Databases with *MYSQL*

2. Data Normalization

Dante – Digital Area for Networking Teachers and Educators

Funded by the Erasmus+ Programme of the European Union

Learning Outcomes

After this lesson the learner will be able to normalize data to the 3rd Normal Form (3FN), by analyzing the user view.

He will learn all the theoretical concepts involving the relational models and how to designg the relational model following all the steps of data normalization (three normal forms).

Funded by the Erasmus+ Programme of the European Union

Data Model

Set of conceptual tools to describe data, relationships between data, data semantics and data constraints.

Funded by the Erasmus+ Programme of the European Union

Data Model

- A representation (usually graphical) of complex data structures – organizes data for different users;
- Communication tool that facilitates interaction between programmers and end users;
- The proper design of the database is carried out through an appropriate data model;
- The model shows the characteristics of the data and the relationships between the collected and stored data.

Funded by the Erasmus+ Programme of the European Union

Relacional Model

The Relational Model is a logical model based on records. In this model, data and the relationships between data are represented by tables, and all tables in the database and their respective attributes and primary keys are described in it.

Item

item_id	item_name	quantity	**supplier_id**
1	pencil	20	**2**
2	paper	35	**3**
3	note book	4	**1**

Supplier

supplier_id	supplier_name	City	Fiscal_id
1	Book Worm	New York	1111111
2	Paper Market	Paris	222222
3	School Supplies	Rome	333333

Funded by the Erasmus+ Programme of the European Union

Relacional Model

- Each column in this example corresponds to a **field** or attribute of the table and each row corresponds to a **record** or row.
- In the relational model, all data, as well as the relationship between them, are represented by a set of tables.
- Each table has a name by which it is referenced.
- Each column has a name, which refers to an attribute or field within it.
- Each row or record in the table represents a single entity or a relationship between entities.

Funded by the Erasmus+ Programme of the European Union

Relacional Model

- The domain of all attributes must consist of atomic values.
- The order in which the attributes (columns) appear is not important and can be changed without changing the meaning of the tuple.
- The order in which the records (lines) appear is not important and can be changed without changing the meaning of the tuple. Each tuple represents an entity (or relationship) and this is only determined by the existing values in each field, and not by its position in the table.

Funded by the Erasmus+ Programme of the European Union

Primary Key

- Each table must have a primary key, which is a field or set of fields that make it possible to uniquely identify any record in the table.
- Primary key fields always have a unique, non-null value for any record in the table.

Supplier

supplier_id	supplier_name	City	Fiscal_id
1	Book Worm	New York	1111111
2	Paper Market	Paris	222222
3	School Supplies	Rome	333333

Primary Key

Funded by the Erasmus+ Programme of the European Union

Candidate key

The different primary key alternatives are called candidate keys.

What are the possible alternatives for the supplier table?

Fiscal_id may be an alternative primary key, therefore it is a candidate key

Funded by the Erasmus+ Programme of the European Union

Foreign Key

Foreign keys identify relationships between entities. A foreign key is a table field that is a primary key in another table

Item

item_id	item_name	quantity	**supplier_id**
1	pencil	20	**2**
2	paper	35	**3**
3	note book	4	**1**

Foreign Key

Supplier

supplier_id	supplier_name	City	Fiscal_id
1	Book Worm	New York	1111111
2	Paper Market	Paris	222222
3	School Supplies	Rome	333333

Primary Key

If the primary key of the supplier table is the supplier_id, then this field makes the relationship between the item and the supplier and is considered a foreign key in the item table. It is this key that determines who is the supplier of each item.

Funded by the Erasmus+ Programme of the European Union

Data Integrity

In a database it is necessary to maintain the integrity of the data.

- **Entity integrity** - attribute values that correspond to the primary key cannot be null or equal to others already existing in the table.
- **Referential integrity** - the foreign key value of one table must always reference an existing primary key in another table. The fact that the foreign key is null means that it does not reference any primary key in another table.
- **Compliance with business rules** - This is ensured by defining situation-specific restrictions to be addressed to ensure that the values to be stored in the tables are valid and meaningful.

