Project Report

CAMPUS ADMINISTRATION

Contents

[1. Project description 1](#_Toc468213230)

[1.1. Short overview 1](#_Toc468213231)

[1.2. List of features 1](#_Toc468213232)

[2. Use case 3](#_Toc468213233)

[3. Overview of technologies used 5](#_Toc468213234)

[3.1. PostgeSQL 5](#_Toc468213235)

[3.2. JavaFX 5](#_Toc468213236)

[3.3. JDBC 5](#_Toc468213237)

[4. ER-diagram 6](#_Toc468213238)

[5. Database design schema 7](#_Toc468213239)

[6. Description of normalization 10](#_Toc468213240)

[7. List of queries 10](#_Toc468213241)

[8. Database Creation queries 21](#_Toc468213242)

# Project description

## Short overview

The students' dormitory is the main place of university students' daily life, so the students' dormitory management is an important part of university management. With the increasing number of students living in dormitories, the necessity of an application for managing all tasks related with the application, allocation and monitoring of the students’ housing facilities becomes clear. Therefore, our team decided to develop Campus Administration System with the aim to cater to the needs of the dormitories and the residence hall managers in terms of easier data input and processing, as well as printing output and manipulation of data. Specifically, the system aims at helping the dormitory administrator managing the information during daily work, since he/she is in charge of all kinds of things in the dormitory. So, by using the system it would make the dormitory administrator work easier, more efficient, and make fewer mistakes. Furthermore, Campus Administration System maintains data of campus personnel including security guard, cleaning service representatives and others.

## List of features

Below key features of the Campus Administration System are presented:

* **Record and management of the students’ and employees’ information**

The students’ information includes student’s name, id number, gender, date of birth and scholarship. As for employees, they also have name, id number, gender, date of birth, salary and position. These data can be modified, updated and deleted.

* **Personnel attendance control**

This function gives an opportunity to manage what time the employee checked in and out. Based on the attendance statistics the salary for personnel will be calculated.

* **Keep visitor records**

Anyone who does not live in the dormitory is regarded as a guest and the system records the visit. Personal information including name, date of birth and gender are necessary. The visiting time, reception person are required. When the visitor leaves, the departure time is recorded.

* **Students’ and employees’ documents maintenance**

Such important documents as passport, academic contract, medical insurance will be stored in the system database.

* **Payment control**

The following feature will help to control payments made by students and guest for room renting.

* **Apartments’ occupation control**

With the help of implemented queries and functions it will be possible to search rooms which have not been occupied yet. Moreover, the system provide information about rooms by gender compatibility.

* **Access control**

According to the role in the university campus, people have access to rooms within the campus territory. For instance, students can not enter to the staff room. There are entry checking machines which will check an entry permission of the person.

# Use case

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Use Case name | Actor | Precondition | Postcondition | Event flow |
| 1 | Add student | Administrator | Student should provide information to the admin | Student is added | Admin inserts required information about the student into the system |
| 2 | Delete student | Administrator | Student should leave the campus | Student is deleted | Admin deletes student’s information from the system |
| 3 | Add guest | Administrator | Guest should provide information to the admin | Guest is added | Admin inserts required information about the guest into the system |
| 4 | Mark guest as left | Administrator | Guest should leave the campus | Guest is marked as left | Admin deletes guest’s information from the system |
| 5 | Add employee | Administrator | Employee should provide information to the admin | Employee is added | Admin inserts required information about the employee into the system |
| 6 | Delete employee | Administrator | Employee should leave his job | Employee id deleted | Admin deletes employee’s information from the system |
| 7 | Search for available room | Administrator | The list of all existing rooms should be available | Available rooms are displayed |  |
| 8 | Search occupied rooms | Administrator | The list of all existing rooms should be available | Occupied rooms are displayed |  |
| 9 | View person’s balance | Administrator | The list of all persons’ balance should be available | Person’s balance is displayed | Admin |
| 10 | View debtors | Administrator | The list of transactions should be available | Debtors are displayed |  |
| 11 | View people inside the campus | Administrator |  | People inside the campus is displayed |  |
| 12 | Add document | Administrator | Person should provide the document to the administrator | Document is added | Admin scans the document and upload into the system |

# Overview of technologies used

## PostgeSQL



As a database server, we used PostgeSQL, since it stores data securely, and to allows for retrieval at the request of other software applications. Moreover, PostgreSQL manages concurrency through a system known as multiversion concurrency control (MVCC), which gives each transaction a "snapshot" of the database, allowing changes to be made without being visible to other transactions until the changes are committed. This largely eliminates the need for read locks, and ensures the database maintains the ACID (atomicity, consistency, isolation, durability) principles in an efficient manner.

## JavaFX

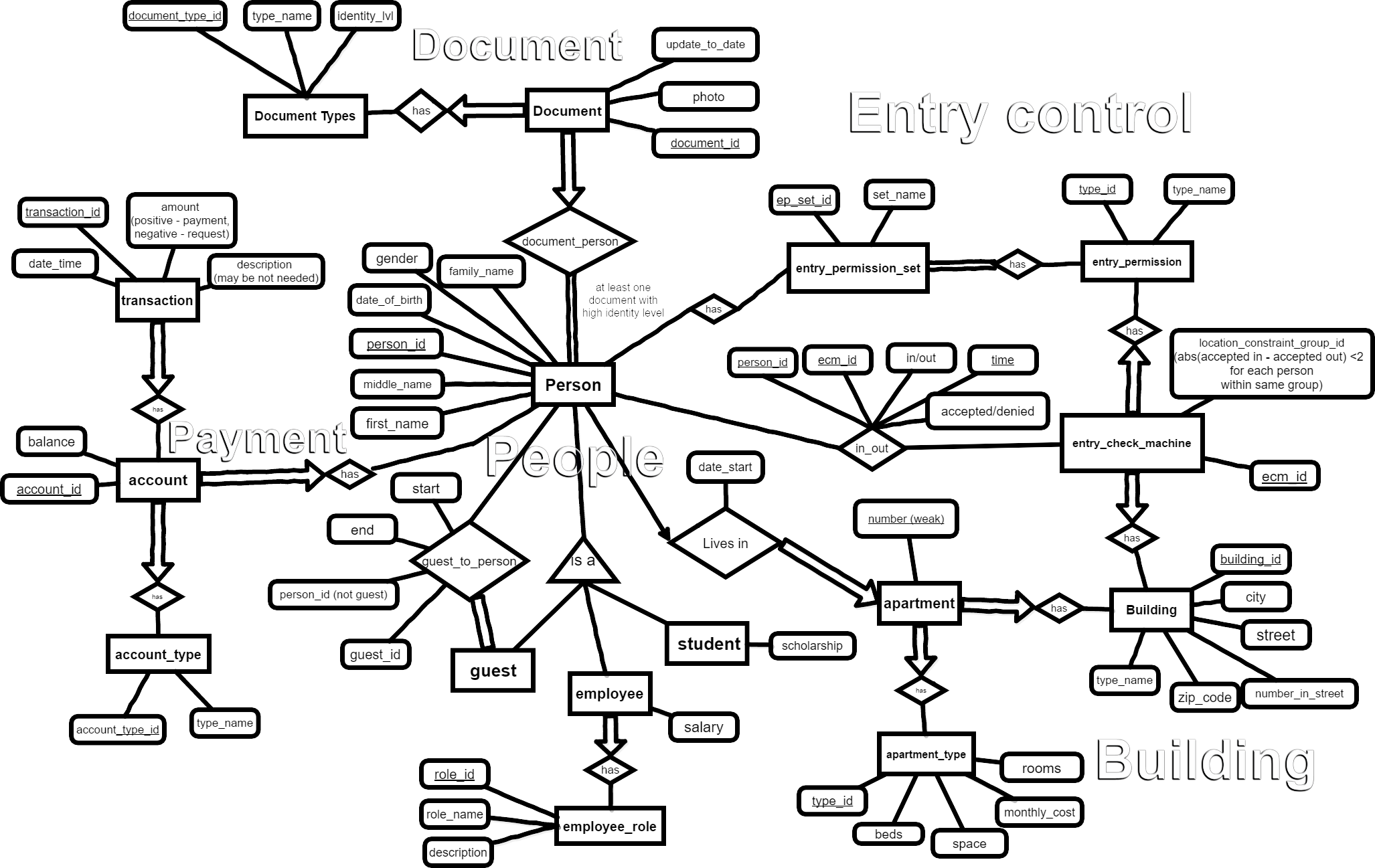
As a software platform for creating and delivering client side application we chose JavaFX. It provides a clear and clean architecture and features many enhancements such as styling, event management, transitions. Furthermore, it provides all the professional Java tooling required to debug, analyze, profile, and log a client application.

Another advantage of JavaFX is that it enables a simple app-like installation on the client side, without any prerequisites.

## JDBC

Java Database Connectivity (JDBC) was implemented as an application programming interface (API) which defines how a client may access a database. The combination of the Java API and the JDBC API made our application development easy and cost effective. Also, The JDBC API includes a way to identify and connect to a data source, using a DataSource object. This made code even more portable and easier to maintain.

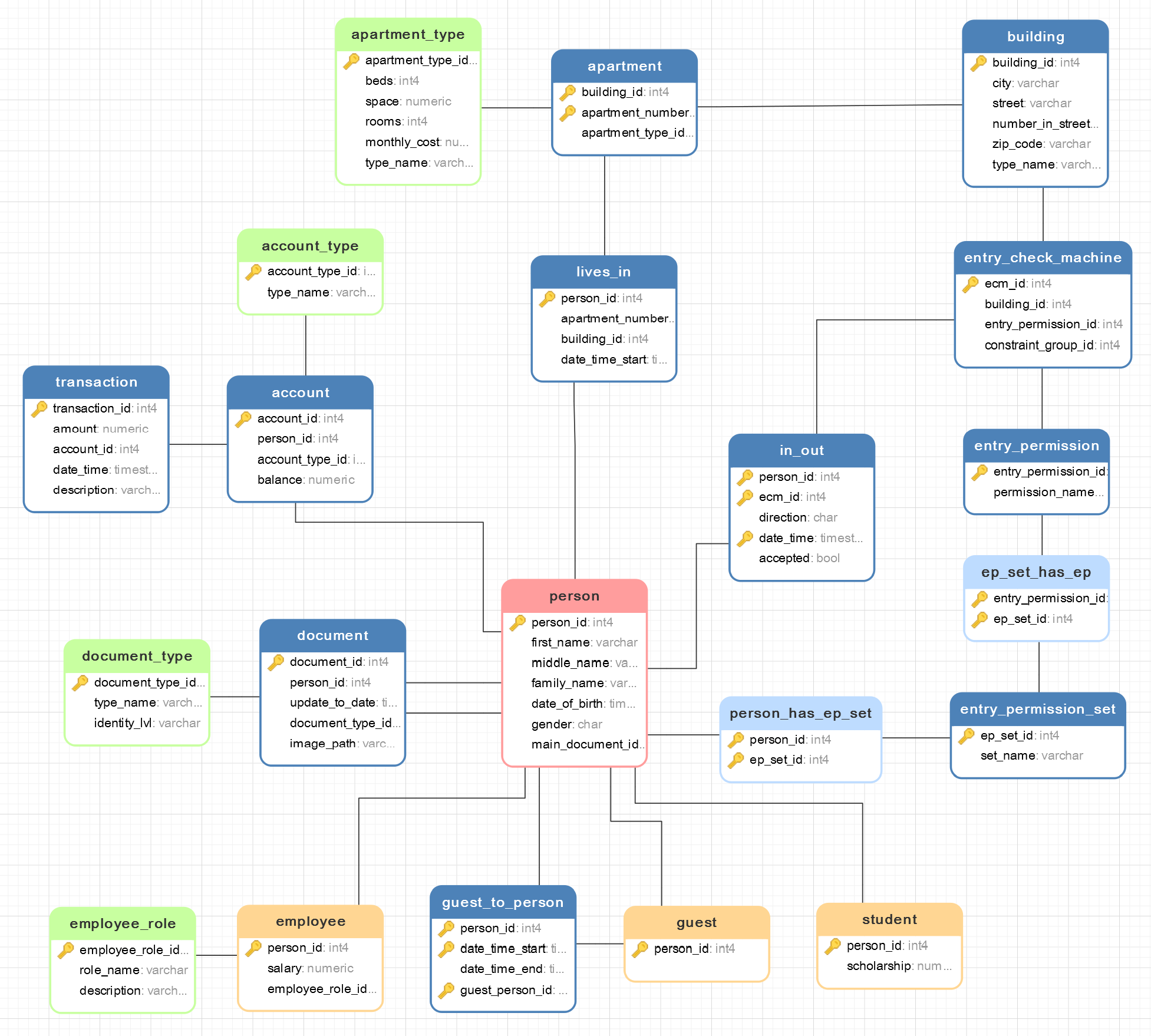
# ER-diagram



Above illustrated Entity-Relationship (ER) model, visually represents the structure of Campus Administration database, where data equates to entities and objects, which are linked by defined relationships expressing dependencies and requirements.

