NORMALISATION

Dr. Mohammed A. Mohammed
University of Sulaimani | College of science | Computer dept.

Normalization in Database Design

1. A bottom-up approach- difficult with a large number of attributes.
2. In conjunction with ER modelling (top-down approach), check whether the tables are well designed.

Normalization-table optimization

- Normalization reduces data redundancy in a database
- Is bottom-up DB design to ensure correctness of up-down design done by ERM
- It eliminates serious manipulation anomalies.
- Normalization is ultimately just rearranging propositions to a better structure.
- This is done by identifying and removing damaging functional dependencies.

Normalization

- Normalization is the process of efficiently organizing data in a database with two goals in mind

- **First goal:** Eliminate redundant data
  - for example, storing the same data in more than one table

- **Second Goal:** Ensure data dependencies make sense
  - for example, only storing related data in a table

Benefits of Normalization

- Less storage space
- Quicker updates
- Less data inconsistency
- Clearer data relationships
- Easier to add data
- Flexible Structure

Redundancy and Data Anomalies

Redundant data is where we have stored the same 'information' more than once. i.e., the redundant data could be removed without the loss of information.

Example: We have the following relation that contains staff and department details:

staffNo	job	dept	dname	city
SL10	Salesman	10	Sales	Stratford
SA51	Manager	20	Accounts	Barking
DS40	Clerk	20	Accounts	Barking
OS45	Clerk	30	Operations	Barking

Such 'redundancy' could lead to the following 'anomalies'

Insert Anomaly: We can't insert a dept without inserting a member of staff that works in that department

Update Anomaly: We could change the name of the dept that **SA51** works in without simultaneously changing the dept that **DS40** works in.

Deletion Anomaly: By removing employee **SL10** we have removed all information relating to the Sales dept.

Anomaly (Irregularity Abnormality Inconsistency)

Repeating Groups

A repeating group is an attribute (or set of attributes) that can have more than one value for a primary key value.

Example: We have the following relation that contains staff and department details and a list of telephone contact numbers for each member of staff.

staffNo	job	dept	dname	city	contact number
SL10	Salesman	10	Sales	Stratford	**018111777, 018111888, 079311122**
SA51	Manager	20	Accounts	Barking	**017111777**
DS40	Clerk	20	Accounts	Barking	
OS45	Clerk	30	Operations	Barking	**079311555**

Repeating Groups are not allowed in a relational design, since all attributes have to be '**atomic**' - i.e., there can only be one value per cell in a table!

Functional Dependency

Formal Definition: Attribute B is functionally dependant upon attribute A *(or a collection of attributes)* if a value of A determines a single value of attribute B at any one time.

Formal Notation: **A → B** This should be read as **'A determines B'** or **'B is functionally dependant on A'**. A is called the *determinant* and B is called the *object of the determinant*.

Example:

staffNo	job	dept	dname
SL10	Salesman	10	Sales
SA51	Manager	20	Accounts
DS40	Clerk	20	Accounts
OS45	Clerk	30	Operations

Functional Dependencies

staffNo → job
staffNo → dept
staffNo → dname
dept → dname

Functional Dependency

Compound Determinants: If more than one attribute is necessary to determine another attribute in an entity, then such a determinant is termed a composite determinant.

Full Functional Dependency: Only of relevance with composite determinants. This is the situation when it is necessary to use all the attributes of the composite determinant to identify its object uniquely.

Example:

order#	line#	qty	price
A001	001	10	200
A002	001	20	400
A002	002	20	800
A004	001	15	300

Full Functional Dependencies

(Order#, line#) $\rightarrow$ qty

(Order#, line#) $\rightarrow$ price

Functional Dependency

2- **Partial Functional Dependency**: This is the situation that exists if it is necessary to only use a subset of the attributes of the composite determinant to identify its object uniquely.

Example:

student#	unit#	room	grade
9900100	A01	TH224	2
9900010	A01	TH224	14
9901011	A02	JS075	3
9900001	A01	TH224	16

Repetition of data!

Full Functional Dependencies

(student#, unit#) $\rightarrow$ grade

Partial Functional Dependencies

unit# $\rightarrow$ room

Dependencies: Definitions

- ***Partial Dependency*** – when an non-key attribute is determined by a part, but not the whole, of a **COMPOSITE** primary key.

Partial Dependency

CUSTOMER

Cust_ID	**Name**	**Order_ID**
101	AT&T	1234
101	AT&T	156
125	Cisco	1250

Dependencies: Definitions

- ***Transitive Dependency*** – when a non-key attribute determines another non-key attribute.

Transitive Dependency

EMPLOYEE

<u>Emp_ID</u>	F_Name	L_Name	Dept_ID	Dept_Name
111	Mary	Jones	1	Acct
122	Sarah	Smith	2	Mktg

Transitive Dependency

Definition: A transitive dependency exists when there is an intermediate functional dependency.

Formal Notation: If $A \rightarrow B$ and $B \rightarrow C$, then it can be stated that the following transitive dependency exists: $A \rightarrow B \rightarrow C$

Example:

staffNo	job	dept	dname
SL10	Salesman	10	Sales
SA51	Manager	20	Accounts
DS40	Clerk	20	Accounts
OS45	Clerk	30	Operations

Repetition of data!

Transitive Dependencies

staffNo $\rightarrow$ dept
dept $\rightarrow$ dname

staffNo $\rightarrow$ dept $\rightarrow$ dname

Normalisation - Relational Model

In order to comply with the relational model it is necessary to

1) Remove repeating groups
2) Avoid redundancy and data anomalies by removing partial and transitive functional dependencies.

Relational Database Design

All attributes in a table must be:

- Atomic
- Depend upon the fully primary key of that table.

