

BRNO UNIVERSITY OF TECHNOLOGY

FACULTY OF ELECTRICAL ENGINEERING &
COMMUNICATION

Flight System Database

Authors

Petr BRÁBLÍK
Martin HANTÁK

October 31, 2022



Contents

1	Introduction	3
2	PostgreSQL	4
2.1	pgAdmin	4
3	MySQL	5
3.1	phpMyAdmin	5
4	Flight System Description	6
5	ERD Diagram	7
6	Use-case Diagram	8
7	System Context Diagram	9
8	List of functional and non-functional requirements	10
9	3rd normal form	10
10	Brief description of each table	11
10.1	Login	11
10.2	Passenger	12
10.3	Passport	12
10.4	Ticket	13
10.5	Baggage	13
10.6	Flight_places	14
10.7	Boarding_pass	14
10.8	Flight	15
10.9	Airline	15
10.10	Airplane	16
10.11	Flight_crew	16
10.12	Assignment_has_flight_crew	17
10.13	Assignment	17
11	Evidence of our database	18
11.1	PostgreSQL	18
11.2	MySQL	22
12	SQL scripts	24

1 Introduction

The goal of this project for the Security of database systems course is a basic introduction to the database design and the design in PostgreSQL and MySQL languages. When designing the tables, we used the MySQL Workbench program, which generated the bulk of the DDL code. Then we modified the DDL code and applied it to the pgAdmin 4 database system. We also remodeled this database into the phpMyAdmin system. Once the tables were created, we filled the tables with relevant data.

Our project is focused on creating a system that can be used as an airport database. Passenger will be able to register in our system, where he will be able to order a ticket. The tables store the relevant information about the person and the flight they will travel with. A total of 13 tables were created that use all database relationships such as 1:1, 1:N, M:N. The database is designed to meet the 3rd normal form.

This document contains a theoretical part in which we describe the individual SQL languages and platforms that we will use in this project. Furthermore, schemes and diagrams of our system, where you can see in detail the structure of individual tables and their connections. You can find the individual DDL and DML codes in the attachments.

2 PostgreSQL

PostgreSQL, often simply Postgres, is a free and open source object-relational database system (PostgreSQL (2022)).

It is 100% ACID compliant, fully supports foreign keys, JOIN operations, views, triggers and stored procedures. Contains most data types such as SERIAL, INTEGER, NUMERIC, BOOLEAN, CHAR, VARCHAR, DATE, INTERVAL, and TIMESTAMP. It is designed to handle a wide range of workloads, from single machines to data centers or web services with many concurrent users.

PostgreSQL is distributed under the BSD license, which is the most liberal of all open source licenses. This license allows unlimited free use, modification and distribution of PostgreSQL for both commercial and non-commercial use.

PostgreSQL is primarily developed for Linux, it is also available for macOS server, Microsoft Windows and is released under the MIT license.

This language is given the greatest emphasis in the project, so we had to study and apply it in detail. (figure 1: pgAdmin (2022))

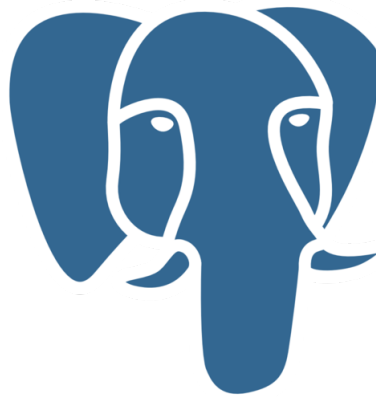


Figure 1: pgAdmin icon

2.1 pgAdmin

PgAdmin is a popular and feature-rich Open Source administration and development platform for PostgreSQL. PgAdmin can be used on Linux, Unix, macOS, and Windows to manage PostgreSQL and EDB Advanced Server version 10 and higher.

PgAdmin is used for monitoring and managing multiple PostgreSQL and EDB Advanced Server database servers, local and remote, through a single graphical interface.

We use the PgAdmin 4 platform in our project to create an aviation database. Working with this platform is very simple and clear, which is why we chose this platform.

3 MySQL

MySQL is a free and open source system for creating relational databases. The system is created by the Swedish company MySQL AB, now owned by Oracle Corporation. Its main authors were Michael "Monty" Widenius and David Axmark.

Due to its easy implementation (it can be installed on GNU/Linux, MS Windows, as well as other operating systems. MySQL was optimized primarily for speed, even at the cost of some simplifications: it only has simple backup methods; since 2005, it also supports so-called views, triggers, and stored procedures.

This language is used in the project to highlight the differences between the two languages. (figure 1: mySQL (2022))



Figure 2: MySQL icon

3.1 phpMyAdmin

The phpMyAdmin program system is a tool written in the PHP language that enables simple management of the content of the MySQL database

through a web interface (phpMyAdmin (2022)). It currently allows simple SQL statements to be executed. It is one of the most popular database management tools. It is available in 72 languages.

We used this platform to create an aviation database in MySQL.