The ER diagram of Campus Administration consists of five parts, which describe roles of building, people, payment, entry control and documents. Specifically, the campus consists of several dormitory buildings. Our campus offers maximum one apartment for each person. Apartments (weak entity – depends on building) differ by such parameters as monthly cost, number of beds etc. Moreover, each building can have an entry checking machines, which control all check ins/outs. An entry-checking machine has an entry permission, and if entry permission sets of the person contains this permission then he can access the building or some specific zone inside the building. In addition, entry-checking machines could form groups; inside each group there is a constraint, that person cannot go out/into this group more than one time consecutively (for example: if person have entered campus, he must out from campus, to enter again, because entry-checking machines on the perimeter of campus form a group). There are three types of person: student, guest and employee. Each guest should be invited by a person (who are not guest) and attached to that person (each guest should have at least one guest\_to\_person relation). As for employee, he has a role that describes his position in the campus. Student and employee can have accounts like scholarship, rent payment, salary etc. with the help of which he can make transactions according to his need. Balance of each account calculated by trigger. Each transaction must be attached to the specific account. Furthermore, each person must have at least one document with high identity level.

# Database design schema



Many-to-many relations becomes separate tables, other constraints from ER implemented by PK, FK and Unique:   
*-- Uniques structure for table account***ALTER TABLE** "public"."account" **ADD UNIQUE** (**"person\_id"**, **"account\_type\_id"**);  
*-- Primary Key structure for table account***ALTER TABLE** "public"."account" **ADD PRIMARY KEY** (**"account\_id"**);  
*-- Uniques structure for table account\_type***ALTER TABLE** "public"."account\_type" **ADD UNIQUE** (**"type\_name"**) **DEFERRABLE**;  
*-- Primary Key structure for table account\_type***ALTER TABLE** "public"."account\_type" **ADD PRIMARY KEY** (**"account\_type\_id"**);  
*-- Uniques structure for table apartment***ALTER TABLE** "public"."apartment" **ADD UNIQUE** (**"building\_id"**, **"apartment\_number"**);  
*-- Primary Key structure for table apartment***ALTER TABLE** "public"."apartment" **ADD PRIMARY KEY** (**"building\_id"**, **"apartment\_number"**);  
*-- Uniques structure for table apartment\_type***ALTER TABLE** "public"."apartment\_type" **ADD UNIQUE** (**"type\_name"**) **DEFERRABLE**;  
*-- Primary Key structure for table apartment\_type***ALTER TABLE** "public"."apartment\_type" **ADD PRIMARY KEY** (**"apartment\_type\_id"**);  
*-- Uniques structure for table building***ALTER TABLE** "public"."building" **ADD UNIQUE** (**"city"**, **"street"**, **"number\_in\_street"**);  
*-- Primary Key structure for table building***ALTER TABLE** "public"."building" **ADD PRIMARY KEY** (**"building\_id"**);  
*-- Primary Key structure for table document***ALTER TABLE** "public"."document" **ADD PRIMARY KEY** (**"document\_id"**);  
*-- Uniques structure for table document\_type***ALTER TABLE** "public"."document\_type" **ADD UNIQUE** (**"type\_name"**) **DEFERRABLE**;  
*-- Primary Key structure for table document\_type***ALTER TABLE** "public"."document\_type" **ADD PRIMARY KEY** (**"document\_type\_id"**);  
*-- Primary Key structure for table employee***ALTER TABLE** "public"."employee" **ADD PRIMARY KEY** (**"person\_id"**);  
*-- Uniques structure for table employee\_role***ALTER TABLE** "public"."employee\_role" **ADD UNIQUE** (**"role\_name"**) **DEFERRABLE**;  
*-- Primary Key structure for table employee\_role***ALTER TABLE** "public"."employee\_role" **ADD PRIMARY KEY** (**"employee\_role\_id"**);  
*-- Primary Key structure for table entry\_check\_machine***ALTER TABLE** "public"."entry\_check\_machine" **ADD PRIMARY KEY** (**"ecm\_id"**);  
*-- Uniques structure for table entry\_permission***ALTER TABLE** "public"."entry\_permission" **ADD UNIQUE** (**"permission\_name"**) **DEFERRABLE**;  
*-- Primary Key structure for table entry\_permission***ALTER TABLE** "public"."entry\_permission" **ADD PRIMARY KEY** (**"entry\_permission\_id"**);  
*-- Uniques structure for table entry\_permission\_set***ALTER TABLE** "public"."entry\_permission\_set" **ADD UNIQUE** (**"set\_name"**) **DEFERRABLE**;  
*-- Primary Key structure for table entry\_permission\_set***ALTER TABLE** "public"."entry\_permission\_set" **ADD PRIMARY KEY** (**"ep\_set\_id"**);  
*-- Primary Key structure for table ep\_set\_has\_ep***ALTER TABLE** "public"."ep\_set\_has\_ep" **ADD PRIMARY KEY** (**"entry\_permission\_id"**, **"ep\_set\_id"**);  
*-- Primary Key structure for table guest***ALTER TABLE** "public"."guest" **ADD PRIMARY KEY** (**"person\_id"**);  
*-- Primary Key structure for table guest\_to\_person***ALTER TABLE** "public"."guest\_to\_person" **ADD PRIMARY KEY** (**"person\_id"**, **"date\_time\_start"**, **"guest\_person\_id"**);  
*-- Primary Key structure for table in\_out***ALTER TABLE** "public"."in\_out" **ADD PRIMARY KEY** (**"date\_time"**, **"person\_id"**, **"ecm\_id"**);  
*-- Indexes structure for table lives\_in***CREATE INDEX** "apartment\_clustered\_index" **ON** "public"."lives\_in" **USING btree** (**"apartment\_number"**, **"building\_id"**);  
**ALTER TABLE** "public"."lives\_in" **CLUSTER ON** "apartment\_clustered\_index";  
*-- Primary Key structure for table lives\_in***ALTER TABLE** "public"."lives\_in" **ADD PRIMARY KEY** (**"person\_id"**);  
*-- Uniques structure for table person***ALTER TABLE** "public"."person" **ADD UNIQUE** (**"main\_document\_id"**) **DEFERRABLE**;  
**ALTER TABLE** "public"."person" **ADD UNIQUE** (**"first\_name"**, **"family\_name"**, **"date\_of\_birth"**, **"middle\_name"**);  
*-- Primary Key structure for table person***ALTER TABLE** "public"."person" **ADD PRIMARY KEY** (**"person\_id"**);  
*-- Primary Key structure for table person\_has\_ep\_set***ALTER TABLE** "public"."person\_has\_ep\_set" **ADD PRIMARY KEY** (**"person\_id"**, **"ep\_set\_id"**);  
*-- Primary Key structure for table student***ALTER TABLE** "public"."student" **ADD PRIMARY KEY** (**"person\_id"**);  
*-- Indexes structure for table transaction***CREATE INDEX** "transaction\_account\_id\_index" **ON** "public"."transaction" **USING btree** (**"account\_id"**);  
**ALTER TABLE** "public"."transaction" **CLUSTER ON** "transaction\_account\_id\_index";

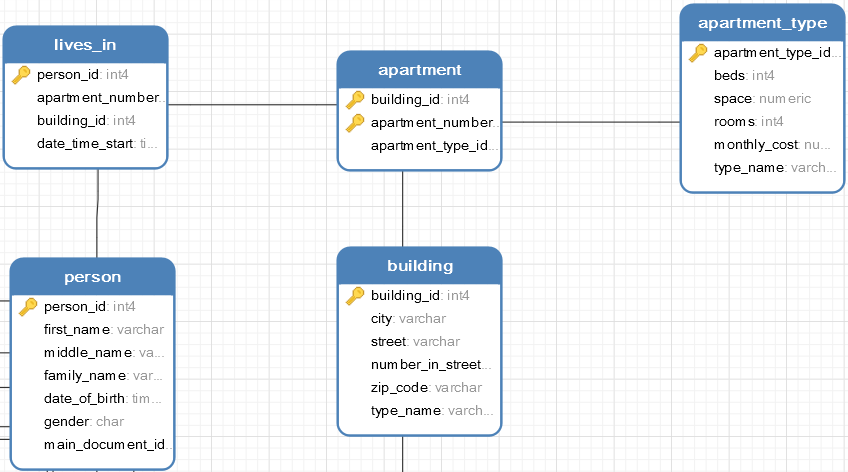
*-- Uniques structure for table transaction***ALTER TABLE** "public"."transaction" **ADD UNIQUE** (**"account\_id"**, **"date\_time"**);  
*-- Primary Key structure for table transaction***ALTER TABLE** "public"."transaction" **ADD PRIMARY KEY** (**"transaction\_id"**);  
*-- Foreign Key structure for table "public"."account"***ALTER TABLE** "public"."account" **ADD FOREIGN KEY** (**"person\_id"**) **REFERENCES** "public"."person" (**"person\_id"**) **ON DELETE RESTRICT ON UPDATE CASCADE DEFERRABLE**;  
**ALTER TABLE** "public"."account" **ADD FOREIGN KEY** (**"account\_type\_id"**) **REFERENCES** "public"."account\_type" (**"account\_type\_id"**) **ON DELETE RESTRICT ON UPDATE CASCADE**;  
*-- Foreign Key structure for table "public"."apartment"***ALTER TABLE** "public"."apartment" **ADD FOREIGN KEY** (**"building\_id"**) **REFERENCES** "public"."building" (**"building\_id"**) **ON DELETE RESTRICT ON UPDATE CASCADE DEFERRABLE**;  
**ALTER TABLE** "public"."apartment" **ADD FOREIGN KEY** (**"apartment\_type\_id"**) **REFERENCES** "public"."apartment\_type" (**"apartment\_type\_id"**) **ON DELETE RESTRICT ON UPDATE CASCADE DEFERRABLE**;  
*-- Foreign Key structure for table "public"."document"***ALTER TABLE** "public"."document" **ADD FOREIGN KEY** (**"document\_type\_id"**) **REFERENCES** "public"."document\_type" (**"document\_type\_id"**) **ON DELETE RESTRICT ON UPDATE CASCADE DEFERRABLE**;  
**ALTER TABLE** "public"."document" **ADD FOREIGN KEY** (**"person\_id"**) **REFERENCES** "public"."person" (**"person\_id"**) **ON DELETE CASCADE ON UPDATE CASCADE DEFERRABLE**;  
*-- Foreign Key structure for table "public"."employee"***ALTER TABLE** "public"."employee" **ADD FOREIGN KEY** (**"employee\_role\_id"**) **REFERENCES** "public"."employee\_role" (**"employee\_role\_id"**) **ON DELETE RESTRICT ON UPDATE CASCADE DEFERRABLE**;  
**ALTER TABLE** "public"."employee" **ADD FOREIGN KEY** (**"person\_id"**) **REFERENCES** "public"."person" (**"person\_id"**) **ON DELETE RESTRICT ON UPDATE CASCADE DEFERRABLE**;  
*-- Foreign Key structure for table "public"."entry\_check\_machine"***ALTER TABLE** "public"."entry\_check\_machine" **ADD FOREIGN KEY** (**"entry\_permission\_id"**) **REFERENCES** "public"."entry\_permission" (**"entry\_permission\_id"**) **ON DELETE RESTRICT ON UPDATE CASCADE DEFERRABLE**;  
**ALTER TABLE** "public"."entry\_check\_machine" **ADD FOREIGN KEY** (**"building\_id"**) **REFERENCES** "public"."building" (**"building\_id"**) **ON DELETE RESTRICT ON UPDATE CASCADE DEFERRABLE**;  
*-- Foreign Key structure for table "public"."ep\_set\_has\_ep"***ALTER TABLE** "public"."ep\_set\_has\_ep" **ADD FOREIGN KEY** (**"ep\_set\_id"**) **REFERENCES** "public"."entry\_permission\_set" (**"ep\_set\_id"**) **ON DELETE RESTRICT ON UPDATE CASCADE DEFERRABLE**;  
**ALTER TABLE** "public"."ep\_set\_has\_ep" **ADD FOREIGN KEY** (**"entry\_permission\_id"**) **REFERENCES** "public"."entry\_permission" (**"entry\_permission\_id"**) **ON DELETE RESTRICT ON UPDATE CASCADE DEFERRABLE**;  
*-- Foreign Key structure for table "public"."guest"***ALTER TABLE** "public"."guest" **ADD FOREIGN KEY** (**"person\_id"**) **REFERENCES** "public"."person" (**"person\_id"**) **ON DELETE RESTRICT ON UPDATE CASCADE DEFERRABLE**;  
*-- Foreign Key structure for table "public"."guest\_to\_person"***ALTER TABLE** "public"."guest\_to\_person" **ADD FOREIGN KEY** (**"guest\_person\_id"**) **REFERENCES** "public"."guest" (**"person\_id"**) **ON DELETE RESTRICT ON UPDATE CASCADE DEFERRABLE**;  
**ALTER TABLE** "public"."guest\_to\_person" **ADD FOREIGN KEY** (**"person\_id"**) **REFERENCES** "public"."person" (**"person\_id"**) **ON DELETE RESTRICT ON UPDATE CASCADE DEFERRABLE**;  
*-- Foreign Key structure for table "public"."in\_out"***ALTER TABLE** "public"."in\_out" **ADD FOREIGN KEY** (**"person\_id"**) **REFERENCES** "public"."person" (**"person\_id"**) **ON DELETE RESTRICT ON UPDATE CASCADE DEFERRABLE**;  
**ALTER TABLE** "public"."in\_out" **ADD FOREIGN KEY** (**"ecm\_id"**) **REFERENCES** "public"."entry\_check\_machine" (**"ecm\_id"**) **ON DELETE RESTRICT ON UPDATE CASCADE DEFERRABLE**;  
*-- Foreign Key structure for table "public"."lives\_in"***ALTER TABLE** "public"."lives\_in" **ADD FOREIGN KEY** (**"person\_id"**) **REFERENCES** "public"."person" (**"person\_id"**) **ON DELETE RESTRICT ON UPDATE CASCADE DEFERRABLE**;  
**ALTER TABLE** "public"."lives\_in" **ADD FOREIGN KEY** (**"building\_id"**, **"apartment\_number"**) **REFERENCES** "public"."apartment" (**"building\_id"**, **"apartment\_number"**) **ON DELETE RESTRICT ON UPDATE CASCADE DEFERRABLE**;  
*-- Foreign Key structure for table "public"."person"***ALTER TABLE** "public"."person" **ADD FOREIGN KEY** (**"main\_document\_id"**) **REFERENCES** "public"."document" (**"document\_id"**) **ON DELETE RESTRICT ON UPDATE CASCADE DEFERRABLE**;

