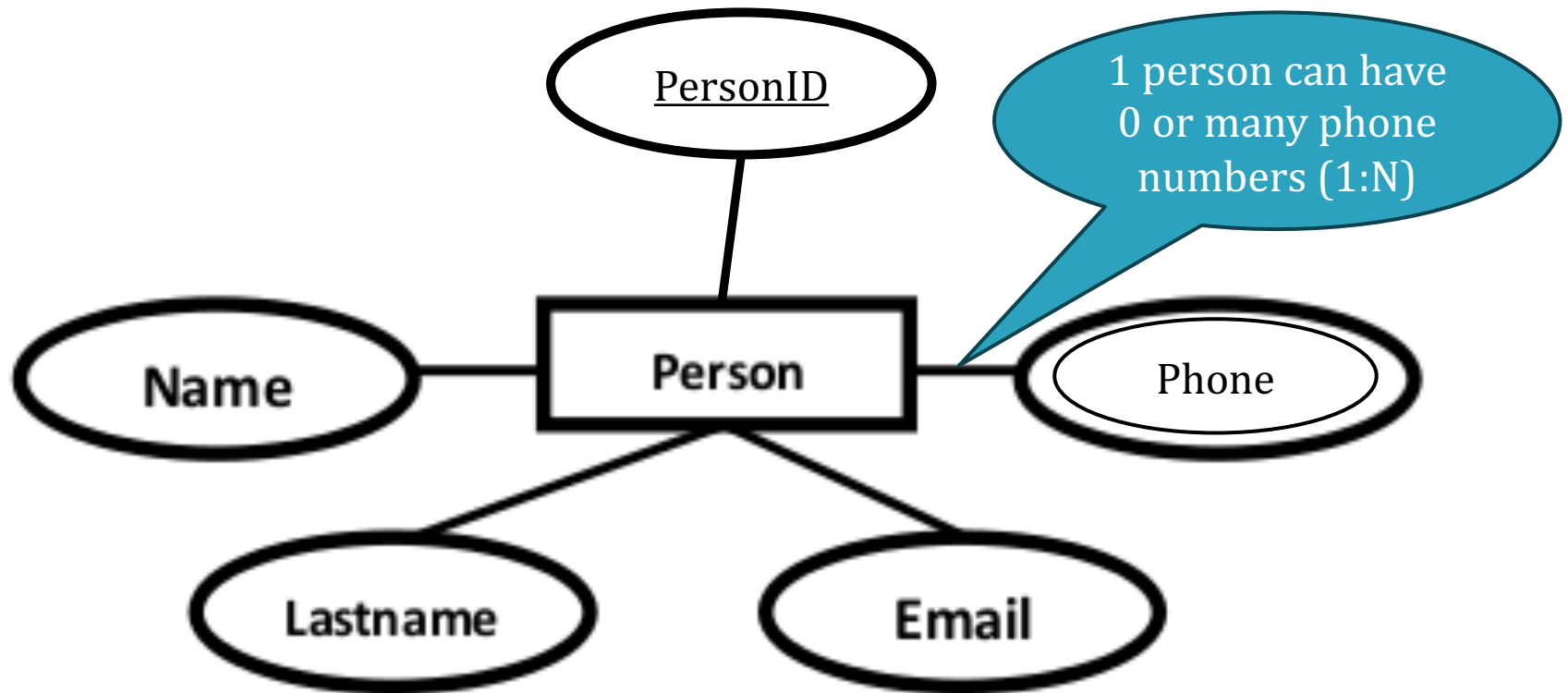
Mapping Algorithm - Entities and Simple Attributes



Persons(personid , name, lastname, email)

What happens to Phone-multivalued attribute?

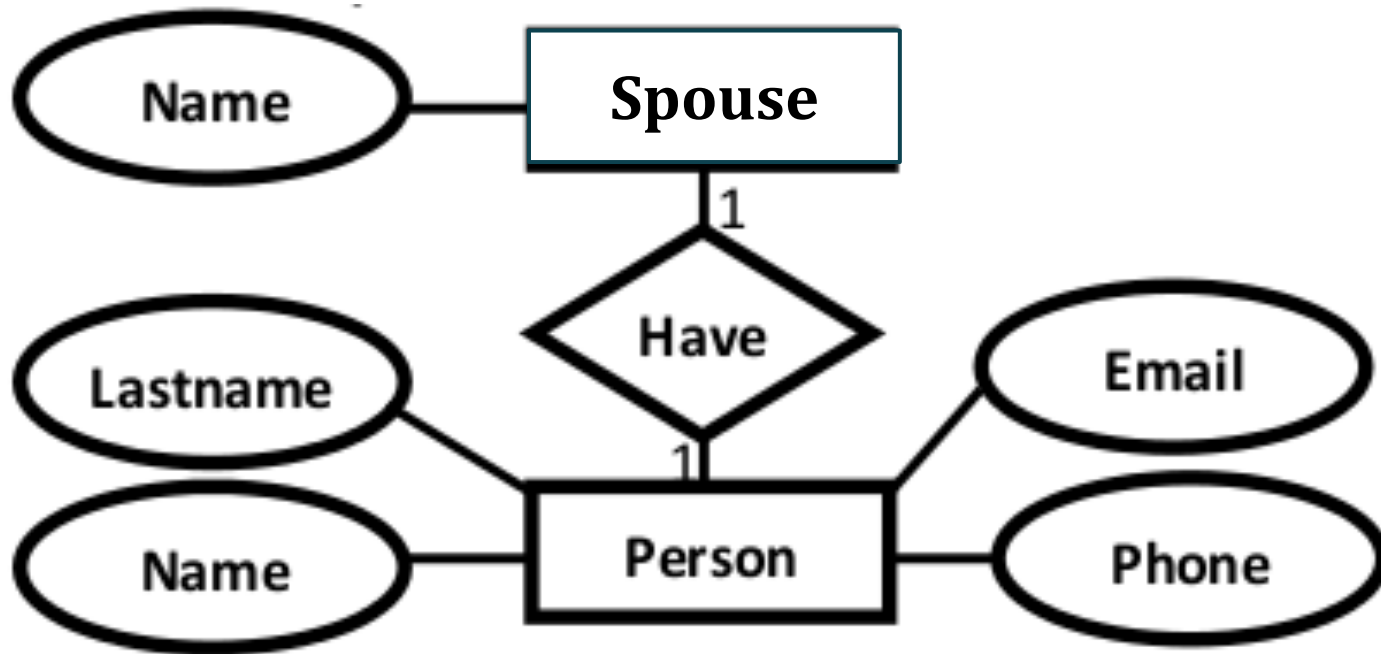
Mapping Algorithm – Multivalued attribute



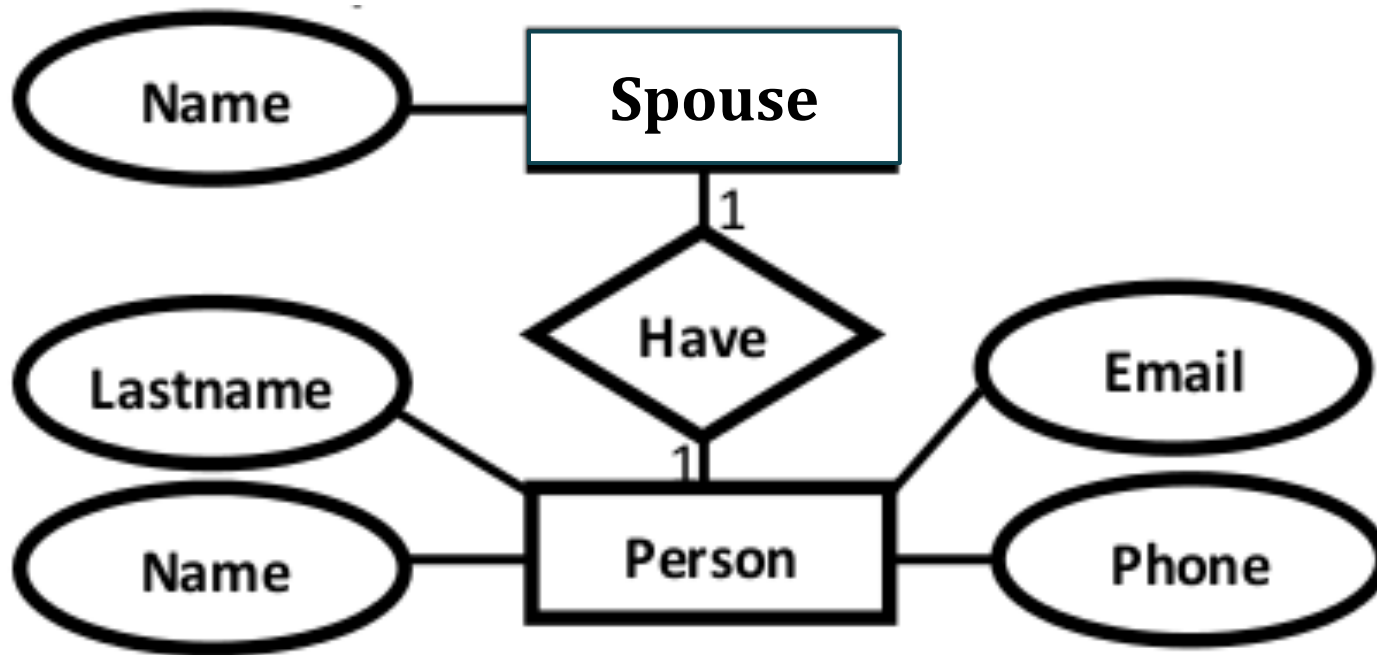
Persons(personid , name, lastname, email)

Phones (phoneid , **personid**, phone)

Mapping Algorithm – 1:1 Relationships



Mapping Algorithm – 1:1 Relationships

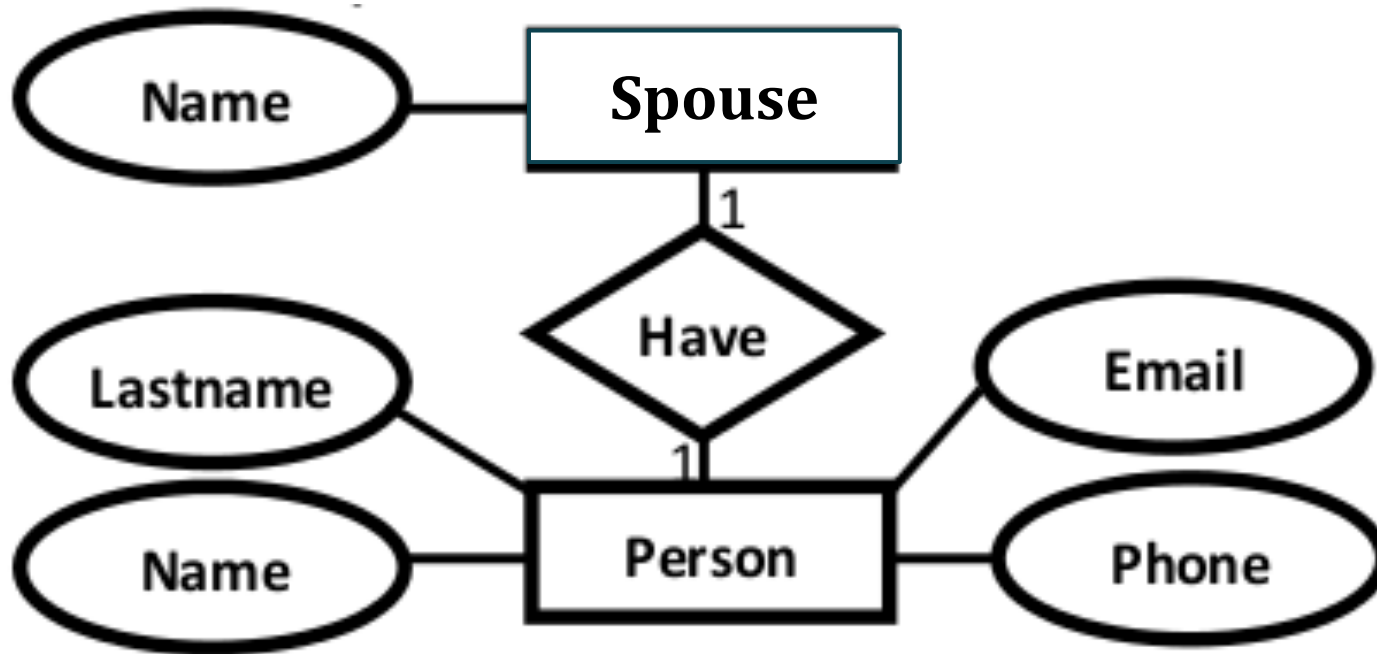


Persons(personid{PK}, name, lastname, email, spouseid{FK})

Spouse (spouseid{PK}, **name**)

Can you think of an alternative approach?

