

Unit 4: Normalization.

2022/2023

Contents

- 4.1. Introduction.
- 4.2. Normalization.
- 4.3. Functional dependencies.
- 4.5. Fully functional dependencies.
- 4.5. 1st Normal Form (1NF).
- 4.6. 2nd Normal Form (2NF).
- 4.7. 3rd Normal Form (3NF).
- 4.8. Boyce-Codd Normal Form (BCNF).
- 4.9. Multivalued dependency.
- 4.10. 4th Normal Forms (4NF).
- 4.11. 5th Normal Forms (5NF).
- 4.12. Conclusions.

4.1. Introduction (I).

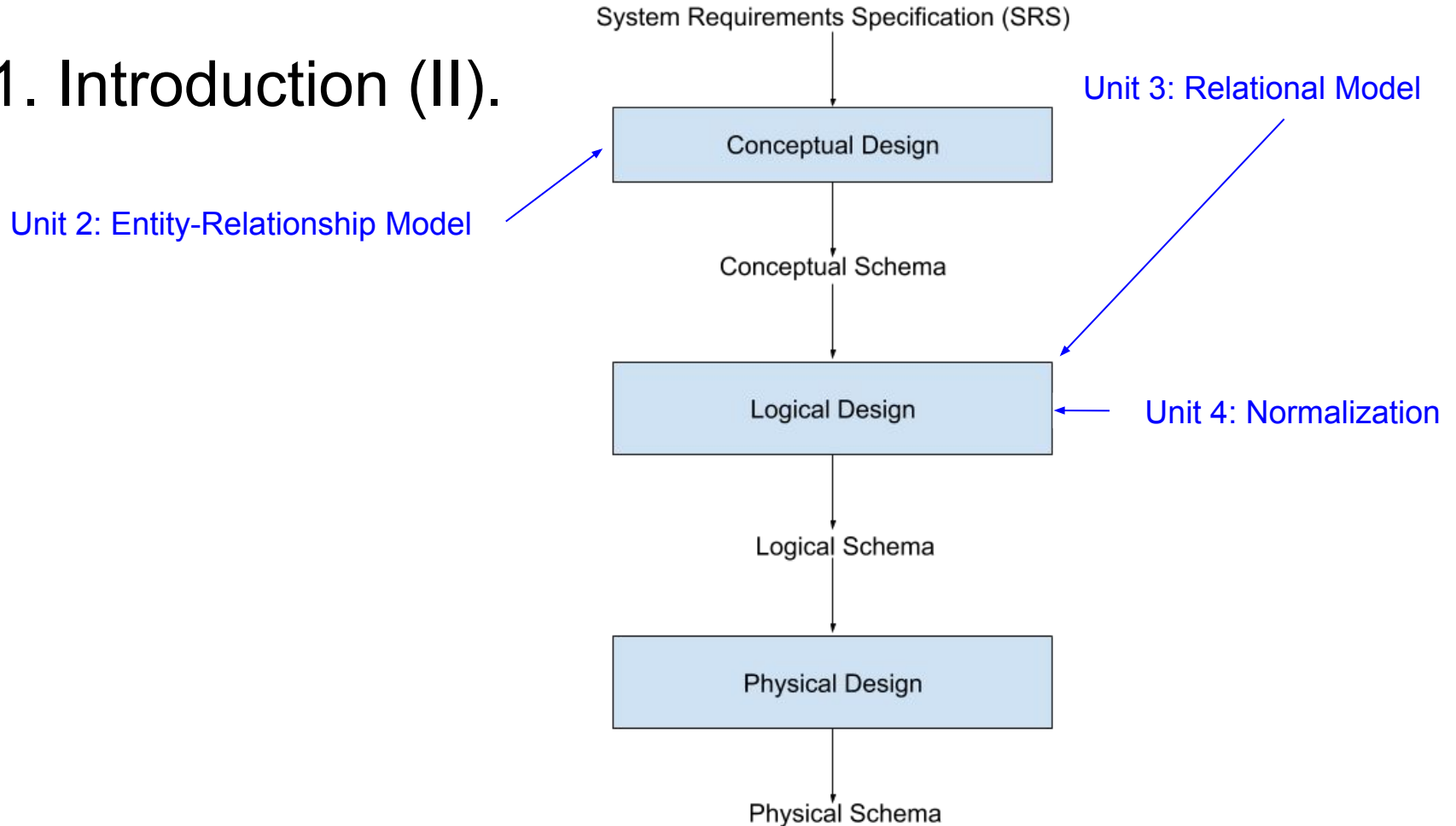
In **entity relationship model** unit we were working with conceptual design, and in **relational model** unit we were working with the logical design of a database.

Firstly, we must define the **logical design** (independent to the DBMS used). After that, **physical design** can be considered (logical data structures are translated into the physical data structures handled by a specific DBMS).

Physical design by definition is **dependent on the DBMS used**, while the logical design must be completely independent to the physical design.

In this unit we will continue with the **logical design**.

4.1. Introduction (II).



4.1. Introduction (III).

- We will try to **design application data independent to the apps**:
 - At **design time** of a system, our **customers will explain to us the way to use their data**: they will explain their work procedures. But, procedures change frequently (change of managers, change of workers, small changes in the business, etc.).
 - We want a **robust data design**: based on the nature of data and being able to **support changes in work procedures (they are less stable than the nature of data)**.
- We are interested in creating a **logical data design, independent of: computer equipment, operating system, DBMS, manipulation language, users**, etc. All this is valid for U2 and U3.

