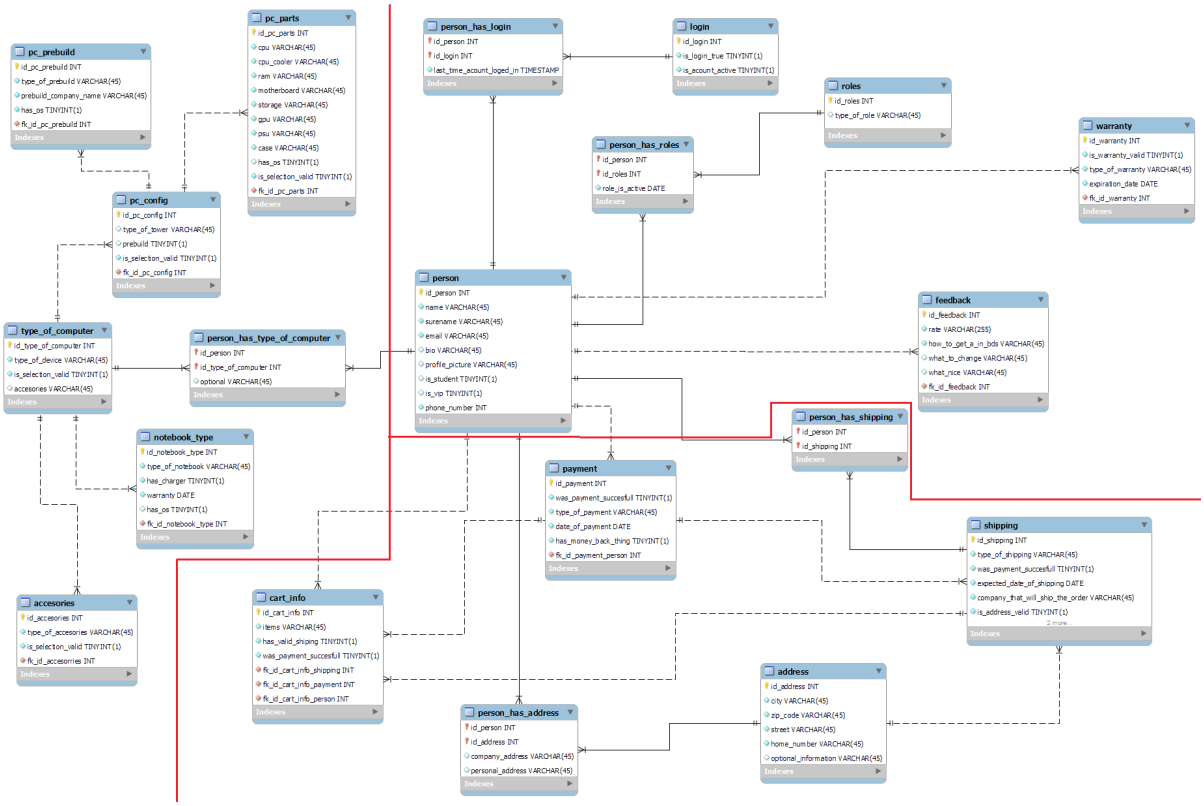
Documentation about the ‘computer store’ database

This is a really strange attempt to make a complex (in some ways), but not broken or unresponsive database, we really tried…

Function (purpose):

The idea behind this database, that I will try to explain, is that we wanted to do some sort of online shop, where a customer can come in and buy his desired tech machine he wanted for a really long time. So logically the first step is to make a “person” table, that can store some initial data and some information about customers account. we make that and i kind of think that there should be separated table for “persons_online_acount” but we decided not to. Then the second thing we did was to make a “roles” table to separate the maintainer, admin and customer himself. “Maintainer” is here for keeping the data itself in right order. Nobody wants a database full of random pointless information that takes space. Second role is an “admin”, he makes sure that the core and effectiveness of the database are at it’s peak. And lastly, there is a “customer” who is here just to “bring the moneyyy” as I would say. Then we created tables on the left part of the diagram. These are used for storing the order and selecting the things customers want to buy. They are connected to the person by “person_has_type_of_computer” table. The bottom right part is used to process the order, store shipping addresses and type of shipping itself. All customers have assigned address via table “person_has_address” that connects users to their individual address. Multiple people can have the same address. Lastly, top right part of our database contains basic user interactions, such as assigning a role via “person_has_role” table. It also contains information about warranties and feedback provided by customers. More details about each table are shown further down in this document.

We think our database is in the 3rd normal form, because we tried to take our knowledge from lectures and use it as best as we could. Non-key attributes are not functionally dependent on another non-key attributes. Atomic domains are divided into elements that are indivisible units.



- Left part is for storing the order itself, that means, what customer (or the company) wants to buy, which product it is.
- Bottom right part is dedicated for storing shipment, payment and cart info. So we can store all the things we need for the most painless process of shipping goodies to customers.
- And lastly, the top right part is just the generic stuff and other stuff that is not in common with each other, such as feedback.

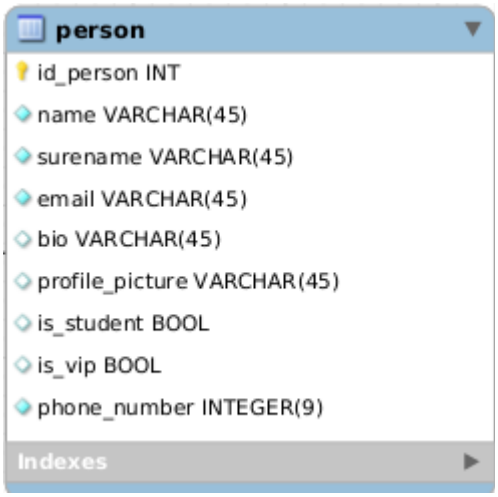
Sidenotes:

- *Main issue that I think is in our scheme is the majority of the FK keys in one table. "Cart_info" has three whole FK pointing to other tables and takes information from them.*
- *I have tried to minimize repeating the same value in multiple tables, but I've come to realization, that I need to have "is_valid" in many tables. And I honestly don't think that the solution should be to make another table with the "is_valid" variable (I am probably wrong).*

Description of the tables:

Brief description of each table and it's contents.

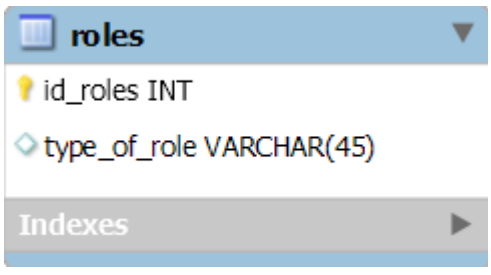
First table: "Person"



This table stores general information about people that use the store:

- *id_person,* id of person, serial and auto incrementation
- *name* , person's name, varchar because it's a string
- *surname,* person's surname, varchar because it's a string
- *email*, person's email, varchar because it's a string
- *bio*, person's bio (optional), varchar because it's a string
- *profile_picture*, link to person's profile picture (optional), varchar because it's a string
- *is_student*, tells if user is student or no, BOOL because it's true/false
- *is_vip*, tells if user is vip or no, BOOL because it's true/false
- *phone_number*, person's phone number, integer because it's a number

Second table: "roles"



This table is about roles in the database. That means that it stores a person's privilege:

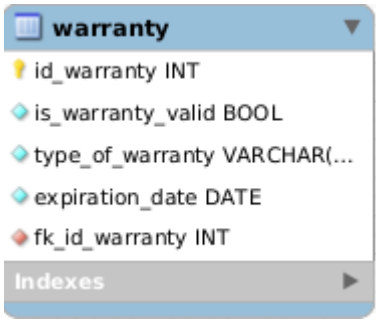
- *id_roles*, id of roles, serial and auto incrementation
- *type_of_role,* specifies what role it is, varchar because it's string