4 Flight System Description

The main task of this project was to come up with a topic on which we demonstrate the design of tables and the subsequent application of SQL functions. We chose a flight system theme because we like to travel to foreign countries by the already mentioned plane. Note: Statistics show that travelling by airplane is one of the safest means of transportation in the world.

The user logs into our system using a login, where he enters his unique e-mail and a secure password. For easier logging in, the user can also enter a nickname. Once the user is logged in (registered), he fills in his first name, last name and relevant information about his passport. The passport is used for checking and better identification of the person before boarding the plane.

The main function of the database is to order tickets online. The user chooses where, where, when and at what price he will travel. When buying tickets, the user chooses the place where he will sit. As some passengers demand luxury, we offer individual classes such as standard, relax and business. In the plane, you can also find seats with more legroom, which are especially suitable for tall people. Of course, these seats have a higher price.

Especially for longer flights, there is the possibility to buy food on board. This service is becoming more and more popular mainly in business class, where nowadays the crew can also prepare multi-course meals. When ordering a ticket, we must not forget the passengers' luggage. In our system, the price depends on the weight of the luggage.

As soon as the passenger buys a ticket, a boarding pass is automatically generated for him, where he can find information such as the gate number, where he will board the plane. Check-in and boarded values are written as True and False in the database. It is checked if the passenger performs the given action.

In the last part of the database, we created tables where you can find what kind of plane you fly and what company owns the plane. We must also not forget the crew of the aircraft and their functions such as pilot or

flight attendant.

5 ERD Diagram

Entity-relationship diagram is used for abstract and conceptual representation of data. As you can already see in the image below, it is a massive amount of blocks connected by lines. Individual blocks represent attribute tables. Connecting lines represent relationships that occur between tables. In our system we use all relationships such as 1:1, 1:N, M:N. Note We use BOOLEAN instead of TINYINT as mentioned in the diagram because Workbench cannot use this data type.