4.2. Normalization (I).

- **Codd (1972)** introduced the concept of **normalization**.
- We may say about normalization: “the **relational database** is **'well-formed'**”.
- **Normalization** is a database design technique which organizes tables in a way that **redundancy and dependency of data is reduced**.
- A database will be well-formed if we reach –at least– the **3NF**.

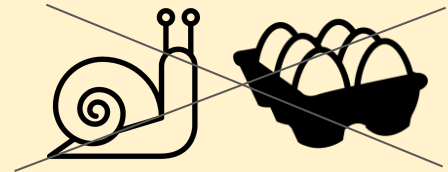
4.2. Normalization (II).

- In other words, **normalization is the process of reorganizing the tables of a relational database** so that it accomplish two basic requirements:

1. There is **no redundant data**, and

2. data **dependencies are logical** (all related data items are stored together).

Normalization



4.2. Normalization (III).

Normalization is also known as **data normalization**. History:


- Codd: 1NF, 2NF and 3NF.
- Boyce and Codd: BCNF.
- Fagin: 4NF and 5NF.



***Rarely used
and difficult!***

4.2. Normalization (IV).

*This doesn't belong to
an EMPLOYEE
relation*

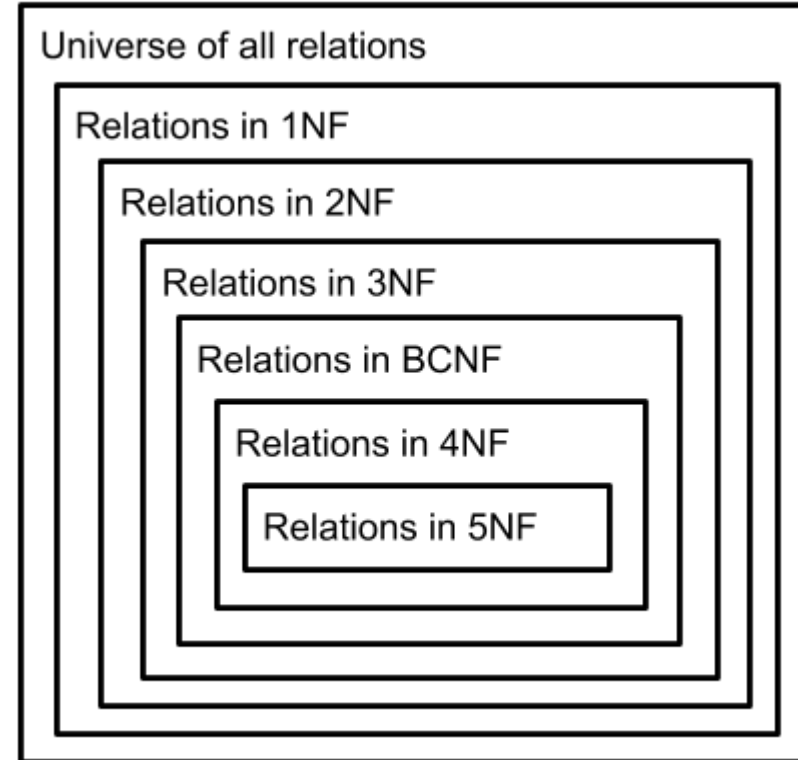


- **Example:**
 - EMPLOYEE (id_card, name, department_name, department_location)

id_card	name	department_name	department_location
11111111A	Brad Pitt	Sales	Palo Alto
22222222B	George Clooney	Research	Silicon Valley
33333333C	Jane Fonda	Accounting	Santa Barbara
44444444D	Jennifer Lawrence	Sales	Palo Alto

4.2. Normalization (V).

- A relation is in a certain normal form if it satisfies a certain set of restrictions.
- Interpretation of the picture:
 - If a relation is in 3NF, it is also in 2NF and in 1NF.
 - Originally Codd defined 1NF, 2NF, and 3NF. The designer of a BD, in general, should try to make all relations **at least in 3NF**.

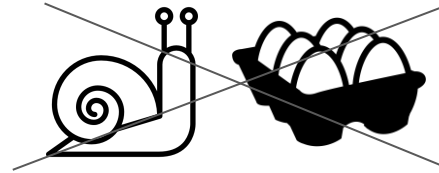


4.2. Normalization (VI).

Without normalization, three types of **data anomalies** can occur: **update, deletion and insertion anomalies** (very undesirable in any database!). **Anomalies are avoided by the process of normalization.** Let's consider the next relation:

Employee_ID	Name	Department	Student_Group
123	S. Gomis	Maths	Primary school
234	L. Huerta	Computing	ASIX
234	L. Huerta	Computing	DAW
456	M. Garcies	History	High school
456	M. Garcies	History	Primary school

4.2. Normalization (VII).

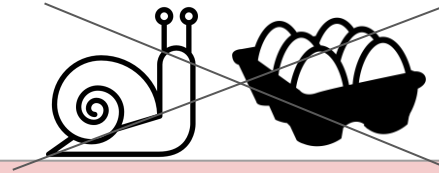


- An **update anomaly** is a data inconsistency that results from data redundancy and a partial update. For example, each employee in a company has a department associated with him/her as well as the student group he/she participates in.
- If M. Garcies' department is wrong it must be updated at least 2 times or there will be inconsistent data in the database (the same thing happens with the field 'Name'). If a user performs an update and he/she does not realize that data is stored redundantly data will be inconsistent.