Third table: “person_has_roles”

This is the connection table between “person” and “roles”. :

- *id_person,* foreign key, int because id_person is int
- *id_roles,* foreign key, int because id_roles is int
- *role_is_active,* marks the date until which is role active, date

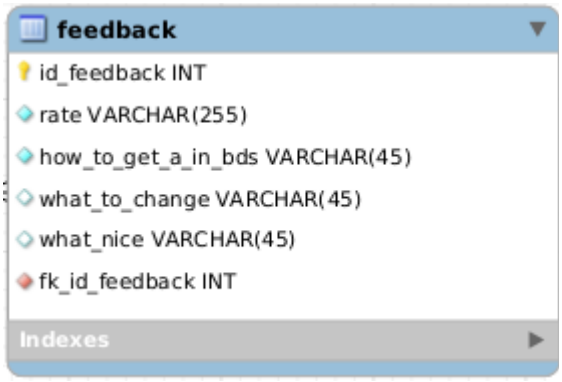
Fourth table: “warranty”



This table stores data about person’s warranty :

- *id_warranty,* id of warranty, serial and auto incrementation
- *is_warranty_valid,* shows if the warranty is valid, BOOL because it’s true/false
- *type_of_warranty,* what kind of warranty it is, varchar because it’s string
- *expiration_date,* marks the date when warranty expires, date
- *fk_id_warranty*, foreign key, connects to “person” table and uses “id_person” as FK, int

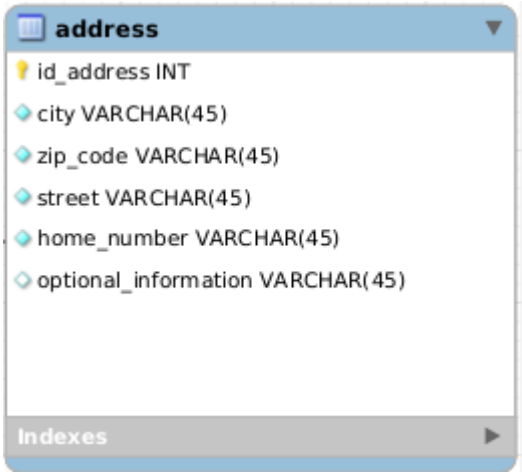
Fifth table: “feedback”



The “feedback” table is for storing data from the customers input. Here he can submit his rating and what to change on the eshop:

- *id_feedback,* id of warranty, serial and auto incrementation
- *rate*, anything customer wants to write, varchar because it’s string
- *how_to_get_a_in_bds*, customer can tell us how to get A in this subject , varchar because it’s string
- *what_to_change*, anything customer wants to see changed, varchar because it’s string
- *what_nice*, anything customer liked, varchar because it’s string
- *fk_id_feedback,* foreign key that points to “person” table and uses “id_person” as FK, int

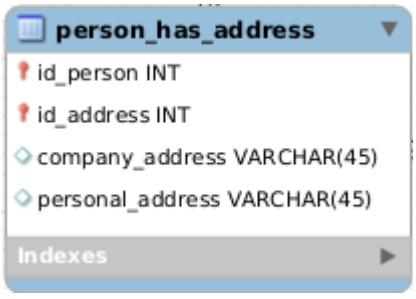
Sixth table: “address”



This table is used to store information about address that belongs to the customer or it can belong to the company:

- *id_address,* id of address, serial and auto incrementation
- *city,* varchar because it’s string
- *zip_code,* varchar because it’s string (also could have been int)
- *street,* varchar because it’s string
- *home_number,* varchar because it’s string (also could have been int)
- *optional_information,* option info about address, can be empty, varchar

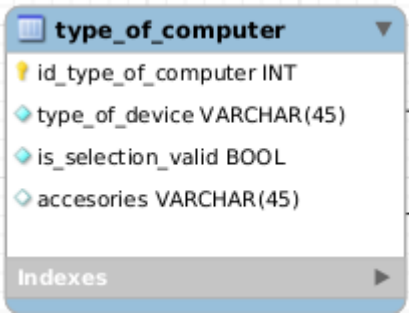
Seventh table: “person_has_address”



This specific table is a connection table between person and address tables:

- *id_person,* foreign key, int because id_person is int
- *id_address,* foreign key, int because id_address is int
- *company_address/personal_address,* can be specified what address it is, not needed and can be empty, varchar

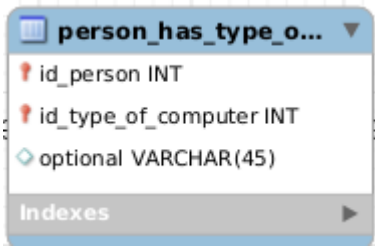
Eight table: “type_of_computer”



This table is the “entry” table as I would call it, here it stores what type of computer a person wants to buy and based on this information, it will send data to its specific tables:

- *id_type_of_computer,* id of type of computer, serial and auto incrementation
- *type_of_device*, specifies what is customer buying, varchar
- *is_selection_valid*, says if the selected product is valid, BOOL
- *accessories*, optional if customer is also buying some kind of accessory, can be null, varchar

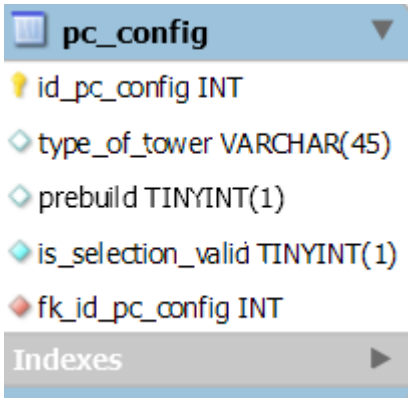
Ninth table: “person_has_type_of_computer”



This is the connecting table between “person” and “type_of_computer”:

- *id_person,* foreign key, int because id_person is int
- *id_type_of_computer,* foreign key, int because id_type_of_computer is int
- *optional,* optional information value, that can be used to closer specification of the computer purchase, if needed, varchar

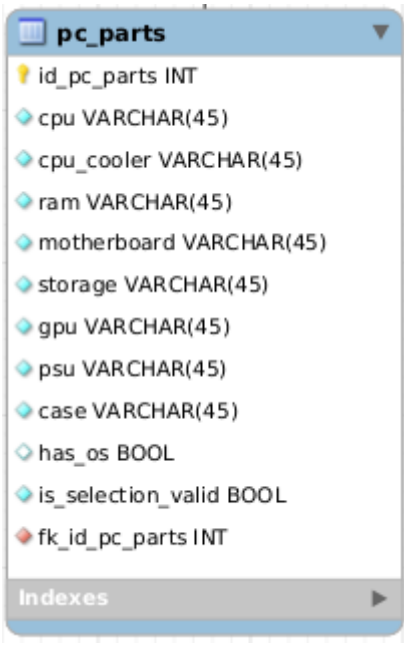
Tenth table: “pc_config”



This table is for further specification of the purchased item. If the customer decides to buy a computer:

- *id_pc_config,* id of pc config, serial and auto incrementation
- *type_of_tower*, value where customer can choose between mini, mid or big tower that fits his needs.
- *prebuild*, value,if the computer was prebuilt or not, BOOL
- *is_selection_valid*, check if the selection is valid and can be stored, BOOL
- *fk_id_pc_config*, foreign key, connects to “type_of_computer” table and uses “id_type_of_computer” as FK, int