*-- Foreign Key structure for table "public"."person\_has\_ep\_set"***ALTER TABLE** "public"."person\_has\_ep\_set" **ADD FOREIGN KEY** (**"ep\_set\_id"**) **REFERENCES** "public"."entry\_permission\_set" (**"ep\_set\_id"**) **ON DELETE RESTRICT ON UPDATE CASCADE DEFERRABLE**;  
**ALTER TABLE** "public"."person\_has\_ep\_set" **ADD FOREIGN KEY** (**"person\_id"**) **REFERENCES** "public"."person" (**"person\_id"**) **ON DELETE CASCADE ON UPDATE CASCADE DEFERRABLE**;

*-- Foreign Key structure for table "public"."student"***ALTER TABLE** "public"."student" **ADD FOREIGN KEY** (**"person\_id"**) **REFERENCES** "public"."person" (**"person\_id"**) **ON DELETE RESTRICT ON UPDATE CASCADE DEFERRABLE**;  
*-- Foreign Key structure for table "public"."transaction"***ALTER TABLE** "public"."transaction" **ADD FOREIGN KEY** (**"account\_id"**) **REFERENCES** "public"."account" (**"account\_id"**) **ON DELETE RESTRICT ON UPDATE CASCADE DEFERRABLE**;

# Description of normalization

In our database schema in 3rd NF, except building table.

Here building table not in 3rd NF, because it has transitive dependencies, but we make a decision not to separate this table because it has few number of rows (5), in future their count likely will be less than 10 and likely will not be changed in future. Also creating address classification - not the aim of this database.

Other tables has no partial and transitive dependencies.

# List of queries

As queries in our project we used views for several reasons:

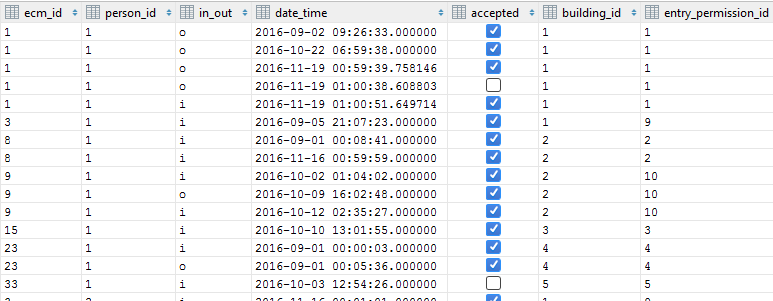
* name of view explains what exactly happens in this query;
* this allows to separate work with queries and user interface, because if query inside view was changed – no need to change code inside app;
* this allows to create complex queries in more simple way.

*-- 1)View structure for apartment\_occupation***CREATE OR REPLACE VIEW** "public"."apartment\_occupation" **AS  
 SELECT** *count*(li.**person\_id**) **AS** beds\_occupied,  
 a.**apartment\_number**,  
 a.**building\_id**,  
 **at**.**beds**,  
 (**at**.**beds** - *count*(li.**person\_id**)) **AS** free\_beds  
 **FROM** ((apartment a  
 **LEFT JOIN** lives\_in li **ON** (((li.**building\_id** = a.**building\_id**) **AND** (li.**apartment\_number** = a.**apartment\_number**))))  
 **JOIN** apartment\_type **at ON** ((a.**apartment\_type\_id** = **at**.**apartment\_type\_id**)))  
 **GROUP BY** a.**apartment\_number**, a.**building\_id**, **at**.**beds**;  
*-- 2)View structure for apartments\_with\_employee***CREATE OR REPLACE VIEW** "public"."apartments\_with\_employee" **AS  
 SELECT** a.**apartment\_number**,  
 a.**building\_id  
 FROM** ((apartment a  
 **JOIN** lives\_in li **ON** (((a.**building\_id** = li.**building\_id**) **AND** (a.**apartment\_number** = li.**apartment\_number**))))  
 **JOIN** employee e **ON** ((li.**person\_id** = e.**person\_id**)));  
*-- 3)View structure for apartments\_with\_female\_persons***CREATE OR REPLACE VIEW** "public"."apartments\_with\_female\_persons" **AS  
 SELECT** a.**apartment\_number**,  
 a.**building\_id  
 FROM** ((apartment a  
 **JOIN** lives\_in li **ON** (((a.**building\_id** = li.**building\_id**) **AND** (a.**apartment\_number** = li.**apartment\_number**))))  
 **JOIN** person p **ON** ((li.**person\_id** = p.**person\_id**)))  
 **WHERE** (p.**gender** = **'F'**::bpchar);  
*-- 4)View structure for apartments\_with\_free\_beds***CREATE OR REPLACE VIEW** "public"."apartments\_with\_free\_beds" **AS  
 SELECT** apartment\_occupation.beds\_occupied,  
 apartment\_occupation.**apartment\_number**,  
 apartment\_occupation.**building\_id**,  
 apartment\_occupation.**beds**,  
 apartment\_occupation.**free\_beds  
 FROM** apartment\_occupation  
 **WHERE** (apartment\_occupation.**free\_beds** > 0);  
*-- 5)View structure for apartments\_with\_male\_persons***CREATE OR REPLACE VIEW** "public"."apartments\_with\_male\_persons" **AS  
 SELECT** a.**apartment\_number**,  
 a.**building\_id  
 FROM** ((apartment a  
 **JOIN** lives\_in li **ON** (((a.**building\_id** = li.**building\_id**) **AND** (a.**apartment\_number** = li.**apartment\_number**))))  
 **JOIN** person p **ON** ((li.**person\_id** = p.**person\_id**)))  
 **WHERE** (p.**gender** = **'M'**::bpchar);  
*-- 6)View structure for apartments\_with\_students***CREATE OR REPLACE VIEW** "public"."apartments\_with\_students" **AS  
 SELECT** a.**apartment\_number**,  
 a.**building\_id  
 FROM** ((apartment a  
 **JOIN** lives\_in li **ON** (((a.**building\_id** = li.**building\_id**) **AND** (a.**apartment\_number** = li.**apartment\_number**))))  
 **JOIN** student s **ON** ((li.**person\_id** = s.**person\_id**)));  
*-- 7)View structure for apartments\_for\_female\_employee***CREATE OR REPLACE VIEW** "public"."apartments\_for\_female\_employee" **AS  
 SELECT** awfb.**beds\_occupied**,  
 awfb.**apartment\_number**,  
 awfb.**building\_id**,  
 awfb.**beds**,  
 awfb.**free\_beds  
 FROM** apartments\_with\_free\_beds awfb  
 **WHERE** ((**NOT** (*EXISTS* ( **SELECT** 1  
 **FROM** apartments\_with\_male\_persons awmp  
 **WHERE** ((awfb.**apartment\_number** = awmp.**apartment\_number**) **AND** (awfb.**building\_id** = awmp.**building\_id**))))) **AND** (**NOT** (*EXISTS* ( **SELECT** 1  
 **FROM** apartments\_with\_students aws  
 **WHERE** ((awfb.**apartment\_number** = aws.**apartment\_number**) **AND** (awfb.**building\_id** = aws.**building\_id**))))));

*-- often when need to find apartment for person there is a constraint that female couldn’t live with male and employee couldn’t live with student in one apartment. However, if they want, we could settle them together – there is no restriction in our database, it is just recommendation and help views for administrator.*  
*-- 8)View structure for apartments\_for\_female\_students***CREATE OR REPLACE VIEW** "public"."apartments\_for\_female\_students" **AS  
 SELECT** awfb.**beds\_occupied**,  
 awfb.**apartment\_number**,  
 awfb.**building\_id**,  
 awfb.**beds**,  
 awfb.**free\_beds  
 FROM** apartments\_with\_free\_beds awfb  
 **WHERE** ((**NOT** (*EXISTS* ( **SELECT** 1  
 **FROM** apartments\_with\_male\_persons awmp  
 **WHERE** ((awfb.**apartment\_number** = awmp.**apartment\_number**) **AND** (awfb.**building\_id** = awmp.**building\_id**))))) **AND** (**NOT** (*EXISTS* ( **SELECT** 1  
 **FROM** apartments\_with\_employee awe  
 **WHERE** ((awfb.**apartment\_number** = awe.**apartment\_number**) **AND** (awfb.**building\_id** = awe.**building\_id**))))));  
*-- 9)View structure for apartments\_for\_male\_employee***CREATE OR REPLACE VIEW** "public"."apartments\_for\_male\_employee" **AS  
 SELECT** awfb.**beds\_occupied**,  
 awfb.**apartment\_number**,  
 awfb.**building\_id**,  
 awfb.**beds**,  
 awfb.**free\_beds  
 FROM** apartments\_with\_free\_beds awfb  
 **WHERE** ((**NOT** (*EXISTS* ( **SELECT** 1  
 **FROM** apartments\_with\_female\_persons awfp  
 **WHERE** ((awfb.**apartment\_number** = awfp.**apartment\_number**) **AND** (awfb.**building\_id** = awfp.**building\_id**))))) **AND** (**NOT** (*EXISTS* ( **SELECT** 1  
 **FROM** apartments\_with\_students aws  
 **WHERE** ((awfb.**apartment\_number** = aws.**apartment\_number**) **AND** (awfb.**building\_id** = aws.**building\_id**))))));  
*-- 10)View structure for apartments\_for\_male\_students***CREATE OR REPLACE VIEW** "public"."apartments\_for\_male\_students" **AS  
 SELECT** awfb.**beds\_occupied**,  
 awfb.**apartment\_number**,  
 awfb.**building\_id**,  
 awfb.**beds**,  
 awfb.**free\_beds  
 FROM** apartments\_with\_free\_beds awfb  
 **WHERE** ((**NOT** (*EXISTS* ( **SELECT** 1  
 **FROM** apartments\_with\_female\_persons awfp  
 **WHERE** ((awfb.**apartment\_number** = awfp.**apartment\_number**) **AND** (awfb.**building\_id** = awfp.**building\_id**))))) **AND** (**NOT** (*EXISTS* ( **SELECT** 1  
 **FROM** apartments\_with\_employee awe  
 **WHERE** ((awfb.**apartment\_number** = awe.**apartment\_number**) **AND** (awfb.**building\_id** = awe.**building\_id**))))));  
*-- 11)View structure for employees\_without\_apartments***CREATE OR REPLACE VIEW** "public"."employees\_without\_apartments" **AS  
 SELECT** p.**person\_id**,  
 p.**first\_name**,  
 p.**middle\_name**,  
 p.**family\_name**,  
 p.**date\_of\_birth**,  
 p.**gender  
 FROM** ((person p  
 **JOIN** employee e **ON** ((p.**person\_id** = e.**person\_id**)))  
 **LEFT JOIN** lives\_in li **ON** ((p.**person\_id** = li.**person\_id**)))  
 **WHERE** (li.**apartment\_number IS NULL**);