Inconsistency = data differs

	Employee_ID	Name	Department	Student_Group
<i>1 time!</i>	456	M. Garcies	History Geography	High school
<i>2 times!</i>	456	M. Garcies	History	Primary school

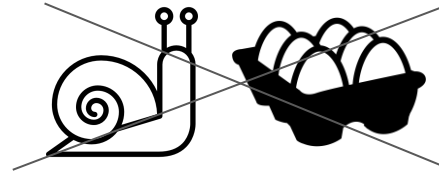
4.2. Normalization (VIII).



- A **deletion anomaly** is the unintended loss of data due to deletion of other data. For example, if the student group 'Primary school' disbanded, and it is deleted from the table above, S. Gomis and the Maths department would cease to exist. This results in database inconsistencies, and it is an example of how **combining information that does not really fit together into the same table may cause problems.**

Employee_ID	Name	Department	Student_Group
123	S. Gomis	Maths	Primary school
234	L. Huerta	Computing	ASIX
234	L. Huerta	Computing	DAW
456	M. Garcies	History	High school
456	M. Garcies	History	Primary school

4.2. Normalization (IX).



- An **insertion anomaly** is the inability to add data to the database due to absence of other data. For example, **assume Student_Group is defined as NOT NULL**. If a new employee is hired but not immediately assigned to a Student_Group then this employee can not be inserted into the database. This results in database inconsistencies due to omission. The problem is that you must know attributes that doesn't depend directly to the concept of 'teacher'.

Employee_ID	Name	Department	Student_Group
665	A. Banderas	Art	???

ERROR: Can't insert data. NULL values are not allowed in field Student_Group.

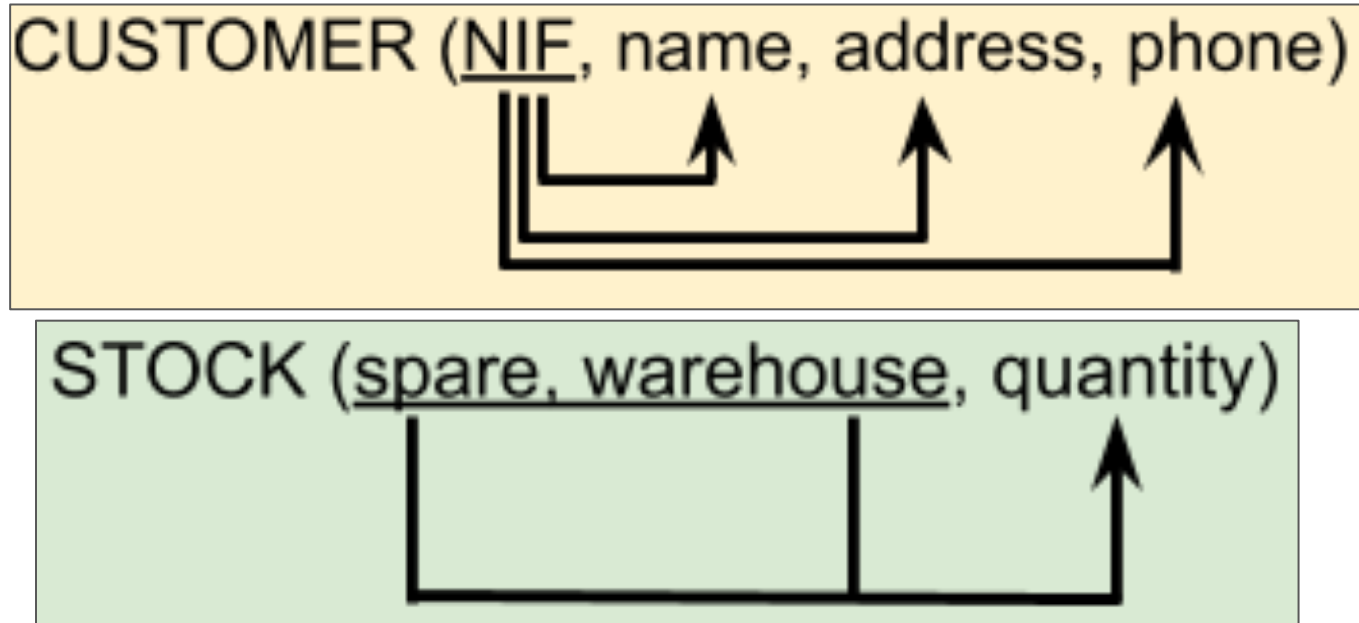
4.3. Functional dependencies (I).



- Let R be a relation, and X and Y two attributes of R . **X determines functionally Y or Y depends functionally on X if and only if:**
 - For each value of X there is one and only one image of Y (X is called determinant).
- X and Y can be composed. For instance: $(X,Z) \rightarrow Y$

4.3. Functional dependencies (II).

Examples:



4.3. Functional dependencies (III).

Examples:

- EMPLOYEE (id_card, name, address, telephone_number, INSS)

- $\text{id_card} \rightarrow \text{name}$
- $\text{id_card} \rightarrow \text{address}$
- $\text{id_card} \rightarrow \text{telephone_number}$
- $\text{id_card}, \text{name} \rightarrow \text{address}$
- $\text{address} \not\rightarrow \text{name}$
- $\text{INSS} \rightarrow \text{name}$
- etc.