Stages of Normalisation

Unnormalised **(UDF)**

↓ Remove repeating groups

First normal form **(1NF)**

↓ Remove partial dependencies

Second normal form **(2NF)**

↓ Remove transitive dependencies

Third normal form **(3NF)**

↓ Remove remaining functional dependency anomalies

Boyce-Codd normal form **(BCNF)**

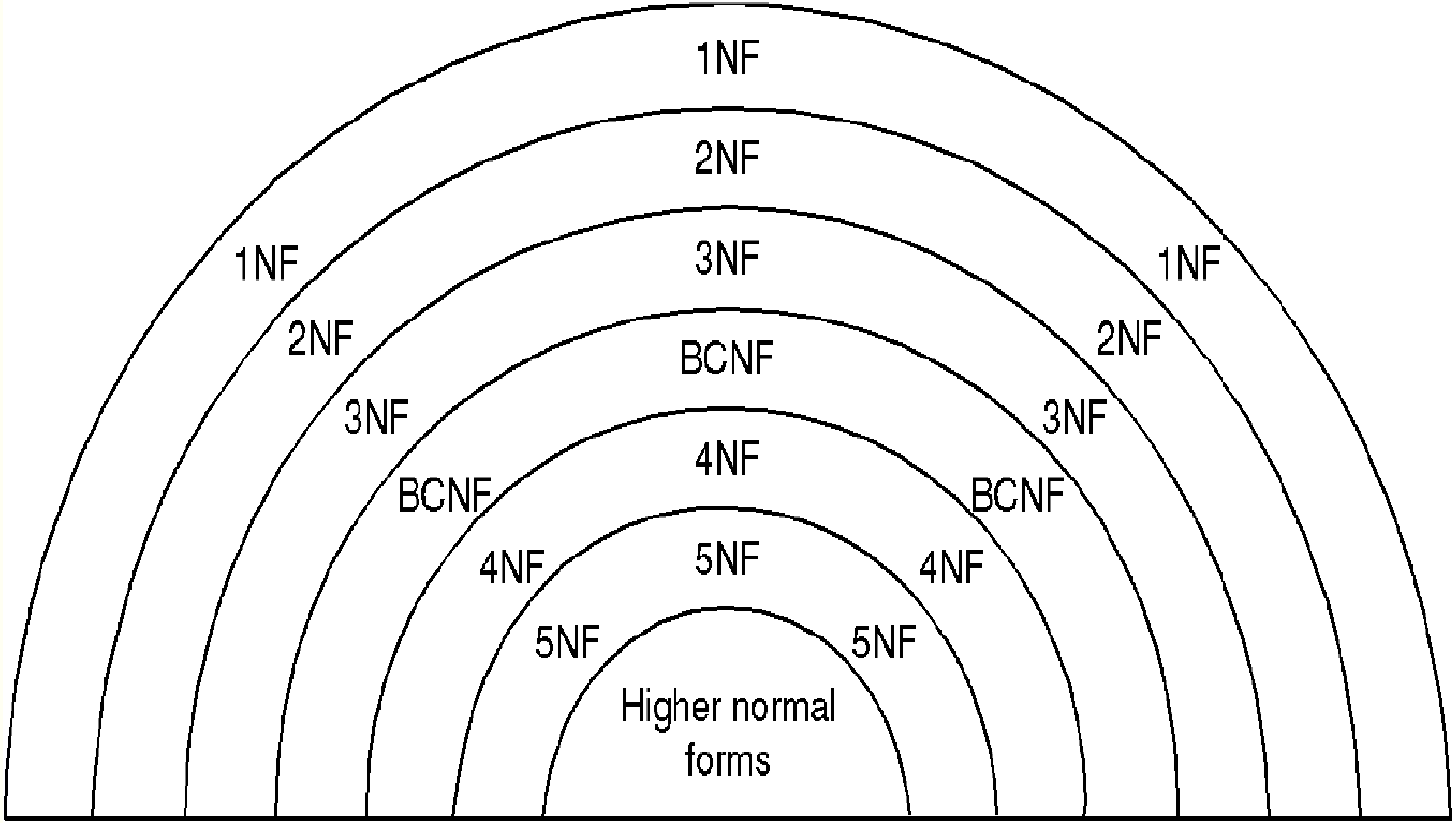
↓ Remove multivalued dependencies

Fourth normal form **(4NF)**

↓ Remove remaining anomalies

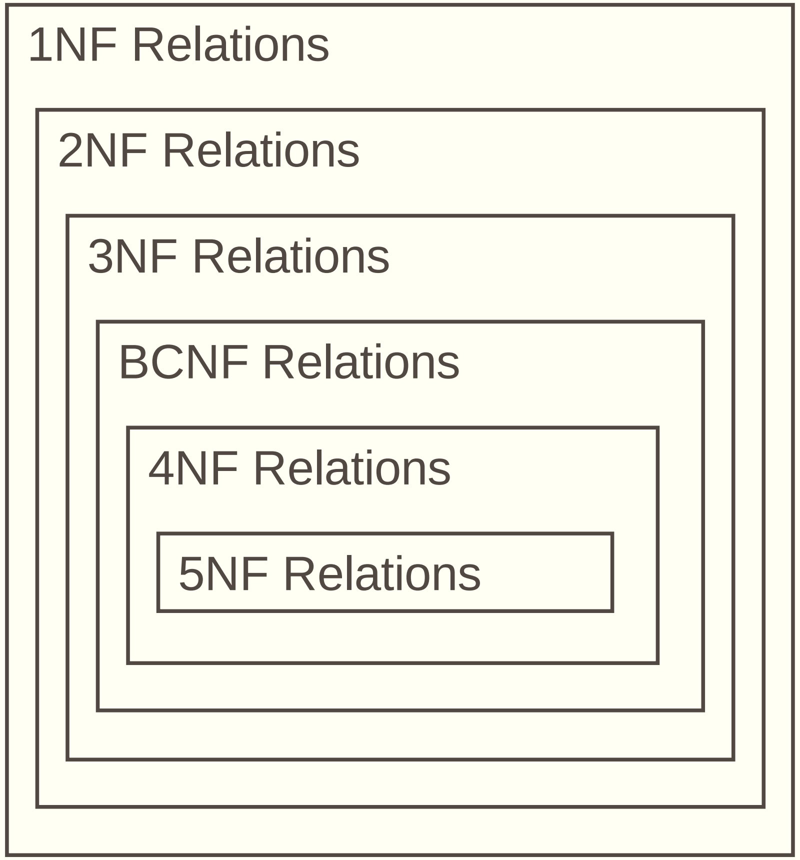
Fifth normal form **(5NF)**

Relationship Between Normal Form



Higher Normal Forms

- **Boyce-Code Normal Form (BCNF)** is as far as we can go with Functional Dependencies (**FD**)s
- Higher normal forms are based on other sorts of dependency
- Fourth normal form removes multi-valued dependencies
- Fifth normal form removes join dependencies



Problematic Table

Project number	Project name	Empl_ number	Employee name	Rate category	Hourly rate
1023	Madagascar travel site	11	Ahmed Karwan	A	$60
		12	Karzan Kamaran	B	$50
		16	Raz Azad	C	$40
1056	Online estate agency	11	Kurda Rzgar	A	$60
		17	Shagull Sarkawt	B	$50

Problematic Table

Project number	Project name	Employee number	Employee name	Rate category	Hourly rate
1023	Madagascar travel site	11	Ahmed Karwan	A	$60
1023	Madagascar travel site	12	Karzan Kamaran	B	$50
1023	Madagascar travel site	16	Raz Azad	C	$40
1056	Online estate agency	11	Kurda Rzgar	A	$60
1056	Online estate agency	17	Shagull Sarkawt	B	$50

1NF Requirements

- Each table must contain a primary key, a candidate key that identifies a tuple in a table.
- The domains of the attributes must include only atomic (simple) values.
- The value of any attribute in a tuple must be a single value from the domain of the attribute.
- There are no repeating groups in the table.

1NF Example

Name	<u>EmpId</u>	Address
Susan	205	525 Mabury Rd. San Jose, CA 95133
Susan	206	875 Gridley St. San Jose, CA 95127

repeating Group

This table is not in 1NF….why?

1NF Example cont.

This is how the table should look like:

Name	EmpId	Street	City	State	Zip
Susan	205	525 Mabury Rd.	San Jose	CA	95133
Susan	206	Gridley St.	San Jose	CA	95127

2NF Requirements

- A relation is in **2NF** if it is in **1NF** and any one of these is true:
  - All the attributes are part of the primary key (there are no non-key attributes)
  - Every non-key attribute is fully functionally dependent on the primary key.