Funded by the Erasmus+ Programme of the European Union

Data Integrity

- The database designer can ensure that the integrity rules are complied with in a database by validating data when entering, deleting or changing records.
- In *Oracle, MySQL* and in other databases, some of these rules can be defined through the creation of tables.
- If the primary keys, foreign keys and constraints on each table are well defined, the database engine itself does not allow handling data in the database that violates the integrity rules.

Funded by the Erasmus+ Programme of the European Union

Objectives to be achieved in a database design project

1. Allow all relevant data to be saved in the database.

- Determine all objectives that are relevant in the database;
- Determine which relationships are necessary;
- Decide which attributes to place in each relation (table).

Funded by the Erasmus+ Programme of the European Union

Objectives to be achieved in a database design project
2. Eliminate all redundant data.

There may be duplicate data in a table that is not necessarily redundant.

Supplier		
supplier_id	supplier_name	City
1	Book Worm	New York
2	Paper Market	Paris
3	School Supplies	Rome
4	XYZ	New York

Although there are various suppliers in the same city, it is not redundant data because it is necessary data.

Funded by the Erasmus+ Programme of the European Union

Objectives to be achieved in a database design project

2. Eliminate all redundant data.

In this case, the city field in the supplier table is redundant, since the zip_code may determine the city

Supplier

supplier_id	supplier_name	city	zip_code
1	Book Worm	New York	1234
2	Paper Market	Paris	1111
3	School Supplies	Rome	2222
4	XYZ	New York	1234

Redundant Data
Through zip_code in the supplier table, the city in the T_zip_code table can be consulted.

Funded by the Erasmus+ Programme of the European Union

Objectives to be achieved in a database design project
2. Eliminate all redundant data.

How to eliminate redundant data:

If there was only the supplier table, the most correct solution to eliminate redundant data would be to split the table in two (supplier and T_zip_code).

Supplier

supplier_id	supplier_name	zip_code
1	Book Worm	1234
2	Paper Market	1111
3	School Supplies	2222
4	XYZ	1234

T_zip_code

zip_code	city
1234	New York
1111	Paris
2222	Rome

Funded by the Erasmus+ Programme of the European Union

Objectives to be achieved in a database design project

3. Keep the number of relationships to a minimum.

Funded by the Erasmus+ Programme of the European Union

Objectives to be achieved in a database design project
4. Normalize all data relations.

Formal process of eliminating redundancy in tables.

- This process identifies the correct location of each attribute and the correct structure of relationships in a database.
- In the data normalization process, there are several normal forms that represent the different rules.
- In most cases, normalization to 3NF is sufficient, but there are cases where it is necessary to reach higher normal forms or even stop before 3NF.