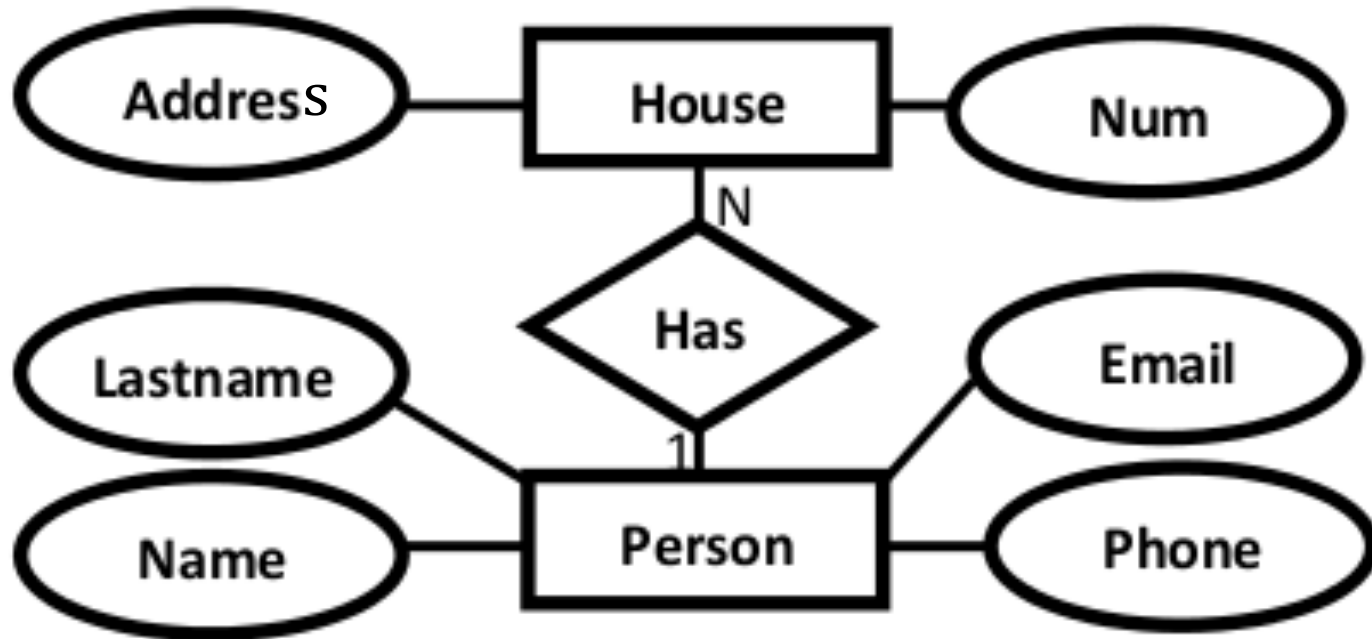
Mapping Algorithm – 1:1 Relationships



Persons(personid{PK}, name, lastname, email)

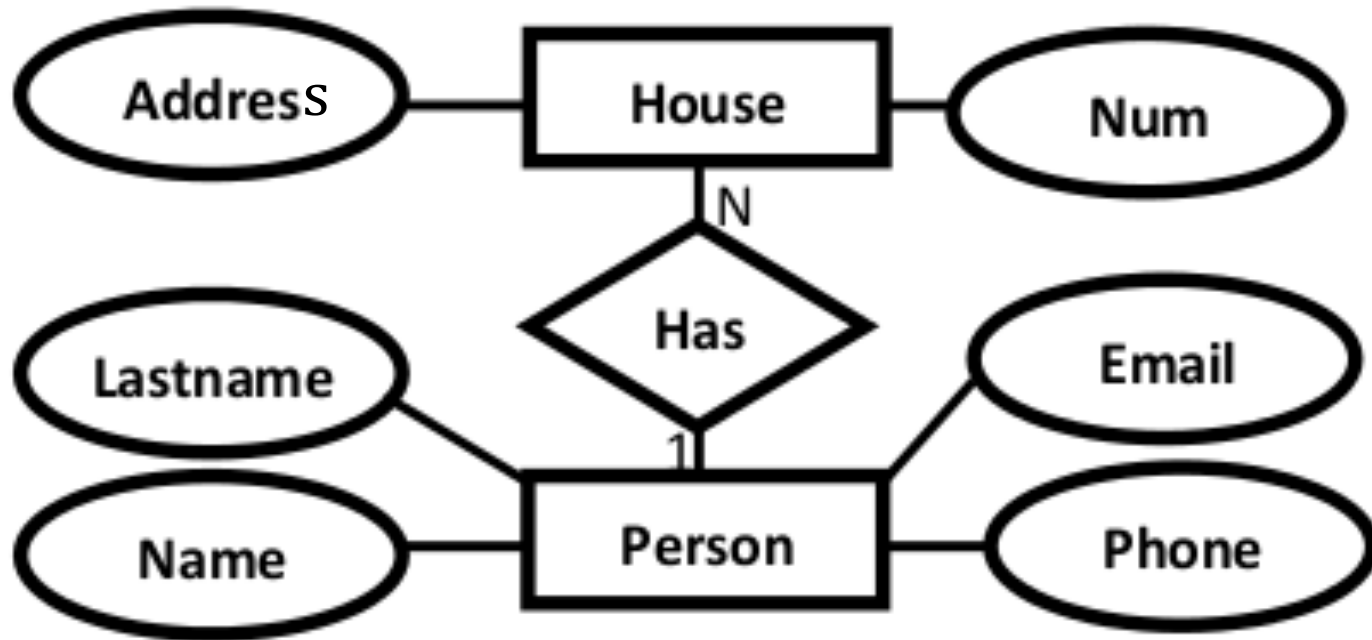
Spouse (spouseid{PK}, **name**, **personid**{FK})

Mapping Algorithm – 1:N Relationships



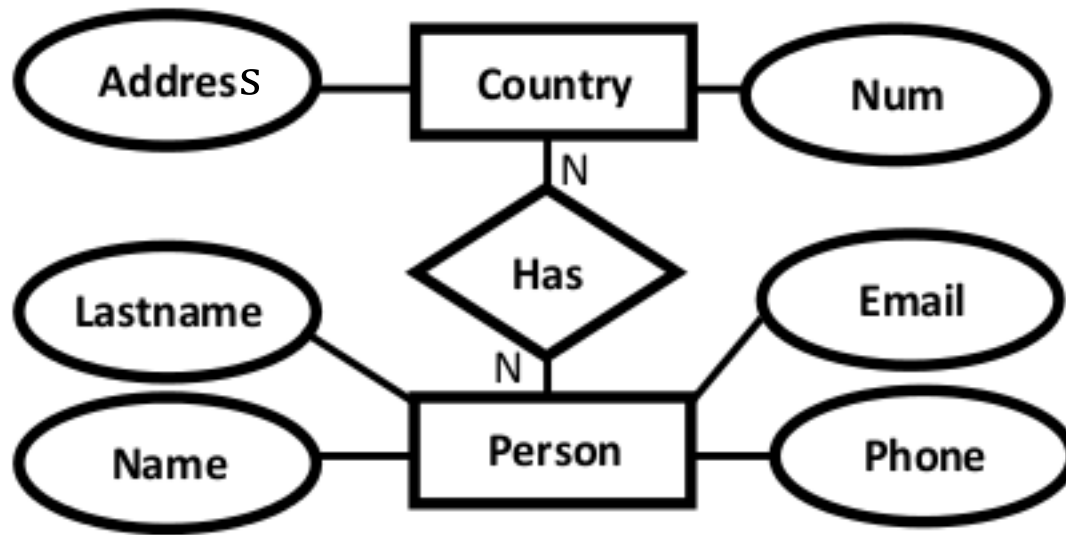
How is this converted to?
Give a try...

Mapping Algorithm – 1:N Relationships



Persons(personid , name, lastname, email)
House (houseid , num , address, *personid*)

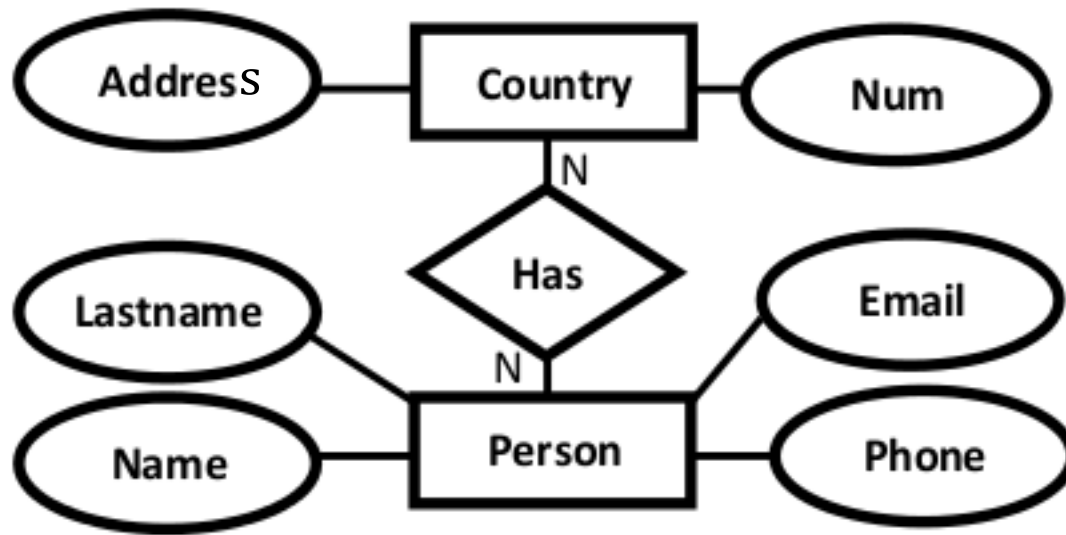
Mapping Algorithm – N:N Relationships



How is this converted?

Give a try...