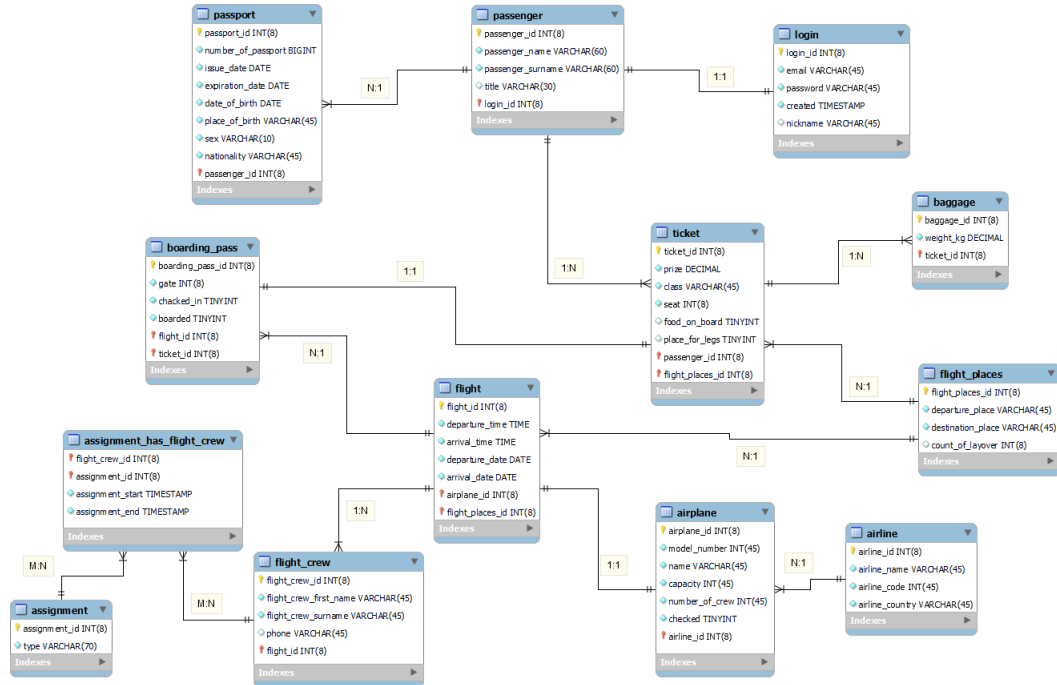


Figure 3: ERD diagram

6 Use-case Diagram

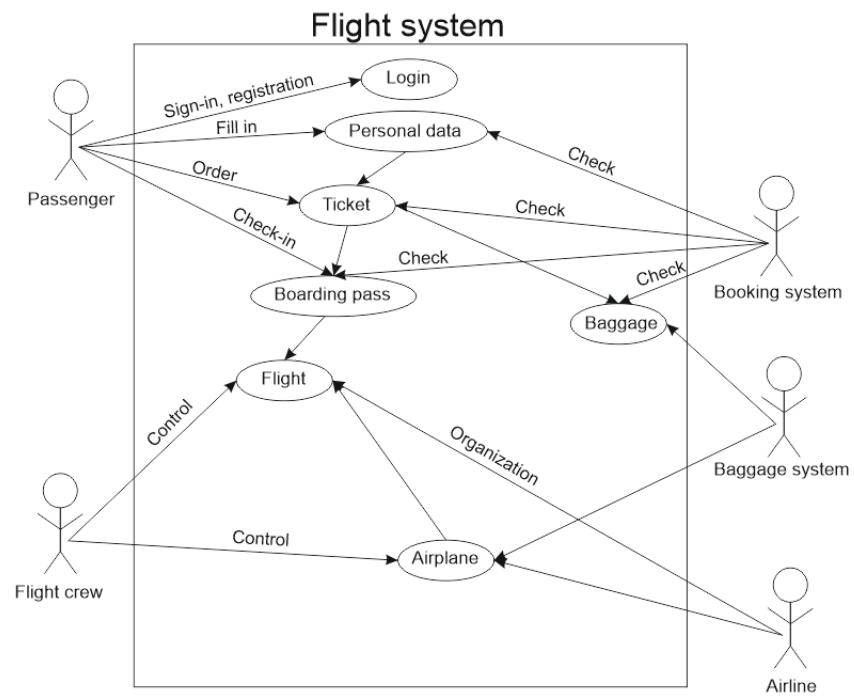


Figure 4: Use case diagram

7 System Context Diagram

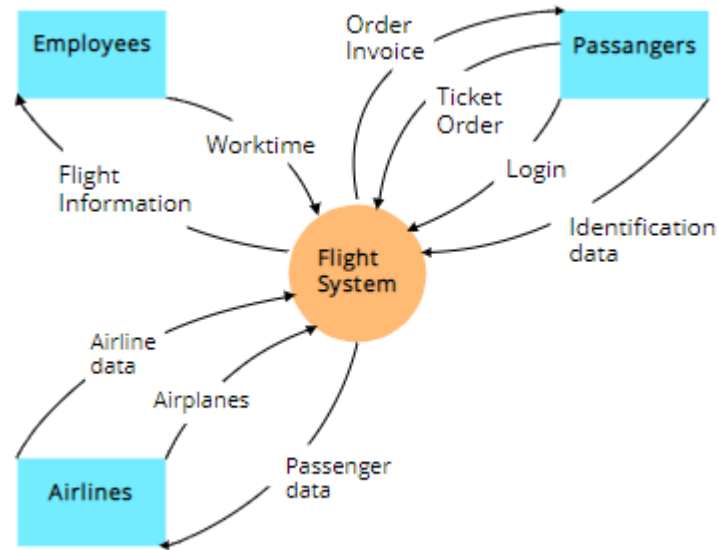


Figure 5: System context diagram

8 List of functional and non-functional requirements

- **F:** The system must enable the purchase and further administration of services entirely via the Internet.
N-F: The system can be registered in using an email and has to allow the setting of a nickname to simplify the login process.
- **F:** Login must be secure and has to allow at least one other form of security measure in addition to a password.
N-F: Login password must be at least 10 characters long and enable 2-factor authentication.
- **F:** When purchasing a product or service on the website, a confirmation has to be displayed to the account and sent to an email
N-F: Confirmation to the account must arrive within 10 minutes of performing the activity and confirmation to the email must arrive within 30 minutes of the purchase.
- **F:** The system must inform the customer about changes in departure times or other technical difficulties.
N-F: The system will always send an email to the customer in the event of technical problems within 10 minutes at the most from the detection of the defect or to the phone in the form of a SMS message, if the customer requests it in advance in the system, with the same time period.
- **F:** The system has to allow to change or cancel a ticket.
N-F: The customer can cancel or change the ticket up to 24 hours before departure. After this time, the customer must use the ticket or let it be forfeited.

9 3rd normal form

I'd start by describing if our database is in first normal form, since the other normal forms are derived from this (Simple Tutorials (2022)).

The first normal form says that each column should contain only atomic or indivisible data. We meet this form because most columns contain only one word or an indivisible unit (for example, a phone number). For example, we split the name column into first name and last name just to meet this standard. We satisfy the second normal form trivially because we have

a single column primary key in each table and do not use composite primary keys.

The third normal form has two conditions:

1. fulfill the second normal form
2. the table must not contain any transitive dependencies

We satisfy the second normal form, as I mentioned earlier. Next we look at all the dependencies. We do not need to solve the login, airline and assignment tables from the perspective of the third normal form, because they do not depend on anything. The remaining tables fulfill this form because we have attributes directly related to the given entity. For example, our database did not meet this form when we had the departure and destination place attributes in the flight and ticket entities, however, we fixed this error by adding another entity that contained only places - flight_places. I don't see any other errors of this type in our diagram, so I'd say it satisfies 3rd normal form.

10 Brief description of each table

10.1 Login

The login table is used for the user to log into the system.