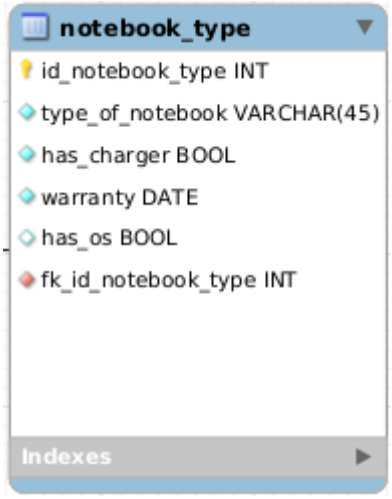
Eleventh table: “pc_parts”



This is the biggest table. Its main purpose is to store whole setup options, if the customer selects any option of a tower:

- *id_pc_parts,* id of pc parts, serial and auto incrementation
- basically all of the varchars are types of components that need to be selected and stored, it’s pointless to write them out because they are exactly the same, only difference is that it is a different component
- two BOOL variables are for checking if the customer wants an operating system and if all of his previous selections are valid
- *fk_id_pc_parts*, foreign key, connects to “pc_config” table and uses “id_pc_config” as FK, int

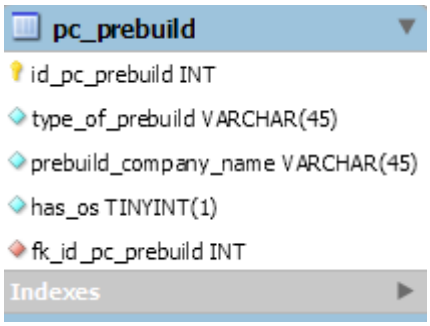
Twelfth table: “notebook_type”



This table is designed for storing notebook options, if the person selected a “notebook” in an eshop page:

- *id_notebook_type,* id of notebook type, serial and auto incrementation
- *type_of_notebook,* type of notebook the person wants to buy, varchar
- *has_charger,* does notebook have charger, BOOL
- *warranty,* date until which notebook has warranty, DATE
- *has_os*, does notebook have OS, BOOL
- *fk_id_notebook_type*, foreign key, connects to “type_of_computer” table and uses “id_type_of_computer” as FK, int

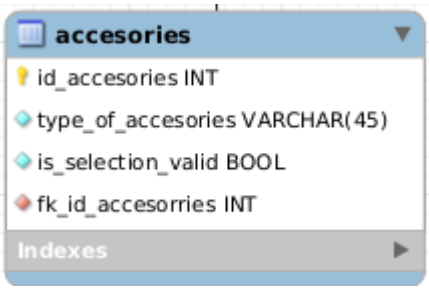
Thirteenth table: “pc_prebuild”



One of the options that customers can make when purchasing a device is prebuilt one:

- *id_pc_prebuild,* id of prebuilt pc, serial and auto incrementation
- *type_of_prebuild,* what kind of pc it is (eg. Gaming, All in one), varchar
- *prebuild_company_name,* what company built this PC, varchar
- *has_os*, does prebuild have OS, BOOL
- *fk_id_pc_prebuild*, foreign key, connects to “pc_config” table and uses “id_pc_condig” as FK, int

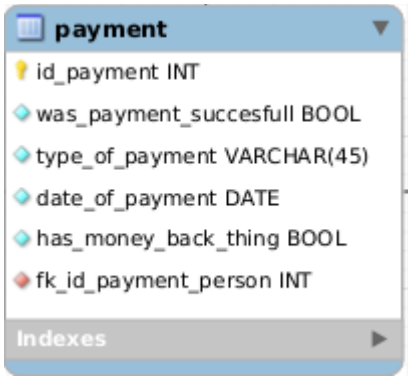
Fourteenth table: “accessories”



Accessories are an optionable table that can store things, if the customer wants to buy for example a keyboard:

- *id_accesories,* id of accessory, serial and auto incrementation
- *type_of_accesories,* what kind of accessory it is, varchar
- *is_selection_valid*, check if the selection is valid and can be stored, BOOL
- *fk_id_accesorries*, foreign key, connects to “type_of_computer” table and uses “id_type_of_computer” as FK, int

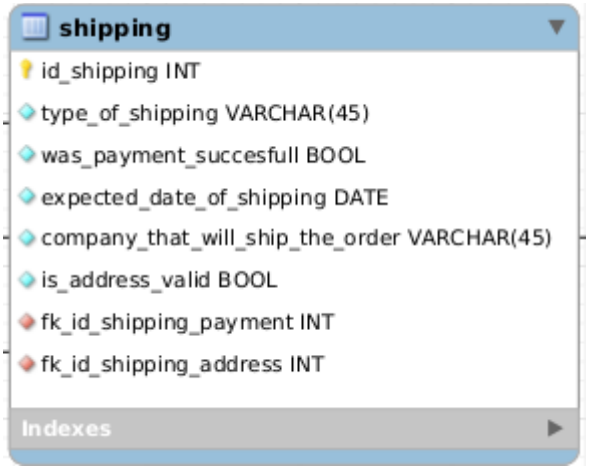
Fifteenth table: “payment”



Main purpose of this table is to store and validate payment:

- *id_payment,* id of payment, serial and auto incrementation
- *was_payment_succesfull,* confirms if payment was successful, BOOL
- *type_of_payment,* what payment method will person use, var
- *date_of_payment,* when was the payment done, DATE
- *has_money_back_thing,* value that can be true if the customer decides he wants to return product and have money back, BOOL
- *fk_id_payment_person*, foreign key, connects to “person” table and uses “id_person” as FK, int

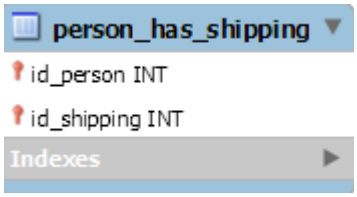
Sixteenth table: “shipping”



Shipping table is for storing the shipment information, without this table, we will never know where to ship our goods:

- *id_shipping,* id of payment, serial and auto incrementation
- *type_of_shipping*, type of shipping (eg. standard, express), varchar
- *was_payment_succesfull,* stores if payment was successful, BOOL
- *expected_date_of_shipping,* date when the shipping will be done, DATE
- *company_that_will_ship_the_order,* stores what company will ship the order, varchar
- *is_address_valid,* checks if address is valid or not, BOOL
- *fk_id_shipping_payment*, foreign key, connects to “payment” table and uses “id_payment” as FK, int
- *fk_id_shipping_address*, foreign key, connects to “address” table and uses “id_address” as FK, int

Seventeenth table: “person_has_shipping”



This is the connecting table between “person” and “shipping”:

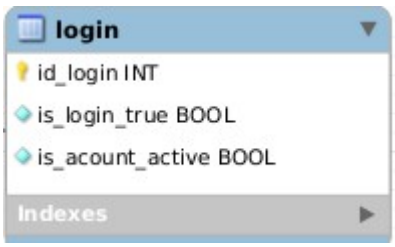
- *id_person,* foreign key, int because id_person is int
- *id_shipping,* foreign key, int because id_shipping is int

Eighteenth table: “cart_info”

This table stores cart info, shipping address and payment (by far the most scuffed table in our database):

- *id_cart_info,* id of payment, serial and auto incrementation
- *items,* what items are in the cart, varchar
- *has_valid_shipping,* does person have valid shipping address, BOOL
- *was_payment_succesfull,* was payment successful, BOOL
- *fk_id_cart_info_shipping*, foreign key, connects to “shipping” table and uses “id_shipping” as FK, int
- *fk_id_cart_info_payment*, foreign key, connects to “payment” table and uses “id_payment” as FK, int
- *fk_id_cart_info_person*, foreign key, connects to “person” table and uses “id_person” as FK, int

Nineteenth table: “login”



Login purpose table. It stores login information:

- *id_login,* id of payment, serial and auto incrementation
- *is_login_true,* checks if login info is correct or not, BOOL
- *is_acount_active,* checks if account is active or not, BOOL