Mapping Algorithm – N:N Relationships

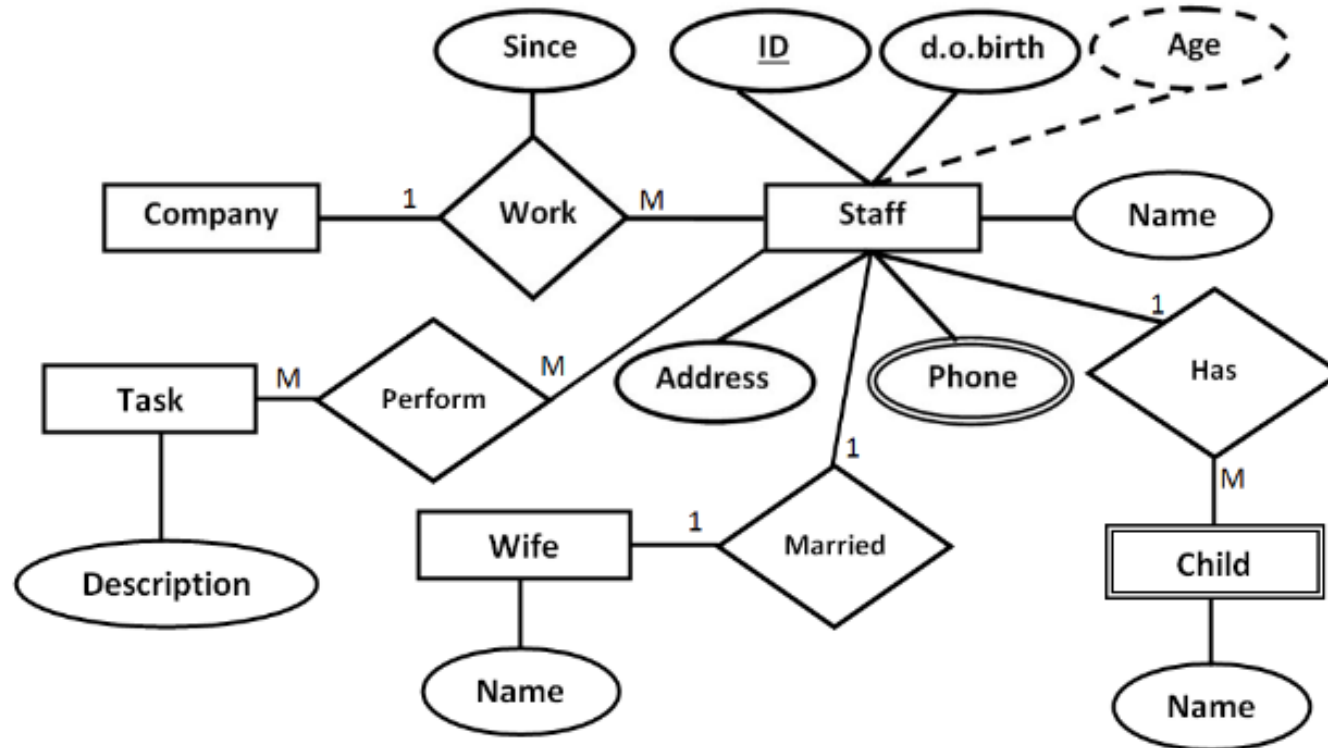


Persons(personid , name, lastname, email)

Countries (countryid , name, code)

HasRelat (hasrelatid , *personid* , *countryid*)

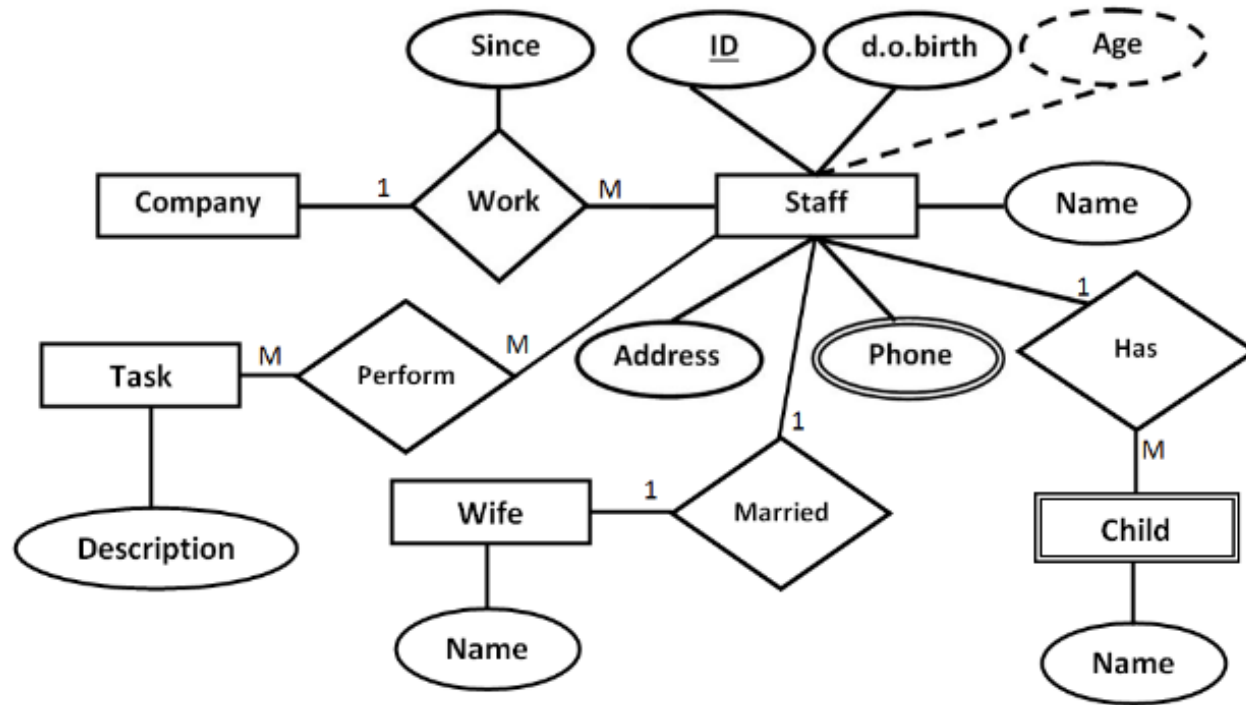
Mapping Algorithm – Activity 1



<https://forms.office.com/e/iLvRbzBiXQ>



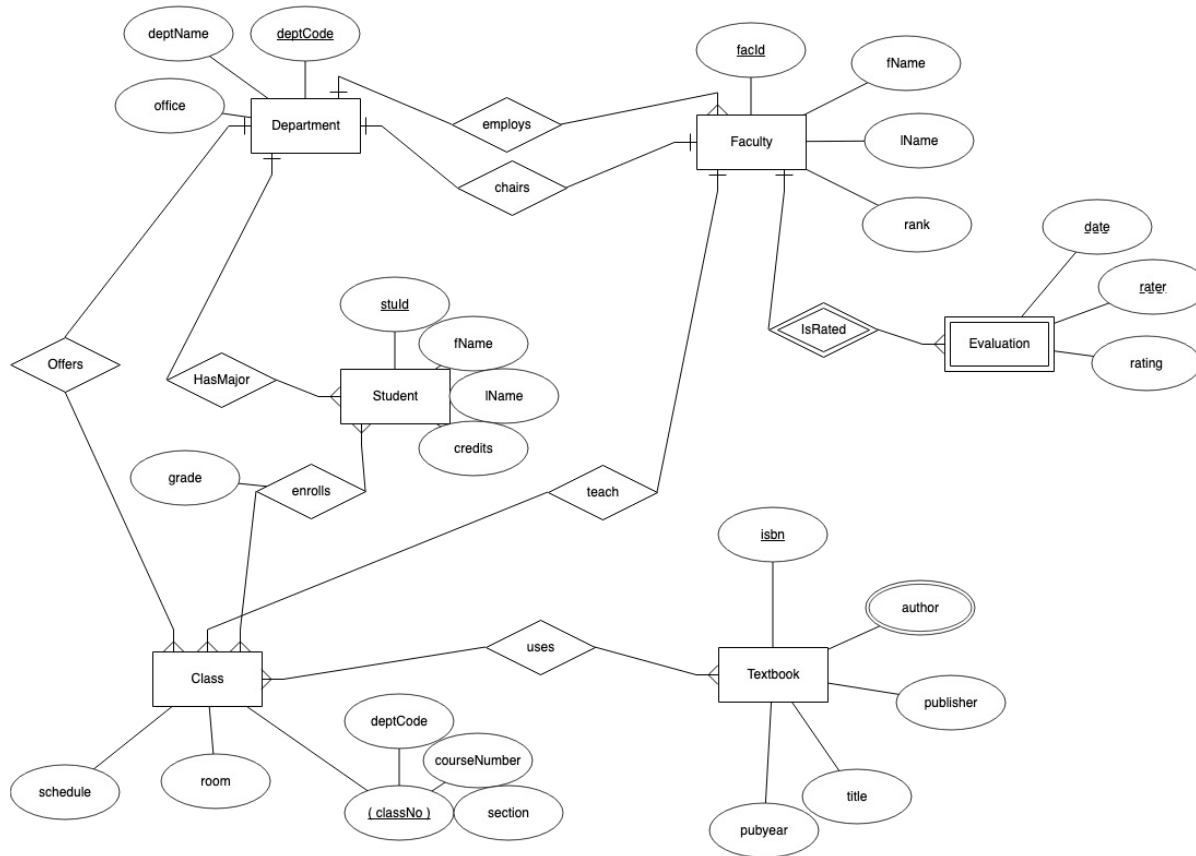
Mapping Algorithm – Activity 1



<https://forms.office.com/e/iLvRbzBiXQ>

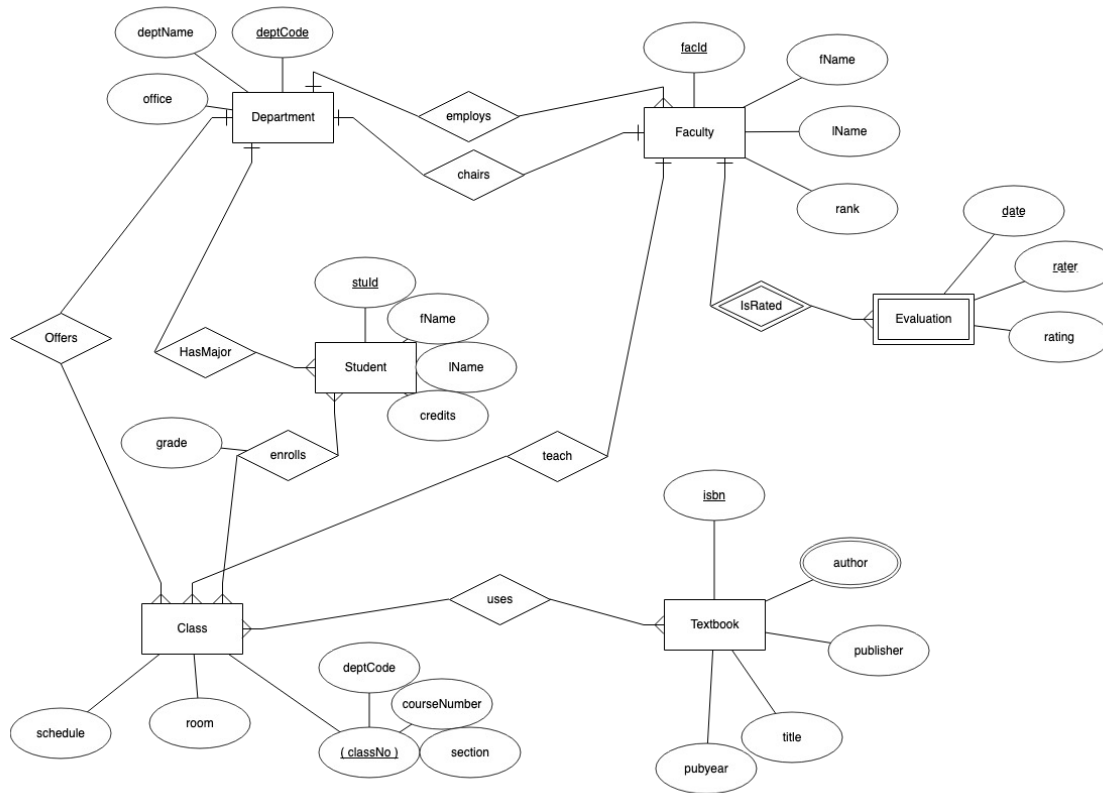
Company(CompanyID , name , address)
Staff(StaffID , dob , address , **WifeID**)
Child(ChildID , name , **StaffID**)
Wife (WifeID , name)
Phone(PhoneID , phoneNumber , **StaffID**)
Task (TaskID , description)
Work(WorkID , **CompanyID** , **StaffID** , since)
Perform(PerformID , **StaffID** , **TaskID**)