    - To achieve this, remove partial functional dependencies, so that no non-key attribute depends on just part of the key.

2NF Example

<u>Professor</u>	<u>Subject</u>	Office
Jones	Math42	MH 410
Jones	CS49C	MH 410
Smith	Chem1A	DH 211
Smith	Chem100W	DH 211
Lee	Math161A	MH 320

Can you see why this table is not in 2NF? (Office is only dependent on Professor)

2NF Example cont.

To make the previous table in 2NF, the table must be split into two separate tables:

Professor	Subject
Jones	Math42
Jones	CS49C
Smith	Chem1A
Smith	Chem100W
Lee	Math161A

Professor	Office
Jones	MH 410
Smith	DH 211
Lee	MH 320

No dependencies on non-key attributes

Inventory			
Description	Supplier	Cost	Supplier Address

There are two non-key fields. So, here are the questions:

•If I know just Description, can I find out Cost?

- No, because we have more than one supplier for the same product.

•If I know just Supplier, can I find out Cost?

- No, because I need to know what the Item is as well.

Therefore, Cost is fully, functionally dependent upon the **ENTIRE PK** (Description-Supplier) for its existence.

(Description, Supplier) → Cost

Inventory		
Description	Supplier	Cost

CONTINUED…

Inventory			
<u>Description</u>	<u>Supplier</u>	Cost	Supplier Address

- **If I know just Description, can I find out Supplier Address?**
  - No, because we have more than one supplier for the same product.
- **If I know just Supplier, can I find out Supplier Address?**
  - Yes, The Address does not depend upon the description of the item.
- **Therefore, Supplier Address is NOT fully, functionally dependent upon the ENTIRE PK (Description-Supplier) for its existence.**

Supplier	
<u>Name</u>	Supplier Address

So putting things together

Inventory			
<u>Description</u>	<u>Supplier</u>	Cost	Supplier Address

Inventory		
<u>Description</u>	<u>Supplier</u>	Cost

Supplier	
<u>Name</u>	Supplier Address

The above relation is now in 2NF.

3NF Requirements

- A relation is in **third normal form** if it is in **2NF**, and *no transitive dependencies* exist.

- A transitive dependency is a functional dependency between non-key attributes.

3NF Example

<u>EmpId</u>	Employee Name	Rate Group	Hourly Rate
100	Charles	A	$20
101	James	B	$25
102	Jennifer	A	$20

This table is not in 3NF because hourly rate is determined by rate group, and so there is transitive dependency between two non-key attributes.

3NF Example Cont.