- SALE (id_seller, id_customer, date, name_seller, name_customer, item)

- $\text{id_seller} \rightarrow \text{name_seller}$
- $\text{id_customer} \rightarrow \text{name_customer}$
- $\text{id_seller}, \text{id_customer}, \text{date} \rightarrow \text{item}$

4.3. Functional dependencies (IV).

Note that:

1. All attributes must depend functionally on the primary key.

2. Transitive property:

$$\left. \begin{array}{l} A \rightarrow B \\ B \rightarrow C \end{array} \right\} A \rightarrow C$$

4.4. Fully functional dependencies (I).

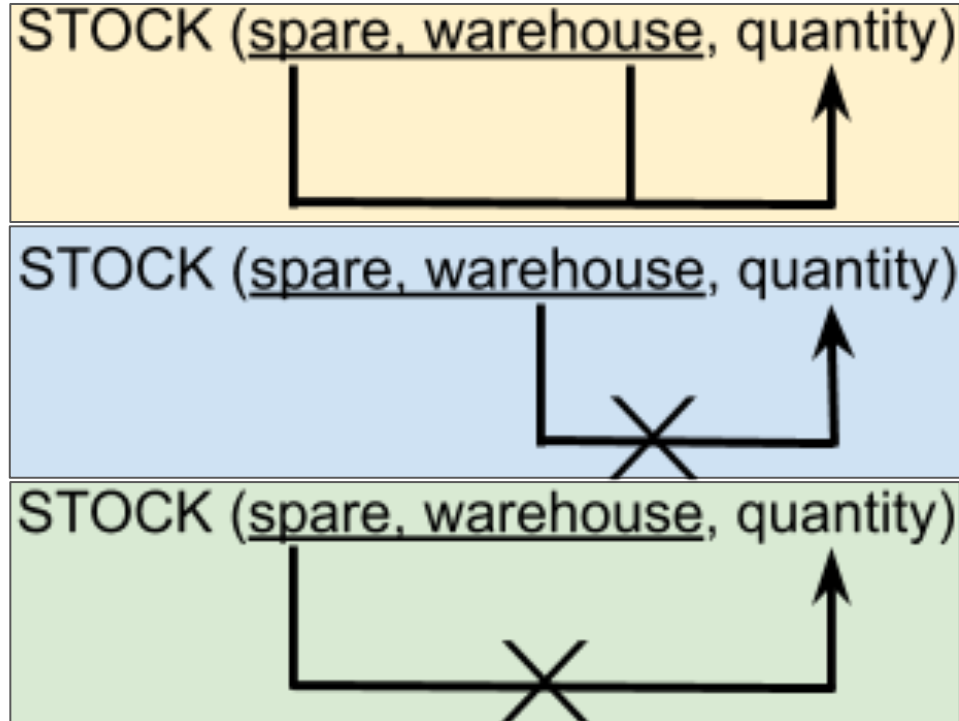


Let R be a relation, and X and Y two attributes of R . X determines fully functionally Y or Y depends fully functionally on X if and only if:

- Y depends functionally on X , and
- Y does not depend functionally on any of the subsets proper to X . For instance: $(X, Z) \Rightarrow Y$ (but $X \nrightarrow Y$ and $Z \nrightarrow Y$).

4.4. Fully functional dependencies (II).

Examples:



4.5. Fully functional dependencies (III).

The functional dependency of data is a **semantic concept**. It is not an algorithmic process.

Semantics of each organization that indicates this dependency.

4.5. 1st Normal Form (1NF).

A relation is in **1NF** if and only if:

- **the domain of each attribute contains only atomic** (indivisible) **values** (the value of each attribute contains only a single value from that domain).

With 1NF we can **avoid redundancy**.

4.5. 1st Normal Form (1NF).

Example: CUSTOMER (NIF, name, address, telephone, invoice_list, amount_invoice, date_invoice)

- This relation **is not** in 1FN because invoice_list is not an atomic attribute.
- The resulting relations in 1FN are:
 - CUSTOMER (NIF, name, address, telephone)
 - INVOICE (num_invoice, amount, date, NIF*)

4.5. 1st Normal Form (1NF).

SOURCE:

<https://www.ionos.es/digitalguide/hosting/cuestiones-tecnicas/normalizacion/>

Example: CUSTOMER (NIF, name, address, telephone)
INVOICE (num_invoice, amount, date, NIF*)

- The resulting relations in 1FN are:
 - CUSTOMER (NIF, name, street, number, door, telephone)
 - INVOICE (num_invoice, amount, date, NIF*)

4.5. 1st Normal Form (1NF).

In other words, a relation is in 1NF if and only if:

- All its attributes contain atomic values (there are no repetitive groups).

Example:

PACIENT	NURSE
111	1
112	3, 4
115	1, 2, 8

*That's
not 1NF!!*



PACIENT	NURSE
111	1
112	3
112	4
115	1
115	2
115	8

4.5. 1st Normal Form (1NF).