Mapping Algorithm – Activity 2



<https://forms.office.com/e/WQ3B6TQwy3>

Mapping Algorithm – Activity 2



- Student (Student_ID, First_Name, Last_Name, Date_of_Birth, Gender)
- Course (Course_ID, Course_Name)
- Teacher (Teacher_ID, First_Name, Last_Name, Date_of_Birth, Gender)
- Enrollment (Enrollment_ID, Student_ID, Course_ID)
- Department (Department_ID, Department_Name)

ERD to Relational Tables: Mapping Algorithm

Correspondence between ER and Relational Models

ER MODEL

Entity type

1:1 or 1:N relationship type

M:N relationship type

n -ary relationship type

Simple attribute

Composite attribute

Multivalued attribute

Value set

Key attribute

RELATIONAL MODEL

Entity relation

Foreign key (or *relationship* relation)

Relationship relation and *two* foreign keys

Relationship relation and n foreign keys

Attribute

Set of simple component attributes

Relation and foreign key

Domain

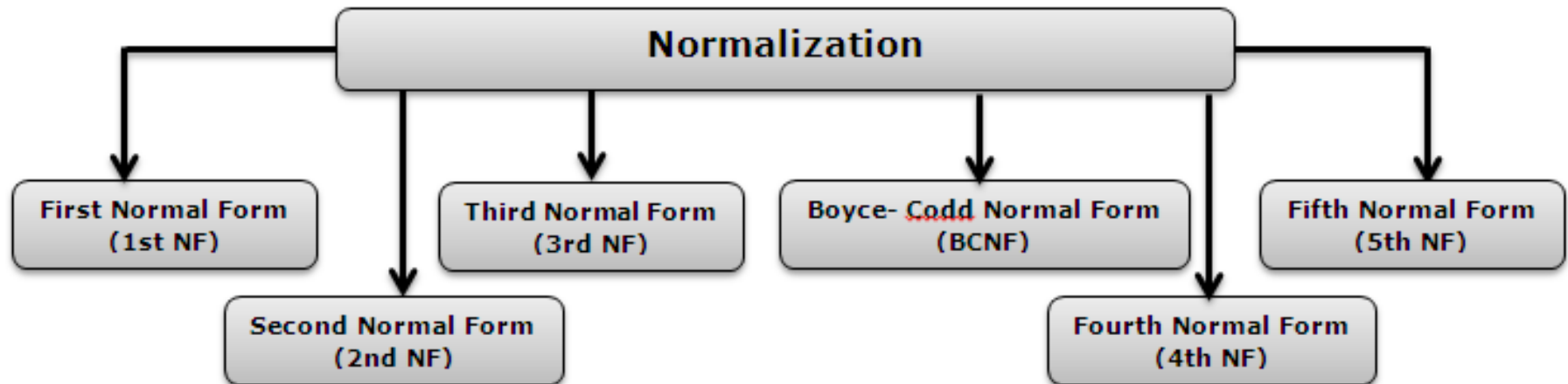
Primary (or secondary) key

Properties of Database Relations

- ❑ Each relation name is distinct from all other relation names in the relational schema.
- ❑ Each attribute has a distinct name.
- ❑ Each cell of a relation contains exactly one atomic (single) value.
- ❑ Values of an attribute all come from the same domain.
- ❑ Each tuple is distinct, there are no duplicate tuples.
- ❑ The order of attributes has no significance.
- ❑ The order of tuples has no significance theoretically.

What is Normalisation?

- **Normalisation** is a technique for producing a set of *suitable relations that support the data requirements* of an enterprise.



Normalisation

Characteristics of a Suitable Set of Relations

Characteristics of a suitable set of relations include:

- ❑ ***minimal number of attributes;***
- ❑ attributes with a close logical relationship (described as functional dependency)
- ❑ ***minimal redundancy*** with each attribute

The Purpose of Normalisation

- ❑ The main aim of a relational database design is to group attributes into relations to ***minimise data redundancy***.
 - ▣ Normalisation, helps to achieve this.
- ❑ Benefits of using a database that has a *suitable* set of relations is that the database will be:
 - ▣ easier for the user to access and maintain the data.
 - ▣ take up minimal storage space on the computer.

Problems of not properly normalised tables

- There are two main problems associated with a relation that is not appropriately normalised:
 - ▣ Data Redundancy
 - ▣ Update Anomalies for:
 - Insertions (adding new),
 - Deletions, and
 - Updates (modifications).

Data Redundancy and Update Anomalies

- ❑ What will be the primary key for this table?
- ❑ Can you spot potential problems with this table design?

CampusStaff Table

staffNo	staffName	position	salary	campusID	campusAddress
SW101	John Smith	Lecturer	30000	C002	Swansea IQ, SA1 8PH
SW102	Diane Lee	PVC	70000	C001	Carmarthen Campus, Carmarthen, SA31 3EP
SW103	Ann Beech	Lecturer	18000	C002	Swansea IQ, SA1 8PH
SW105	David Ford	Snr Lecturer	41000	C001	Carmarthen Campus, Carmarthen, SA31 3EP
SW106	Mary Howe	Dean, Humanities	55000	C003	Lampeter Campus Ceredigion SA48 7ED
SW107	Susan Brand	Receptionist	18000	C001	Swansea IQ, SA1 8PH

Redundancy

- ❑ Swansea IQ address is repeated for 3 staff members
- ❑ Carmarthen is repeated for 2 staff members

CampusStaff Table

staffNo	staffName	position	salary	campusID	campusAddress
SW101	John Smith	Lecturer	30000	C002	Swansea IQ, SA1 8PH
SW102	Diane Lee	PVC	70000	C001	Carmarthen Campus, Carmarthen, SA31 3EP
SW103	Ann Beech	Lecturer	18000	C002	Swansea IQ, SA1 8PH
SW105	David Ford	Snr Lecturer	41000	C001	Carmarthen Campus, Carmarthen, SA31 3EP
SW106	Mary Howe	Dean, Humanities	55000	C003	Lampeter Campus Ceredigion SA48 7ED
SW107	Susan Brand	Receptionist	18000	C001	Swansea IQ, SA1 8PH

Insertion (Add New) Anomalies

- There are two types of insertion anomalies.
 - ▣ **Problem 1:** To add the details of any new members of staff, we must include the details of the campus the staff is located at.

staffNo	staffName	position	salary	campusID	campusAddress
SW101	John Smith	Lecturer	30000	C002	Swansea IQ, SA1 8PH
SW102	Diane Lee	PVC	70000	C001	Carmarthen Campus, Carmarthen, SA31 3EP
SW103	Ann Beech	Lecturer	18000	C002	Swansea IQ, SA1 8PH
SW105	David Ford	Snr Lecturer	41000	C001	Carmarthen Campus, Carmarthen, SA31 3EP
SW106	Mary Howe	Dean, Humanities	55000	C003	Lampeter Campus Ceredigion SA48 7ED
SW107	Susan Brand	Receptionist	18000	C001	Swansea IQ, SA1 8PH

Insertion (Add New) Anomalies

- There are two types of insertion anomalies.
 - ▣ **Problem 2:** To add a new campus that currently has no staff assigned (there is no separate campus table), it is necessary to fill the staff related columns with NULL values.

staffNo	staffName	position	salary	campusID	campusAddress
SW101	John Smith	Lecturer	30000	C002	Swansea IQ, SA1 8PH
SW102	Diane Lee	PVC	70000	C001	Carmarthen Campus, Carmarthen, SA31 3EP
SW103	Ann Beech	Lecturer	18000	C002	Swansea IQ, SA1 8PH
SW105	David Ford	Snr Lecturer	41000	C001	Carmarthen Campus, Carmarthen, SA31 3EP
SW106	Mary Howe	Dean, Humanities	55000	C003	Lampeter Campus Ceredigion SA48 7ED
SW107	Susan Brand	Receptionist	18000	C001	Swansea IQ, SA1 8PH
NULL?	No staff yet				New campus

Update (Modification) Anomalies

- Suppose a campus moves to another address, we need to update the campus address details.

staffNo	staffName	position	salary	campusID	campusAddress
SW101	John Smith	Lecturer	30000	C002	Swansea IQ, SA1 8PH
SW102	Diane Lee	PVC	70000	C001	Carmarthen Campus, Carmarthen, SA31 3EP
SW103	Ann Beech	Lecturer	18000	C002	Swansea IQ, SA1 8PH
SW105	David Ford	Snr Lecturer	41000	C001	Carmarthen Campus, Carmarthen, SA31 3EP
SW106	Mary Howe	Dean, Humanities	55000	C003	Lampeter Campus Ceredigion SA48 7ED
SW107	Susan Brand	Receptionist	18000	C001	Swansea IQ, SA1 8PH

Deletion Anomalies

- ❑ If we delete a row from the CampusStaff table representing the last member of staff at the campus, the details about that campus are lost.

CampusStaff Table

staffNo	staffName	position	salary	campusID	campusAddress
SW101	John Smith	Lecturer	30000	C002	Swansea IQ, SA1 8PH
SW102	Diane Lee	PVC	70000	C001	Carmarthen Campus, Carmarthen, SA31 3EP
SW103	Ann Beech	Lecturer	18000	C002	Swansea IQ, SA1 8PH
SW105	David Ford	Snr Lecturer	41000	C001	Carmarthen Campus, Carmarthen, SA31 3EP
SW106	Mary Howe	Dean, Humanities	55000	C003	Lampeter Campus Ceredigion SA48 7ED
SW107	Susan Brand	Receptionist	18000	C001	Swansea IQ, SA1 8PH

The Better Design

- The anomaly problems of the Staff table can be eliminated by *decomposing this relation into smaller relations*.

staffNo staffName position salary campusID campusAddress

- ▣ **Campus** (campusID, campusAddress)
- ▣ **Staff** (staffNo, staffName, position, salary, campusID)

Decomposing Into Smaller Tables

- There are two important properties associated with the decomposition of a larger table into smaller ones:
 - ▣ **Loss-less join:** enables us to find any instance of the original larger relation from corresponding instances in the smaller relations.
 - ▣ **Dependency preservation:** ensures that a constraint on the original relation can be maintained by simply enforcing some constraint on each of the smaller relations.

Loss-less join

CustomerOrders

OrderID	CustomerID	OrderDate	TotalAmount
101	1001	2023-10-05	500.00
102	1002	2023-10-06	300.00
103	1001	2023-10-07	700.00

1.OrderInfo (OrderID, CustomerID, TotalAmount)

OrderID	CustomerID	TotalAmount
101	1001	500.00
102	1002	300.00
103	1001	700.00

2.OrderDateInfo (OrderID, OrderDate)

OrderID	OrderDate
101	2023-10-05
102	2023-10-06
103	2023-10-07

Loss-less join

1.OrderInfo (OrderID, CustomerID, TotalAmount)

OrderID	CustomerID	TotalAmount
101	1001	500.00
102	1002	300.00
103	1001	700.00

2.OrderDateInfo (OrderID, OrderDate)

OrderID	OrderDate
101	2023-10-05
102	2023-10-06
103	2023-10-07

Reconstructed CustomerOrders

OrderID	CustomerID	OrderDate	TotalAmount
101	1001	2023-10-05	500.00
102	1002	2023-10-06	300.00
103	1001	2023-10-07	700.00

Dependency preservation

1.OrderInfo (OrderID, CustomerID, TotalAmount)

OrderID	CustomerID	TotalAmount
101	1001	500.00
102	1002	300.00
103	1001	700.00

2.OrderDateInfo (OrderID, OrderDate)

OrderID	OrderDate
101	2023-10-05
102	2023-10-06
103	2023-10-07

Reconstructed CustomerOrders

OrderID	CustomerID	OrderDate	TotalAmount
101	1001	2023-10-05	500.00
102	1002	2023-10-06	300.00
103	1001	2023-10-07	700.00

Functional Dependencies

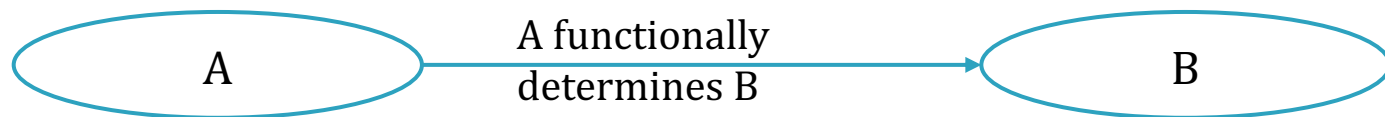
- ❑ Important concept associated with normalisation.
- ❑ Put simply, it *describes the **relationships** between attributes (columns) in a relation (table)*.
 - ▣ Aims to identify table attributes such that the chosen candidate key (PK) identifies exactly one record.