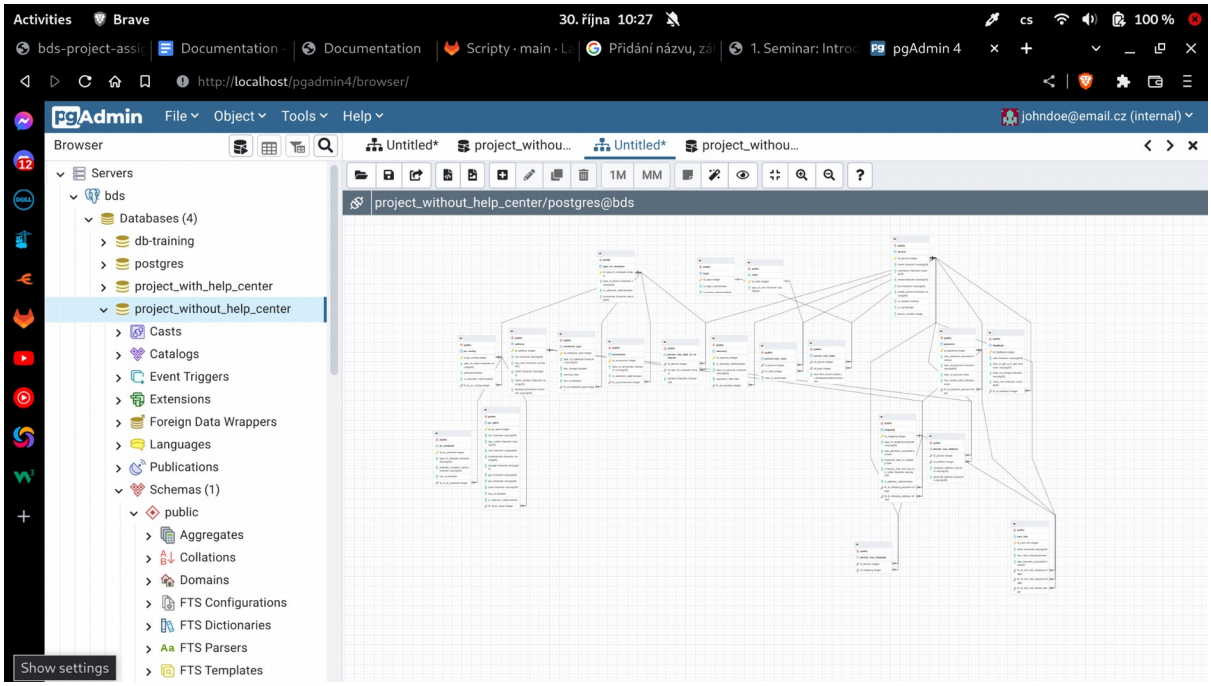
Twentieth table: “person_has_login”

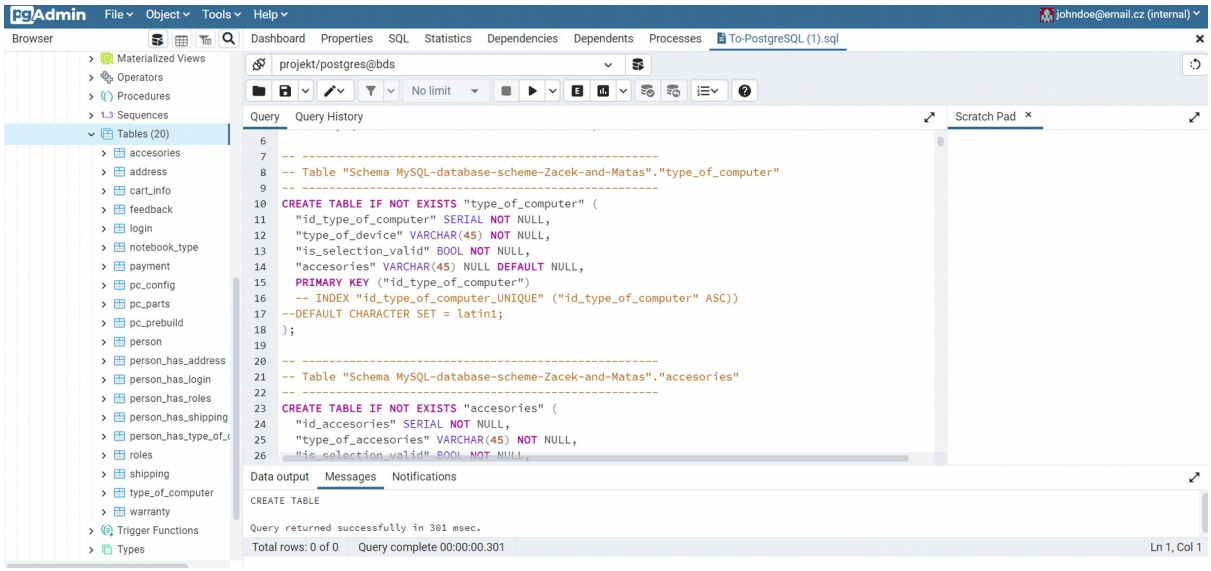
This is the connecting table between “person” and “login”:

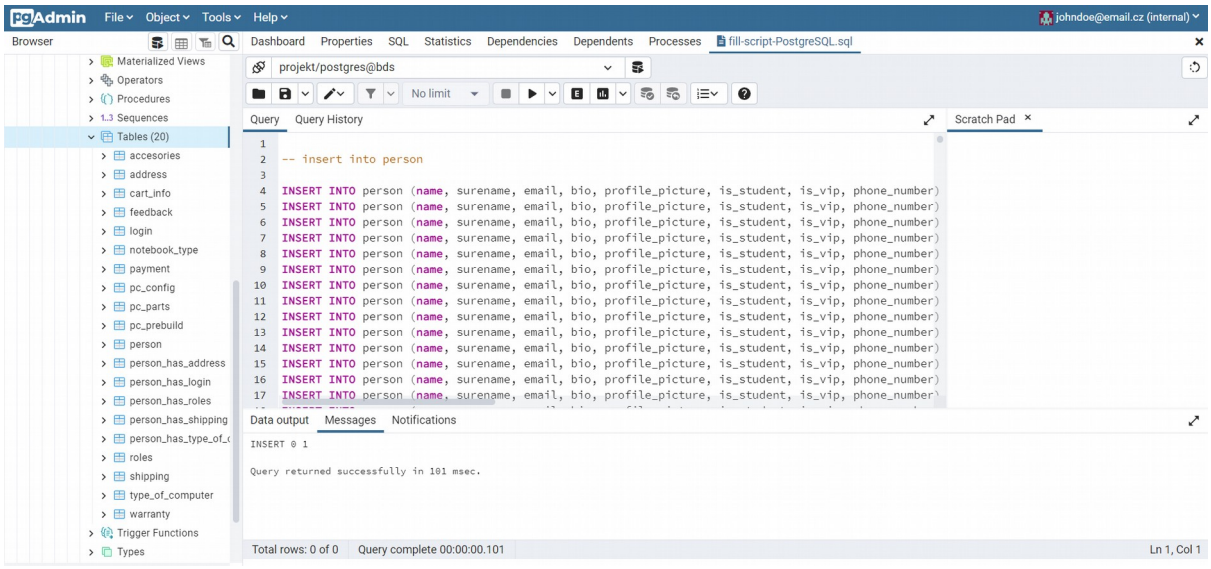
- *id_person,* foreign key, int because id_person is int
- *id_login,* foreign key, int because id_login is int
- *last_time_acount_loged_in,* saves the last time when user logged in, TIMESTAMP