- **login_id** – primary key of the table, type BIGSERIAL – 8 byte number, auto-incrementing integer
- **email** – the user enters his unique email, the main link to his account, type VARCHAR(45) – the length of email characters is limited to 45
- **password** – the user's security password, type VARCHAR(45) – the user has the option to create a password with a length of 45 characters
- **created** – recording the time of account creation, type TIMESTAMP – date and time of creation
- **nickname** – optional, used for easier login to the system, type VARCHAR(45) – length of nickname characters limited to 45

10.2 Passenger

The table serves to identify the passenger on board the aircraft.

- **passenger_id** - primary key of the table, type BIGSERIAL – 8 byte number, auto-incrementing integer
- **passenger_name** – user name, used to identify a person, type VARCHAR(60) – length of name characters limited to 60
- **passenger_surname** - user's surname, used to identify a person, type VARCHAR(60) - length of surname characters limited to 60
- **title** – optional, if the user has a title (such as Ing., PhDr., etc.), then write it here, type VARCHAR(30) – character length 30 in case the user has multiple titles
- **login_id** – foreign key referring to the login table, type BIGSERIAL – 8 byte number

10.3 Passport

Detailed identification of the passenger and can be also used to verify the person's identity. A person can have multiple passports.

- **passport_id** - primary key of the table, type BIGSERIAL – 8 byte number, auto-incrementing integer
- **number_of_passport** – unique passport number of each citizen of the state, type BIGINT – for storing a large number
- **issue_date** – passport issue date, type DATE – day, month, year
- **expiration_date** – passport expiration date, type DATE – day, month, year
- **date_of_birth** – date of birth of the user, type DATE – day, month, year
- **place_of_birth** – place where the user was born, type DATE – day, month, year
- **sex** – male/female, type VARCHAR(10) – 10 characters, because the user only enters either male/female

- **nationality** – country where the user comes from, type VARCHAR(45)
– country, max length 45 characters
- **passenger_id** - foreign key referring to the passenger table, type BIGSERIAL – 8 byte number

10.4 Ticket

The ticket serves as an order for flight services.

- **ticket_id** - primary key of the table, type BIGSERIAL – 8 byte number, auto-incrementing integer
- **prize** – the price for which the user buys the ticket, type DECIMAL – the price is in decimal numbers
- **class** - degrees of comfort on board the plane - standard, relax, business, type VARCHAR(45) – 45 characters to indicate the class
- **seat** – seat number, type VARCHAR(8) – integer indicating the seat
- **foof_on_board** – optional, the user specifies whether he wants to order food on board the plane, type BOOLEAN – true or false
- **place_for_legs** – optional, user chooses if he wants a special place with more space for legs, more expensive price, type BOOLEAN – true or false
- **passenger_id** - foreign key referring to the passenger table, type BIGSERIAL – 8 byte number
- **flight_places_id** - foreign key referring to the flight_places table, type BIGSERIAL – 8 byte number

10.5 Baggage

The baggage table is used for check-in before the flight.

- **baggage_id** - primary key of the table, type BIGSERIAL – 8 byte number, auto-incrementing integer
- **weight_kg** – baggage weight in kilograms, type DECIMAL – weight in kilograms, then the number is rounded. After rounding, the luggage is assigned the category to which it belongs.

- **ticket_id** - foreign key referring to the ticket table, type BIGSERIAL – 8 byte number

10.6 Flight_places

Places from and to which the passenger is flying.

- **flight_places_id** - primary key of the table, type BIGSERIAL – 8 byte number, auto-incrementing integer
- **departuere_place** – place of departure, type VARCHAR(45) – name of the airport of departure, length 45 characters
- **destination_place** - destination place of the flight, type VARCHAR(45) - the name of the destination airport with a length of 45 characters
- **count_of_layover** - number of layovers to reach the destination where the passenger wants to fly, type INT(8) - number of layovers

10.7 Boarding_pass

This pass is generated automatically after purchasing a ticket. It is mainly used for movement at the airport, so that the passenger gets on the plane through various controls.

- **boarding_pass_id** - primary key of the table, type BIGSERIAL – 8 byte number, auto-incrementing integer
- **gate** - the number from which the passenger departs is shown, type INT(8) - whole number
- **checked_in** – information on whether the passenger passed the check. It is checked whether he is carrying sharp objects or any weapons. Type BOOLEAN - true or false
- **boarded** – information if the passenger is present on board, type BOOLEAN – true or false
- **flight_id** - foreign key referring to the flight table, type BIGSERIAL – 8 byte number
- **ticket_id** - foreign key referring to the ticket table, type BIGSERIAL – 8 byte number

10.8 Flight

Information on what time and date the flights will take place.