Functional Dependency

- If A and B are attributes of a table R:
 - ▣ B is functionally dependent on A (denoted $A \rightarrow B$), if each value of A in R is associated with **exactly one** value of B in R
 - *A is a candidate key (unique).*
- **Note:** A and B can be composed of one/more attributes, i.e. two attributes (A_1, A_2) in R determine the values of the other attributes (B_1, B_2, B_3) in R.
$$(A_1, A_2) \rightarrow (B_1, B_2, B_3)$$
 - ▣ Thus, from the candidate key, A, you can determine the values of the other attributes, B (from one row only).

Functional Dependencies

- In functional dependency ($A \rightarrow B$), the attribute(s) on the left-side (LHS) functionally determine the values of the attribute(s) on the right-side (RHS).
- The attribute(s) on the left-side are called the **determinant** of the functional dependency.



- A is any attribute(s) that can be a **key for some or all the other attributes**.
- B is made up of the **other attributes determined (identified) by the key, A**.

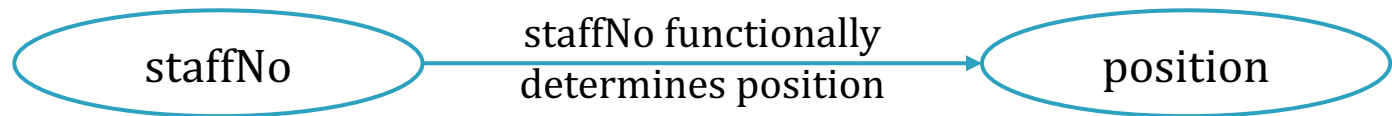
Functional Dependency Examples

Considering our smaller Staff table, we will look at some examples of functional dependencies.

staffNo	staffName	position	salary	campusID
SW101	John Smith	Lecturer	30000	C002
SW102	Diane Lee	PVC	70000	C001
SW103	Ann Beech	Lecturer	18000	C002
SW105	David Ford	Snr Lecturer	41000	C001
SW106	Mary Howe	Dean, Humanities	55000	C003
SW107	Susan Brand	Receptionist	18000	C001

Functional Dependency Examples

- In this example:

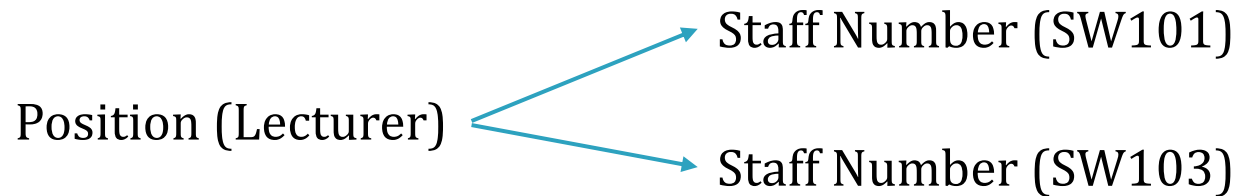


Staff Number SW106 → Dean, Humanities

staffNo	staffName	position	salary	campusID
SW101	John Smith	Lecturer	30000	C002
SW102	Diane Lee	PVC	70000	C001
SW103	Ann Beech	Lecturer	18000	C002
SW105	David Ford	Snr Lecturer	41000	C001
SW106	Mary Howe	Dean, Humanities	55000	C003
SW107	Susan Brand	Receptionist	18000	C001

Functional Dependency Examples

- In this case, however:



staffNo	staffName	position	salary	campusID
SW101	John Smith	Lecturer	30000	C002
SW102	Diane Lee	PVC	70000	C001
SW103	Ann Beech	Lecturer	18000	C002
SW105	David Ford	Snr Lecturer	41000	C001
SW106	Mary Howe	Dean, Humanities	55000	C003
SW107	Susan Brand	Receptionist	18000	C001

Full Functional Dependency

- If we consider the current data in the Staff table, the following functional dependencies appear to exist:
 - ▣ $\text{staffNo} \rightarrow \text{staffName}$
 - ▣ $\text{staffName} \rightarrow \text{staffNo}$
- But do these functional dependencies hold for all time?
- What if another member of staff with the name John Smith joins?
 - ▣ The functional dependency $\text{staffName} \rightarrow \text{staffNo}$ no longer holds.

Full Functional Dependency

- When designing tables:
 - ▣ We want to identify functional dependencies that hold for **all** possible attribute values of a relation.
 - ▣ As these represent the types of integrity constraints we need to identify.
- Determinants (the left-side) should have the **minimal** number of attributes to maintain the functional dependency with attributes on the right-side **for all time**.
 - ▣ This requirement is called **full functional dependency**.

Question

- ❑ Looking at the Staff table again:
 - ❑ Is the following a valid functional dependency?
 - ❑ If yes, is it a full functional dependency?
 - ❑ If no, why not?

staffNo, staffName $\rightarrow$ *campusID*

staffNo	staffName	position	salary	campusID
SW101	John Smith	Lecturer	30000	C002
SW102	Diane Lee	PVC	70000	C001
SW103	Ann Beech	Lecturer	18000	C002
SW105	David Ford	Snr Lecturer	41000	C001
SW106	Mary Howe	Dean, Humanities	55000	C003
SW107	Susan Brand	Receptionist	18000	C001

Answer

- Yes, it is a valid functional dependency.
 - ▣ Each combination of staffNo and staffName is associated with a single campus value.
- It is, however, not a full functional dependency.
 - ▣ For full functional dependency, the left side should have the **minimal** number of attributes to maintain the functional dependency.
 - ▣ This is an example of a ***partial dependency***.
 - ▣ campusID, the right side, is also functionally dependent on a ***subset*** of the left side determinant (staffNo, staffName):

staffNo $\rightarrow$ campusID

You can determine the campusID from just the staffNo alone.

Fully Functional vs Partial Dependency

EmployeeInfo			
EmployeeID	EmployeeName	Salary	Department
101	John Smith	50000	HR
102	Mary Johnson	60000	Finance
103	David Clark	55000	Operations

- Identify the Functional Dependencies

Fully Functional vs Partial Dependency

EmployeeInfo			
EmployeeID	EmployeeName	Salary	Department
101	John Smith	50000	HR
102	Mary Johnson	60000	Finance
103	David Clark	55000	Operations

□ Identify the Functional Dependencies

- EmployeeID → EmployeeName
- EmployeeID → Salary
- EmployeeID → Department
- EmployeeID, EmployeeName → Department
- EmployeeID, Salary → Department
- EmployeeName → Department

Fully Functional vs Partial Dependency

EmployeeInfo			
EmployeeID	EmployeeName	Salary	Department
101	John Smith	50000	HR
102	Mary Johnson	60000	Finance
103	David Clark	55000	Operations

□ Identify the Functional Dependencies

- EmployeeID → EmployeeName **FF**
- EmployeeID → Salary **FF**
- EmployeeID → Department **FF**
- EmployeeID, EmployeeName → Department **PD**
- EmployeeID, Salary → Department **PD**
- EmployeeName → Department **NO FUNCTIONAL DEPENDENCY**

Transitive Dependencies

- ❑ One last type of dependency to consider because its existence in a relation can potentially cause update anomalies.
- ❑ Transitive Dependencies
 - A condition where if A, B and C are attributes of a relation such that: $A \rightarrow B$ and $B \rightarrow C$
That is, A determines B, and B determines C.
 - Then C is transitively dependent on A via B (provided that A is not functionally dependent on B or C).

Transitive Dependency Example

- If we refer to the original CampusStaff table:
 StaffBranch (staffNo, staffName, position, salary, campusID, campusAddress)
- The following functional dependencies exist:
 - ▣ staffNo → staffName, staffName, position, salary, campusID, campusAddress)
 - ▣ campusID → campusAddress
- But campusAddress is transitively dependent on staffNo via campusID.
 - ▣ The staffNo functionally determines the campus address via the campusID
 - ▣ And neither campusID nor campusAddress functionally determines staffNo.

Why Functional Dependencies?

- The main purpose of identifying a set of functional dependencies for a relation is to:
 - ▣ specify the set of integrity constraints that must hold for a relation.
- The first constraint considered in a table is the identification of **candidate keys** in order to select a primary key.
- Functional dependencies are particularly useful in identifying the **primary keys** in a relation.
- Transitive dependencies, on the other hand, help to eliminate update anomalies.

Functional Dependency In Normalisation

- There is a **one-to-one** relationship between the LHS (determinant) attributes and the RHS attributes.
 - ▣ That is the LHS attribute(s) will identify exactly one row.
 - ▣ In other words, the LHS attribute(s) is a candidate key.
- The reverse RHS attributes can have both a one-to-one and one-to-many relationship with the determinant (LHS).
- They hold for ***all time***.
- The determinant has the ***minimal*** number of attributes necessary to maintain the dependency with the attributes on the right-hand-side.

The Process of Normalisation

Finally!!