EmpId	Employee Name	Rate Group
100	Charles	A
101	James	B
102	Jennifer	A

Rate Group	Hourly Rate
A	$20
B	$25

Boyce/Codd Normal Form

- A relation is BCNF ⇔ every determinant is a candidate key
- A determinant is an attribute, possibly composite, on which some other attribute is fully functionally dependent

Boyce/Codd Normal Form

S	J	T
Smith	Math	Prof. White
Smith	Physics	Prof. Green
Jones	Math	Prof. White
Jones	Physics	Prof. Brown

Relation SJT

1. For each subject (J), each student (S) of that subject taught by only one teacher (T): FD: S, J → T
2. Each teacher (T) teaches only one subject (J): FD: T → J
3. Each subject (J) is taught by several teacher: MVD: J→ → T

- There exists a relation SJT with attributes S (student), J (subject) and T (teacher). The meaning of SJT tuple is that the specified student is taught the specified subject by the specified teacher.

Boyce/Codd Normal Form

- There are two determinants: (S, J) and T in functional dependency
- Anomalies in update:
  - If the fact that Jones studies physics is deleted, the fact that Professor Brown teaches physics is also lost. It is because T is a determinant but not a candidate key

Boyce/Codd Normal Form

S	J
Smith	Math
Smith	Physics
Jones	Math
Jones	Physics

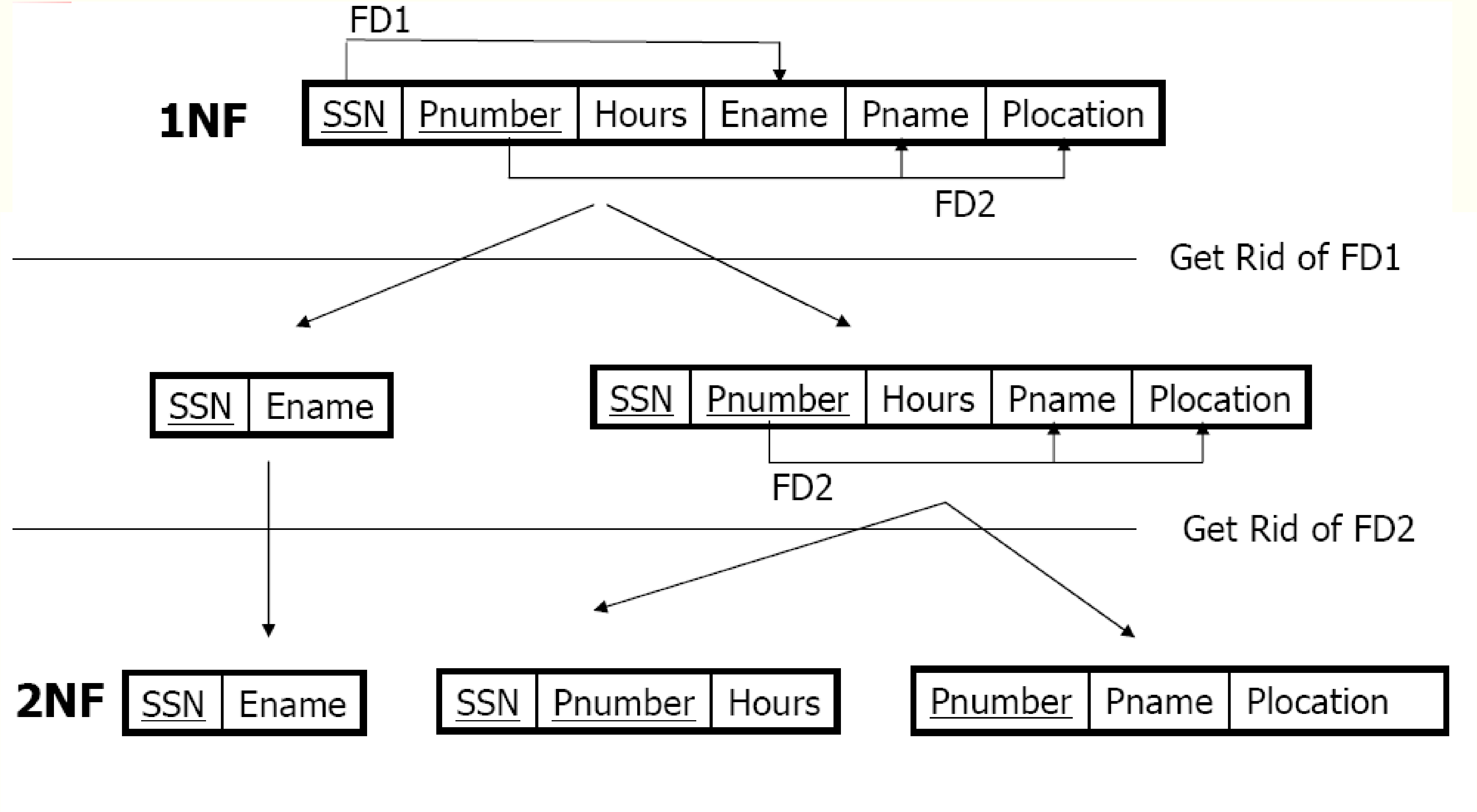
Relation ST

T	J
Prof. White	Math
Prof. Green	Physics
Prof. Brown	Physics

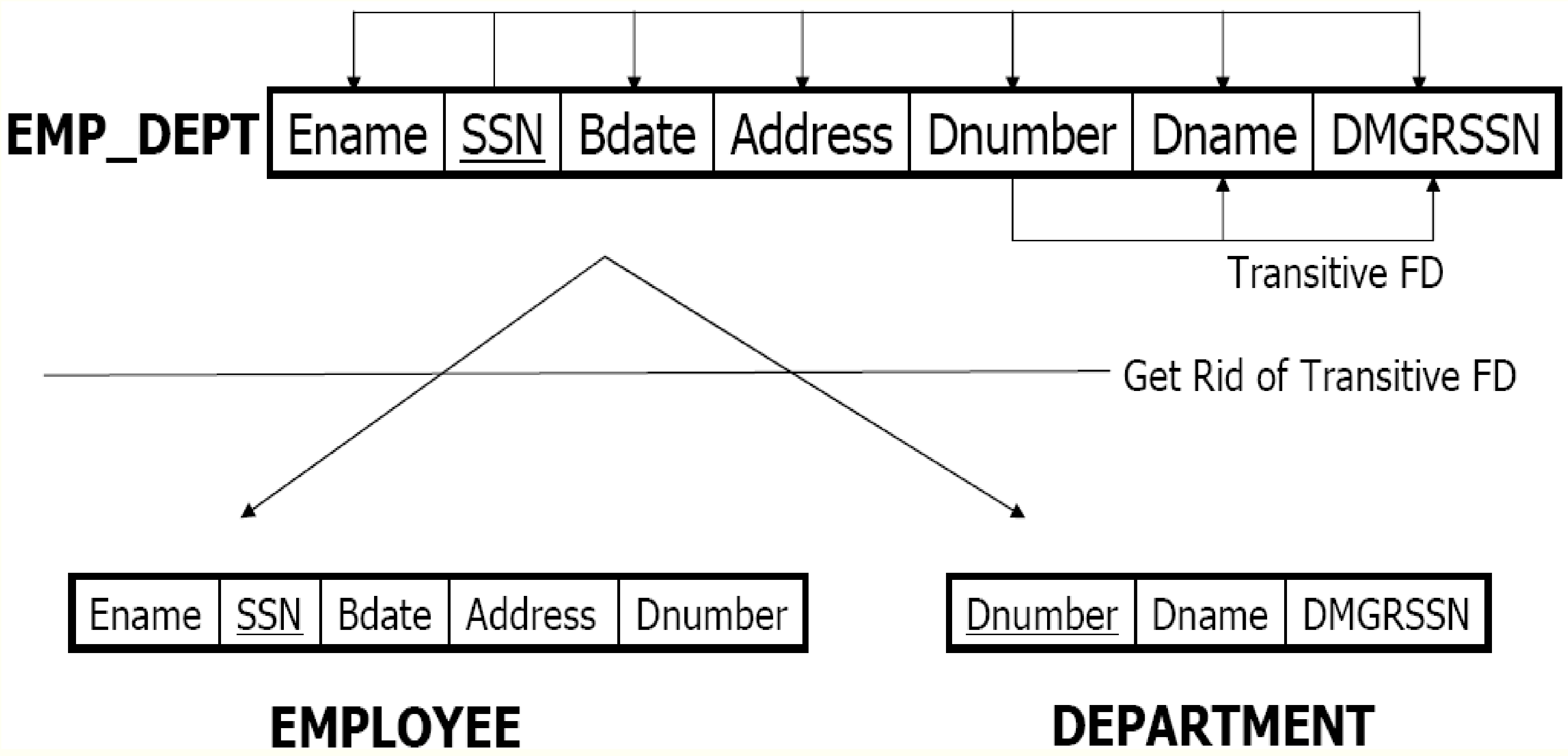
Relation TJ

Relations (S, J) and (T, J) are in BCNF because all determinants are candidate keys.

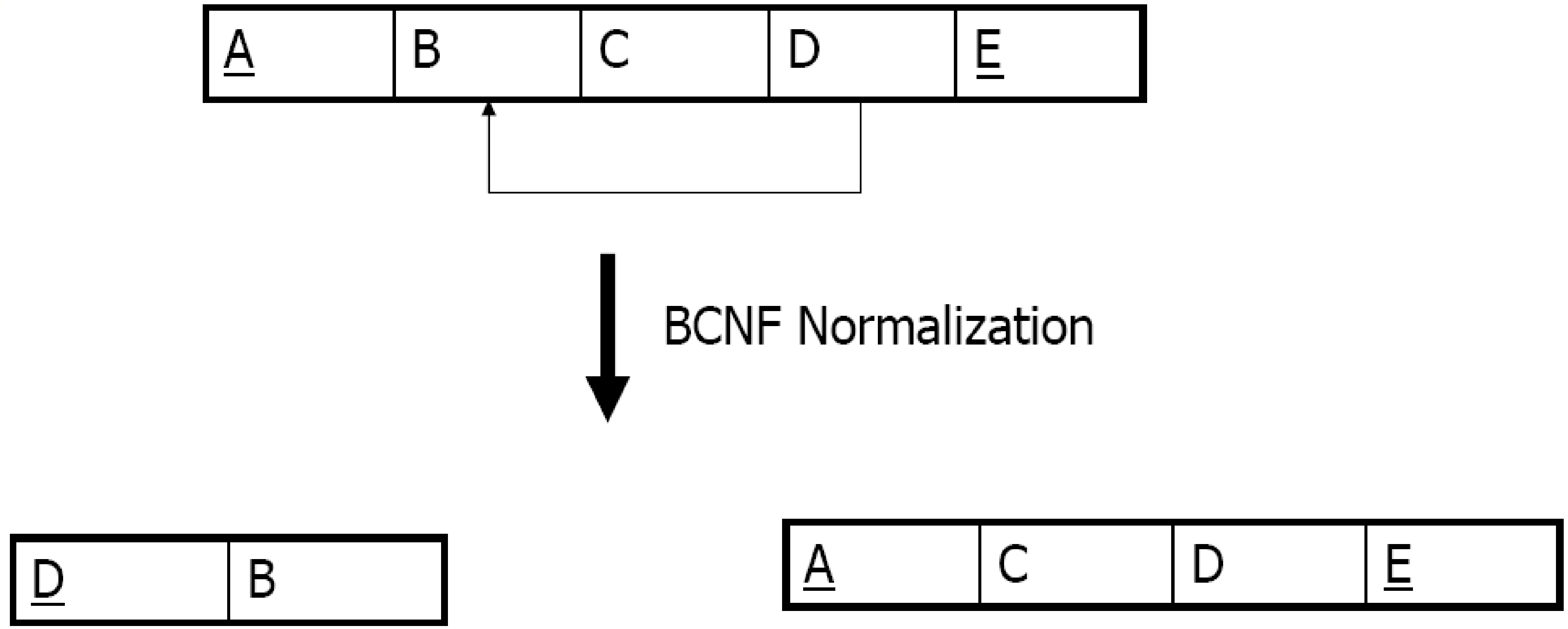
Example: 1NF to 2NF



Example: 2NF to 3NF



Example: 3NF to BCNF



Denormalize, Always Normalize or Not?