*-- next views search for guests who are staying in campus more than 24 hours. Often host needs to pay for such guests*  
*-- 12)View structure for guest\_control***CREATE OR REPLACE VIEW** "public"."guest\_control" **AS  
 SELECT** guest.**person\_id AS** guest\_id,  
 person.**first\_name**,  
 person.**family\_name**,  
 (g.**date\_time\_end** - g.**date\_time\_start**) **AS** stay\_time  
 **FROM** ((guest  
 **JOIN** person **ON** ((guest.**person\_id** = person.**person\_id**)))  
 **JOIN** guest\_to\_person g **ON** ((guest.**person\_id** = g.**guest\_person\_id**)))  
 **WHERE** ((g.**date\_time\_end** - g.**date\_time\_start**) > **'24:00:00'**::**interval**);  
*-- 13)View structure for guest\_control\_with\_host***CREATE OR REPLACE VIEW** "public"."guest\_control\_with\_host" **AS  
 SELECT** guest.**person\_id AS** guest\_person\_id,  
 p1.**first\_name AS** guest\_name,  
 p1.**family\_name AS** guest\_family\_name,  
 (g.date\_time\_end - g.date\_time\_start) **AS** stay\_time,  
 p2.**person\_id AS** host\_person\_id,  
 p2.**first\_name AS** host\_first\_name,  
 p2.**family\_name AS** host\_family\_name  
 **FROM** (((guest  
 **JOIN** person p1 **ON** ((guest.**person\_id** = p1.**person\_id**)))  
 **JOIN** guest\_to\_person g **ON** ((guest.person\_id = g.guest\_person\_id)))  
 **JOIN** person p2 **ON** ((g.person\_id = p2.**person\_id**)))  
 **WHERE** ((g.date\_time\_end - g.date\_time\_start) > **'24:00:00'**::**interval**);

*-- next view show all the information of passing the entrance guard machine when person have a permission (interesting thing here is “not accepted” entries, because this means that person tries to enter/escape to/from campus territory twice or more times consecutively. This means that person gave his pass card to someone else, that is against the rules)*  
*-- 14)View structure for have\_entry\_permissions***CREATE OR REPLACE VIEW** "public"."have\_entry\_permissions" **AS  
 SELECT** in\_out.**ecm\_id**,  
 in\_out.**person\_id**,  
 in\_out.**direction AS** in\_out,  
 in\_out.**date\_time**,  
 in\_out.**accepted**,  
 entry\_check\_machine.**building\_id**,  
 entry\_check\_machine.**entry\_permission\_id  
 FROM** (in\_out  
 **JOIN** entry\_check\_machine **USING** (**ecm\_id**))  
 **WHERE** (entry\_check\_machine.**entry\_permission\_id IN** ( **SELECT** ep.**entry\_permission\_id  
 FROM** ((((person  
 **JOIN** person\_has\_ep\_set **USING** (**person\_id**))  
 **JOIN** entry\_permission\_set **USING** (ep\_set\_id))  
 **JOIN** ep\_set\_has\_ep **USING** (ep\_set\_id))  
 **JOIN** entry\_permission ep **USING** (**entry\_permission\_id**))  
 **WHERE** (in\_out.**person\_id** = person.**person\_id**)));



*-- select the last time a person get through a entrance guard machine. It will be useful for checking if a person didn’t active for a long time.*

*-- 15)View structure for last\_time\_person\_accepted\_in\_out***CREATE OR REPLACE VIEW** "public"."last\_time\_person\_accepted\_in\_out" **AS  
 SELECT** in\_out.**person\_id**,  
 *max*(in\_out.**date\_time**) **AS** last\_accepted\_entry  
 **FROM** in\_out  
 **WHERE** (in\_out.**accepted** = **true**)  
 **GROUP BY** in\_out.**person\_id**;

*-- select the information about the unsuccessful passing (when person tries to enter a place and hi haven’t got permissions for this)*  
*-- 16)View structure for no\_entry\_permission\_for\_in\_out***CREATE OR REPLACE VIEW** "public"."no\_entry\_permission\_for\_in\_out" **AS  
 SELECT** in\_out.**ecm\_id**,  
 in\_out.**person\_id**,  
 in\_out.**direction AS** in\_out,  
 in\_out.**date\_time**,  
 in\_out.**accepted**,  
 entry\_check\_machine.**building\_id**,  
 entry\_check\_machine.**entry\_permission\_id  
 FROM** (in\_out  
 **JOIN** entry\_check\_machine **USING** (**ecm\_id**))  
 **WHERE** (**NOT** (entry\_check\_machine.**entry\_permission\_id IN** ( **SELECT** ep.**entry\_permission\_id  
 FROM** ((((person  
 **JOIN** person\_has\_ep\_set **USING** (**person\_id**))  
 **JOIN** entry\_permission\_set **USING** (ep\_set\_id))  
 **JOIN** ep\_set\_has\_ep **USING** (ep\_set\_id))  
 **JOIN** entry\_permission ep **USING** (**entry\_permission\_id**))  
 **WHERE** (in\_out.**person\_id** = person.**person\_id**))));  
*-- 17)View structure for outdated\_documents***CREATE OR REPLACE VIEW** "public"."outdated\_documents" **AS  
 SELECT document**.**person\_id**,  
 **document**.**document\_id**,  
 **document**.**document\_type\_id**,  
 document\_type.**type\_name**,  
 document\_type.**identity\_lvl**,  
 (*now*() - (**document**.**update\_to\_date**)::**timestamp with time zone**) **AS** overdue\_for,  
 person.**first\_name**,  
 person.**family\_name  
 FROM** ((**document  
 JOIN** document\_type **USING** (**document\_type\_id**))  
 **JOIN** person **USING** (**person\_id**))  
 **WHERE** (**document**.**update\_to\_date** < *now*());  
*-- 18)View structure for outdated\_documents\_of\_students\_and\_employee***CREATE OR REPLACE VIEW** "public"."outdated\_documents\_of\_students\_and\_employee" **AS  
 SELECT document**.**person\_id**,  
 **document**.**document\_id**,  
 **document**.**document\_type\_id**,  
 document\_type.**type\_name**,  
 document\_type.**identity\_lvl**,  
 (*now*() - (**document**.**update\_to\_date**)::**timestamp with time zone**) **AS** overdue\_for,  
 person.**first\_name**,  
 person.**family\_name  
 FROM** ((**document  
 JOIN** document\_type **USING** (**document\_type\_id**))  
 **JOIN** person **USING** (**person\_id**))  
 **WHERE** ((**document**.**update\_to\_date** < *now*()) **AND** (**document**.**person\_id IN** ( **SELECT** student.**person\_id  
 FROM** student  
 **UNION  
 SELECT** employee.**person\_id  
 FROM** employee)));  
*-- 19)View structure for personnel\_attendance\_control***CREATE OR REPLACE VIEW** "public"."personnel\_attendance\_control" **AS  
 SELECT** person.**person\_id**,  
 person.**first\_name**,  
 person.**family\_name**,  
 (*now*() - (t.last\_accepted\_entry)::**timestamp with time zone**) **AS** no\_action\_for  
 **FROM** (( **SELECT** in\_out.**person\_id**,  
 *max*(in\_out.**date\_time**) **AS** last\_accepted\_entry  
 **FROM** in\_out  
 **WHERE** (((*now*() - (in\_out.**date\_time**)::**timestamp with time zone**) > **'168:00:00'**::**interval**) **AND** (in\_out.**accepted** = **true**))  
 **GROUP BY** in\_out.**person\_id  
 ORDER BY** (*max*(in\_out.**date\_time**))) t  
 **JOIN** person **ON** ((t.**person\_id** = person.**person\_id**)));  
*-- 20)View structure for persons\_inside\_campus\_now***CREATE OR REPLACE VIEW** "public"."persons\_inside\_campus\_now" **AS  
 SELECT** in\_out.**person\_id**,  
 in\_out.**ecm\_id**,  
 in\_out.**direction**,  
 in\_out.**date\_time**,  
 in\_out.**accepted**,  
 entry\_check\_machine.**building\_id**,  
 entry\_check\_machine.**entry\_permission\_id**,  
 entry\_check\_machine.**constraint\_group\_id**,  
 person.**first\_name**,  
 person.**middle\_name**,  
 person.**family\_name**,  
 person.**date\_of\_birth**,  
 person.**gender**,  
 person.**main\_document\_id  
 FROM** ((in\_out  
 **JOIN** entry\_check\_machine **USING** (**ecm\_id**))  
 **JOIN** person **USING** (**person\_id**))  
 **WHERE** ((in\_out.**person\_id IN** ( **SELECT** last\_time\_person\_accepted\_in\_out.**person\_id  
 FROM** last\_time\_person\_accepted\_in\_out)) **AND** (in\_out.**date\_time** = ( **SELECT** last\_time\_person\_accepted\_in\_out.**last\_accepted\_entry  
 FROM** last\_time\_person\_accepted\_in\_out  
 **WHERE** (last\_time\_person\_accepted\_in\_out.**person\_id** = in\_out.**person\_id**))) **AND** ((entry\_check\_machine.**constraint\_group\_id** <> 1) **OR** (in\_out.**direction** <> **'o'**::bpchar)));