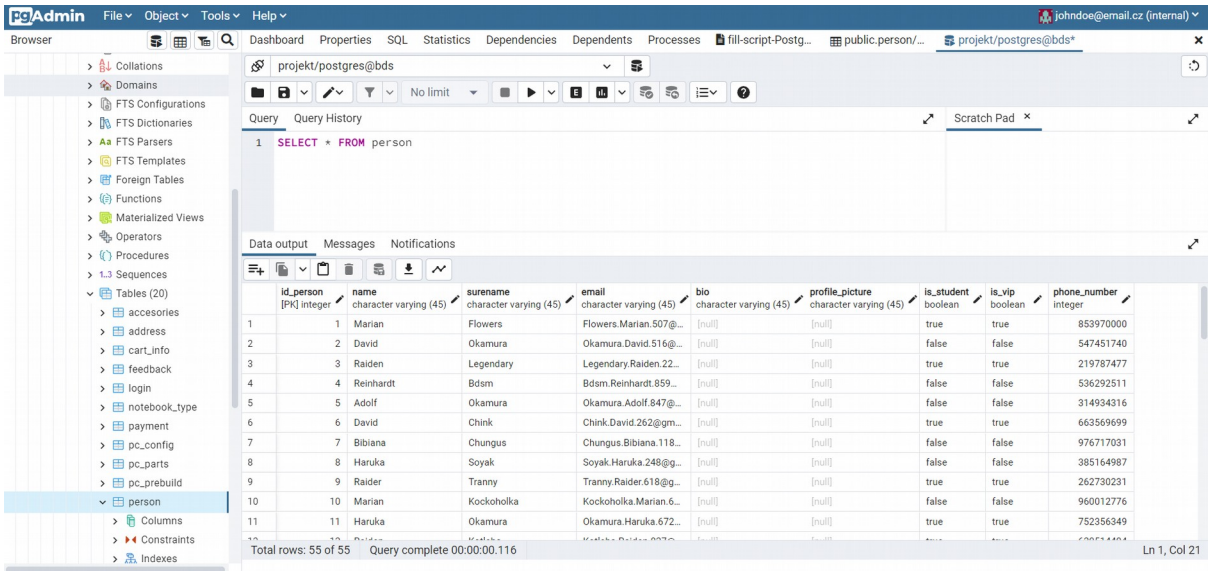
Proof of our working databases in PostgreSQL and MySQL