Funded by the Erasmus+ Programme of the European Union

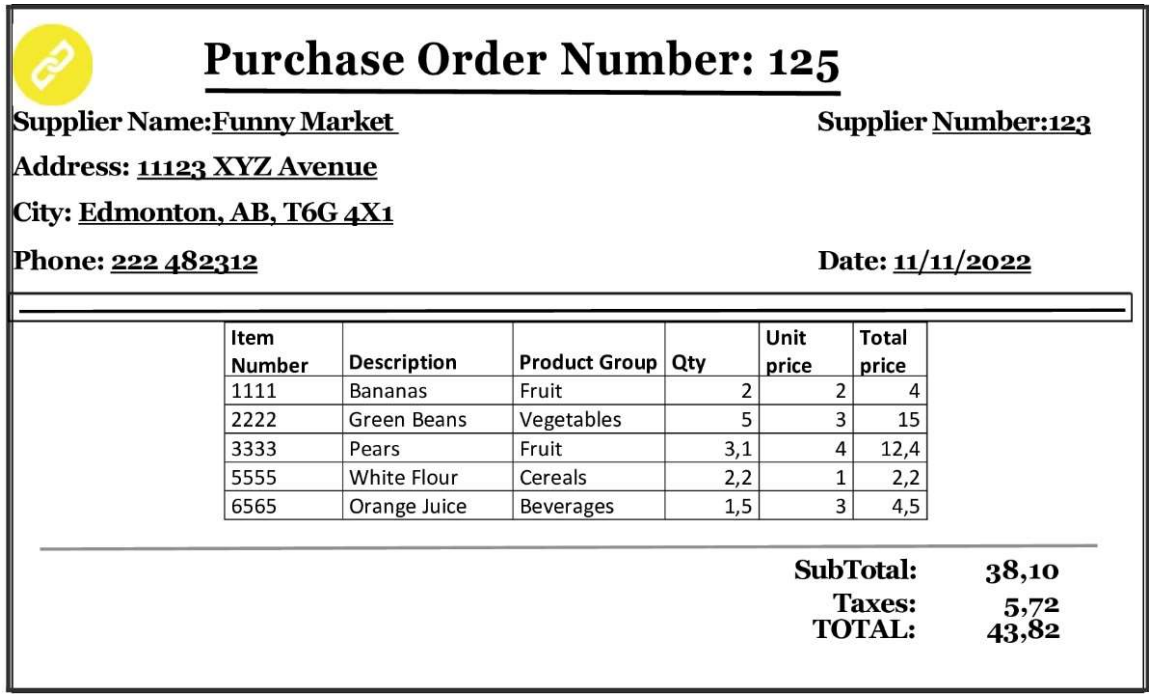
Objectives to be achieved in a database design project
4. Normalize all data relations.

- The technique used to analyze data is called normalization, which is the database logical design.
- Process from the **particular to general -** all attributes are gathered in order to identify the database tables;
- Details are obtained by analyzing samples of **user views** in applications, or prototypes.
- Applying the normalization technique to **user view** allows data presented in a complex way to be represented in a form of two-dimensional relations (tables).

Funded by the Erasmus+ Programme of the European Union

Data Normalization
User View

- User view is the view of data presented to the user by the application, ex. Set of requirements necessary to support user operations;
- User view can include input screen, output screen, input form, detailed report, summary report, etc.
- User View example:



Purchase Order Number: 125

Supplier Name: Funny Market — **Supplier Number:** 123

Address: 11123 XYZ Avenue

City: Edmonton, AB, T6G 4X1

Phone: 222 482312 — **Date:** 11/11/2022

Item Number	Description	Product Group	Qty	Unit price	Total price
1111	Bananas	Fruit	2	2	4
2222	Green Beans	Vegetables	5	3	15
3333	Pears	Fruit	3,1	4	12,4
5555	White Flour	Cereals	2,2	1	2,2
6565	Orange Juice	Beverages	1,5	3	4,5

SubTotal: 38,10
Taxes: 5,72
TOTAL: 43,82

Funded by the Erasmus+ Programme of the European Union

Data Normalization

- Data Normalization is a rigorous technique used to divide user view data into database tables, where all attributes are represented in tables;
- Normalization requires that a primary key be identified for each relation;
- The purpose of normalization is to make all table attributes **functionally dependent only on its primary key.**

Funded by the Erasmus+ Programme of the European Union

Data Normalization
Funcional Dependency *Example*

For the **SUPPLIER** table:

- The supplier_address is functionally dependent on the supplier_id.

supplier_id → ***supplier_address***

- The **supplier_id is the primary key** which is unique, its address can be determined using the supplier_id, since each supplier can only have one address.

- The supplier_address is functionally dependent on the supplier_id, but the supplier_id is not functionally dependent on the address. There may be several suppliers with the same address, although each one has a different supplier_id.

Funded by the Erasmus+ Programme of the European Union

Data Normalization
User View

- User view is the view of data presented to the user by the application, ex. Set of requirements necessary to support user operations;
- User view can include input screen, output screen, input form, detailed report, summary report;
- The user view should display a sample of the data;
- It can be a prototype or a real sample;
- Several user views must be collected in order to cover all application components;
- Data normalization is applied to each user view to determine relationships or tables.