*-- next two views used for data generation and in future will be helpful for queries like how long person was inside the campus (total or in some time interval)*  
*-- 21)View structure for persons\_with\_first\_accepted\_campus\_entry\_equals\_out***CREATE OR REPLACE VIEW** "public"."persons\_with\_first\_accepted\_campus\_entry\_equals\_out" **AS  
 SELECT** in\_out.**person\_id**,  
 in\_out.**date\_time**,  
 in\_out.**ecm\_id**,  
 in\_out.**direction**,  
 in\_out.**accepted  
 FROM** (in\_out  
 **JOIN** entry\_check\_machine **USING** (**ecm\_id**))  
 **WHERE** ((in\_out.**accepted** = **true**) **AND** (entry\_check\_machine.**constraint\_group\_id** = 1) **AND** (in\_out.**direction** = **'o'**::bpchar) **AND** (in\_out.**date\_time** <= **ALL** ( **SELECT** io.**date\_time  
 FROM** (in\_out io  
 **JOIN** entry\_check\_machine ecm **USING** (**ecm\_id**))  
 **WHERE** ((io.**accepted** = **true**) **AND** (ecm.**constraint\_group\_id** = 1) **AND** (io.**person\_id** = in\_out.**person\_id**)))))  
 **GROUP BY** in\_out.**person\_id**, in\_out.**date\_time**, in\_out.**ecm\_id**, in\_out.**direction**, in\_out.**accepted**;  
*-- 22)View structure for persons\_with\_last\_accepted\_campus\_entry\_equals\_in***CREATE OR REPLACE VIEW** "public"."persons\_with\_last\_accepted\_campus\_entry\_equals\_in" **AS  
 SELECT** in\_out.**person\_id**,  
 in\_out.**date\_time**,  
 in\_out.**ecm\_id**,  
 in\_out.**direction**,  
 in\_out.**accepted**,  
 *now*() **AS** now  
 **FROM** (in\_out  
 **JOIN** entry\_check\_machine **USING** (**ecm\_id**))  
 **WHERE** ((in\_out.**accepted** = **true**) **AND** (entry\_check\_machine.**constraint\_group\_id** = 1) **AND** (in\_out.**direction** = **'i'**::bpchar) **AND** (in\_out.**date\_time** >= **ALL** ( **SELECT** io.**date\_time  
 FROM** (in\_out io  
 **JOIN** entry\_check\_machine ecm **USING** (**ecm\_id**))  
 **WHERE** ((io.**accepted** = **true**) **AND** (ecm.**constraint\_group\_id** = 1) **AND** (io.**person\_id** = in\_out.**person\_id**)))))  
 **GROUP BY** in\_out.**person\_id**, in\_out.**date\_time**, in\_out.**ecm\_id**, in\_out.**direction**, in\_out.**accepted**;  
*-- 23)View structure for rental\_fee\_balance***CREATE OR REPLACE VIEW** "public"."rental\_fee\_balance" **AS  
 SELECT** account.**person\_id**,  
 person.**first\_name**,  
 person.**family\_name**,  
 account.**balance AS** rental\_fee\_balance  
 **FROM** ((account  
 **JOIN** person **USING** (**person\_id**))  
 **JOIN** account\_type **ON** ((account\_type.**account\_type\_id** = account.**account\_type\_id**)))  
 **WHERE** ((account\_type.**type\_name**)::**text** = **'Rental Fee'**::**text**);  
*-- 24)View structure for rental\_fee\_balance\_negative***CREATE OR REPLACE VIEW** "public"."rental\_fee\_balance\_negative" **AS  
 SELECT** account.**person\_id**,  
 person.**first\_name**,  
 person.**family\_name**,  
 account.**balance AS** rental\_fee\_balance  
 **FROM** ((account  
 **JOIN** person **USING** (**person\_id**))  
 **JOIN** account\_type **ON** ((account\_type.**account\_type\_id** = account.**account\_type\_id**)))  
 **WHERE** (((account\_type.**type\_name**)::**text** = **'Rental Fee'**::**text**) **AND** (account.**balance** < (0)::**numeric**));  
*-- 25)View structure for students\_without\_apartment***CREATE OR REPLACE VIEW** "public"."students\_without\_apartment" **AS  
 SELECT** p.**person\_id**,  
 p.**first\_name**,  
 p.**middle\_name**,  
 p.**family\_name**,  
 p.**date\_of\_birth**,  
 p.**gender  
 FROM** ((person p  
 **JOIN** student s **ON** ((p.**person\_id** = s.**person\_id**)))  
 **LEFT JOIN** lives\_in li **ON** ((p.**person\_id** = li.**person\_id**)))  
 **WHERE** (li.**apartment\_number IS NULL**);  
*-- 26)View structure for tuition\_fee\_balance***CREATE OR REPLACE VIEW** "public"."tuition\_fee\_balance" **AS  
 SELECT** account.**person\_id**,  
 person.**first\_name**,  
 person.**family\_name**,  
 account.**balance AS** rental\_fee\_balance  
 **FROM** ((account  
 **JOIN** person **USING** (**person\_id**))  
 **JOIN** account\_type **ON** ((account\_type.**account\_type\_id** = account.**account\_type\_id**)))  
 **WHERE** ((account\_type.**type\_name**)::**text** = **'Tuition Fee'**::**text**);  
*-- 27)View structure for tuition\_fee\_balance\_negative***CREATE OR REPLACE VIEW** "public"."tuition\_fee\_balance\_negative" **AS  
 SELECT** account.**person\_id**,  
 person.**first\_name**,  
 person.**family\_name**,  
 account.**balance AS** rental\_fee\_balance  
 **FROM** ((account  
 **JOIN** person **USING** (**person\_id**))  
 **JOIN** account\_type **ON** ((account\_type.**account\_type\_id** = account.**account\_type\_id**)))  
 **WHERE** (((account\_type.**type\_name**)::**text** = **'Tuition Fee'**::**text**) **AND** (account.**balance** < (0)::**numeric**));

*-- 28…)Functions (sorry for text highlighting – MS Word have some problems with it)*

**CREATE OR REPLACE FUNCTION** *insert\_into\_in\_out*(\_person\_id **INT4**, \_ecm\_id **INT4**, \_direction **CHAR**(1)) **RETURNS BOOL AS $$  
 DECLARE const\_g\_id INT4;  
 DECLARE last\_direction CHAR(1);  
 BEGIN  
 IF (***exists***(SELECT ep.entry\_permission\_id, ecm.ecm\_id FROM person  
 NATURAL JOIN person\_has\_ep\_set  
 NATURAL JOIN entry\_permission\_set  
 NATURAL JOIN ep\_set\_has\_ep  
 NATURAL JOIN entry\_permission ep  
 JOIN entry\_check\_machine ecm ON ep.entry\_permission\_id = ecm.entry\_permission\_id  
 WHERE person.person\_id = \_person\_id AND ecm.ecm\_id = \_ecm\_id))  
 THEN  
 SELECT constraint\_group\_id FROM entry\_check\_machine  
 WHERE ecm\_id = \_ecm\_id  
 INTO const\_g\_id;  
 IF ((const\_g\_id) ISNULL )  
 THEN  
 INSERT INTO in\_out (person\_id, ecm\_id, direction, date\_time, accepted) VALUES (\_person\_id,\_ecm\_id,\_direction,***now***(),TRUE);  
 RETURN TRUE;  
 ELSE  
 SELECT direction FROM in\_out  
 NATURAL JOIN entry\_check\_machine  
 WHERE in\_out.person\_id = \_person\_id  
 AND entry\_check\_machine.constraint\_group\_id = const\_g\_id  
 AND in\_out.accepted = TRUE  
 ORDER BY date\_time DESC  
 LIMIT 1  
 INTO last\_direction;  
 IF (last\_direction = \_direction)  
 THEN  
 RAISE NOTICE 'several entry in one direction';  
 INSERT INTO in\_out (person\_id, ecm\_id, direction, date\_time, accepted) VALUES (\_person\_id,\_ecm\_id,\_direction,***now***(),FALSE );  
 RETURN FALSE;  
 ELSE  
 INSERT INTO in\_out (person\_id, ecm\_id, direction, date\_time, accepted) VALUES (\_person\_id,\_ecm\_id,\_direction,***now***(),TRUE);  
 RETURN TRUE;  
 END IF;  
 END IF;  
 ELSE  
 RAISE NOTICE 'no permission';  
 INSERT INTO in\_out (person\_id, ecm\_id, direction, date\_time, accepted) VALUES (\_person\_id,\_ecm\_id,\_direction,***now***(),FALSE );  
 RETURN FALSE;  
 END IF;  
 END;  
$$ LANGUAGE** plpgsql;  
**CREATE OR REPLACE FUNCTION** *add\_person\_into\_appartment*(person\_id **INTEGER**,  
 apartment\_number **INTEGER**, building\_id\_param **INTEGER**, date\_time\_start\_param **TIMESTAMP**) **RETURNS VOID AS $$  
BEGIN  
 IF (apartment\_number IN ( SELECT a.apartment\_number FROM apartments\_with\_free\_beds AS a WHERE a.building\_id = building\_id\_param))  
 THEN  
 INSERT INTO lives\_in VALUES (person\_id, apartment\_number, building\_id\_param, date\_time\_start\_param);  
 ELSE  
 RAISE EXCEPTION 'No free places'  
 USING HINT = 'Check apt number and building id';  
 END IF ;  
END;  
$$ LANGUAGE** plpgsql;  
  
**CREATE OR REPLACE FUNCTION** *add\_new\_student* (first\_name **VARCHAR**(50),middle\_name **VARCHAR**(50),family\_name **VARCHAR**(200),  
 gender **CHAR**, dob **TIMESTAMP**, image\_path **VARCHAR**(200),  
 scholarship **INTEGER**) **RETURNS VOID AS $$  
DECLARE  
 new\_person\_id INTEGER;  
 new\_doc\_id INTEGER;  
BEGIN  
 SELECT** *nextval***('person\_person\_id\_seq'::REGCLASS) INTO new\_person\_id;  
 BEGIN  
 SET CONSTRAINTS ALL DEFERRED;  
 INSERT INTO document VALUES (***nextval***('document\_document\_id\_seq'::REGCLASS), new\_person\_id,** *current\_timestamp* **+ INTERVAL '1 year', 1, image\_path)  
 RETURNING document\_id INTO new\_doc\_id;  
 INSERT INTO person VALUES (new\_person\_id, first\_name, middle\_name, family\_name, dob, gender, new\_doc\_id);  
 END;  
 INSERT INTO student VALUES (new\_person\_id, scholarship);  
 INSERT INTO account VALUES (***nextval***('account\_account\_id\_seq'::REGCLASS), new\_person\_id,1); -- rental\_fee  
 INSERT INTO account VALUES (***nextval***('account\_account\_id\_seq'::REGCLASS), new\_person\_id,2); -- tuition\_fee  
 INSERT INTO account VALUES (***nextval***('account\_account\_id\_seq'::REGCLASS), new\_person\_id,4); -- Scholarship  
 INSERT INTO person\_has\_ep\_set VALUES (new\_person\_id, 4); -- base for persons  
 INSERT INTO person\_has\_ep\_set VALUES (new\_person\_id, 1); -- base for students  
END;  
$$ LANGUAGE** plpgsql;  
**CREATE OR REPLACE FUNCTION** *add\_new\_employee* (first\_name **VARCHAR**(50),middle\_name **VARCHAR**(50),family\_name **VARCHAR**(200),  
 gender **CHAR**, dob **TIMESTAMP**, salary **INTEGER**,  
 **role VARCHAR**(50), image\_path **VARCHAR**(200)) **RETURNS VOID AS $$  
DECLARE  
 new\_person\_id INTEGER;  
 new\_doc\_id INTEGER;  
 role\_id INTEGER;  
BEGIN  
 SELECT employee\_role.employee\_role\_id INTO role\_id FROM employee\_role WHERE role\_name LIKE role;  
 SELECT** *nextval***('person\_person\_id\_seq'::REGCLASS) INTO new\_person\_id;  
 BEGIN  
 SET CONSTRAINTS ALL DEFERRED;  
 INSERT INTO document VALUES (***nextval***('document\_document\_id\_seq'::REGCLASS), new\_person\_id,** *current\_timestamp* **+ INTERVAL '1 year', 1, image\_path)  
 RETURNING document\_id INTO new\_doc\_id;  
 INSERT INTO person VALUES (new\_person\_id, first\_name, middle\_name, family\_name, dob, gender, new\_doc\_id);  
 END;  
 INSERT INTO employee VALUES (new\_person\_id, salary, role\_id);  
 INSERT INTO account VALUES (***nextval***('account\_account\_id\_seq'::REGCLASS), new\_person\_id, 1); -- rental\_fee  
 INSERT INTO account VALUES (***nextval***('account\_account\_id\_seq'::REGCLASS), new\_person\_id, 3); -- Salary  
 INSERT INTO person\_has\_ep\_set VALUES (new\_person\_id, 4); -- base for persons  
 IF (role\_id = 4 OR role\_id = 5)  
 THEN  
 INSERT INTO person\_has\_ep\_set VALUES (new\_person\_id, 6); -- canteen staff  
 ELSEIF (role\_id = 3 OR role\_id = 6)  
 THEN  
 INSERT INTO person\_has\_ep\_set VALUES (new\_person\_id, 2); -- administrator  
 ELSEIF (role\_id = 2)  
 THEN  
 INSERT INTO person\_has\_ep\_set VALUES (new\_person\_id, 7); -- cleaning  
 END IF;  
END;  
$$ LANGUAGE** plpgsql;  
*-- -----------------------  
-- add guest to person  
-- -----------------------***DROP FUNCTION IF EXISTS** *add\_guest\_to\_person*(**integer**,**integer**);  
**CREATE OR REPLACE FUNCTION public**.*add\_guest\_to\_person*(\_guest\_person\_id **integer**, \_person\_id **integer**)  
 **RETURNS BOOL  
 LANGUAGE** plpgsql  
**AS $function$  
BEGIN  
 IF (NOT** *exists***(SELECT** *\** **FROM guest WHERE guest.person\_id = \_guest\_person\_id))  
 THEN  
 RAISE NOTICE 'No guest person';  
 RETURN FALSE;  
 END IF;  
 IF (not** *exists***(SELECT person\_id from student WHERE student.person\_id = \_person\_id  
 UNION  
 SELECT person\_id FROM employee WHERE employee.person\_id = \_person\_id))  
 THEN  
 RAISE NOTICE 'Host should be students or employee';  
 RETURN FALSE;  
 END IF;  
 IF (***exists***(SELECT** *\** **FROM guest\_to\_person  
 WHERE guest\_to\_person.guest\_person\_id = \_guest\_person\_id  
 AND guest\_to\_person.date\_time\_end IS NULL))  
 THEN  
 RAISE NOTICE 'Guest should leave before come in again';  
 RETURN FALSE;  
 END IF;  
 INSERT INTO guest\_to\_person (guest\_person\_id, person\_id, date\_time\_start, date\_time\_end ) VALUES (\_guest\_person\_id,\_person\_id,***now***(),NULL);  
 RETURN TRUE;  
END;  
$function$  
  