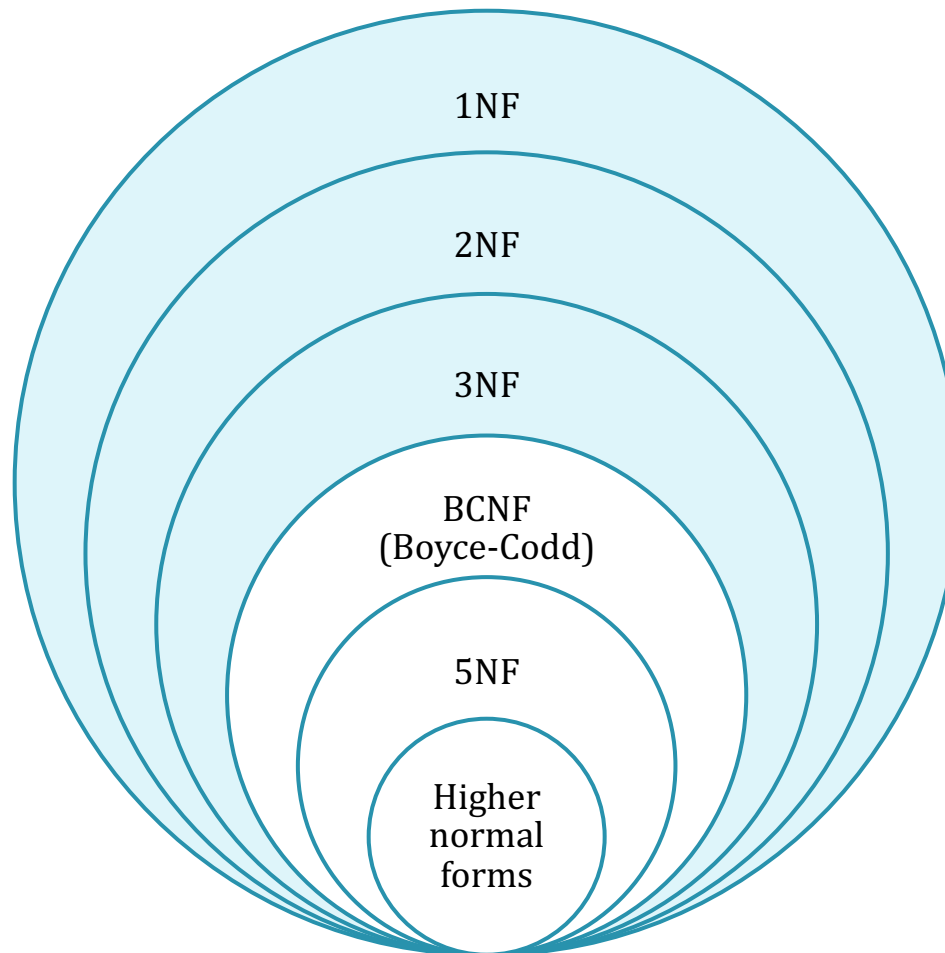


The Process of Normalisation

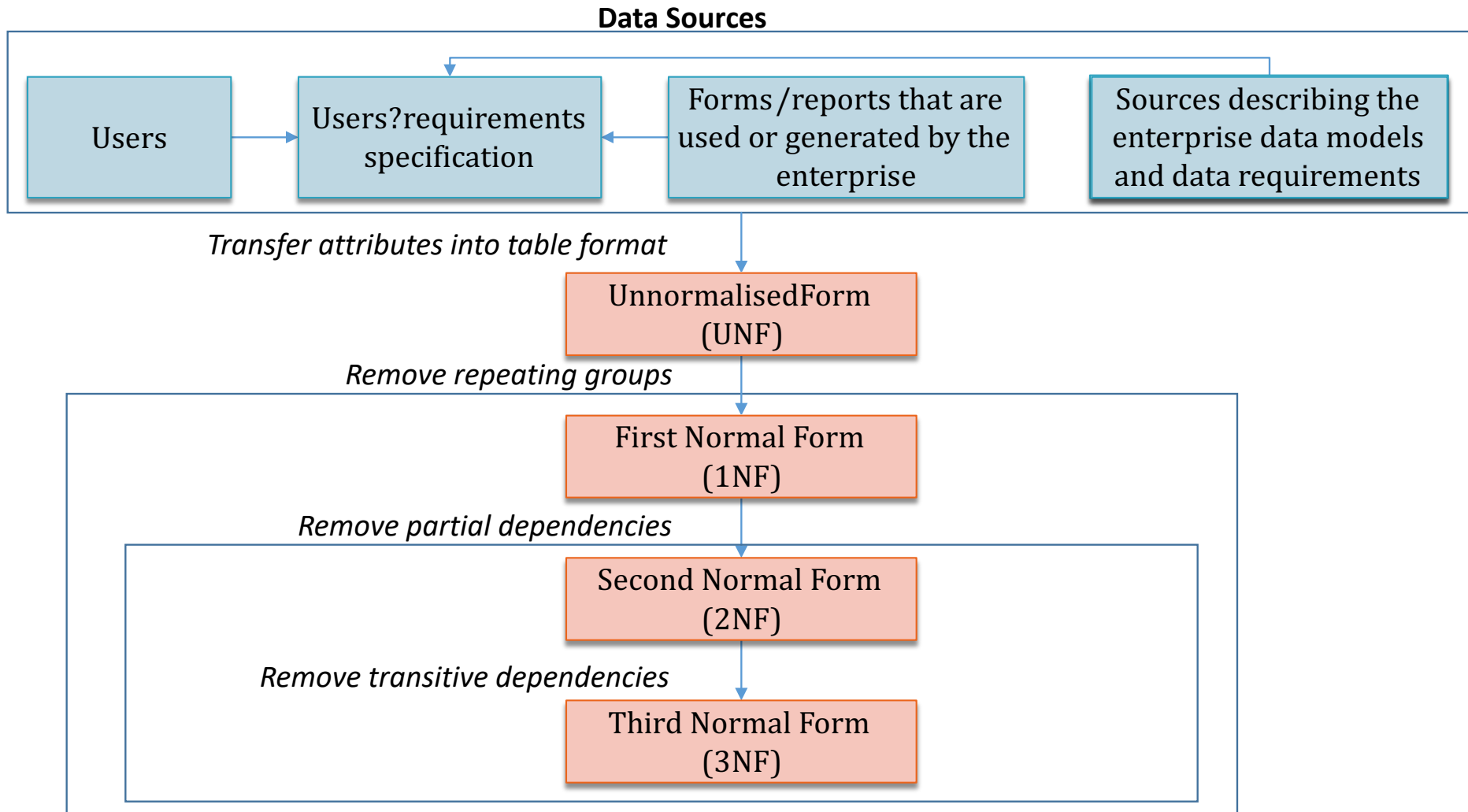
- ❑ A formal technique for analysing relations based on their keys and the functional dependencies (close logical relationship) among the attributes of that relation.
- ❑ Executed as a series of steps.
 - ▣ Each step corresponds to a specific normal form that has known properties.
 - ▣ As each step evolves, the relations become stronger in format and less vulnerable to update anomalies.

Normal Forms (NFs)

At the minimum, aim for **Third Normal Form (3NF)**



The Process of Normalisation



Sample Data: Student Module Report

student ID	student Name	module Code	moduleName	credits	mark1	mark2	lecturer ID	lecturerName
P100001	John Kay	M5X0110	Database Systems	30	75.00	65.00	L040	Tina Murphy
		M5X0105	Web Development	20	79.50	69.00	L093	Tony Shaw
P100004	Aline Stewart	M5X0110	Database Systems	30	55.00	52.00	L040	Tina Murphy
		M5X0105	Web Development	20	41.00	39.00	L093	Tony Shaw
		M5X0104	Internet Security	20	62.50	67.00	L093	Tony Shaw

- Assume each module has two assessments and the marks are stored in mark1 and mark2.

Unnormalised Form (UNF) and First Normal Form (1NF)

- Unnormalised Form (UNF)
 - ▣ A table that contains one or more repeating groups.
 - ▣ A repeating group is a group of attributes with one or more these attributes containing two or more values for a cell in a table.
- First Normal Form (1NF)
 - ▣ A relation in which the intersection of each row and column *contains one and only one value*.
 - ▣ That is, there are no repeating groups.

UNF to 1NF: Step 1

To unnormalise (i.e. convert UNF to 1NF):

- ❑ Identify and nominate an attribute or group of attributes as the key for the unnormalised table.
- ❑ Identify the repeating group(s) in the unnormalised table which repeats for the key attribute(s) identified in last step.

UNF to 1NF: Step 2

- ❑ **Method 1:** Flatten the table
 - ▣ Create separate rows for each value in the repeating group of data.
 - ▣ For the repeating group values (attributes with single values), fill the values down in the new rows created.
- ❑ **Method 2:** Splitting the table by placing the repeating data along with the original key attribute(s) into a separate table.
 - ▣ Note: If there are multiple repeating groups, repeat this process till there are no repeating groups remain.

UNF to 1NF: Step 1

- Identify and Nominate Key: **studentID**
- What are the repeating groups?

student ID	student Name	module Code	moduleName	credits	mark1	mark2	lecturer ID	lecturerName
P100001	John Kay	M5X0110	Database Systems	30	75.00	65.00	L040	Tina Murphy
		M5X0105	Web Development	20	79.50	69.00	L093	Tony Shaw
P100004	Aline Stewart	M5X0110	Database Systems	30	55.00	52.00	L040	Tina Murphy
		M5X0105	Web Development	20	41.00	39.00	L093	Tony Shaw
		M5X0104	Internet Security	20	62.50	67.00	L093	Tony Shaw

UNF to 1NF Method 1:

Step 2 -Flattening the Table

- Fill the group values down in the empty cells.

student ID	student Name	module Code	moduleName	credits	mark1	mark2	lecturer ID	lecturerName
P100001	John Kay	M5X0110	Database Systems	30	75.00	65.00	LO40	Tina Murphy
P100001	John Kay	M5X0105	Web Development	20	79.50	69.00	LO93	Tony Shaw
P100004	Aline Stewart	M5X0110	Database Systems	30	55.00	52.00	LO40	Tina Murphy
P100004	Aline Stewart	M5X0105	Web Development	20	41.00	39.00	LO93	Tony Shaw
P100004	Aline Stewart	M5X0104	Internet Security	20	62.50	67.00	LO93	Tony Shaw

UNF to 1NF Method1:

Step 2 - Creating Separate Table

- Recall that the key field identified was **studentID**.
- Splitting the table by placing the repeating data with this key field (studentID) into a separate table produces the following relations.

Table 1		Table 2 + key field column too, studentID						
student ID	student Name	module Code	moduleName	credits	mark1	mark2	lecturer ID	lecturerName
P100001	John Kay	M5X0110	Database Systems	30	75.00	65.00	L040	Tina Murphy
		M5X0105	Web Development	20	79.50	69.00	L093	Tony Shaw
P100004	Aline Stewart	M5X0110	Database Systems	30	55.00	52.00	L040	Tina Murphy
		M5X0105	Web Development	20	41.00	39.00	L093	Tony Shaw
		M5X0104	Internet Security	20	62.50	67.00	L093	Tony Shaw

1NF Tables (from Method 2)

□ Student

student ID	student Name
P100001	John Kay
P100004	Aline Stewart

□ StudentModule

student ID	module Code	moduleName	credits	mark1	mark2	lecturer ID	lecturerName
P100001	M5X0110	Database Systems	30	75.00	65.00	L040	Tina Murphy
P100001	M5X0105	Web Development	20	79.50	69.00	L093	Tony Shaw
P100004	M5X0110	Database Systems	30	55.00	52.00	L040	Tina Murphy
P100004	M5X0105	Web Development	20	41.00	39.00	L093	Tony Shaw
P100004	M5X0104	Internet Security	20	62.50	67.00	L093	Tony Shaw

Second Normal Form (2NF)

- A relation that is in **1NF** and every non-primary key attribute is ***fully functionally dependent*** on the primary key.
- This step involves removing partial dependencies.

2NF Steps

- ❑ Identify the primary key for each 1NF relation.
- ❑ Identify the functional dependencies in the relation.
- ❑ Check for partial dependencies
 - If partial dependencies exist on the primary key, remove them by placing them in a new relation along with a copy of their determinant (left side).
 - And remove the duplicate rows in the new table.

1NF to 2NF: Student Table

student ID	student Name
P100001	John Kay
P100004	Aline Stewart

- ❑ What is the primary key? **studentID**
- ❑ What are the functional dependencies?
 - ▣ $\text{studentID} \rightarrow \text{studentName}$
- ❑ Are there any partial dependencies?
 - ▣ No, this is a simple functional dependency involving single attributes and it is clear that studentName is fully functionally dependent on studentNo
- ❑ The Student table can remain as it is.

1NF to 2NF: StudentModule Table

student ID	module Code	moduleName	credits	mark1	mark2	lecturer ID	lecturerName
P100001	M5X0110	Database Systems	30	75.00	65.00	L040	Tina Murphy
P100001	M5X0105	Web Development	20	79.50	69.00	L093	Tony Shaw
P100004	M5X0110	Database Systems	30	55.00	52.00	L040	Tina Murphy
P100004	M5X0105	Web Development	20	41.00	39.00	L093	Tony Shaw
P100004	M5X0104	Internet Security	20	62.50	67.00	L093	Tony Shaw

- ❑ What is the primary key?
- ❑ What are the functional dependencies?

StudentModule Functional Dependencies

- Primary key: studentID, moduleCode, so we have:
studentID, moduleCode $\rightarrow$ moduleName, credits, mark1, mark2, lecturerID, lecturerName
- Recall: partial dependencies involves identifying attributes that are functionally dependent on a **subset** of the determinant (left-side).
 - ▣ Nothing is functionally dependent on studentID alone (you need both studentID and moduleCode to determine other values).
 - ▣ But what about moduleCode?
moduleCode $\rightarrow$ moduleName, credits, lecturerID, lecturerName
 - ▣ If we extract this dependency from the table, leaving the remaining fields for functional dependency (studentID, moduleCode), we get the two tables on the next slide.