- **flight_id** - primary key of the table, type BIGSERIAL – 8 byte number, auto-incrementing integer
- **departure_time** - information about the departure of the aircraft, at what time, type TIME - hours, minutes, seconds, Note. Time and date are split into multiple columns for better data handling. E.g. delay of departure time
- **arrival_time** - information about the arrival of the aircraft, at what time, type TIME - hours, minutes, seconds
- **departure_date** - information about the departure of the aircraft, which day, DATE type - year, month, day
- **arrival_date** - information about the arrival of the aircraft, which day, DATE type - year, month, day
- **airplane_id** - foreign key referring to the airplane table, type BIGSERIAL – 8 byte number
- **flight_places_id** - foreign key referring to the flight_places table, type BIGSERIAL – 8 byte number

10.9 Airline

Information about the company that owns the individual aircraft.

- **airline_id** - primary key of the table, type BIGSERIAL – 8 byte number, auto-incrementing integer
- **airline_name** - name of the company that owns the individual aircraft, type VARCHAR(45) - 45 characters
- **airline_code** – identification of the company by number, type INT(45) – integers of size 45
- **airline_country** – the country the company is from, type VARCHAR(45) – 45 characters

10.10 Airplane

Information about a specific aircraft before departure.

- **airplane_id** - primary key of the table, type BIGSERIAL – 8 byte number, auto-incrementing integer
- **model_number** – unique identification of the aircraft, type INT(45) - an integer of size 45
- **name** - name of the aircraft, type VARCHAR(45) – characters of size 45
- **capacity** – indicates how many people the plane can accommodate, type INT(45) – an integer of size 45
- **number_of_crew** - the necessary amount of crew required to operate the aircraft, type INT(45) - an integer of size 45
- **checked** – confirmation if the plane was checked before the flight, type BOOLEAN – value true or false
- **airline_id** - foreign key referring to the airline table, type BIGSERIAL – 8 byte number

10.11 Flight_crew

Information about the crew that operates a particular aircraft.

- **flight_crew_id** - primary key of the table, type BIGSERIAL – 8 byte number, auto-incrementing integer
- **flight_crew_first_name** – name of the crew member, type VARCHAR(45) – 45 characters
- **flight_crew_surname** – crew member's surname, type VARCHAR(45) – 45 characters
- **phone** - optional, crew member contact, phone number, type VARCHAR(45) - 45 characters, due to e.g. + before the phone number
- **flight_id** - foreign key referring to the flight table, type BIGSERIAL – 8 byte number

10.12 Assignment_has_flight_crew

Intermediate table connecting flight_crew and assignment. It is created for the M:N session.

- **flight_crew_id** - foreign key referring to the flight_crew table, type BIGSERIAL – 8 byte number
- **assignment_id** - foreign key referring to the assignment table, type BIGSERIAL – 8 byte number
- **assignment_start** - start time when the given task occurs for a crew member for a certain flight, basically it is a check if the crew is at work, type TIMESTAMP - date and time
- **assignment_end** - end time, control of departure from work, type TIMESTAMP - date and time

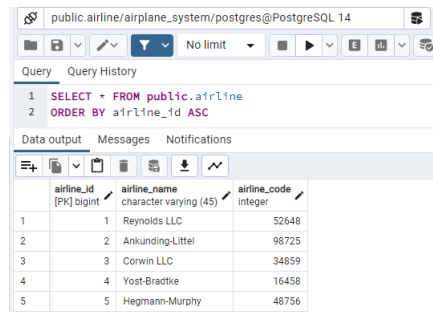
10.13 Assignment

A description of what role the crew member is waiting for. E.g. pilot, flight attendant, aircraft condition controller, etc.

- **assignment_id** - primary key of the table, type BIGSERIAL – 8 bytes number, autoincrementing
- **type** – description of the role of the crew member, as I already mentioned pilot, flight attendant, controller, person who loads luggage onto the plane, etc., type VARCHAR(70) – characters of size 70

11 Evidence of our database

11.1 PostgreSQL



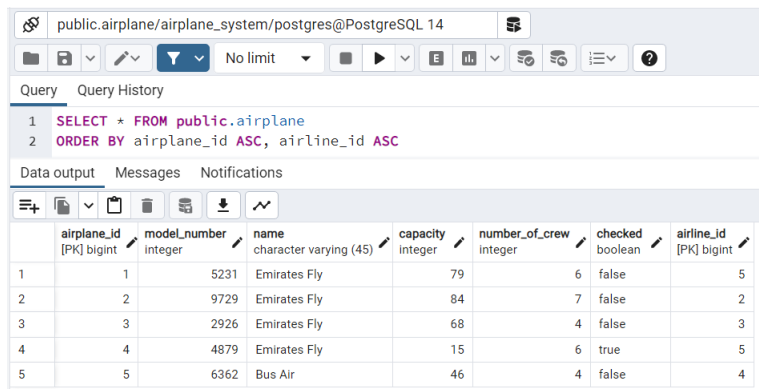
The screenshot shows a PostgreSQL query window with the following SQL query:

```
1 SELECT * FROM public.airline
2 ORDER BY airline_id ASC
```