Funded by the Erasmus+ Programme of the European Union

How to analyze user view

Steps to transform a user view into a group of normalized relations (tables):

1. Universal Relation (UNF)
2. Normalize relationships.

Funded by the Erasmus+ Programme of the European Union

Universal Relation (UNF)

The domains of all its attributes contain only atomic values;

There are no repetitive attributes describing the same characteristic, attributes that can exist more than once in each record.

Supplier

supplier_id	supplier_name	items	prices
1	Book Worm	pencil, eraser	10, 50
2	Paper Market	book, pen	11, 22
3	School Supplies	staples, clips	21, 31

Non-atomic values

Supplier

supplier_id	supplier_name	items1	item2	price1	price 2
1	Book Worm	pencil	eraser	10	50
2	Paper Market	book	pen	11	22
3	School Supplies	staples	clips	21	31

Repeated attributes

Funded by the Erasmus+ Programme of the European Union

User View
Product Order

Purchase Order Number: 125

Supplier Name:Funny Market **Supplier Number:123**

Address: 11123 XYZ Avenue

City: Edmonton, AB, T6G 4X1

Phone: 222 482312 **Date: 11/11/2022**

Item Number	Description	Product Group	Qty	Unit price	Total price
1111	Bananas	Fruit	2	2	4
2222	Green Beans	Vegetables	5	3	15
3333	Pears	Fruit	3,1	4	12,4
5555	White Flour	Cereals	2,2	1	2,2
6565	Orange Juice	Beverages	1,5	3	4,5

SubTotal: 38,10
Taxes: 5,72
TOTAL: 43,82

Funded by the Erasmus+ Programme of the European Union

Universal Relation (UNF) (Not normalized)

- Identify all attributes present in the user view

```
purchase_order_number
supplier_name
supplier_number
supplier_address
supplier_city
supplier_phone
purchase_order_date
item_number
item_description
Product_group_description
quantity
unit_price
total_price
sub_total
taxes
total
```

Funded by the Erasmus+ Programme of the European Union

Universal Relation (UNF) (Not Normalized)

Choose a designation for the relationship and list all of its attributes separated by comma ","

Put all the attributes within parenthesis "["

```
Purchase_Order[purchase_order_number,supplier_name
supplier_number, supplier_address, supplier_city,
supplier_phone, purchase_order_date, item_number,
item_description, product_group_description, quantity,
unit_price,
total_price, sub_total, taxes, total]
```

Funded by the Erasmus+ Programme of the European Union

Universal Relation (UNF)

Identify a primary key (consisting of one or several attributes) that represents the description of the user's view and underline it.

Consider all the alternative primary keys (candidate keys)

```
Purchase_Order[purchase_order_number,supplier_name
supplier_number, supplier_address, supplier_city,
supplier_phone, purchase_order_date, item_number,
item_description, product_group_description, quantity,
unit_price,
total_price, sub_total, taxes, total]
```