1NF to 2NF: StudentModule Table

□ ModuleLecturer

moduleCode	moduleName	credits	lecturerID	lecturerName
M5X0110	Database Systems	30	L040	Tina Murphy
M5X0105	Web Development	20	L093	Tony Shaw
M5X0104	Internet Security	20	L093	Tony Shaw

□ StudentModule

studentID	moduleCode	mark1	mark2
P100001	M5X0110	75.00	65.00
P100001	M5X0105	79.50	69.00
P100004	M5X0110	55.00	52.00
P100004	M5X0105	41.00	39.00
P100004	M5X0104	62.50	67.00

2NF Tables

Student

student ID	student Name
P100001	John Kay
P100004	Aline Stewart

ModuleLecturer

moduleCode	moduleName	credits	lecturerID	lecturerName
M5X0110	Database Systems	30	L040	Tina Murphy
M5X0105	Web Development	20	L093	Tony Shaw
M5X0104	Internet Security	20	L093	Tony Shaw

StudentModule

studentID	moduleCode	mark1	mark2
P100001	M5X0110	75.00	65.00
P100001	M5X0105	79.50	69.00
P100004	M5X0110	55.00	52.00
P100004	M5X0105	41.00	39.00
P100004	M5X0104	62.50	67.00

Third Normal Form (3NF)

- A relation that is in both 1NF and 2NF **and** in which no non-primary key is transitively dependent on the primary key.
- Based on the concept of transitive dependency.
- Recall that transitive dependency is a condition where:
 - A, B and C are attributes of a relation such that if $A \rightarrow B$ and $B \rightarrow C$,
 - then C is transitively dependent on A through B (provided that A is not functionally dependent on B or C).

2NF to 3NF

- ❑ Identify the primary key in the 2NF relation
- ❑ Identify functional dependencies in the relation.
- ❑ Check if transitive dependencies exist on the primary key and remove them by placing them in a new relation with a copy of their determinant.

2NF to 3NF

Student and StudentModule Tables

- All the non-primary keys in the Student and StudentModule tables are functionally dependent on only their primary keys.
- They do not have any transitive dependencies.

Student

student ID	student Name
P100001	John Kay
P100004	Aline Stewart

StudentModule

studentID	moduleCode	mark1	mark2
P100001	M5X0110	75.00	65.00
P100001	M5X0105	79.50	69.00
P100004	M5X0110	55.00	52.00
P100004	M5X0105	41.00	39.00
P100004	M5X0104	62.50	67.00

2NF to 3NF

ModuleLecturer Table

moduleCode	moduleName	credits	lecturerID	lecturerName
M5X0110	Database Systems	30	L040	Tina Murphy
M5X0105	Web Development	20	L093	Tony Shaw
M5X0104	Internet Security	20	L093	Tony Shaw

- ❑ What is the primary key?
- ❑ What are the functional dependencies?
- ❑ Any transitive dependencies?

2NF to 3NF

ModuleLecturer Table

moduleCode	moduleName	credits	lecturerID	lecturerName
M5X0110	Database Systems	30	L040	Tina Murphy
M5X0105	Web Development	20	L093	Tony Shaw
M5X0104	Internet Security	20	L093	Tony Shaw

- ❑ What is the primary key?
- ❑ What are the functional dependencies?
- ❑ Any transitive dependencies?

moduleCode → lecturerID → lecturerName

moduleCode → lecturerName

2NF to 3NF

ModuleLecturer Table

moduleCode	moduleName	credits	lecturerID	lecturerName
M5X0110	Database Systems	30	L040	Tina Murphy
M5X0105	Web Development	20	L093	Tony Shaw
M5X0104	Internet Security	20	L093	Tony Shaw

- ❑ What is the primary key? **moduleCode**
- ❑ What are the functional dependencies?
 - $\text{moduleCode} \rightarrow \text{moduleName}, \text{credits}, \text{lecturerID}, \text{lecturerName}$
 - $\text{lecturerID} \rightarrow \text{lecturerName}$
- ❑ Any transitive dependencies?
 - Yes: $\text{lecturerID} \rightarrow \text{lecturerName}$
 moduleCode functionally determines lecturerName via the lecturerID .
 - Remove this transitive dependency into a separate relation.

2NF to 3NF

ModuleLecturer Table

- Create two tables from the ModuleLecturer table

- Lecturer

lecturerID	lecturerName
LO40	Tina Murphy
LO93	Tony Shaw

- Module

moduleCode	moduleName	credits	lecturerID
M5X0110	Database Systems	30	LO40
M5X0105	Web Development	20	LO93
M5X0104	Internet Security	20	LO93

The Normalised 3NF Relations

Student

studentID	studentName
P100001	John Kay
P100004	Aline Stewart

Module

moduleCode	moduleName	credits	lecturerID
M5X0110	Database Systems	30	L040
M5X0105	Web Development	20	L093
M5X0104	Internet Security	20	L093

Lecturer

lecturerID	lecturerName
L040	Tina Murphy
L093	Tony Shaw

StudentModule

studentID	moduleCode	mark1	mark2
P100001	M5X0110	75.00	65.00
P100001	M5X0105	79.50	69.00
P100004	M5X0110	55.00	52.00
P100004	M5X0105	41.00	39.00
P100004	M5X0104	62.50	67.00

Lab Exercise



- ❑ Download Instructions from Moodle

Further reading - Determinants

Consider the table occurrence Stock:

Table: Stock

<u>Stock</u>	<u>Description</u>
P4	Nut
P2	Pin
P7	Nut
P1	Bolt
P5	Bolt

Enterprise Rule: A given Stock# has exactly one associated description (a given description as one or more Stock No's associated with it).

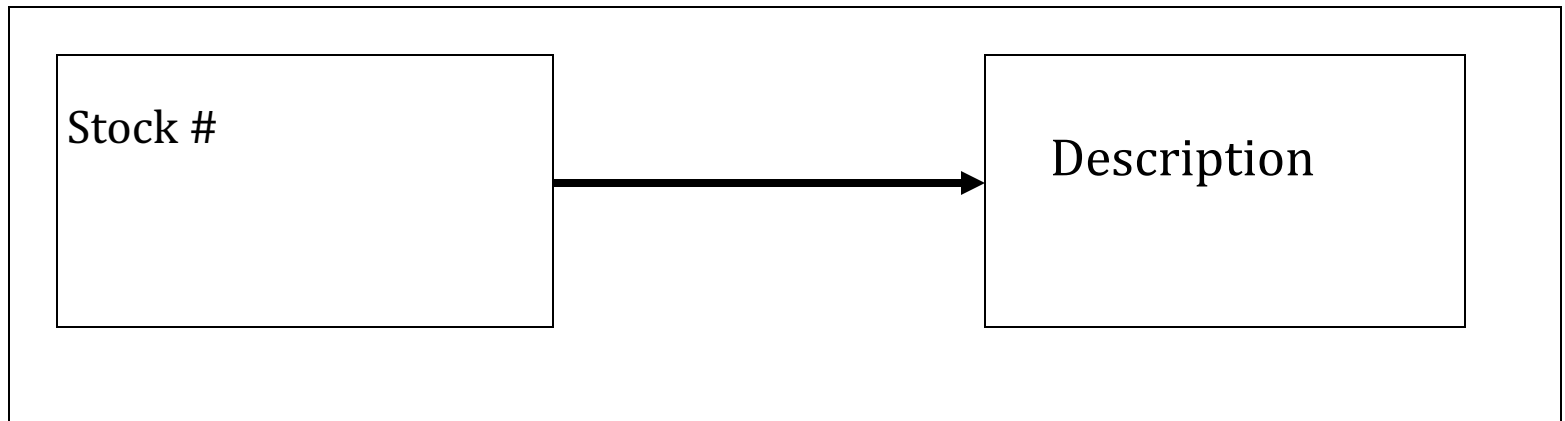
Stock# is a determinant of Description, i.e.

Stock# determines description, and

Description is determined by/dependent on Stock#

determinacy Diagrams

These can be used to show determinants and the attributes they determine. Thus, for the previous example the determinacy diagram is:



The arrow shows the dependency. The direction of the arrow head shows which attribute is the determinant.

Determinacy Diagrams

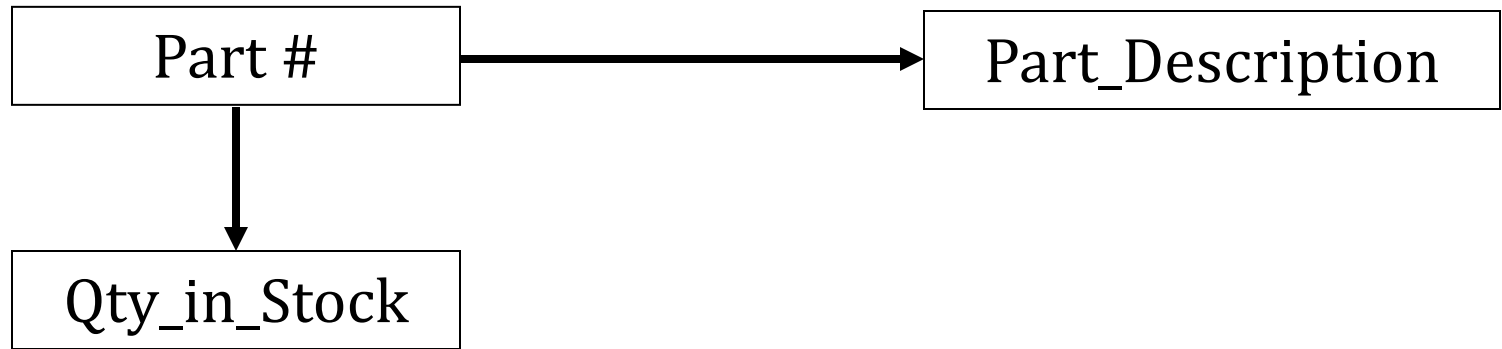
Consider the following Enterprise Rules:

1. A given Consultant# has exactly one associated Project#.
2. A given Project# has exactly one associated Consultant#.

The resultant determinacy diagram is:



Examples of Determinacy Diagrams



Part# is the determinant of Part_Description and Qty_in_Stock.

Each Part# is associated with only one Part_Description and one Qty_in_Stock.

A given Qty_in_Stock may be associated with many Part# values.

Examples of determinacy Diagrams

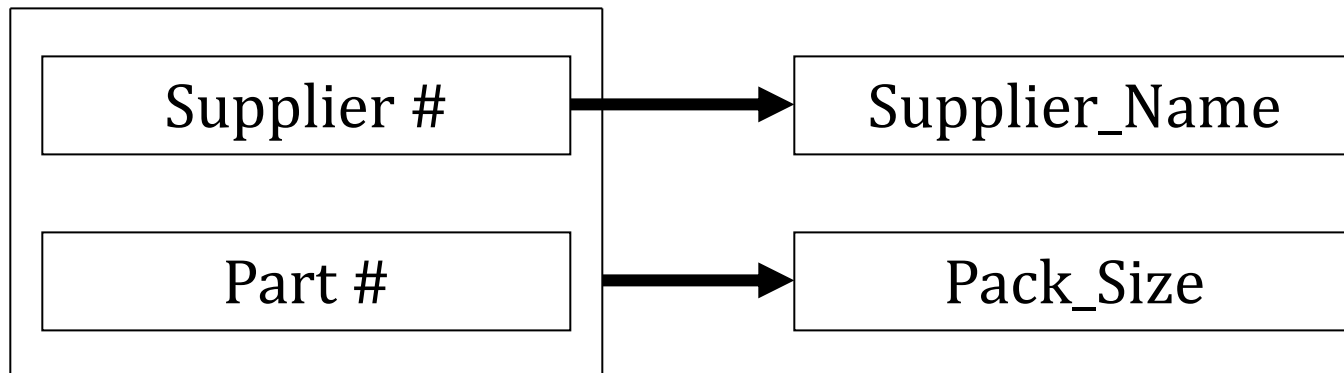
A given Part_Description value may be associated with many Part# values.