The query results are displayed in a table with the following columns: `airline_id` (PK), `airline_name`, and `airline_code`. The data is as follows:

airline_id	airline_name	airline_code
1	Reynolds LLC	52648
2	Ankunding-Littel	98725
3	Corwin LLC	34859
4	Yost-Bradtko	16458
5	Hegmann-Murphy	48756

Figure 6: Table airline



The screenshot shows a PostgreSQL query window with the following SQL query:

```
1 SELECT * FROM public.airplane
2 ORDER BY airplane_id ASC, airline_id ASC
```

The query results are displayed in a table with the following columns: `airplane_id` (PK), `model_number`, `name`, `capacity`, `number_of_crew`, `checked`, and `airline_id` (FK). The data is as follows:

airplane_id	model_number	name	capacity	number_of_crew	checked	airline_id
1	5231	Emirates Fly	79	6	false	5
2	9729	Emirates Fly	84	7	false	2
3	2926	Emirates Fly	68	4	false	3
4	4879	Emirates Fly	15	6	true	5
5	6362	Bus Air	46	4	false	4

Figure 7: Table airplane

public.assignment/airplane_system/postgres@P

Query Query History

```
1 SELECT * FROM public.assignment
2 Loading... gnment_id ASC
```

Data output Messages Notifications

	assignment_id [PK] bigint	type character varying (70)
1	1	Pilot
2	2	Engineer
3	3	Flight attendant
4	4	Engineer assistant
5	5	Co-pilot

Figure 8: Table assignment

Query Query History

```
1 SELECT * FROM public.assignment_has_flight_crew
2 Loading... nt_crew_id ASC, assignment_id ASC
```

Data output Messages Notifications

	flight_crew_id [PK] bigint	assignment_id [PK] bigint	assignment_start timestamp without time zone	assignment_end timestamp without time zone
1	1	5	2022-10-24 00:00:00	2022-10-24 12:40:00
2	2	4	2022-10-24 00:00:00	2022-10-24 11:56:48
3	3	1	2022-10-24 00:00:00	2022-10-24 08:32:00
4	4	2	2022-10-24 00:00:00	2022-10-24 19:51:26
5	5	4	2022-10-24 00:00:00	2022-10-24 04:44:17

Figure 9: Table assignment_has_crew

Query Query History

```
1 SELECT * FROM public.baggage
2 Loading... age_id ASC, ticket_id ASC
```

Data output Messages Notifications

	baggage_id [PK] bigint	weight_kg numeric	ticket_id [PK] bigint
1	1	7.61	22
2	2	17.43	21
3	3	3.16	39
4	4	11.28	5
5	5	14.3	13

Figure 10: Table baggage

Query Query History

```
1 SELECT * FROM public.boarding_pass
2 Loading... boarding_pass_id ASC, flight_id ASC, ticket_id ASC
```

Data output Messages Notifications

	boarding_pass_id [PK] bigint	gate integer	checked_in boolean	boarded boolean	flight_id [PK] bigint	ticket_id [PK] bigint
1	1	7	false	true	8	38
2	2	3	false	true	7	21
3	3	1	true	true	28	16
4	4	8	true	true	4	9
5	5	5	false	false	19	45

Figure 11: Table boarding_pass

Query Query History

```
1 SELECT * FROM public.flight_crew
2 Loading... flight_crew_id ASC, flight_id ASC
```

Data output Messages Notifications

	flight_crew_id [PK] bigint	flight_crew_first_name character varying (45)	flight_crew_surname character varying (45)	phone character varying (45)	flight_id [PK] bigint
1	1	Ursola	Louthe	561400514	22
2	2	Leif	Ream	894490081	8
3	3	Welsh	Master	483376387	21
4	4	Kain	Ladell	104967790	3
5	5	Fairlie	Baddoe	991381841	2

Figure 12: Table flight_crew

```
1 SELECT * FROM public.flight_places
2 Loading... flight_places_id ASC
```

Data output Messages Notifications

	flight_places_id [PK] bigint	departure_place character varying (45)	destination_place character varying (45)	count_of_layover integer
1	1	Brno	Paris	1
2	2	Krakow	London	[null]
3	3	Brno	Amsterdam	2
4	4	Prague	New York	[null]
5	5	Bratislava	Moscow	1

Figure 13: Table flight_places

Query Query History

```
1 SELECT * FROM public.flight
2 Loading... nt_id ASC, airplane_id ASC, flight_places_id ASC
```