PostgreSQL

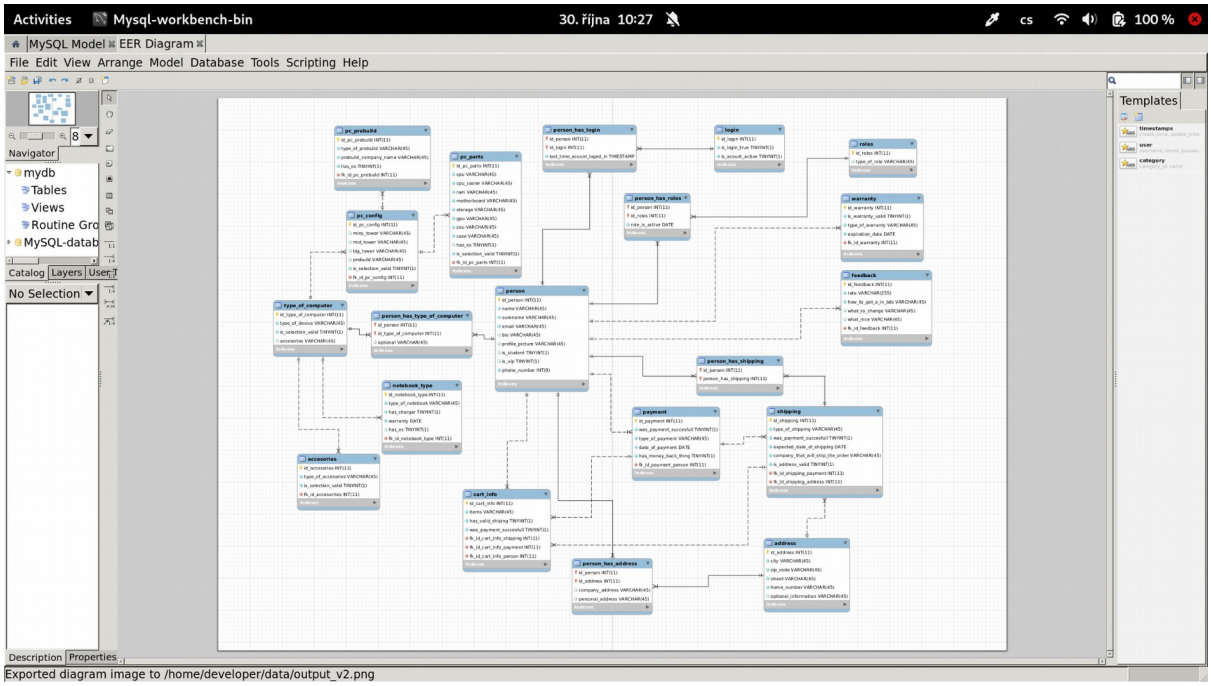


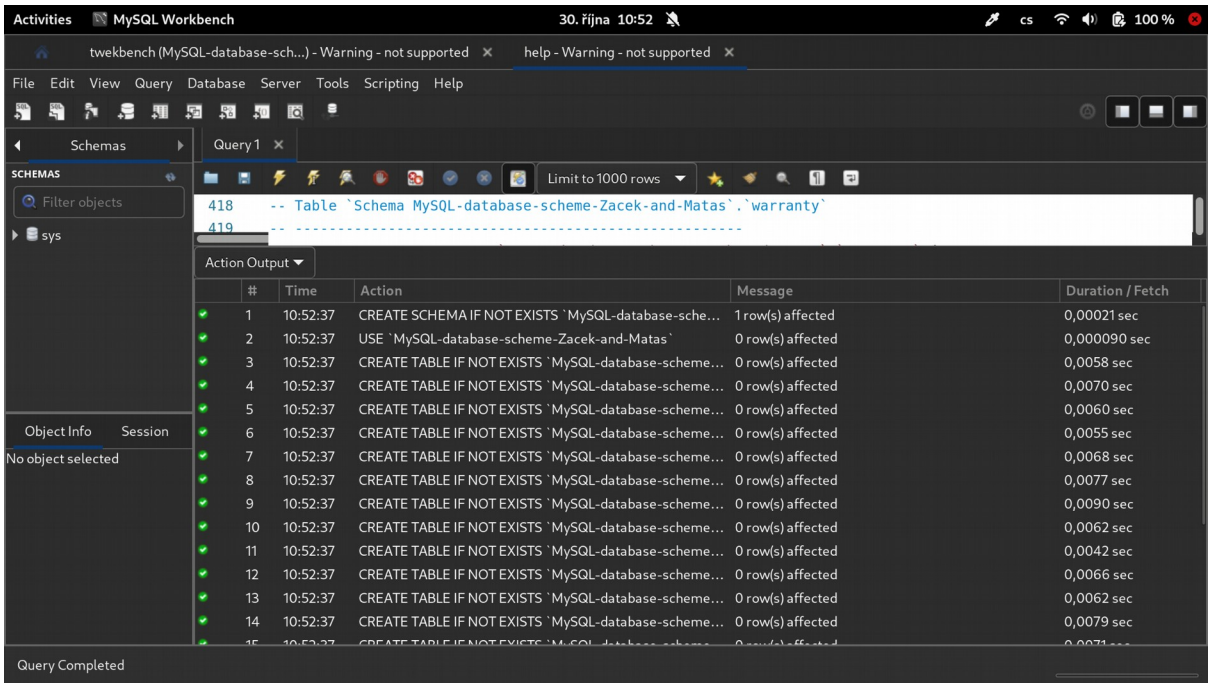

pgAdmin
File
Object
Tools
Help
johndoe@email.cz (internal)
Browser
Dashboard
Properties
SQL
Statistics
Dependencies
Dependents
Processes
To-PostgreSQL (1).sql
projekt/postgres@bds
No limit
Query
Query History
Scratch Pad
Materialized Views
Operators
Procedures
Sequences
Tables (20)
accesories
address
cart_info
feedback
login
notebook_type
payment
pc_config
pc_parts
pc_prebuild
person
person_has_address
person_has_login
person_has_roles
person_has_shipping
person_has_type_of_
roles
shipping
type_of_computer
warranty
Trigger Functions
Types
-- Table "Schema MySQL-database-scheme-Zacek-and-Matas"."type_of_computer"
CREATE TABLE IF NOT EXISTS "type_of_computer" (
"id_type_of_computer" SERIAL NOT NULL,
"type_of_device" VARCHAR(45) NOT NULL,
"is_selection_valid" BOOL NOT NULL,
"accesories" VARCHAR(45) NULL DEFAULT NULL,
PRIMARY KEY ("id_type_of_computer")
-- INDEX "id_type_of_computer_UNIQUE" ("id_type_of_computer" ASC))
--DEFAULT CHARACTER SET = latin1;
);
-- Table "Schema MySQL-database-scheme-Zacek-and-Matas"."accesories"
CREATE TABLE IF NOT EXISTS "accesories" (
"id_accesories" SERIAL NOT NULL,
"type_of_accesories" VARCHAR(45) NOT NULL,
Data output
Messages
Notifications
CREATE TABLE
Query returned successfully in 301 msec.
Total rows: 0 of 0
Query complete 00:00:00.301
Ln 1, Col 1

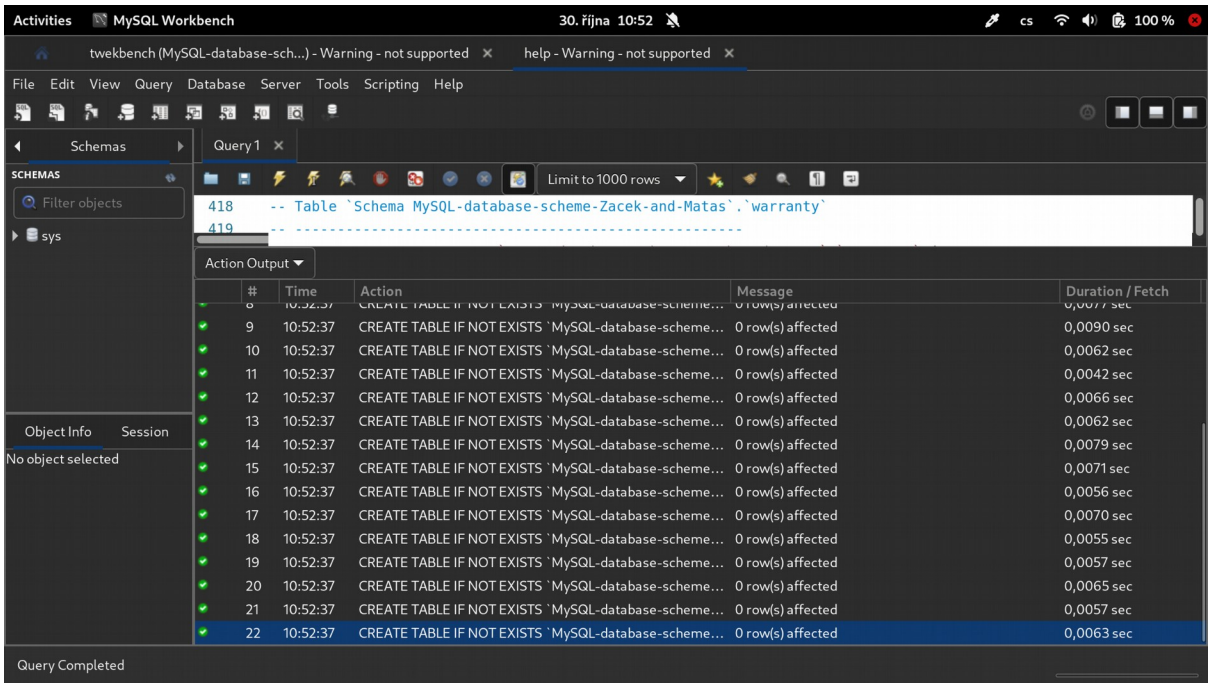

pgAdmin
File
Object
Tools
Help
johndoe@email.cz (internal)
Browser
Dashboard
Properties
SQL
Statistics
Dependencies
Dependents
Processes
fill-script-PostgreSQL.sql
projekt/postgres@bds
No limit
Query
Query History
Scratch Pad
Materialized Views
Operators
Procedures
Sequences
Tables (20)
accesories
address
cart_info
feedback
login
notebook_type
payment
pc_config
pc_parts
pc_prebuild
person
person_has_address
person_has_login
person_has_roles
person_has_shipping
person_has_type_of_
roles
shipping
type_of_computer
warranty
Trigger Functions
Types
-- insert into person
INSERT INTO person (name, surename, email, bio, profile_picture, is_student, is_vip, phone_number)
Data output
Messages
Notifications
INSERT 0 1
Query returned successfully in 101 msec.
Total rows: 0 of 0
Query complete 00:00:00.101
Ln 1, Col 1


PgAdmin
File
Object
Tools
Help
johndoe@email.cz (internal)
Browser
Dashboard
Properties
SQL
Statistics
Dependencies
Dependents
Processes
fill-script-Postg...
public.person/...
projekt/postgres@bds*
projekt/postgres@bds
No limit
Query
Query History
Scratch Pad
1 SELECT * FROM person
Data output
Messages
Notifications
Collations
Domains
FTS Configurations
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Operators
Procedures
Sequences
Tables (20)
accesories
address
cart_info
feedback
login
notebook_type
payment
pc_config
pc_parts
pc_prebuild
person
Columns
Constraints
Indexes
id_person [PK] integer
name character varying (45)
surename character varying (45)
email character varying (45)
bio character varying (45)
profile_picture character varying (45)
is_student boolean
is_vip boolean
phone_number integer
1 1 Marian Flowers Flowers.Marian.507@... [null] [null] true true 853970000
2 2 David Okamura Okamura.David.516@... [null] [null] false false 547451740
3 3 Raiden Legendary Legendary.Raiden.22... [null] [null] true true 219787477
4 4 Reinhardt Bdsm Bdsm.Reinhardt.859... [null] [null] false false 536292511
5 5 Adolf Okamura Okamura.Adolf.847@... [null] [null] false false 314934316
6 6 David Chink Chink.David.262@gm... [null] [null] true true 663569699
7 7 Bibiana Chungus Chungus.Bibiana.118... [null] [null] false false 976717031
8 8 Haruka Soyak Soyak.Haruka.248@g... [null] [null] false false 385164987
9 9 Raider Tranny Tranny.Raider.618@g... [null] [null] true true 262730231
10 10 Marian Kockoholka Kockoholka.Marian.6... [null] [null] false false 960012776
11 11 Haruka Okamura Okamura.Haruka.672... [null] [null] true true 752356349
Total rows: 55 of 55
Query complete 00:00:00.116
Ln 1, Col 21