- In order to meet performance requirements, you may have to Denormalize portions of the database design.

- Sometimes it is inconvenient to normalize. The semantics of the design will probably become clearer.
  - For instance in an address directory it might be good to have both zip code and city in the same table as the street address despite its violation of the 3NF.

Summary

- ✓ A table is in 1NF when all the key attributes are defined (no repeating groups in the table) and when all remaining attributes are dependent on the primary key.
- ✓ A table is in 2NF when it is in 1NF and it includes no partial dependencies.
- ✓ A table is in 3NF when it is in 2NF and it contains no transitive dependencies.
- ✓ A table is in Boyce-Codd Normal Form (BCNF) when it is in 3NF and every determinant in the table is a candidate key.

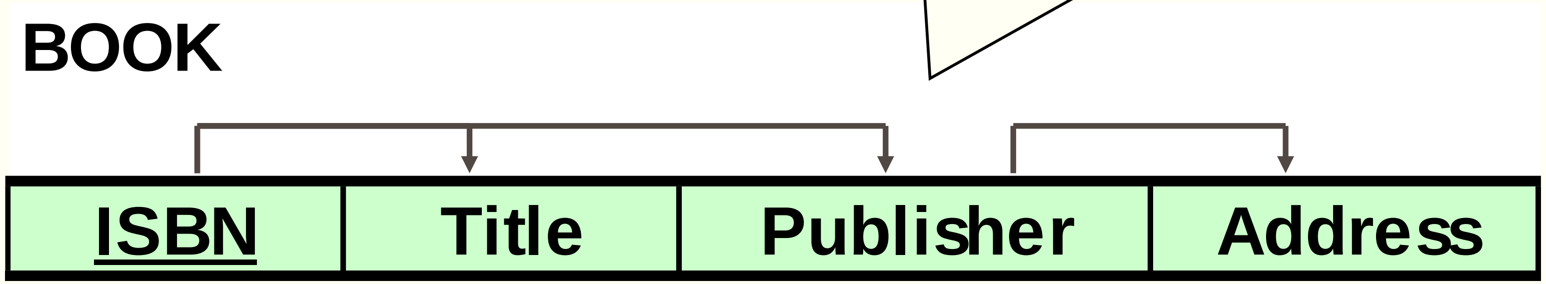
Normal Forms: Review

- Unnormalized – There are multivalued attributes or repeating groups
- 1 NF – No multivalued attributes or repeating groups.
- 2 NF – 1 NF plus no partial dependencies
- 3 NF – 2 NF plus no transitive dependencies

Example 1: Determine NF

- ISBN → Title
- ISBN → Publisher
- Publisher → Address

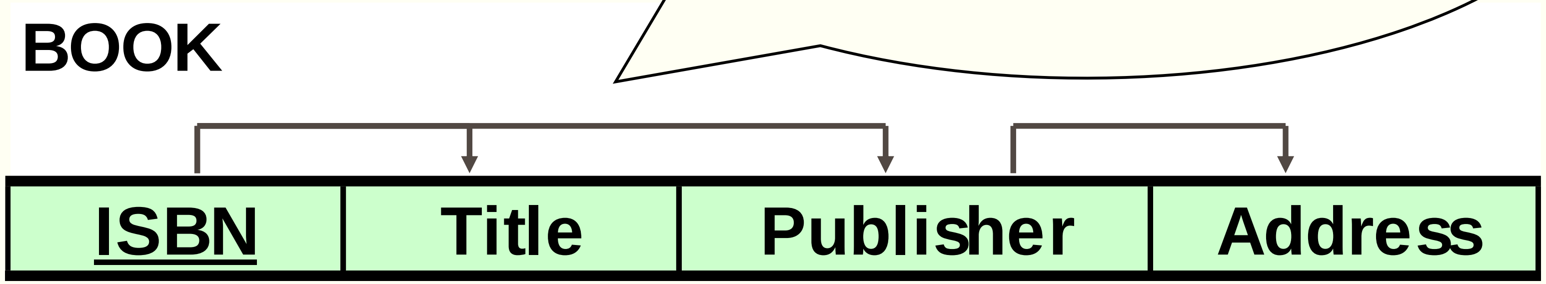
All attributes are directly or indirectly determined by the primary key; therefore, the relation is at least in 1 NF



Example 1: Determine NF

- ISBN → Title
- ISBN → Publisher
- Publisher → Address

The relation is at least in 1NF. There is no COMPOSITE primary key, therefore there can't be partial dependencies. Therefore, the relation is at least in 2NF



Example 1: Determine NF

- ISBN → Title
- ISBN → Publisher
- Publisher → Address

Publisher is a non-key attribute, and it determines Address, another non-key attribute. Therefore, there is a transitive dependency, which means that the relation is NOT in 3 NF.