Data output Messages Notifications

	flight_id [PK] bigint	departure_time time without time zone	arrival_time time without time zone	departure_date date	arrival_date date	airplane_id [PK] bigint	flight_places_id [PK] bigint
1	1	00:20:33	14:39:32	2022-11-24	2022-11-24	2	4
2	2	00:53:28	17:29:30	2022-11-24	2022-11-24	3	5
3	3	00:18:32	17:56:49	2022-11-24	2022-11-24	3	1
4	4	00:33:32	20:38:29	2022-11-24	2022-11-24	1	1
5	5	00:41:20	20:36:43	2022-11-24	2022-11-24	3	5
6	6	00:27:40	18:47:35	2022-11-24	2022-11-24	3	1
7	7	00:36:57	22:12:00	2022-11-24	2022-11-24	1	4
8	8	00:41:24	13:31:59	2022-11-24	2022-11-24	3	5
9	9	00:10:38	13:55:10	2022-11-24	2022-11-24	5	2
10	10	00:41:28	16:50:50	2022-11-24	2022-11-24	2	5
11	11	00:46:23	18:27:25	2022-11-24	2022-11-24	1	5
12	12	00:46:29	19:08:27	2022-11-24	2022-11-24	5	1
13	13	00:28:52	18:25:26	2022-11-24	2022-11-24	1	4
14	14	00:36:46	23:27:25	2022-11-24	2022-11-24	4	4
15	15	00:30:23	18:15:17	2022-11-24	2022-11-24	4	2
16	16	00:52:50	13:35:01	2022-11-24	2022-11-24	1	4
17	17	00:11:10	20:12:32	2022-11-24	2022-11-24	4	2
18	18	00:26:18	19:22:30	2022-11-24	2022-11-24	4	2

Successfully run. Total query run...

Figure 14: Table flight

```
1 SELECT * FROM public.login
2 Loading... n_id ASC
```

Data output Messages Notifications

	login_id [PK] bigint	email character varying (45)	password character varying (45)	created timestamp without time zone	nickname character varying (45)
1	1	vsprade0@ask.com	lUcwFetV	2016-02-09 13:06:19	crawlingson0
2	2	vroyle1@abc.net.au	WrdkIO	2017-06-02 23:54:32	jbrawson1
3	3	ndionsett12@google.cn	iy3YTEuGW	2011-01-10 06:07:28	lmorrice2
4	4	scocksedge3@cisco...	vUthhgAXU	2021-12-19 11:59:35	jwynes3
5	5	noxbury4@springer.c...	CMtMXIsuY	2020-02-11 20:45:24	rwixon4
6	6	naveries5@tinypic.com	3NzAGP	2021-07-24 02:45:56	mfullman5
7	7	gosmund6@ehow.com	VKqcxQe2MSyE	2015-12-12 11:18:38	jodowgaine6
8	8	lgransden7@senate.g...	JIUG34	2021-10-24 09:56:33	klaville7
9	9	pesland8@economist...	DvgfgV89biaB	2011-01-17 10:51:35	cstillmann8
10	10	gdomsalla9@google...	pISieEm	2019-05-10 20:38:46	lbigland9
11	11	kbarga@usda.gov	TlGapoqlQ	2016-12-22 01:48:00	bblytha
12	12	lbardwellb@de.vu	tduPKfn5rl	2015-01-03 17:49:36	hchastagnierb
13	13	pmacphersonc@pagl...	4XGJsd6	2021-04-24 17:12:35	ltrathanc
14	14	sdugmored@ustream...	AQ1WZUVKznF3	2013-04-27 08:26:32	pupchurchd
15	15	ishilladaye@artisteer...	l30nhyPdbN	2018-01-15 22:45:46	jjunkine
16	16	cperacof@dell.com	nMLTeXNoC	2022-03-26 20:38:20	jcridlonef
17	17	trutedgeg@arstechni...	GCbLhJ	2017-09-19 17:49:03	ndelaneyg
18	18	mmckyrrellyh@npr.org	VSFISzx48n	2011-03-21 08:38:36	

Successfully run. Total query run...