Example:

INVOICES (num, date, {item, quantity, price})

<1, '02/02/2020', {<5,3,99.99>, <6,1,20.00>, <10,1,125.50>}>

1NF:

INVOICES (num_invoice, date, item, quantity, price)

INVOICES (1, '02/02/2020', 5, 3, 99.99)

(1, '02/02/2020', 6, 1, 20.00)

(1, '02/02/2020', 10, 1, 125.50)

*But it is not in the 3NF,
and we want at least
3NF!!*

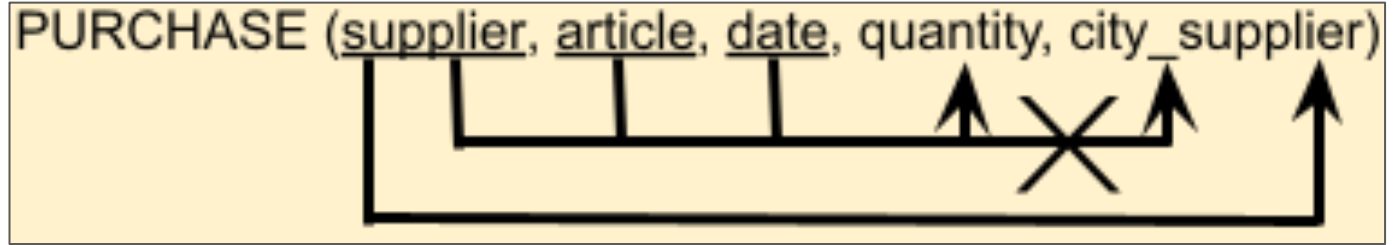
4.6. 2nd Normal Form (2NF).

A relationship is in **2NF** if and only if:

- It is in **1NF**.
- **Any attribute** that is **not part** of the **primary key** depends **fully functionally** on the **entire primary key**.

4.6. 2nd Normal Form (2NF).

Example:



SUPPLIERS (supplier, city_supplier)

PURCHASES (supplier*, article, date, quantity)

In other words, considering R (A, B, C, D, E):

- R1 (A, E) $A \rightarrow E$
- R2 (A*, B, C, D) $A, B, C \Rightarrow D$

4.6. 2nd Normal Form (2NF).

2NF theorem:

- R (A, B, C, D) and $A \rightarrow D$ thus:
 - R1 (A, D)
 - R2 (A*, B, C)
- This theorem is only a clue... You must think about the translation in every single case...

4.6. 2nd Normal Form (2NF).

Exercise:

- Transform to 2NF (identify functional dependencies):
 1. STOCK (item_code, warehouse, quantity, warehouse_phone_num)
 2. BOOKS (isbn, library_code, quantity, library_address, title)

4.7. 3rd Normal Form (3NF).

A relation is in **3NF** if and only if:

- It is in **2NF**.
- **Any attribute** that is **not part** of the **primary key** does not functionally depend on any other attribute that is not key (each non-key attribute depends only on the PK or there are no transitive dependencies among non-key attributes).

4.7. 3rd Normal Form (3NF).

Example:



It is not in 3NF because location_department also functionally depends on the department, which is a non-key attribute. The resulting relationships in 3FN are:

EMPLOYEE (employee, department*)
DEPARTMENT (department, location)

In other words, considering $R(\underline{A}, B, C)$:

- $R1(\underline{A}, B^*) \quad A \rightarrow B$
- $R2(\underline{B}, C) \quad B \rightarrow C$

4.7. 3rd Normal Form (3NF).

3NF theorem:

- $R(\underline{A}, B, C)$ and $B \rightarrow C$ thus:
 - $R1(\underline{A}, B^*)$
 - $R2(\underline{B}, C)$
- This theorem is only a clue... You must think about the translation in every single case...

4.7. 3rd Normal Form (3NF).

Exercise:

- Transform to 3NF (identify functional dependencies):
 - a. HOSPITALS (name, town, county, num_of_beds, company, hospital_address, company_address)

4.8. Boyce-Codd Normal Form (BCNF).

A relation is in **BCNF** if and only if:

- It is in **3NF**, and
 - **Every determinant is a candidate key.**
-
- Recall, a **determinant** is any attribute (simple or composite) on which some other attribute is fully functionally dependent.
 - BCNF accomplish 3NF, but 3FN may not accomplish BCNF.

It happens when candidates keys are overlapped.

4.8. Boyce-Codd Normal Form (BCNF).

Example 1:

It is not in BCNF because teacher (=determinant) is not a candidate key.

- STU_TEA_SUB (student, subject, teacher) and candidate key also {student, teacher},
 - student, subject \Rightarrow teacher
 - student, teacher \Rightarrow subject
 - teacher \rightarrow subject

R1 (teacher, subject)
R2 (student, teacher*)

In other words, considering R (A, B, C) with

- A, B \Rightarrow C
- A, C \Rightarrow B
- C \rightarrow B

In BCNF:

R1 (C, B)

R2 (A, C*)

4.8. Boyce-Codd Normal Form (BCNF).

Example 2:

It is not in BCNF because postal_code (=determinant) is not a candidate key.