CREATE OR REPLACE FUNCTION public**.*create\_guest\_person*(\_first\_name **character varying**, \_middle\_name **character varying**, \_family\_name **character varying**, \_gender **character**, \_date\_of\_birth **timestamp without time zone**, \_update\_to\_date **timestamp without time zone**, \_document\_type\_id **integer**, \_image\_path **character varying**, \_host\_person\_id **integer**)  
 **RETURNS boolean  
 LANGUAGE** plpgsql  
**AS $function$  
DECLARE  
 new\_person\_id INTEGER;  
 new\_guest\_id INTEGER;  
 new\_doc\_id INTEGER;  
BEGIN  
 SELECT person.person\_id FROM person WHERE person.date\_of\_birth = \_date\_of\_birth  
 AND person.family\_name = \_family\_name  
 AND person.first\_name = \_first\_name  
 INTO new\_person\_id;  
 IF (new\_person\_id IS NOT NULL )  
 THEN  
 SELECT person\_id FROM guest WHERE person\_id = new\_person\_id INTO new\_guest\_id;  
 IF (new\_guest\_id IS NULL )  
 THEN  
 RAISE EXCEPTION 'Person is exist but not as guest'  
 USING HINT = 'You can not create a guest who already working here';  
 ELSE  
 IF (SELECT add\_guest\_to\_person(new\_guest\_id,\_host\_person\_id) = TRUE)  
 THEN  
 RETURN TRUE;  
 ELSE  
 RETURN FALSE;  
 END IF;  
 END IF;  
 ELSE  
 SELECT** *nextval***('person\_person\_id\_seq'::REGCLASS) INTO new\_person\_id;  
 SELECT** *nextval***('document\_document\_id\_seq'::REGCLASS) INTO new\_doc\_id;  
 BEGIN  
 SET CONSTRAINTS ALL DEFERRED;  
 INSERT INTO document VALUES (new\_doc\_id, new\_person\_id, \_update\_to\_date, \_document\_type\_id, \_image\_path);  
 INSERT INTO person VALUES (new\_person\_id, \_first\_name, \_middle\_name, \_family\_name, \_date\_of\_birth, \_gender, new\_doc\_id);  
 INSERT INTO guest VALUES (new\_person\_id);  
 END;  
 IF (SELECT add\_guest\_to\_person(new\_person\_id,\_host\_person\_id) = TRUE)  
 THEN  
 RETURN TRUE;  
 ELSE  
 BEGIN  
 SET CONSTRAINTS ALL DEFERRED;  
 DELETE FROM person WHERE person\_id = new\_person\_id;  
 DELETE FROM document WHERE document\_id = new\_doc\_id;  
 DELETE FROM guest WHERE person\_id = new\_person\_id;  
 END;  
 RETURN FALSE;  
 END IF;  
 END IF;  
END;  
$function$  
DROP FUNCTION IF EXISTS** *guest\_left\_from\_person*(**integer**,**integer**);  
**CREATE OR REPLACE FUNCTION public**.*guest\_left\_from\_person*(\_guest\_person\_id **integer**, \_person\_id **integer**)  
 **RETURNS boolean  
 LANGUAGE** plpgsql  
**AS $function$  
BEGIN  
 IF (NOT** *exists***(SELECT** *\** **FROM guest WHERE guest.person\_id = \_guest\_person\_id))  
 THEN  
 RAISE NOTICE 'No guest person';  
 RETURN FALSE;  
 END IF;  
 IF (not** *exists***(SELECT person\_id from student WHERE student.person\_id = \_person\_id  
 UNION  
 SELECT person\_id FROM employee WHERE employee.person\_id = \_person\_id))  
 THEN  
 RAISE NOTICE 'Host should be students or employee';  
 RETURN FALSE;  
 END IF;  
 IF (NOT** *exists***(SELECT** *\** **FROM guest\_to\_person  
 WHERE guest\_to\_person.guest\_person\_id = \_guest\_person\_id  
 AND guest\_to\_person.person\_id = \_person\_id  
 AND guest\_to\_person.date\_time\_end IS NULL))  
 THEN  
 RAISE NOTICE 'Not registered';  
 RETURN FALSE;  
 ELSE  
 UPDATE guest\_to\_person SET date\_time\_end =** *now***()  
 WHERE guest\_person\_id = \_guest\_person\_id  
 AND person\_id = \_person\_id  
 AND date\_time\_end IS NULL;  
 RETURN TRUE;  
 END IF;  
END;  
$function$**

# Database Creation queries

*-- Sequence structure for account\_account\_id\_seq***DROP SEQUENCE IF EXISTS** "public"."account\_account\_id\_seq" **CASCADE**;  
**CREATE SEQUENCE** "public"."account\_account\_id\_seq"  
 **INCREMENT** 1  
 **MINVALUE** 1  
 **MAXVALUE** 9223372036854775807  
 **START** 64  
 **CACHE** 1;  
**SELECT** *setval*(**'"public"."account\_account\_id\_seq"'**, 64, **true**);  
*-- Sequence structure for apartment\_type\_apartment\_type\_id\_seq***DROP SEQUENCE IF EXISTS** "public"."apartment\_type\_apartment\_type\_id\_seq" **CASCADE**;  
**CREATE SEQUENCE** "public"."apartment\_type\_apartment\_type\_id\_seq"  
 **INCREMENT** 1  
 **MINVALUE** 1  
 **MAXVALUE** 9223372036854775807  
 **START** 4  
 **CACHE** 1;  
*-- Sequence structure for building\_building\_id\_seq***DROP SEQUENCE IF EXISTS** "public"."building\_building\_id\_seq" **CASCADE**;  
**CREATE SEQUENCE** "public"."building\_building\_id\_seq"  
 **INCREMENT** 1  
 **MINVALUE** 1  
 **MAXVALUE** 9223372036854775807  
 **START** 6  
 **CACHE** 1;  
*-- Sequence structure for document\_document\_id\_seq***DROP SEQUENCE IF EXISTS** "public"."document\_document\_id\_seq" **CASCADE**;  
**CREATE SEQUENCE** "public"."document\_document\_id\_seq"  
 **INCREMENT** 1  
 **MINVALUE** 1  
 **MAXVALUE** 9223372036854775807  
 **START** 27  
 **CACHE** 1;  
*-- Sequence structure for document\_type\_document\_type\_id\_seq***DROP SEQUENCE IF EXISTS** "public"."document\_type\_document\_type\_id\_seq" **CASCADE**;  
**CREATE SEQUENCE** "public"."document\_type\_document\_type\_id\_seq"  
 **INCREMENT** 1  
 **MINVALUE** 1  
 **MAXVALUE** 9223372036854775807  
 **START** 10  
 **CACHE** 1;  
**SELECT** *setval*(**'"public"."document\_type\_document\_type\_id\_seq"'**, 10, **true**);  
*-- Sequence structure for employee\_role\_employee\_role\_id\_seq***DROP SEQUENCE IF EXISTS** "public"."employee\_role\_employee\_role\_id\_seq" **CASCADE**;  
**CREATE SEQUENCE** "public"."employee\_role\_employee\_role\_id\_seq"  
 **INCREMENT** 1  
 **MINVALUE** 1  
 **MAXVALUE** 9223372036854775807  
 **START** 7  
 **CACHE** 1;  
*-- Sequence structure for entry\_check\_machine\_ecm\_id\_seq***DROP SEQUENCE IF EXISTS** "public"."entry\_check\_machine\_ecm\_id\_seq" **CASCADE**;  
**CREATE SEQUENCE** "public"."entry\_check\_machine\_ecm\_id\_seq"  
 **INCREMENT** 1  
 **MINVALUE** 1  
 **MAXVALUE** 9223372036854775807  
 **START** 84  
 **CACHE** 1;  
**SELECT** *setval*(**'"public"."entry\_check\_machine\_ecm\_id\_seq"'**, 84, **true**);  
*-- Sequence structure for entry\_permission\_entry\_permission\_id\_seq***DROP SEQUENCE IF EXISTS** "public"."entry\_permission\_entry\_permission\_id\_seq" **CASCADE**;  
**CREATE SEQUENCE** "public"."entry\_permission\_entry\_permission\_id\_seq"  
 **INCREMENT** 1  
 **MINVALUE** 1  
 **MAXVALUE** 9223372036854775807  
 **START** 23  
 **CACHE** 1;  
**SELECT** *setval*(**'"public"."entry\_permission\_entry\_permission\_id\_seq"'**, 23, **true**);  
*-- Sequence structure for entry\_permission\_set\_ep\_set\_id\_seq***DROP SEQUENCE IF EXISTS** "public"."entry\_permission\_set\_ep\_set\_id\_seq" **CASCADE**;  
**CREATE SEQUENCE** "public"."entry\_permission\_set\_ep\_set\_id\_seq"  
 **INCREMENT** 1  
 **MINVALUE** 1  
 **MAXVALUE** 9223372036854775807  
 **START** 8  
 **CACHE** 1;  
*-- Sequence structure for person\_person\_id\_seq***DROP SEQUENCE IF EXISTS** "public"."person\_person\_id\_seq" **CASCADE**;  
**CREATE SEQUENCE** "public"."person\_person\_id\_seq"  
 **INCREMENT** 1  
 **MINVALUE** 1  
 **MAXVALUE** 9223372036854775807  
 **START** 26  
 **CACHE** 1;  
*-- Sequence structure for transaction\_transaction\_id\_seq***DROP SEQUENCE IF EXISTS** "public"."transaction\_transaction\_id\_seq" **CASCADE**;  
**CREATE SEQUENCE** "public"."transaction\_transaction\_id\_seq"  
 **INCREMENT** 1  
 **MINVALUE** 1  
 **MAXVALUE** 9223372036854775807  
 **START** 1175  
 **CACHE** 1;  
*-- Sequence structure for transaction\_type\_transaction\_type\_id\_seq***DROP SEQUENCE IF EXISTS** "public"."transaction\_type\_transaction\_type\_id\_seq" **CASCADE**;  
**CREATE SEQUENCE** "public"."transaction\_type\_transaction\_type\_id\_seq"  
 **INCREMENT** 1  
 **MINVALUE** 1  
 **MAXVALUE** 9223372036854775807  
 **START** 5  
 **CACHE** 1;  
*-- Table structure for account***DROP TABLE IF EXISTS** "public"."account" **CASCADE**;  
**CREATE TABLE** "public"."account" (  
**"account\_id" int4 DEFAULT** *nextval*(**'account\_account\_id\_seq'**::**regclass**) **NOT NULL**,  
**"person\_id" int4 NOT NULL**,  
**"account\_type\_id" int4 NOT NULL**,  
**"balance" numeric**(255));  
*-- Table structure for account\_type***DROP TABLE IF EXISTS** "public"."account\_type" **CASCADE**;  
**CREATE TABLE** "public"."account\_type" (  
**"account\_type\_id" int4 DEFAULT** *nextval*(**'transaction\_type\_transaction\_type\_id\_seq'**::**regclass**) **NOT NULL**,  
**"type\_name" varchar**(50) **COLLATE** "default" **NOT NULL**);  
*-- Table structure for apartment***DROP TABLE IF EXISTS** "public"."apartment" **CASCADE**;  
**CREATE TABLE** "public"."apartment" (  
**"building\_id" int4 NOT NULL**,  
**"apartment\_number" int4 NOT NULL**,  
**"apartment\_type\_id" int4 NOT NULL**);  
*-- Table structure for apartment\_type***DROP TABLE IF EXISTS** "public"."apartment\_type" **CASCADE**;  
**CREATE TABLE** "public"."apartment\_type" (  
**"apartment\_type\_id" int4 DEFAULT** *nextval*(**'apartment\_type\_apartment\_type\_id\_seq'**::**regclass**) **NOT NULL**,  
**"beds" int4**,  
**"space" numeric**(4,2),  
**"rooms" int4**,  
**"monthly\_cost" numeric**(8,2),  
**"type\_name" varchar**(50) **COLLATE** "default");  
*-- Table structure for building***DROP TABLE IF EXISTS** "public"."building" **CASCADE**;  
**CREATE TABLE** "public"."building" (  
**"building\_id" int4 DEFAULT** *nextval*(**'building\_building\_id\_seq'**::**regclass**) **NOT NULL**,  
**"city" varchar**(50) **COLLATE** "default" **NOT NULL**,  
**"street" varchar**(100) **COLLATE** "default" **NOT NULL**,  
**"number\_in\_street" varchar**(6) **COLLATE** "default" **NOT NULL**,  
**"zip\_code" varchar**(30) **COLLATE** "default",  
**"type\_name" varchar**(50) **COLLATE** "default");  
*-- Table structure for document***DROP TABLE IF EXISTS** "public"."document" **CASCADE**;  
**CREATE TABLE** "public"."document" (  
**"document\_id" int4 DEFAULT** *nextval*(**'document\_document\_id\_seq'**::**regclass**) **NOT NULL**,  
**"person\_id" int4 NOT NULL**,  
**"update\_to\_date" timestamp**(6) **NOT NULL**,  
**"document\_type\_id" int4 NOT NULL**,  
**"image\_path" varchar**(255) **COLLATE** "default" **NOT NULL**);  
*-- Table structure for document\_type***DROP TABLE IF EXISTS** "public"."document\_type" **CASCADE**;  
**CREATE TABLE** "public"."document\_type" (  
**"document\_type\_id" int4 DEFAULT** *nextval*(**'document\_type\_document\_type\_id\_seq'**::**regclass**) **NOT NULL**,  
**"type\_name" varchar**(50) **COLLATE** "default" **NOT NULL**,  
**"identity\_lvl" varchar**(255) **COLLATE** "default" **NOT NULL**);  
*-- Table structure for employee***DROP TABLE IF EXISTS** "public"."employee" **CASCADE**;  
**CREATE TABLE** "public"."employee" (  
**"person\_id" int4 NOT NULL**,  
**"salary" numeric**(8,2) **NOT NULL**,  
**"employee\_role\_id" int4 NOT NULL**);  
*-- Table structure for employee\_role***DROP TABLE IF EXISTS** "public"."employee\_role" **CASCADE**;  
**CREATE TABLE** "public"."employee\_role" (  
**"employee\_role\_id" int4 DEFAULT** *nextval*(**'employee\_role\_employee\_role\_id\_seq'**::**regclass**) **NOT NULL**,  
**"role\_name" varchar**(50) **COLLATE** "default" **NOT NULL**,  
**"description" varchar**(255) **COLLATE** "default");  
*-- Table structure for entry\_check\_machine***DROP TABLE IF EXISTS** "public"."entry\_check\_machine" **CASCADE**;  
**CREATE TABLE** "public"."entry\_check\_machine" (  
**"ecm\_id" int4 DEFAULT** *nextval*(**'entry\_check\_machine\_ecm\_id\_seq'**::**regclass**) **NOT NULL**,  
**"building\_id" int4 NOT NULL**,  
**"entry\_permission\_id" int4 NOT NULL**,  
**"constraint\_group\_id" int4**);  
*-- Table structure for entry\_permission***DROP TABLE IF EXISTS** "public"."entry\_permission" **CASCADE**;  
**CREATE TABLE** "public"."entry\_permission" (  
**"entry\_permission\_id" int4 DEFAULT** *nextval*(**'entry\_permission\_entry\_permission\_id\_seq'**::**regclass**) **NOT NULL**,  
**"permission\_name" varchar**(100) **COLLATE** "default" **NOT NULL**);  
*-- Table structure for entry\_permission\_set***DROP TABLE IF EXISTS** "public"."entry\_permission\_set" **CASCADE**;  
**CREATE TABLE** "public"."entry\_permission\_set" (  
**"ep\_set\_id" int4 DEFAULT** *nextval*(**'entry\_permission\_set\_ep\_set\_id\_seq'**::**regclass**) **NOT NULL**,  
**"set\_name" varchar**(50) **COLLATE** "default" **NOT NULL**);  
*-- Table structure for ep\_set\_has\_ep***DROP TABLE IF EXISTS** "public"."ep\_set\_has\_ep" **CASCADE**;  
**CREATE TABLE** "public"."ep\_set\_has\_ep" (  
**"entry\_permission\_id" int4 NOT NULL**,  
**"ep\_set\_id" int4 NOT NULL**);