Figure 15: Table login

11.2 MySQL

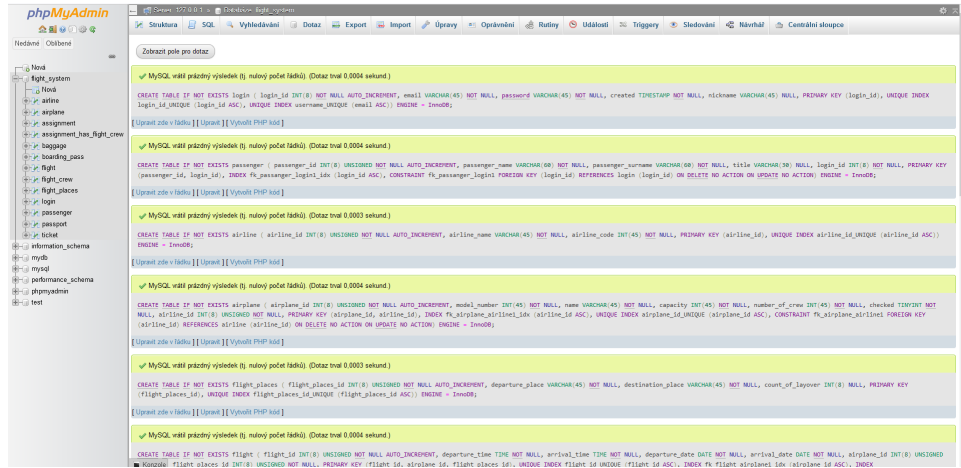


Figure 16: Table inserts into our database in phpMyAdmin

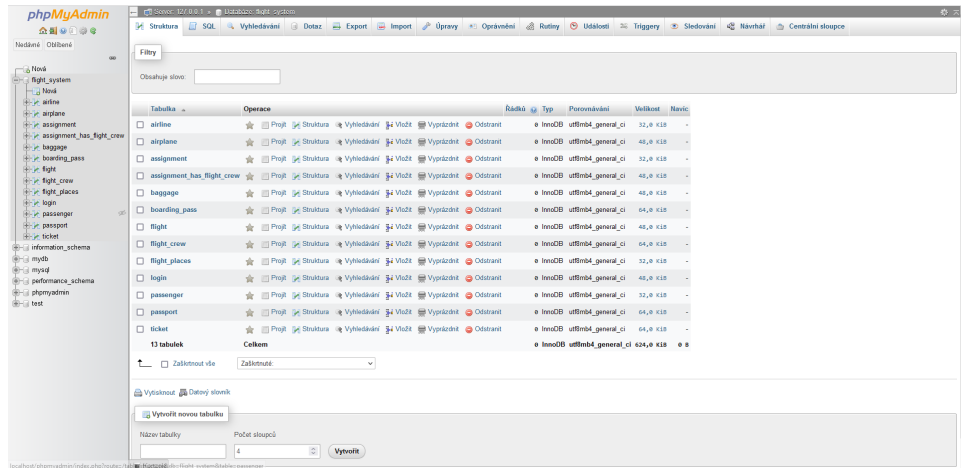


Figure 17: Overview of the tables in phpMyAdmin

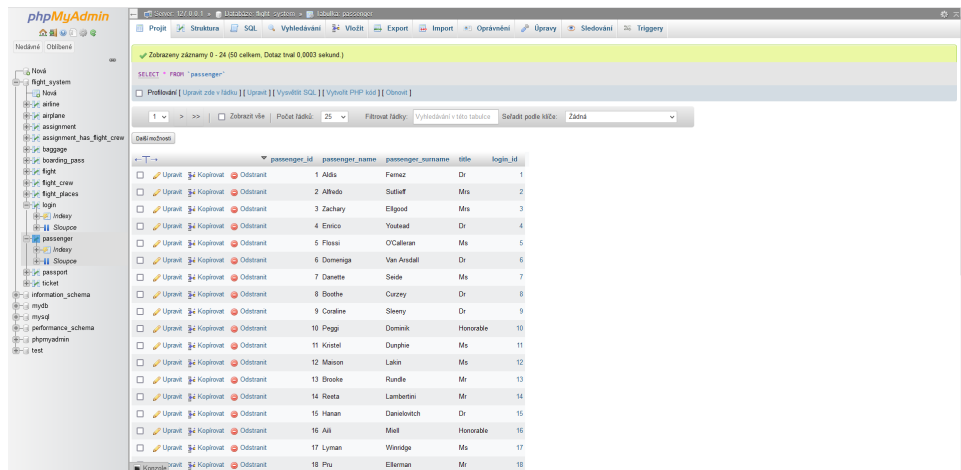


Figure 18: Row insert into our database in phpMyAdmin

12 SQL scripts

- DDL script for PostgreSQL – ./ddl_postgreSQL.sql
- DDL script for MySQL – ./ddl_mySQL.sql
- DML script for PostgreSQL – ./dml_postgreSQL.sql
- DML script for MySQL – ./dml_mySQL.sql

13 Conclusion

In the first part of the project, we got acquainted with the Workbench program, where we gradually created individual tables and their attributes. This part was the most difficult for us, because our system was constantly being optimized, so creating the design of the database took the most time. As soon as our design was completed, we generated the DDL code, which we then modified into PostgreSQL and MySQL.

After the DDL was completed, the codes were applied to the pgAdmin 4 and phpMyAdmin programs. We then filled these databases with data and generated DML codes. We weren't so keen on the last part of the project, which was the documentation.

We rate this project very positively and think we have gained experience in database and SQL programming.

References

- mySQL (2022). mySQL: Mysql is a free and open source system for creating relational databases. Picture taken on October 30, 2022. <https://cs.wikipedia.org/wiki/MySQL>.
- pgAdmin (2022). pgAdmin: pgadmin is a web-based database manager. Picture taken on October 30, 2022. <https://www.pgadmin.org/>.
- phpMyAdmin (2022). phpMyAdmin: Mysql is a free and open source system for creating relational databases. Read on October 30, 2022. <https://cs.wikipedia.org/wiki/PhpMyAdmin>.
- PostgreSQL (2022). PostgreSQL: Postgresql is a free and open source object-relational database system. Read on October 30, 2022. <https://postgres.cz/wiki/PostgreSQL>.
- Simple Tutorials (2022). Simple Tutorial: Simple tutorials is a web page mainly for education. Read on October 30, 2022. <https://simplesqltutorials.com/third-normal-form/>.