MySQL

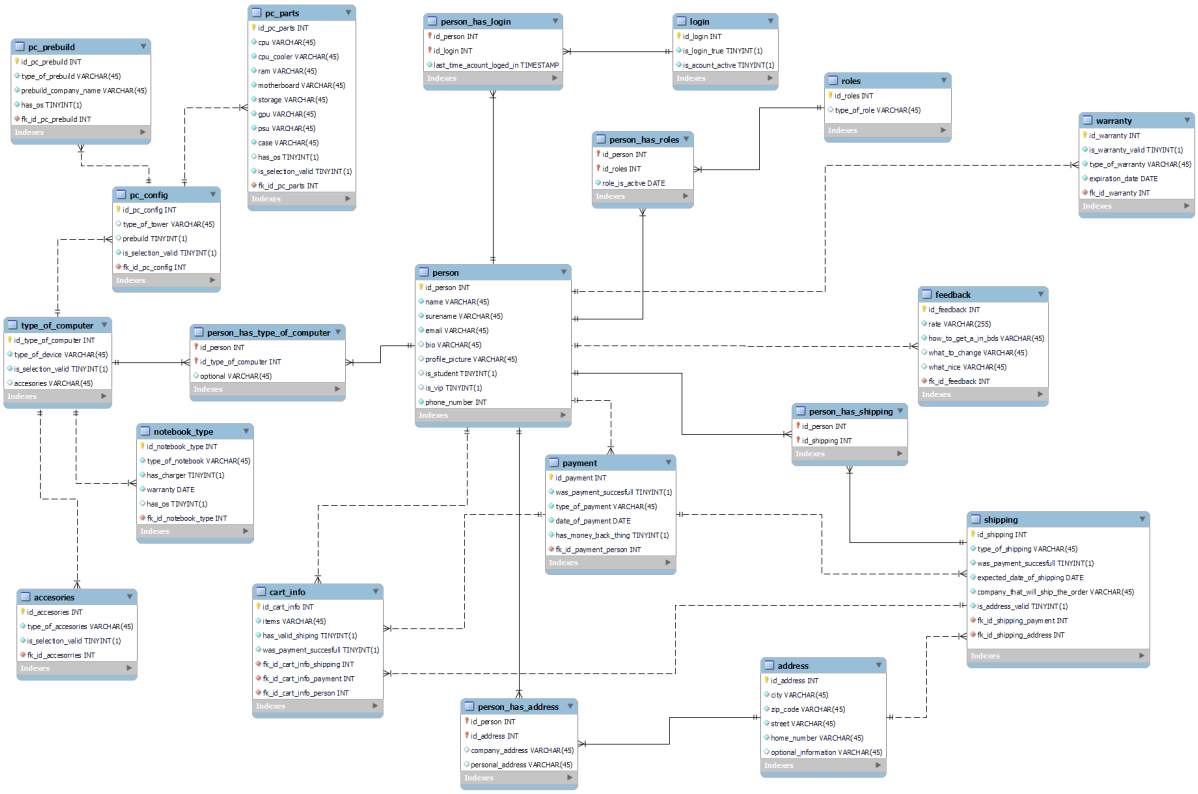

Activities
Mysql-workbench-bin
30. října 10:27
cs
100 %
MySQL Model
EER Diagram
File Edit View Arrange Model Database Tools Scripting Help
Navigator
mydb
Tables
Views
Routine Gro
MySQL-datab
Catalog
Layers
No Selection
Templates
Description
Properties
Exported diagram image to /home/developer/data/output_v2.png


Activities
MySQL Workbench
30. října 10:52
cs
100 %
twekbench (MySQL-database-sch...) - Warning - not supported
help - Warning - not supported
File Edit View Query Database Server Tools Scripting Help
Schemas
Query 1
SCHEMAS
Filter objects
sys
Limit to 1000 rows
418 -- Table `Schema MySQL-database-scheme-Zacek-and-Matas`.`warranty`
419 -- ---
Action Output
Time Action Message Duration / Fetch
1 10:52:37 CREATE SCHEMA IF NOT EXISTS `MySQL-database-sche... 1 row(s) affected 0,00021 sec
2 10:52:37 USE `MySQL-database-scheme-Zacek-and-Matas` 0 row(s) affected 0,000090 sec
3 10:52:37 CREATE TABLE IF NOT EXISTS `MySQL-database-scheme... 0 row(s) affected 0,0058 sec
4 10:52:37 CREATE TABLE IF NOT EXISTS `MySQL-database-scheme... 0 row(s) affected 0,0070 sec
5 10:52:37 CREATE TABLE IF NOT EXISTS `MySQL-database-scheme... 0 row(s) affected 0,0060 sec
6 10:52:37 CREATE TABLE IF NOT EXISTS `MySQL-database-scheme... 0 row(s) affected 0,0055 sec
7 10:52:37 CREATE TABLE IF NOT EXISTS `MySQL-database-scheme... 0 row(s) affected 0,0068 sec
8 10:52:37 CREATE TABLE IF NOT EXISTS `MySQL-database-scheme... 0 row(s) affected 0,0077 sec
9 10:52:37 CREATE TABLE IF NOT EXISTS `MySQL-database-scheme... 0 row(s) affected 0,0090 sec
10 10:52:37 CREATE TABLE IF NOT EXISTS `MySQL-database-scheme... 0 row(s) affected 0,0062 sec
11 10:52:37 CREATE TABLE IF NOT EXISTS `MySQL-database-scheme... 0 row(s) affected 0,0042 sec
12 10:52:37 CREATE TABLE IF NOT EXISTS `MySQL-database-scheme... 0 row(s) affected 0,0066 sec
13 10:52:37 CREATE TABLE IF NOT EXISTS `MySQL-database-scheme... 0 row(s) affected 0,0062 sec
14 10:52:37 CREATE TABLE IF NOT EXISTS `MySQL-database-scheme... 0 row(s) affected 0,0079 sec
Object Info Session
No object selected
Query Completed