- STREET_MAP (street, town, postal_code) and candidate key also {street, postal_code},
 - street, town \Rightarrow postal_code
 - street, postal_code \Rightarrow town
 - postal_code \rightarrow town

R1 (postal_code, town)

R2 (street, postal_code*)

In other words, considering R (A, B, C) with

- A, B \Rightarrow C
- A, C \Rightarrow B
- C \rightarrow B

In BCNF:

R1 (C, B)

R2 (A, C*)

4.9. Multivalued dependency.

- Multivalued dependency occurs when **two attributes** in a table are **independent of each other but, both depend on a third attribute**.
- A multivalued dependency consists of at least two attributes that are dependent on a third attribute that's why **it always requires at least three attributes**.

Source: <https://www.javatpoint.com/dbms-multivalued-dependency>

4.9. Multivalued dependency.

Example: Suppose there is a bike manufacturer company which produces two colors(white and black) of each model every year.

<u>BIKE_MODEL</u>	<u>MANUF_YEAR</u>	<u>COLOR</u>
M2001	2008	White
M2001	2009	Black
M3001	2013	White
M3001	2014	Black
M4006	2017	White
M4006	2018	Black

Here columns COLOR and MANUF_YEAR are dependent on BIKE_MODEL and independent of each other.

In this case, these two columns can be called as multivalued dependent on BIKE_MODEL. The representation of these dependencies is shown below:
BIKE_MODEL \twoheadrightarrow MANUF_YEAR
BIKE_MODEL \twoheadrightarrow COLOR

This can be read as "BIKE_MODEL multidetermined MANUF_YEAR" and "BIKE_MODEL multidetermined COLOR".

4.10. 4th Normal Forms (4NF).

A relation is in **4NF** (Ronald Fagin, 1977) if and only if:

- It is in **BCNF**
- for every one of its non-trivial multivalued dependencies $X \twoheadrightarrow Y$, X is a superkey—that is, X is either a candidate key or a superset thereof.
- If for a single value of X , multiple values of Y exists, then the relation will be a **multi-valued dependency** ($A \twoheadrightarrow B$).

4.10. 4th Normal Forms (4NF).

Example 1: STUDENT

<u>STU_ID</u>	<u>COURSE</u>	<u>HOBBY</u>
21	Computer	Dancing
21	Math	Singing
34	Chemistry	Dancing
74	Biology	Cricket
59	Physics	Hockey

R(STU_ID, COURSE, HOBBY)

- STU_ID \rightarrow COURSE
- STU_ID \rightarrow HOBBY

The given STUDENT table is in BCNF.

In the STUDENT relation, the student with STU_ID with value 21, contains two courses (Computer and Math) and two hobbies (Dancing and Singing).

4.10. 4th Normal Forms (4NF).

Example 1:

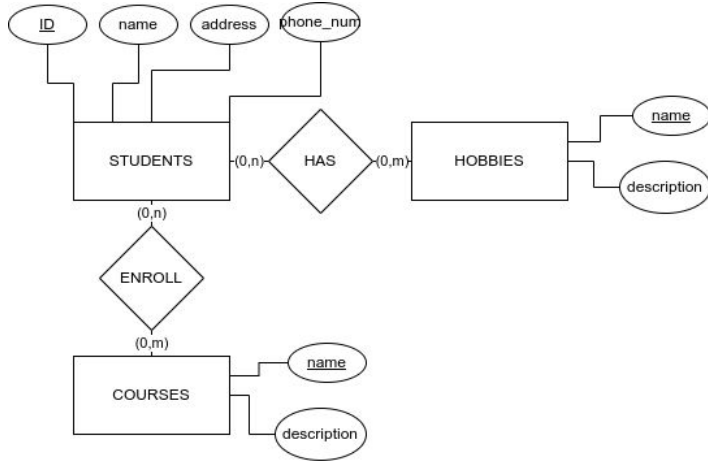
STUDENT_COURSE

<u>STU_ID</u>	<u>COURSE</u>
21	Computer
21	Math
34	Chemistry
74	Biology
59	Physics

STUDENT_HOBBY

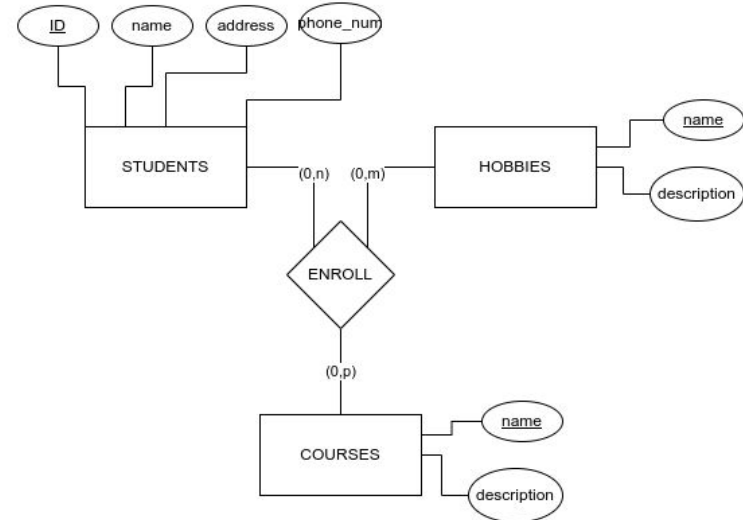
<u>STU_ID</u>	<u>HOBBY</u>
21	Dancing
21	Singing
34	Dancing
74	Cricket
59	Hockey

4.10. 4th Normal Forms (4NF).