*-- Table structure for guest***DROP TABLE IF EXISTS** "public"."guest" **CASCADE**;  
**CREATE TABLE** "public"."guest" (  
**"person\_id" int4 NOT NULL**);  
*-- Table structure for guest\_to\_person***DROP TABLE IF EXISTS** "public"."guest\_to\_person" **CASCADE**;  
**CREATE TABLE** "public"."guest\_to\_person" (  
**"person\_id" int4 NOT NULL**,  
**"date\_time\_start" timestamp**(6) **NOT NULL**,  
**"date\_time\_end" timestamp**(6),  
**"guest\_person\_id" int4 NOT NULL**);  
*-- Table structure for in\_out***DROP TABLE IF EXISTS** "public"."in\_out" **CASCADE**;  
**CREATE TABLE** "public"."in\_out" (  
**"person\_id" int4 NOT NULL**,  
**"ecm\_id" int4 NOT NULL**,  
**"direction" char**(1) **COLLATE** "default" **NOT NULL**,  
**"date\_time" timestamp**(6) **NOT NULL**,  
**"accepted" bool NOT NULL**);  
*-- Table structure for lives\_in***DROP TABLE IF EXISTS** "public"."lives\_in" **CASCADE**;  
**CREATE TABLE** "public"."lives\_in" (  
**"person\_id" int4 NOT NULL**,  
**"apartment\_number" int4 NOT NULL**,  
**"building\_id" int4 NOT NULL**,  
**"date\_time\_start" timestamp**(6) **NOT NULL**);  
*-- Table structure for person***DROP TABLE IF EXISTS** "public"."person" **CASCADE**;  
**CREATE TABLE** "public"."person" (  
**"person\_id" int4 DEFAULT** *nextval*(**'person\_person\_id\_seq'**::**regclass**) **NOT NULL**,  
**"first\_name" varchar**(50) **COLLATE** "default" **NOT NULL**,  
**"middle\_name" varchar**(50) **COLLATE** "default",  
**"family\_name" varchar**(50) **COLLATE** "default" **NOT NULL**,  
**"date\_of\_birth" timestamp**(6) **NOT NULL**,  
**"gender" char**(1) **COLLATE** "default" **NOT NULL**,  
**"main\_document\_id" int4 NOT NULL**);  
*-- Table structure for person\_has\_ep\_set***DROP TABLE IF EXISTS** "public"."person\_has\_ep\_set" **CASCADE**;  
**CREATE TABLE** "public"."person\_has\_ep\_set" (  
**"person\_id" int4 NOT NULL**,  
**"ep\_set\_id" int4 NOT NULL**);  
*-- Table structure for student***DROP TABLE IF EXISTS** "public"."student" **CASCADE**;  
**CREATE TABLE** "public"."student" (  
**"person\_id" int4 NOT NULL**,  
**"scholarship" numeric**(8,2) **NOT NULL**);  
*-- Table structure for transaction***DROP TABLE IF EXISTS** "public"."transaction" **CASCADE**;  
**CREATE TABLE** "public"."transaction" (  
**"transaction\_id" int4 DEFAULT** *nextval*(**'transaction\_transaction\_id\_seq'**::**regclass**) **NOT NULL**,  
**"amount" numeric**(8,2) **NOT NULL**,  
**"account\_id" int4 NOT NULL**,  
**"date\_time" timestamp**(6) **NOT NULL**,  
**"description" varchar**(255) **COLLATE** "default");  
**ALTER SEQUENCE** "public"."account\_account\_id\_seq" **OWNED BY** "account".**"account\_id"**;  
**ALTER SEQUENCE** "public"."apartment\_type\_apartment\_type\_id\_seq" **OWNED BY** "apartment\_type".**"apartment\_type\_id"**;  
**ALTER SEQUENCE** "public"."building\_building\_id\_seq" **OWNED BY** "building".**"building\_id"**;  
**ALTER SEQUENCE** "public"."document\_document\_id\_seq" **OWNED BY** "document".**"document\_id"**;  
**ALTER SEQUENCE** "public"."document\_type\_document\_type\_id\_seq" **OWNED BY** "document\_type".**"document\_type\_id"**;  
**ALTER SEQUENCE** "public"."employee\_role\_employee\_role\_id\_seq" **OWNED BY** "employee\_role".**"employee\_role\_id"**;  
**ALTER SEQUENCE** "public"."entry\_check\_machine\_ecm\_id\_seq" **OWNED BY** "entry\_check\_machine".**"ecm\_id"**;  
**ALTER SEQUENCE** "public"."entry\_permission\_entry\_permission\_id\_seq" **OWNED BY** "entry\_permission".**"entry\_permission\_id"**;  
**ALTER SEQUENCE** "public"."entry\_permission\_set\_ep\_set\_id\_seq" **OWNED BY** "entry\_permission\_set".**"ep\_set\_id"**;  
**ALTER SEQUENCE** "public"."person\_person\_id\_seq" **OWNED BY** "person".**"person\_id"**;  
**ALTER SEQUENCE** "public"."transaction\_transaction\_id\_seq" **OWNED BY** "transaction".**"transaction\_id"**;  
**ALTER SEQUENCE** "public"."transaction\_type\_transaction\_type\_id\_seq" **OWNED BY** "account\_type".**"account\_type\_id"**;  
*-- Indexes structure for table account***CREATE UNIQUE INDEX** "account\_person\_id\_account\_type\_id\_idx" **ON** "public"."account" **USING btree** (**"person\_id"**, **"account\_type\_id"**);  
*-- Uniques structure for table account***ALTER TABLE** "public"."account" **ADD UNIQUE** (**"person\_id"**, **"account\_type\_id"**);  
*-- Primary Key structure for table account***ALTER TABLE** "public"."account" **ADD PRIMARY KEY** (**"account\_id"**);  
*-- Uniques structure for table account\_type***ALTER TABLE** "public"."account\_type" **ADD UNIQUE** (**"type\_name"**) **DEFERRABLE**;  
*-- Primary Key structure for table account\_type***ALTER TABLE** "public"."account\_type" **ADD PRIMARY KEY** (**"account\_type\_id"**);  
*-- Indexes structure for table apartment***CREATE UNIQUE INDEX** "apartment\_building\_id\_apartment\_number\_idx" **ON** "public"."apartment" **USING btree** (**"building\_id"**, **"apartment\_number"**);  
*-- Uniques structure for table apartment***ALTER TABLE** "public"."apartment" **ADD UNIQUE** (**"building\_id"**, **"apartment\_number"**);  
*-- Primary Key structure for table apartment***ALTER TABLE** "public"."apartment" **ADD PRIMARY KEY** (**"building\_id"**, **"apartment\_number"**);  
*-- Uniques structure for table apartment\_type***ALTER TABLE** "public"."apartment\_type" **ADD UNIQUE** (**"type\_name"**) **DEFERRABLE**;  
*-- Primary Key structure for table apartment\_type***ALTER TABLE** "public"."apartment\_type" **ADD PRIMARY KEY** (**"apartment\_type\_id"**);  
*-- Uniques structure for table building***ALTER TABLE** "public"."building" **ADD UNIQUE** (**"city"**, **"street"**, **"number\_in\_street"**);  
*-- Primary Key structure for table building***ALTER TABLE** "public"."building" **ADD PRIMARY KEY** (**"building\_id"**);  
*-- Indexes structure for table document***CREATE UNIQUE INDEX** "document\_person\_id\_document\_type\_id\_idx" **ON** "public"."document" **USING btree** (**"person\_id"**, **"document\_type\_id"**);  
**CREATE INDEX** "document\_person\_id\_clust\_index" **ON** "public"."document" **USING btree** (**"person\_id"**);  
**ALTER TABLE** "public"."document" **CLUSTER ON** "document\_person\_id\_clust\_index";  
*-- Primary Key structure for table document***ALTER TABLE** "public"."document" **ADD PRIMARY KEY** (**"document\_id"**);  
*-- Uniques structure for table document\_type***ALTER TABLE** "public"."document\_type" **ADD UNIQUE** (**"type\_name"**) **DEFERRABLE**;  
*-- Primary Key structure for table document\_type***ALTER TABLE** "public"."document\_type" **ADD PRIMARY KEY** (**"document\_type\_id"**);  
*-- Primary Key structure for table employee***ALTER TABLE** "public"."employee" **ADD PRIMARY KEY** (**"person\_id"**);  
*-- Uniques structure for table employee\_role***ALTER TABLE** "public"."employee\_role" **ADD UNIQUE** (**"role\_name"**) **DEFERRABLE**;  
*-- Primary Key structure for table employee\_role***ALTER TABLE** "public"."employee\_role" **ADD PRIMARY KEY** (**"employee\_role\_id"**);  
*-- Primary Key structure for table entry\_check\_machine***ALTER TABLE** "public"."entry\_check\_machine" **ADD PRIMARY KEY** (**"ecm\_id"**);  
*-- Uniques structure for table entry\_permission***ALTER TABLE** "public"."entry\_permission" **ADD UNIQUE** (**"permission\_name"**) **DEFERRABLE**;  
*-- Primary Key structure for table entry\_permission***ALTER TABLE** "public"."entry\_permission" **ADD PRIMARY KEY** (**"entry\_permission\_id"**);  
*-- Uniques structure for table entry\_permission\_set***ALTER TABLE** "public"."entry\_permission\_set" **ADD UNIQUE** (**"set\_name"**) **DEFERRABLE**;  
*-- Primary Key structure for table entry\_permission\_set***ALTER TABLE** "public"."entry\_permission\_set" **ADD PRIMARY KEY** (**"ep\_set\_id"**);  
*-- Primary Key structure for table ep\_set\_has\_ep***ALTER TABLE** "public"."ep\_set\_has\_ep" **ADD PRIMARY KEY** (**"entry\_permission\_id"**, **"ep\_set\_id"**);  
*-- Primary Key structure for table guest***ALTER TABLE** "public"."guest" **ADD PRIMARY KEY** (**"person\_id"**);  
*-- Primary Key structure for table guest\_to\_person***ALTER TABLE** "public"."guest\_to\_person" **ADD PRIMARY KEY** (**"person\_id"**, **"date\_time\_start"**, **"guest\_person\_id"**);  
*-- Primary Key structure for table in\_out***ALTER TABLE** "public"."in\_out" **ADD PRIMARY KEY** (**"date\_time"**, **"person\_id"**, **"ecm\_id"**);  
*-- Indexes structure for table lives\_in***CREATE INDEX** "apartment\_clustered\_index" **ON** "public"."lives\_in" **USING btree** (**"apartment\_number"**, **"building\_id"**);  
**ALTER TABLE** "public"."lives\_in" **CLUSTER ON** "apartment\_clustered\_index";  
*-- Primary Key structure for table lives\_in***ALTER TABLE** "public"."lives\_in" **ADD PRIMARY KEY** (**"person\_id"**);  
*-- Uniques structure for table person***ALTER TABLE** "public"."person" **ADD UNIQUE** (**"main\_document\_id"**) **DEFERRABLE**;  
**ALTER TABLE** "public"."person" **ADD UNIQUE** (**"first\_name"**, **"family\_name"**, **"date\_of\_birth"**, **"middle\_name"**);  
*-- Primary Key structure for table person***ALTER TABLE** "public"."person" **ADD PRIMARY KEY** (**"person\_id"**);  
*-- Primary Key structure for table person\_has\_ep\_set***ALTER TABLE** "public"."person\_has\_ep\_set" **ADD PRIMARY KEY** (**"person\_id"**, **"ep\_set\_id"**);  
*-- Primary Key structure for table student***ALTER TABLE** "public"."student" **ADD PRIMARY KEY** (**"person\_id"**);  
*-- Indexes structure for table transaction***CREATE INDEX** "transaction\_account\_id\_index" **ON** "public"."transaction" **USING btree** (**"account\_id"**);  
**ALTER TABLE** "public"."transaction" **CLUSTER ON** "transaction\_account\_id\_index";  
*-- Triggers structure for table transaction***CREATE OR REPLACE FUNCTION** *update\_account\_balance\_on\_insert*() **RETURNS TRIGGER AS $$  
DECLARE  
BEGIN  
 UPDATE account  
 SET balance = (SELECT** *sum***(amount) FROM transaction WHERE transaction.account\_id =** *new***.account\_id)  
 WHERE account.account\_id =** *new***.account\_id;  
 RETURN** *new***;  
END;  
$$ LANGUAGE** plpgsql;  
  