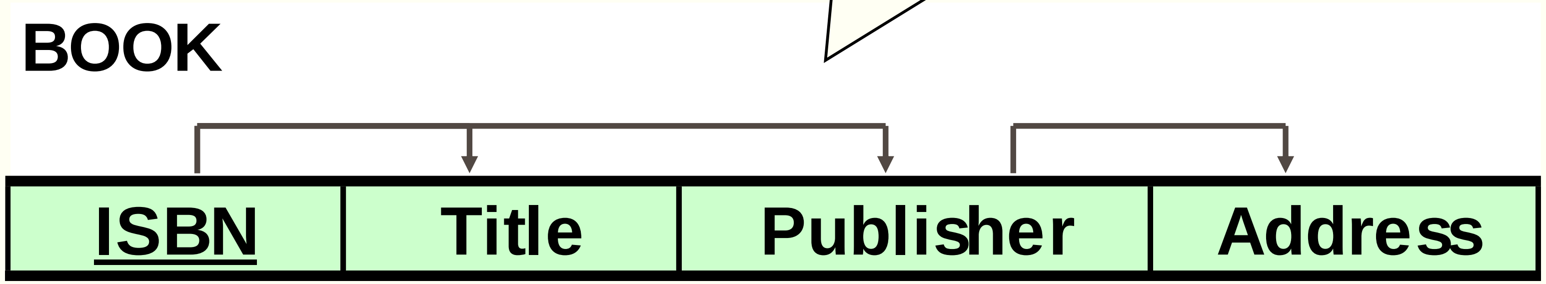
BOOK

<u>ISBN</u>	Title	Publisher	Address

Example 1: Determine NF

- ISBN → Title
- ISBN → Publisher
- Publisher → Address

We know that the relation is at least in 2NF, and it is not in 3 NF. Therefore, we conclude that the relation is in 2NF.



Example 1: Determine NF

- ISBN → Title
- ISBN → Publisher
- Publisher → Address

In your solution you will write the following justification:

1) **No M/V attributes, therefore at least 1NF**
multi/value
2) **No partial dependencies, therefore at least 2NF**
3) **There is a transitive dependency (Publisher → Address), therefore, not 3NF**

Conclusion: The relation is in 2NF

BOOK

<u>ISBN</u>	Title	Publisher	Address

Example 2: Determine NF

- Product_ID → Description

All attributes are directly or indirectly determined by the primary key; therefore, the relation is at least in 1 NF

ORDER

<u>Order_No</u>	<u>Product_ID</u>	Description

Example 2: Determine NF

- Product_ID → Description

The relation is at least in 1NF.
There is a COMPOSITE Primary Key (PK) (Order_No, Product_ID), therefore there can be partial dependencies. Product_ID, which is a part of PK, determines Description; hence, there is a partial dependency. Therefore, the relation is not 2NF. No sense to check for transitive dependencies!

ORDER

Order_No	Product_ID	Description

Example 2: Determine NF

- Product_ID → Description

We know that the relation is at least in 1NF, and it is not in 2 NF. Therefore, we conclude that the relation is in 1 NF.

ORDER

Order_No	Product_ID	Description

Example 2: Determine NF

- Product_ID → Description

In your solution you will write the following justification:

1) No M/V attributes, therefore at least 1NF

2) There is a partial dependency (Product_ID → Description), therefore not in 2NF

Conclusion: The relation is in 1NF

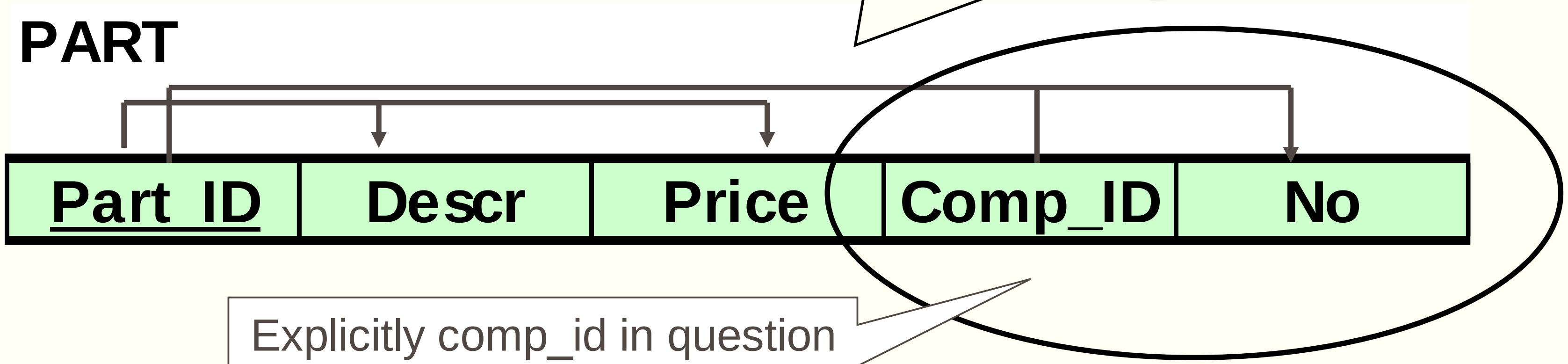
ORDER

Order_No	Product_ID	Description

Example 3: Determine NF

- Part_ID → Description
- Part_ID → Price
- Part_ID, Comp_ID → No

Comp_ID and No are not determined by the primary key; therefore, the relation is NOT in 1 NF. No sense in looking at partial or transitive dependencies.