STUDENTS (ID, name, address, phone_num)
HOBBIES (name, description)
COURSES (name, description)
STU-COU-HOB (STUDENT_ID*, COURSE_ID*, HOBBY_ID*)

STUDENTS (ID, name, address, phone_num)
HOBBIES (name, description)
COURSES (name, description)
STUDENTS-HOBBIES (STUDENT_ID*, HOBBY_ID*)
STUDENTS-COURSES (STUDENT_ID*, COURSE_ID*)



4.10. 4th Normal Forms (4NF).

Example 2: RESTAURANT

<u>Restaurant</u>	<u>Pizza Variety</u>	<u>Delivery Area</u>
A1 Pizza	Thick Crust	Springfield
A1 Pizza	Thick Crust	Shelbyville
A1 Pizza	Thick Crust	Capital City
A1 Pizza	Stuffed Crust	Springfield
A1 Pizza	Stuffed Crust	Shelbyville
A1 Pizza	Stuffed Crust	Capital City
Elite Pizza	Thin Crust	Capital City
Elite Pizza	Stuffed Crust	Capital City
Vincenzo's Pizza	Thick Crust	Springfield
Vincenzo's Pizza	Thick Crust	Shelbyville
Vincenzo's Pizza	Thin Crust	Springfield
Vincenzo's Pizza	Thin Crust	Shelbyville

R(Restaurant, Pizza Variety,
Delivery Area)

- Restaurant → Pizza Variety
- Restaurant → Delivery Area

The given RESTAURANT table is in BCNF.

In the RESTAURANT relation, the restaurant "A1 Pizza", contains two varieties of pizza ("Thick Crust" and "Stuffed Crust") and three delivery areas ("Springfield", "Shelbyville", and "Capital City").

4.10. 4th Normal Forms (4NF).

Example 2:

Varieties By Restaurant

<u>Restaurant</u>	<u>Pizza Variety</u>
A1 Pizza	Thick Crust
A1 Pizza	Stuffed Crust
Elite Pizza	Thin Crust
Elite Pizza	Stuffed Crust
Vincenzo's Pizza	Thick Crust
Vincenzo's Pizza	Thin Crust

Delivery Areas By Restaurant

<u>Restaurant</u>	<u>Delivery Area</u>
A1 Pizza	Springfield
A1 Pizza	Shelbyville
A1 Pizza	Capital City
Elite Pizza	Capital City
Vincenzo's Pizza	Springfield
Vincenzo's Pizza	Shelbyville

4.11. 5th Normal Forms (5NF).

A relation R is in **5NF** (Ronald Fagin, 1979) if and only if:

- It is in **4NF** and
 - every non-trivial join dependency in that table is implied by the candidate keys.
-
- A join dependency $\{A, B, \dots, Z\}$ on R is implied by the candidate key(s) of R if and only if each of A, B, ..., Z is a superkey for R.

Source: https://en.wikipedia.org/wiki/Fifth_normal_form

4.11. 5th Normal Forms (5NF).

Example 1:

R1

<u>Agent</u>	<u>Company</u>	<u>Product</u>
Smith	Ford	car
Smith	Ford	truck
Smith	GM	car
Smith	GM	truck
Jones	Ford	car

If agents represent companies, companies make products, and agents sell products, then we might want to keep a record of which agent sells which product for which company. This information could be kept in a table like R1.

An agent sells a certain product, and **he/she represents a company making that product, then he sells that product for that company.**

4.11. 5th Normal Forms (5NF).

Example 1 (solution):