purchase_order_number identifies uniquely each purchase order

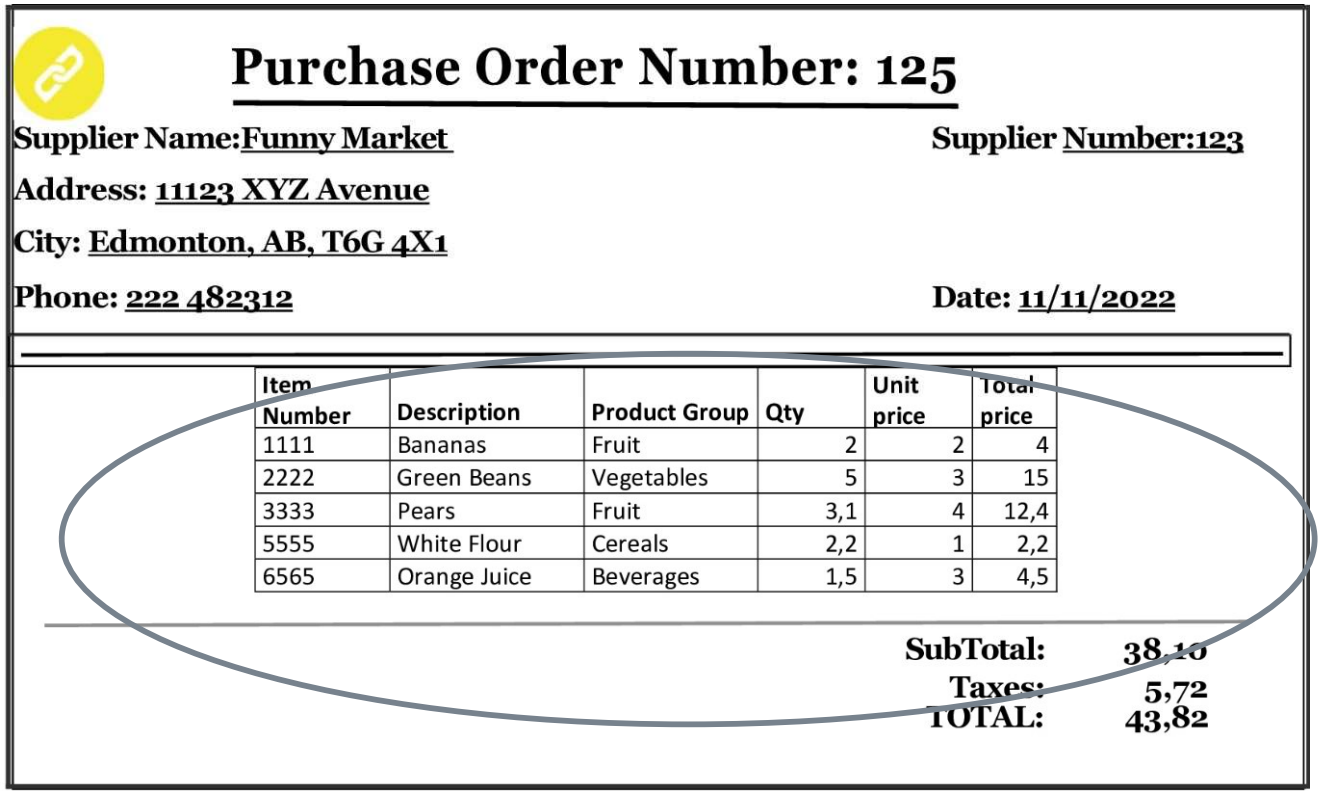
All table attributes are functionally dependente on purchase_order_number

Funded by the Erasmus+ Programme of the European Union

Universal Relation (UNF)

Determining whether the attribute is a single value or is repeated individually or as part of a set of related attributes. Identify an attribute or a group of related attributes that have more than one value.

Each of these attributes have more than one value



Purchase Order Number: 125

Supplier Name:Funny Market **Supplier Number:123**

Address: 11123 XYZ Avenue

City: Edmonton, AB, T6G 4X1

Phone: 222 482312 **Date: 11/11/2022**

Item Number	Description	Product Group	Qty	Unit price	Total price
1111	Bananas	Fruit	2	2	4
2222	Green Beans	Vegetables	5	3	15
3333	Pears	Fruit	3,1	4	12,4
5555	White Flour	Cereals	2,2	1	2,2
6565	Orange Juice	Beverages	1,5	3	4,5

SubTotal: **38,10**

Taxes: **5,72**

TOTAL: **43,82**

Funded by the Erasmus+ Programme of the European Union

Universal Relation (UNF)

Put the attributes that have more than one value within braces { } - referred to as a 'repeating group'

```
Purchase_Order[purchase_order_number,supplier_name
supplier_number, supplier_address, supplier_city,
supplier_phone, purchase_order_date, {item_number,
item_description, product_group_description, quantity,
unit_price,
total_price,} sub_total, taxes, total]
```