Table: Supplier Part

<u>Supplier#</u>	<u>Supplier_Name</u>	<u>Part#</u>	<u>Pack_Size</u>
S2	Jones	P1	10
S7	Smith	P6	25
S2	Jones	P4	40
S5	Jones	P1	20

Rules

1. Supplier identified by supplier#.
2. Part identified by Part#.
3. A Supplier has one Supplier_Name.
4. A Supplier may supply many parts in many Pack_Sizes.
5. A Part may be supplied by different suppliers in different Pack_Sizes.
6. A Supplier supplies a given part in just one Pack_Size.



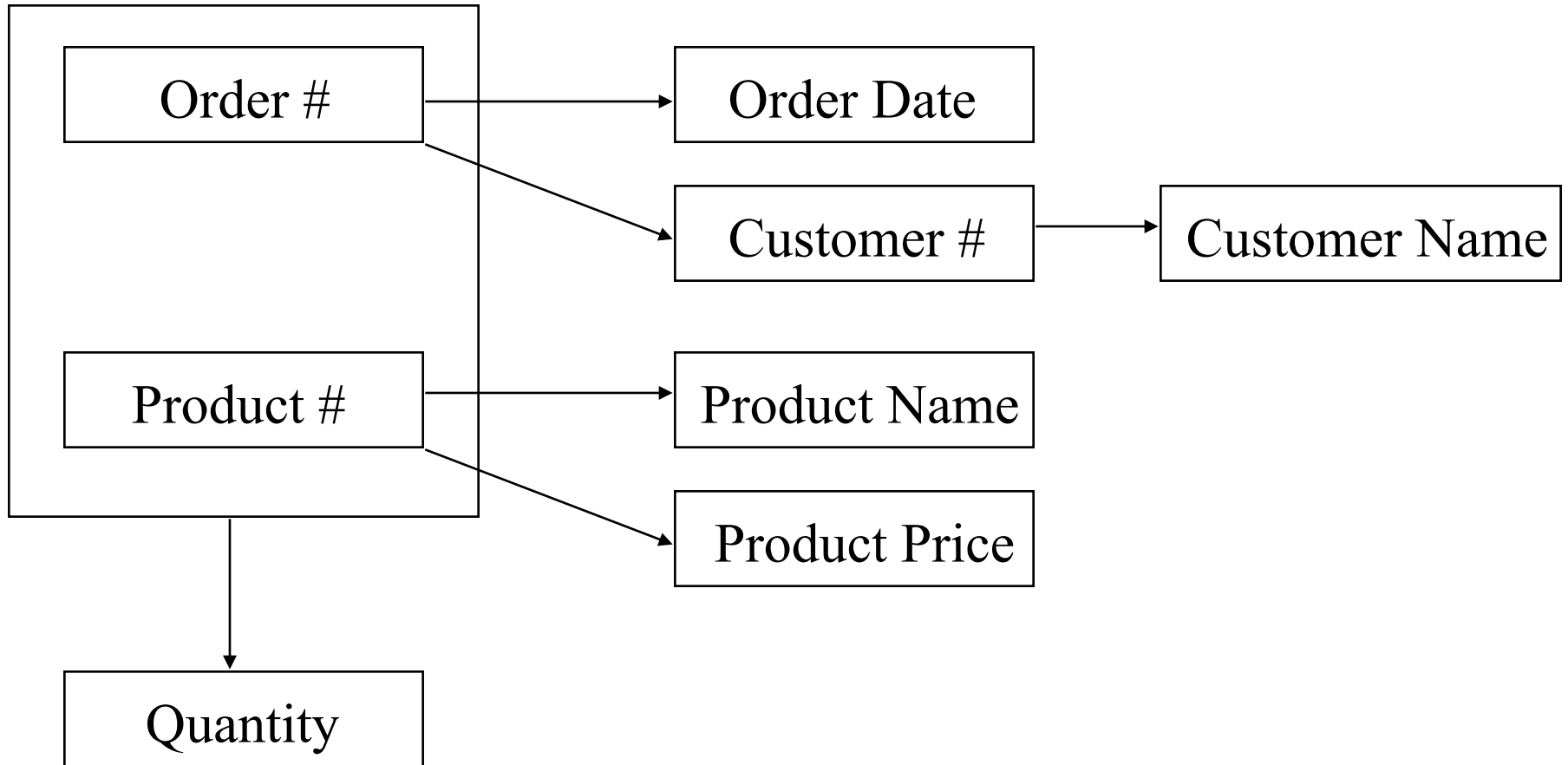
Exercise 1

Table: Order Processing

Order#	Order Date	Cust #	Cust Name	Prod#	Prod Name	Qty	Prod Price
1234	01/02/96	2235	Jenkins	4875	Widget	200	1.00
1385	08/02/96	3444	Jones	2890	Woggle	300	2.00
1567	15/03/96	3444	Jones	4875	Widget	1000	1.00
1789	21/04/96	2890	Jones	3332	Wangle	150	0.50
2045	23/04/96	2235	Jenkins	3332	Wangle	40	0.50
2344	02/03/96	2235	Jenkins	4875	Widget	400	1.00
1385	08/02/96	34444	Jones	4875	Widget	300	1.00

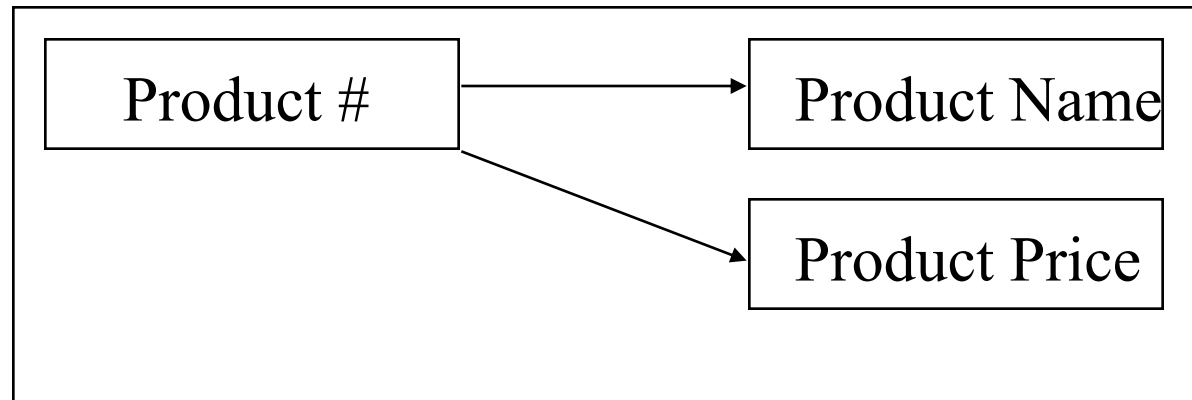
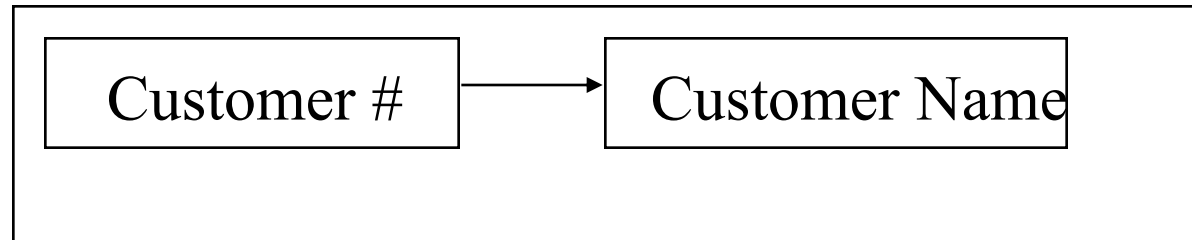
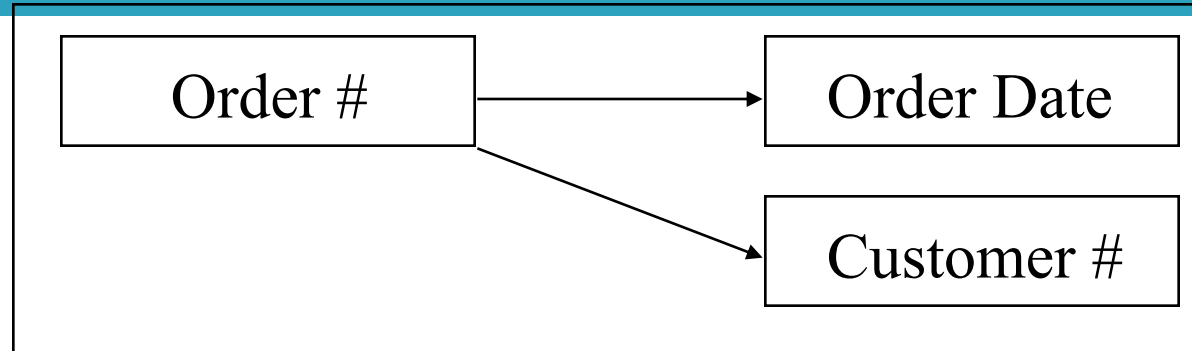
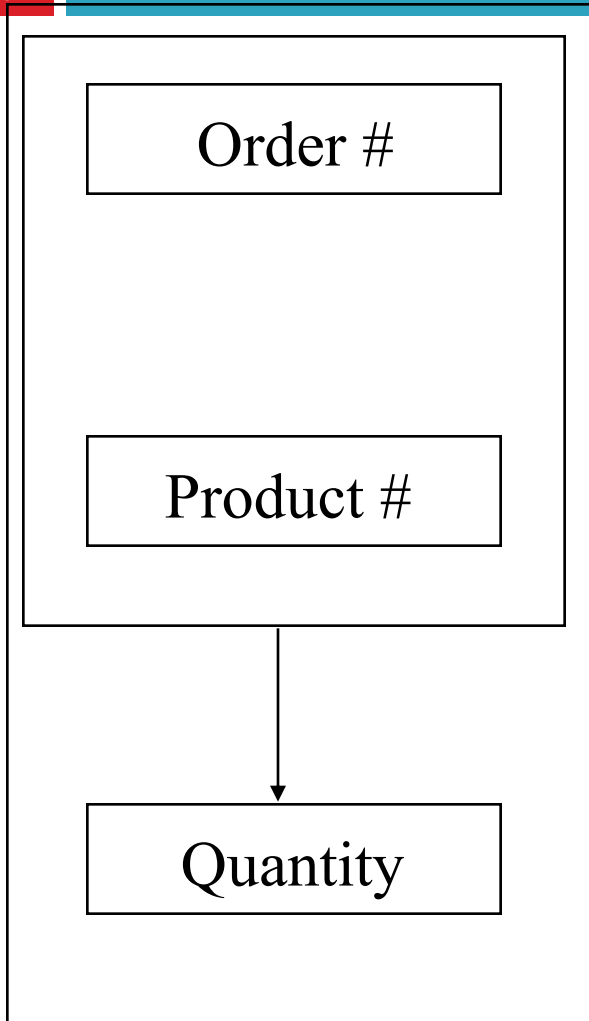
1. Produce a determinacy diagram for the above data.
2. From your diagram construct the resultant tables.

Initial Diagram



Individual Diagrams

90



Construct Tables

91

- Table: Order
 - ▣ Order (Order #, Order Date, Customer #)

- Table: Customer
 - ▣ Customer (Customer #, Customer Name)

- Table: Product
 - ▣ Product (Product #, Product Name, Product Price)

- Table Order Line
 - ▣ Order Line (Order #, Product #, Quantity)

Thank you