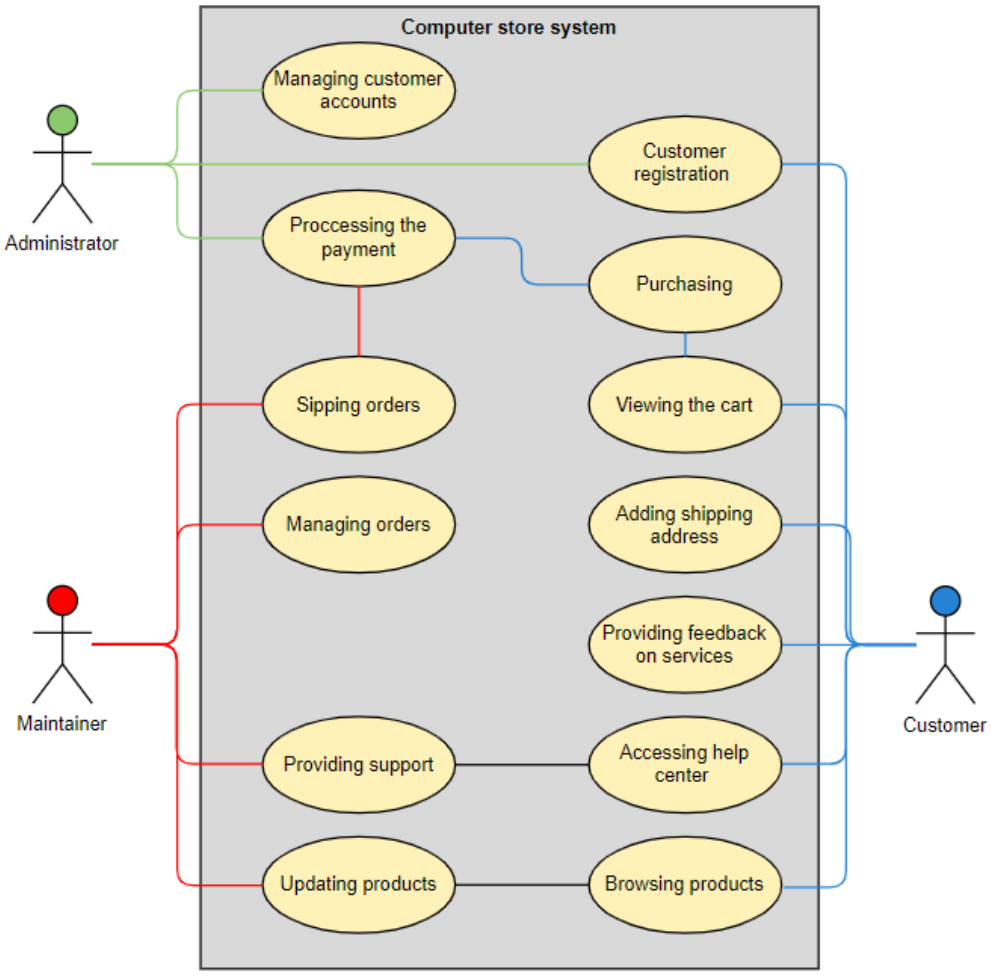

Activities
MySQL Workbench
30. října 10:52
cs
100 %
twekbench (MySQL-database-sch...) - Warning - not supported
help - Warning - not supported
File Edit View Query Database Server Tools Scripting Help
Schemas
Query 1
SCHEMAS
Filter objects
sys
Limit to 1000 rows
418 -- Table `Schema MySQL-database-scheme-Zacek-and-Matas`.`warranty`
419 -- ---
Action Output
Time Action Message Duration / Fetch
9 10:52:37 CREATE TABLE IF NOT EXISTS `MySQL-database-scheme... 0 row(s) affected 0,0090 sec
10 10:52:37 CREATE TABLE IF NOT EXISTS `MySQL-database-scheme... 0 row(s) affected 0,0062 sec
11 10:52:37 CREATE TABLE IF NOT EXISTS `MySQL-database-scheme... 0 row(s) affected 0,0042 sec
12 10:52:37 CREATE TABLE IF NOT EXISTS `MySQL-database-scheme... 0 row(s) affected 0,0066 sec
13 10:52:37 CREATE TABLE IF NOT EXISTS `MySQL-database-scheme... 0 row(s) affected 0,0062 sec
14 10:52:37 CREATE TABLE IF NOT EXISTS `MySQL-database-scheme... 0 row(s) affected 0,0079 sec
15 10:52:37 CREATE TABLE IF NOT EXISTS `MySQL-database-scheme... 0 row(s) affected 0,0071 sec
16 10:52:37 CREATE TABLE IF NOT EXISTS `MySQL-database-scheme... 0 row(s) affected 0,0056 sec
17 10:52:37 CREATE TABLE IF NOT EXISTS `MySQL-database-scheme... 0 row(s) affected 0,0070 sec
18 10:52:37 CREATE TABLE IF NOT EXISTS `MySQL-database-scheme... 0 row(s) affected 0,0055 sec
19 10:52:37 CREATE TABLE IF NOT EXISTS `MySQL-database-scheme... 0 row(s) affected 0,0057 sec
20 10:52:37 CREATE TABLE IF NOT EXISTS `MySQL-database-scheme... 0 row(s) affected 0,0065 sec
21 10:52:37 CREATE TABLE IF NOT EXISTS `MySQL-database-scheme... 0 row(s) affected 0,0057 sec
22 10:52:37 CREATE TABLE IF NOT EXISTS `MySQL-database-scheme... 0 row(s) affected 0,0063 sec
Object Info Session
No object selected
Query Completed

Diagrams

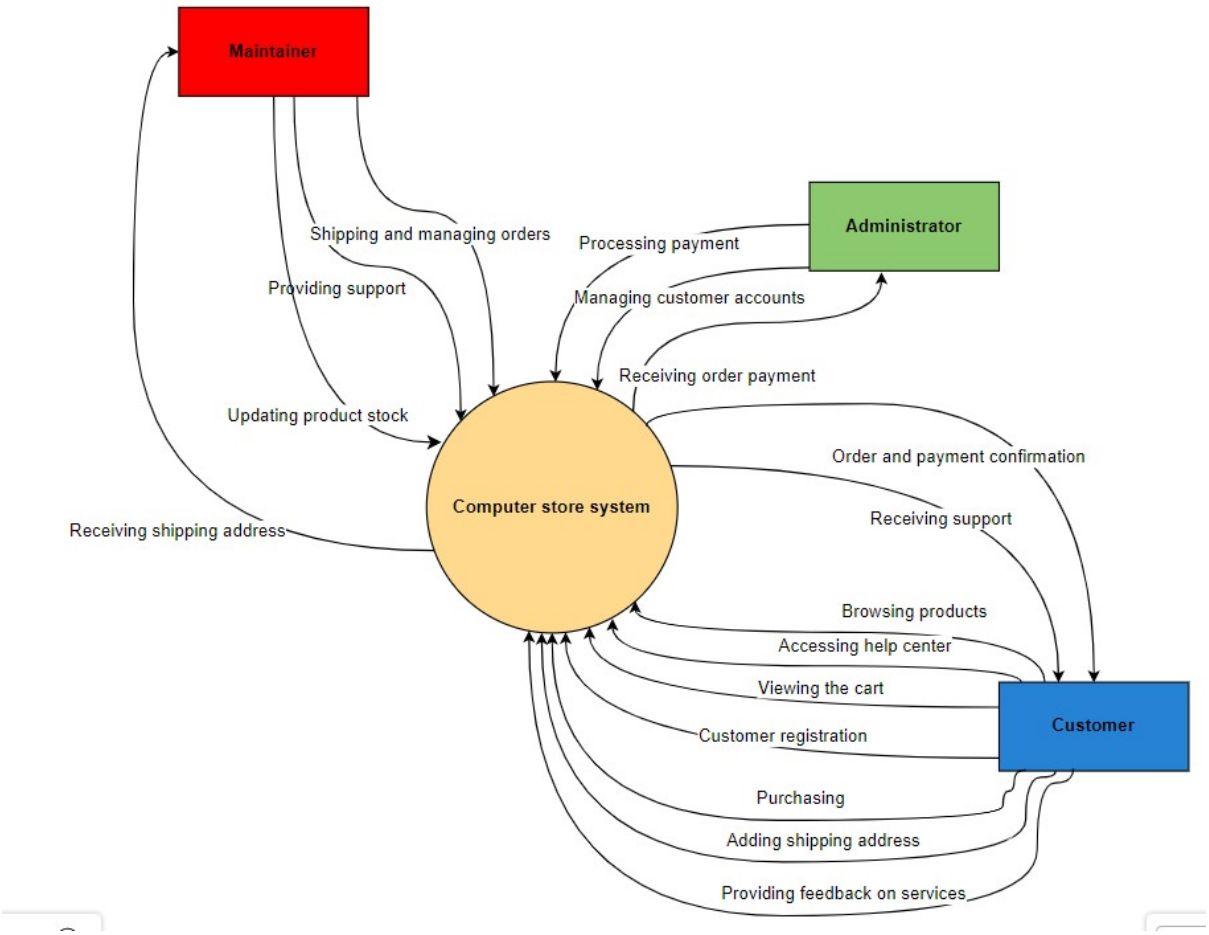
ERD



Use case diagram



System context diagram



List of non-functional and functional requirements

Functional requirements is what a system is **supposed to accomplish**, the main things that the user expects from the software. In our case it may be:

- creating new account (registration)
- updating account information such as shipping address or payment method
- browsing through products
- adding items to cart
- successfully placing orders

The non-functional requirement **elaborates a performance characteristic** of the system. Typically non-functional requirements fall into areas such as accessibility, fault tolerance, efficiency, reliability, security and so on. In our case it may be:

- performance of the system in peak hours
- time taken for the system to retrieve data from database
- security
- ability to handle large amount of users at the same time
- 24/7 availability with no downtime