**CREATE OR REPLACE FUNCTION** *update\_account\_balance\_on\_delete*() **RETURNS TRIGGER AS $$  
DECLARE  
BEGIN  
 UPDATE account  
 SET balance = (SELECT** *sum***(amount) FROM transaction WHERE transaction.account\_id =** *old***.account\_id)  
 WHERE account.account\_id =** *old***.account\_id;  
 RETURN** *old***;  
END;  
$$ LANGUAGE** plpgsql;  
**DROP TRIGGER IF EXISTS** account\_balance\_update\_on\_insert **ON** "public"."transaction" **CASCADE**;  
**CREATE TRIGGER** "account\_balance\_update\_on\_insert" **AFTER INSERT ON** "public"."transaction"  
**FOR EACH ROW  
EXECUTE PROCEDURE** *"update\_account\_balance\_on\_insert"*();  
**DROP TRIGGER IF EXISTS** account\_balance\_on\_delete **ON** "public"."transaction" **CASCADE**;  
**CREATE TRIGGER** "account\_balance\_on\_delete" **AFTER DELETE ON** "public"."transaction"  
**FOR EACH ROW  
EXECUTE PROCEDURE** *"update\_account\_balance\_on\_delete"*();  
*-- Uniques structure for table transaction***ALTER TABLE** "public"."transaction" **ADD UNIQUE** (**"account\_id"**, **"date\_time"**);  
*-- Primary Key structure for table transaction***ALTER TABLE** "public"."transaction" **ADD PRIMARY KEY** (**"transaction\_id"**);  
*-- Foreign Key structure for table "public"."account"***ALTER TABLE** "public"."account" **ADD FOREIGN KEY** (**"person\_id"**) **REFERENCES** "public"."person" (**"person\_id"**) **ON DELETE RESTRICT ON UPDATE CASCADE DEFERRABLE**;  
**ALTER TABLE** "public"."account" **ADD FOREIGN KEY** (**"account\_type\_id"**) **REFERENCES** "public"."account\_type" (**"account\_type\_id"**) **ON DELETE RESTRICT ON UPDATE CASCADE**;  
*-- Foreign Key structure for table "public"."apartment"***ALTER TABLE** "public"."apartment" **ADD FOREIGN KEY** (**"building\_id"**) **REFERENCES** "public"."building" (**"building\_id"**) **ON DELETE RESTRICT ON UPDATE CASCADE DEFERRABLE**;  
**ALTER TABLE** "public"."apartment" **ADD FOREIGN KEY** (**"apartment\_type\_id"**) **REFERENCES** "public"."apartment\_type" (**"apartment\_type\_id"**) **ON DELETE RESTRICT ON UPDATE CASCADE DEFERRABLE**;  
*-- Foreign Key structure for table "public"."document"***ALTER TABLE** "public"."document" **ADD FOREIGN KEY** (**"document\_type\_id"**) **REFERENCES** "public"."document\_type" (**"document\_type\_id"**) **ON DELETE RESTRICT ON UPDATE CASCADE DEFERRABLE**;  
**ALTER TABLE** "public"."document" **ADD FOREIGN KEY** (**"person\_id"**) **REFERENCES** "public"."person" (**"person\_id"**) **ON DELETE CASCADE ON UPDATE CASCADE DEFERRABLE**;  
*-- Foreign Key structure for table "public"."employee"***ALTER TABLE** "public"."employee" **ADD FOREIGN KEY** (**"employee\_role\_id"**) **REFERENCES** "public"."employee\_role" (**"employee\_role\_id"**) **ON DELETE RESTRICT ON UPDATE CASCADE DEFERRABLE**;  
**ALTER TABLE** "public"."employee" **ADD FOREIGN KEY** (**"person\_id"**) **REFERENCES** "public"."person" (**"person\_id"**) **ON DELETE RESTRICT ON UPDATE CASCADE DEFERRABLE**;  
*-- Foreign Key structure for table "public"."entry\_check\_machine"***ALTER TABLE** "public"."entry\_check\_machine" **ADD FOREIGN KEY** (**"entry\_permission\_id"**) **REFERENCES** "public"."entry\_permission" (**"entry\_permission\_id"**) **ON DELETE RESTRICT ON UPDATE CASCADE DEFERRABLE**;  
**ALTER TABLE** "public"."entry\_check\_machine" **ADD FOREIGN KEY** (**"building\_id"**) **REFERENCES** "public"."building" (**"building\_id"**) **ON DELETE RESTRICT ON UPDATE CASCADE DEFERRABLE**;  
*-- Foreign Key structure for table "public"."ep\_set\_has\_ep"***ALTER TABLE** "public"."ep\_set\_has\_ep" **ADD FOREIGN KEY** (**"ep\_set\_id"**) **REFERENCES** "public"."entry\_permission\_set" (**"ep\_set\_id"**) **ON DELETE RESTRICT ON UPDATE CASCADE DEFERRABLE**;  
**ALTER TABLE** "public"."ep\_set\_has\_ep" **ADD FOREIGN KEY** (**"entry\_permission\_id"**) **REFERENCES** "public"."entry\_permission" (**"entry\_permission\_id"**) **ON DELETE RESTRICT ON UPDATE CASCADE DEFERRABLE**;  
*-- Foreign Key structure for table "public"."guest"***ALTER TABLE** "public"."guest" **ADD FOREIGN KEY** (**"person\_id"**) **REFERENCES** "public"."person" (**"person\_id"**) **ON DELETE RESTRICT ON UPDATE CASCADE DEFERRABLE**;  
*-- Foreign Key structure for table "public"."guest\_to\_person"***ALTER TABLE** "public"."guest\_to\_person" **ADD FOREIGN KEY** (**"guest\_person\_id"**) **REFERENCES** "public"."guest" (**"person\_id"**) **ON DELETE RESTRICT ON UPDATE CASCADE DEFERRABLE**;  
**ALTER TABLE** "public"."guest\_to\_person" **ADD FOREIGN KEY** (**"person\_id"**) **REFERENCES** "public"."person" (**"person\_id"**) **ON DELETE RESTRICT ON UPDATE CASCADE DEFERRABLE**;  
*-- Foreign Key structure for table "public"."in\_out"***ALTER TABLE** "public"."in\_out" **ADD FOREIGN KEY** (**"person\_id"**) **REFERENCES** "public"."person" (**"person\_id"**) **ON DELETE RESTRICT ON UPDATE CASCADE DEFERRABLE**;  
**ALTER TABLE** "public"."in\_out" **ADD FOREIGN KEY** (**"ecm\_id"**) **REFERENCES** "public"."entry\_check\_machine" (**"ecm\_id"**) **ON DELETE RESTRICT ON UPDATE CASCADE DEFERRABLE**;  
*-- Foreign Key structure for table "public"."lives\_in"***ALTER TABLE** "public"."lives\_in" **ADD FOREIGN KEY** (**"person\_id"**) **REFERENCES** "public"."person" (**"person\_id"**) **ON DELETE RESTRICT ON UPDATE CASCADE DEFERRABLE**;  
**ALTER TABLE** "public"."lives\_in" **ADD FOREIGN KEY** (**"building\_id"**, **"apartment\_number"**) **REFERENCES** "public"."apartment" (**"building\_id"**, **"apartment\_number"**) **ON DELETE RESTRICT ON UPDATE CASCADE DEFERRABLE**;  
*-- Foreign Key structure for table "public"."person"***ALTER TABLE** "public"."person" **ADD FOREIGN KEY** (**"main\_document\_id"**) **REFERENCES** "public"."document" (**"document\_id"**) **ON DELETE RESTRICT ON UPDATE CASCADE DEFERRABLE**;

*-- Foreign Key structure for table "public"."person\_has\_ep\_set"***ALTER TABLE** "public"."person\_has\_ep\_set" **ADD FOREIGN KEY** (**"ep\_set\_id"**) **REFERENCES** "public"."entry\_permission\_set" (**"ep\_set\_id"**) **ON DELETE RESTRICT ON UPDATE CASCADE DEFERRABLE**;  
**ALTER TABLE** "public"."person\_has\_ep\_set" **ADD FOREIGN KEY** (**"person\_id"**) **REFERENCES** "public"."person" (**"person\_id"**) **ON DELETE CASCADE ON UPDATE CASCADE DEFERRABLE**;

*-- Foreign Key structure for table "public"."student"***ALTER TABLE** "public"."student" **ADD FOREIGN KEY** (**"person\_id"**) **REFERENCES** "public"."person" (**"person\_id"**) **ON DELETE RESTRICT ON UPDATE CASCADE DEFERRABLE**;  
*-- Foreign Key structure for table "public"."transaction"***ALTER TABLE** "public"."transaction" **ADD FOREIGN KEY** (**"account\_id"**) **REFERENCES** "public"."account" (**"account\_id"**) **ON DELETE RESTRICT ON UPDATE CASCADE DEFERRABLE**;