R1

<u>AGENT</u>	<u>COMPANY</u>
Smith	Ford
Smith	GM
Jones	Ford

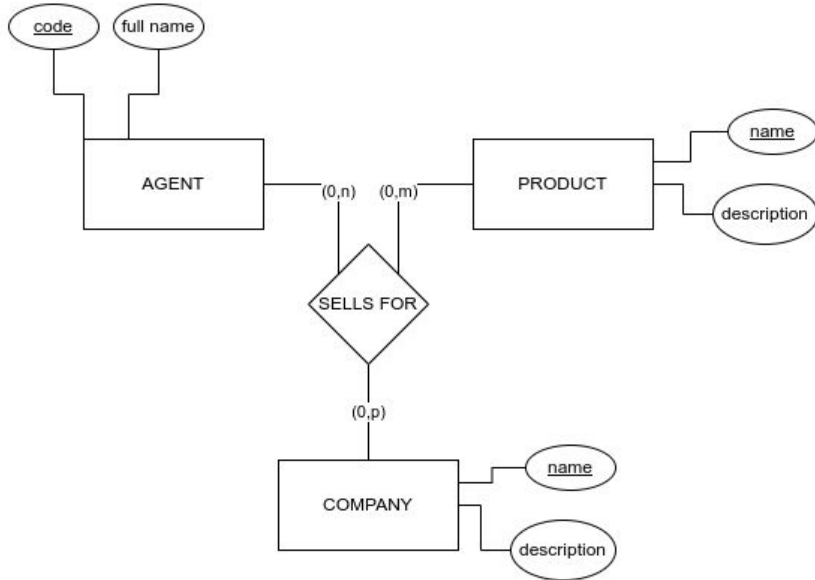
R2

<u>COMPANY</u>	<u>PRODUCT</u>
Ford	car
Ford	truck
GM	car
GM	truck

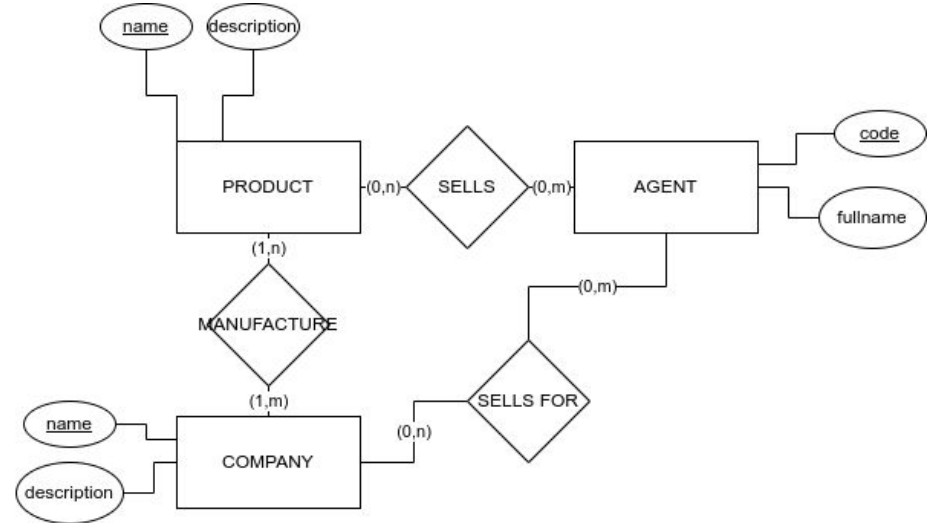
R3

<u>AGENT</u>	<u>PRODUCT</u>
Smith	car
Smith	truck
Jones	car

4.11. 5th Normal Forms (5NF).



COMPANIES (name, description)
PRODUCTS (name, description)
AGENTS (code, fullname)
MANUFACTURES (product name*, company name*)
SELLS_FOR (agent code*, company name*)
SELLS (product name*, agent code*)



COMPANIES (name, description)
PRODUCTS (name, description)
AGENT (code, fullname)
AG-CO-PRO(agent code*, product name*, company name*)

4.11. 5th Normal Forms (5NF).

Example 2:

P0

<u>SUBJECT</u>	<u>LECTURER</u>	<u>SEMESTER</u>
Computer	Anshika	Semester 1
Computer	John	Semester 1
Math	John	Semester 1
Math	Akash	Semester 2
Chemistry	Praveen	Semester 1

In the above table, John takes both Computer and Math class for Semester 1 but he doesn't take Math class for Semester 2. In this case, combination of all these fields required to identify a valid data.

Suppose we add a new Semester as Semester 3 but do not know about the subject and who will be taking that subject so we leave Lecturer and Subject as NULL. But all three columns together acts as a primary key, so we can't leave other two columns blank.

4.11. 5th Normal Forms (5NF).

Example 2 (solution):

P1

<u>SEMESTER</u>	<u>SUBJECT</u>
Semester 1	Computer
Semester 1	Math
Semester 1	Chemistry
Semester 2	Math

P2

<u>SUBJECT</u>	<u>LECTURER</u>
Computer	Anshika
Computer	John
Math	John
Math	Akash
Chemistry	Praveen

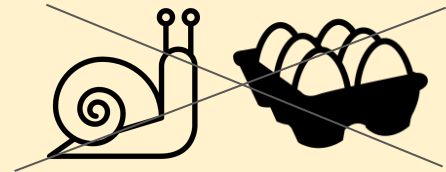
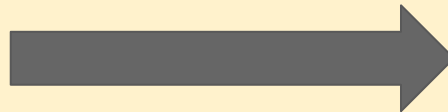
P3

<u>SEMESTER</u>	<u>LECTURER</u>
Semester 1	Anshika
Semester 1	John
Semester 1	John
Semester 2	Akash
Semester 1	Praveen

4.12. Conclusions.

- Normalization guarantees the quality of a database. Any database system should be normalized up to 3NF.
- If you translate from the ER model to the relational model in the way that we saw in this course, your database will be normalized.

Normalization



4.12. Conclusions.

To do the exercises, remember that:

- 1NF: The domain of each attribute contains only atomic (indivisible) values, and the value of each attribute contains only a single value from that domain.
- 2NF: 1NF + any attribute that is not part of the primary key depends fully functionally on the entire primary key.
- 3NF: 2NF + each non-key attribute depends only on the primary key.

Sources.

- **M. J. Ramos, A. Ramos and F. Montero.** Sistemas gestores de bases de datos. McGrawHill: 1st Edition, 2006.
- **Abraham Silberschatz, Henry F. Korth and S. Sudarshan.** *Database System Concepts*. McGrawHill: 6th Edition, 2010.
- Apunts de la UIB del professor Miquel Manresa (1996).
- <https://www.studytonight.com/dbms/>
- <https://www.geeksforgeeks.org/database-normalization-normal-forms/>
- http://databasemanagement.wikia.com/wiki/Category:Data_Anomalies
- https://en.wikipedia.org/wiki/Database_normalization
- <http://www.bkent.net/Doc/simple5.htm>