Funded by the Erasmus+ Programme of the European Union

Relação Universal (UNF)

Redundant data that can be derived or calculated may be eliminated

Purchase_Order[purchase_order_number, supplier_name supplier_number, supplier_address, supplier_city, supplier_phone, purchase_order_date, {item_number, item_description, product_group_description, quantity, unit_price, **total_price,**} **sub_total**, taxes, **total**]

*Total_price = quantity*unit_price ; sub_total=sum(total_price); total= sub_total+taxes*

Although sub_total may be derived, we will keep them in the table for faster processing of data

Purchase_Order[purchase_order_number, supplier_name supplier_number, supplier_address, supplier_city, supplier_phone, purchase_order_date, {item_number, item_description, product_group_description, quantity, unit_price} **sub_total**, taxes]

Funded by the Erasmus+ Programme of the European Union

Universal Relation (UNF)

Determine if atribute values are repeated and add a coded atribute.

Both "bananas" and "pears" are in the product_group "fruit"
All items may be classified in a specific group.
Therefore we may introduce a ***product_group_number*** *since many items will be in the same group.*

```
Purchase_Order[purchase_order_number,supplier_name
supplier_number, supplier_address, supplier_city,
supplier_phone, purchase_order_date, {item_number,
item_description, product_group_number,
product_group_description, quantity, unit_price}
sub_total, taxes]
```

Funded by the Erasmus+ Programme of the European Union

Example user view in UNF

```
UNF
Purchase_Order[purchase_order_number,supplier_name
supplier_number, supplier_address, supplier_city,
supplier_phone, purchase_order_date, {item_number,
item_description, product_group_number,
product_group_description, quantity, unit_price}
sub_total, taxes]
```

Sintax:

- [ ] : contains the list of attributes for the relation
- Underline attributes which are primary keys
- { } : attributes that have more than one value for a primary key value.

Funded by the Erasmus+ Programme of the European Union

Data Normalization
Eliminate other Depencies

- Depending on the primary key chosen for a relation, the following dependencies can exist in that relation:

  multi-valued, partial, transitive and functional.

- A normalized relationship should **only contain functional dependencies**.

- Normalization resolves multi-valued, partial, and transitive dependencies.

Funded by the Erasmus+ Programme of the European Union

1st Normal Form (1FN)

A relation is in 1st normal form (1NF) when:

- The primary key determines a single value for all attributes in the relations (table).
- In other words, the relation does not contain **repeating groups**.

Funded by the Erasmus+ Programme of the European Union

1st Normal Form (1FN)
Eliminate Multi-Value Dependencies

- A multi-value dependency (repeating group) occurs when a primary key value determines more than one value of a non-key attribute;

- For example, If `purchase_order_number` is used as a primary key of the Purchase Order relation, there is more than one value of

```
{item_number, item_description, product_group_number,
product_group_description, quantity, unit_price}
```

Funded by the Erasmus+ Programme of the European Union

Conversion : UNF → 1FN

- Take repeating group out of UNF

1FN
Purchase_Order[purchase_order_number,supplier_name
supplier_number, supplier_address, supplier_city,
supplier_phone, purchase_order_date, sub_total, taxes]

- Create a new relationship with the primary key of the original relationship and with the attributes within the repeating group and add an attribute to the primary key to ensure uniqueness:

1NF
Purchase_Order_item
[**purchase_order_number, item_number**, item_description,
product_group_number, product_group_description,
quantity, unit_price]

Funded by the Erasmus+ Programme of the European Union

Example in 1FN

1FN
Purchase_Order[purchase_order_number,supplier_name
supplier_number, supplier_address, supplier_city,
supplier_phone, purchase_order_date, sub_total, taxes]

Purchase_Order_item
[**purchase_order_number, item_number**, item_description,
product_group_number, product_group_description,
quantity, unit_price]

Funded by the Erasmus+ Programme of the European Union

2nd Normal Form (2FN)

- A table is in 2NF when it is in 1NF and all fields that are not part of the key are functionally dependent on the primary key.