Explicitly comp_id in question not depend on PK

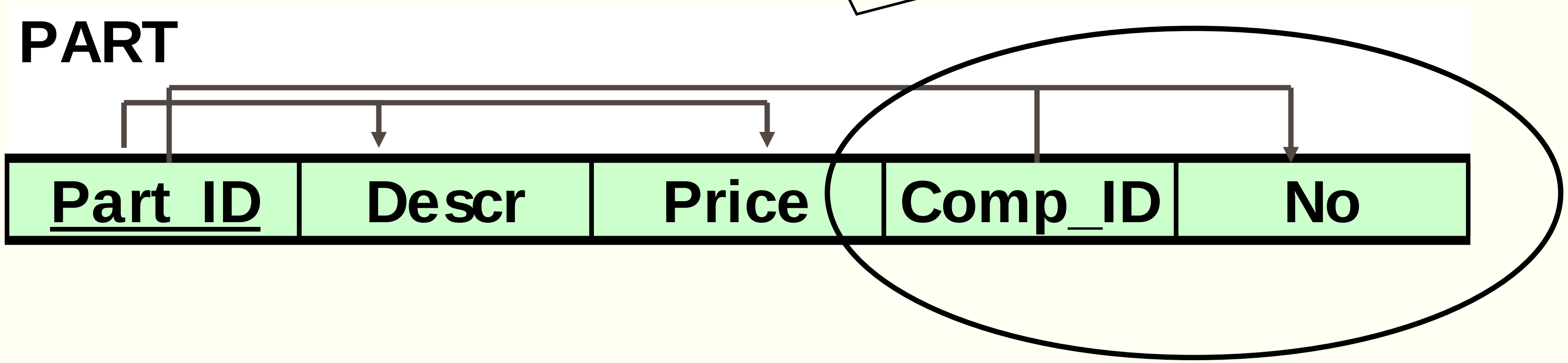
Example 3: Determine NF

- Part_ID → Description
- Part_ID → Price
- Part_ID, Comp_ID → No

In your solution you will write the following justification:

1) There are M/V attributes; therefore, not 1NF

Conclusion: The relation is not normalized.



Bringing a Relation to 1NF

STUDENT

<u>**Stud_ID**</u>	**Name**	**Course_ID**	**Units**
101	Lennon	MSI 250	3.00
101	Lennon	MSI 415	3.00
125	Johnson	MSI 331	3.00

Bringing a Relation to 1NF

- Option 1: Make a determinant of the repeating group (or the multivalued attribute) a part of the primary key.

Composite Primary Key

STUDENT

Stud_ID	**Name**	**Course_ID**	**Units**
101	Lennon	MSI 250	3.00
101	Lennon	MSI 415	3.00
125	Johnson	MSI 331	3.00

Bringing a Relation to 1NF

- Option 2: Remove the entire repeating group from the relation. Create another relation which would contain all the attributes of the repeating group, plus the primary key from the first relation. In this new relation, the primary key from the original relation and the determinant of the repeating group will comprise a primary key.

STUDENT

Stud_ID	**Name**	**Course_ID**	**Units**
101	Lennon	MSI 250	3.00
101	Lennon	MSI 415	3.00
125	Johnson	MSI 331	3.00

Bringing a Relation to 1NF

STUDENT

Stud_ID	Name
101	Lennon
125	Jonson

STUDENT_COURSE

Stud_ID	Course	Units
101	MSI 250	3
101	MSI 415	3
125	MSI 331	3

Bringing a Relation to 2NF

Composite Primary Key

STUDENT

Stud_ID	Name	Course_ID	Units
101	Lennon	MSI 250	3.00
101	Lennon	MSI 415	3.00
125	Johnson	MSI 331	3.00

Bringing a Relation to 2NF

- Goal: Remove Partial Dependencies

Composite Primary Key

Partial Dependencies

STUDENT

<u>**Stud_ID**</u>	**Name**	<u>**Course_ID**</u>	**Units**
101	Lennon	MSI 250	3.00
101	Lennon	MSI 415	3.00
125	Johnson	MSI 331	3.00

Bringing a Relation to 2NF

- Remove attributes that are dependent from the part but not the whole of the primary key from the original relation. For each partial dependency, create a new relation, with the corresponding part of the primary key from the original as the primary key.

STUDENT

<u>**Stud_ID**</u>	**Name**	<u>**Course_ID**</u>	**Units**
101	Lennon	MSI 250	3.00
101	Lennon	MSI 415	3.00
125	Johnson	MSI 331	3.00

Bringing a Relation to 2NF

CUSTOMER

Stud_ID	Name	Course_ID	Units
101	Lennon	MSI 250	3.00
101	Lennon	MSI 415	3.00
125	Johnson	MSI 331	3.00

STUDENT_COURSE

Stud_ID	Course_ID
101	MSI 250
101	MSI 415
125	MSI 331

COURSE

Course_ID	Units
MSI 250	3.00
MSI 415	3.00
MSI 331	3.00

STUDENT

Stud_ID	Name
101	Lennon
125	Johnson

Bringing a Relation to 3NF

- Goal: Get rid of transitive dependencies.

Transitive Dependency

EMPLOYEE

<u>Emp_ID</u>	F_Name	L_Name	Dept_ID	Dept_Name
111	Mary	Jones	1	Acct
122	Sarah	Smith	2	Mktg

Bringing a Relation to 3NF

- Remove the attributes, which are dependent on a non-key attribute, from the original relation. For each transitive dependency, create a new relation with the non-key attribute which is a determinant in the transitive dependency as a primary key, and the dependent non-key attribute as a dependent.

EMPLOYEE

<u>Emp_ID</u>	F_Name	L_Name	Dept_ID	Dept_Name
111	Mary	Jones	1	Acct
122	Sarah	Smith	2	Mktg

Bringing a Relation to 3NF

EMPLOYEE

Emp_ID	**F_Name**	**L_Name**	**Dept_ID**	**Dept_Name**
111	Mary	Jones	1	Acct
122	Sarah	Smith	2	Mktg

EMPLOYEE

Emp_ID	**F_Name**	**L_Name**	**Dept_ID**
111	Mary	Jones	1
122	Sarah	Smith	2

DEPARTMENT

Dept_ID	**Dept_Name**
1	Acct
2	Mktg

Any Question ?