- In other words, the relation cannot have **partial dependencies** – attribute values can be determined with only part of the key.

Funded by the Erasmus+ Programme of the European Union

2st Normal Form (2FN) Eliminate Parcial Dependencies

- A partial dependency exists when only part of a composite key determines the value of a non-key attribute;

- A, B → C but actually B → C

1NF
Purchase_Order_item
[purchase_order_number, item_number, **item_description**, **product_group_number**, **product_group_description**, quantity, **unit_price**]

item_description, product_group_number, product_group_description,and unit_price are determined by item_number

Funded by the Erasmus+ Programme of the European Union

Conversion : 1NF → 2FN

- Create new relations that consist of part of the primary key and all attributes that are determined by that part of the primary key.

2FN
Item[item_number, **item_description, product_group_number, product_group_description, unit_price**]

- Include the original relationships without the attributes partially dependent on the primary key.

2NF
Purchase_Order_item
[purchase_order_number, item_number, quantity]

Funded by the Erasmus+ Programme of the European Union

Example in 2FN

2FN
Purchase_Order[purchase_order_number,supplier_name
supplier_number, supplier_address, supplier_city,
supplier_phone, purchase_order_date, sub_total, taxes]

Item[item_number, **item_description, product_group_number, product_group_description, unit_price**]

Purchase_Order_item
[purchase_order_number, item_number, quantity]

Funded by the Erasmus+ Programme of the European Union

3rd Normal Form (3FN)

- A table is in 3NF when it is in 2NF and all attributes are functionally dependent on the primary key and only on the primary key.
- In other words, there can be no transitive functional dependencies in 3NF.

Funded by the Erasmus+ Programme of the European Union

3rd Normal Form (3FN)
Eliminate Transitive Dependencies

- Transitive dependency means that an attribute is not the primary key of the table but nevertheless identifies another attribute.

2FN
Item[item_number, item_description, **product_group_number**, **product_group_description**, unit_price]

item_number is primary key and determines the value of product_group_number, which in turn determines the product_group_description

Funded by the Erasmus+ Programme of the European Union

Conversion : 2NF → 3NF

- Create new relationships composed of attributes that are not determined by attributes other than primary key (transitive dependencies) and define the primary key of those attributes.

3FN
Product_Group[product_group_number, product_group_description]

- Include the original relationships without the attributes partially with transitive dependencies.

3FN
Item[item_number, item_description, **product_group_number**, unit_price]

Funded by the Erasmus+ Programme of the European Union

Example in 3NF

3FN
Purchase_Order[purchase_order_number,supplier_name
supplier_number, supplier_address, supplier_city,
supplier_phone, purchase_order_date, sub_total, taxes]

Purchase_Order_item
[purchase_order_number, item_number, quantity]

Item[item_number, item_description, product_group_number,
unit_price]

Product_Group[product_group_number,
product_group_description]

Funded by the Erasmus+ Programme of the European Union

Check the final result after data normalization

- The same functional dependency must not appear in more than one relation;
- There should be no redundant relations.

Funded by the Erasmus+ Programme of the European Union

Data Normalization *(conclusions)*

- Each user view always results in 1 or more relations in 1NF;
- Each relation in 1NF relation results in 1 or more relations in 2NF;
- Each relation in 2NF relation results in 1 or more relations in 3NF;
- Attributes are never lost – will always be included in one of the relations;
- A relation is never lost.

Funded by the Erasmus+ Programme of the European Union

Data Normalization *(conclusions)*

- Although a table is normalized, it can still have unwanted properties that cause the database to malfunction.
- The designer must recognize when the database has unwanted properties to determine which normal form ends the normalization process of the relationship.
- The normalization process rarely goes through all the normal forms. Often the analyst recognizes, from experience, that a given relationship is not normalized, and places it directly in 3NF or FNBC.
- In most cases, normalization to 3NF is sufficient, but there are cases where it is necessary to reach higher normal forms or even stop before 3NF.