# BD1 – projekt


## Paweł Dąbrowski, Maciej Łodziński


Celem niniejszego projektu było stworzenie bazy danych przedstawiającą sieć uczelnianych akademików wraz z prostą aplikacją umożliwiającą wywoływanie poleceń SQL napisanej w JAVIE.

Link do repozytorium: https://gitlab-stud.elka.pw.edu.pl/mlodzins/bd_project.git

Pliki projektowe intelij zawierające aplikaję**: branch master->simple_dorm_app**

Pliki zawierające skrypty sql: **branch master->DB_src->SCRIPTS**

## 1. Struktura bazy danych:

Baza danych miała odzwierciedlać uproszczony model sieci domów studenta.

**Każdy z akademików składał się z:**

- Pokojów – tablica ROOMS,
- Pracowników – tablica STAFF.

Dodatkowo każdy akademik ma swój adres – tablica ADDRESSES, każdy pracownik ma swoją pozycję – tablica POSITIONS.

**Do każdego z pokojów przypisane zostały:**

- Zestaw mebli – tablica FURNITURE,
- Mieszkańcy pokoju, studenci – tablica STUDENTS.

**Oprócz powyżej wymienionych, stworzona została tablica STUDENS_HISTORY przechowująca listę okresów meldunków studentów.**

**1.1 Tablica DORMS:**

- **DORM_ID – klucz główny,**
- **NAME - nazwa akademika,**
- **GENDER – precyzuje, czy dom studenta jest żeński, męski czy koedukacyjny,**
- **MAX_COUNT – ilość miejsc w akademiku,**
- **CURRENT_COUNT – ilość obecnie zajętych miejsc, zaimplementowane w celu szybszego sprawdzania, czy są wolne miejsca w akademiku,**
- **ADDRESS_ID – klucz obcy adresu, pod którym znajduje się akademik (zakładamy możliwość istnienia wielu w jednym budynku)**

**1.2 Tablica ROOMS:**

- **ROOM_ID – klucz główny pokoju,**
- **ROOM_NUM – numer pokoju,**
- **GENDER – analogicznie do DORM.GENDER,**
- **MAX_COUNT – analogicznie do DORM.MAX_COUNT,**
- **CURRENT_COUNT – analogicznie do DORM.CURRENT_COUNT,**
- **DORM_ID – klucz obcy akademika, w którym znajduje się pokój.**

**1.3 Tablica STUDENTS:**

- **INDEX_NUMBER – klucz główny, oznacza nr indeksu,**
- **NAME – imię studenta,**
- **SURNAME – nazwisko studenta,**
- **ROOM_ID – klucz obcy pokoju studenta,**
- **GENDER – płeć studenta.**

**1.4 Tablica STAFF:**

- **STAFF_ID – klucz główny pracownika,**
- **NAME – imię pracownika,**
- **SURNAME – nazwisko pracownika,**
- **SALARY – pensja pracownika,**
- **DORM_ID – klucz obcy akademika, w którym pracuje,**
- **POSITION_ID – klucz obcy stanowiska pracownika.**

**1.5 Tablica FURNITURE:**

- **FURNITURE_ID – klucz główny,**
- **NAME – nazwa – opis mebla,**
- **CONDITION – stan mebla (w skali 0-10),**
- **ROOM_ID – klucz obcy pokoju, w którym się znajduje.**

**1.6 Tablica POSITIONS:**

- **POSITION_ID – klucz główny,**
- **NAME – nazwa stanowiska,**
- **MIN_SALARY – minimalna płaca,**
- **MAX_SALARY – maksymalna płaca.**

**1.7 Tablica ADDRESSES:**

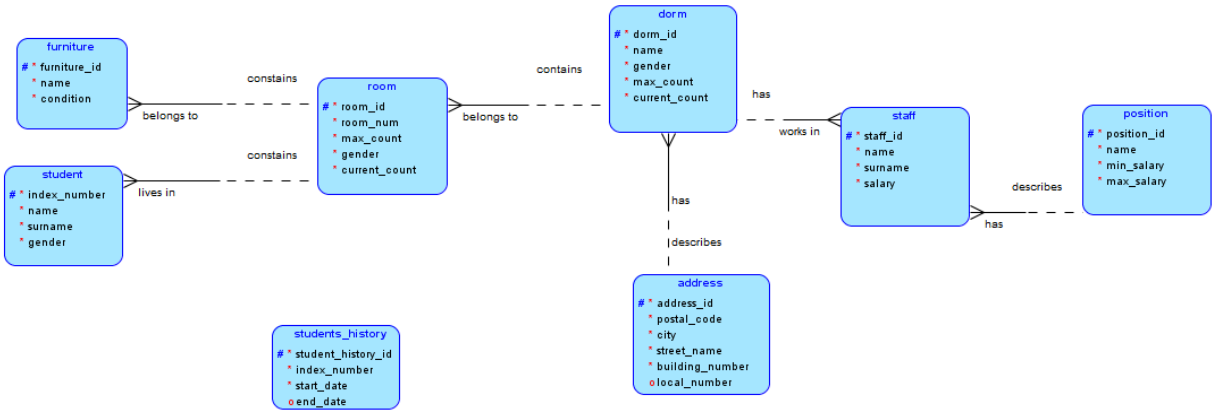
- **ADDRESS_ID – klucz główny,**
- **POSTAL_CODE – kod pocztowy,**
- **CITY – miasto,**
- **STREET_NAME – nazwa ulicy,**
- **BUILDING_NUMBER – numer budynku,**
- **LOCAL_NUMBER – numer lokalu (opcjonalny).**

**1.8 Tablica STUDENTS_HISTORY:**

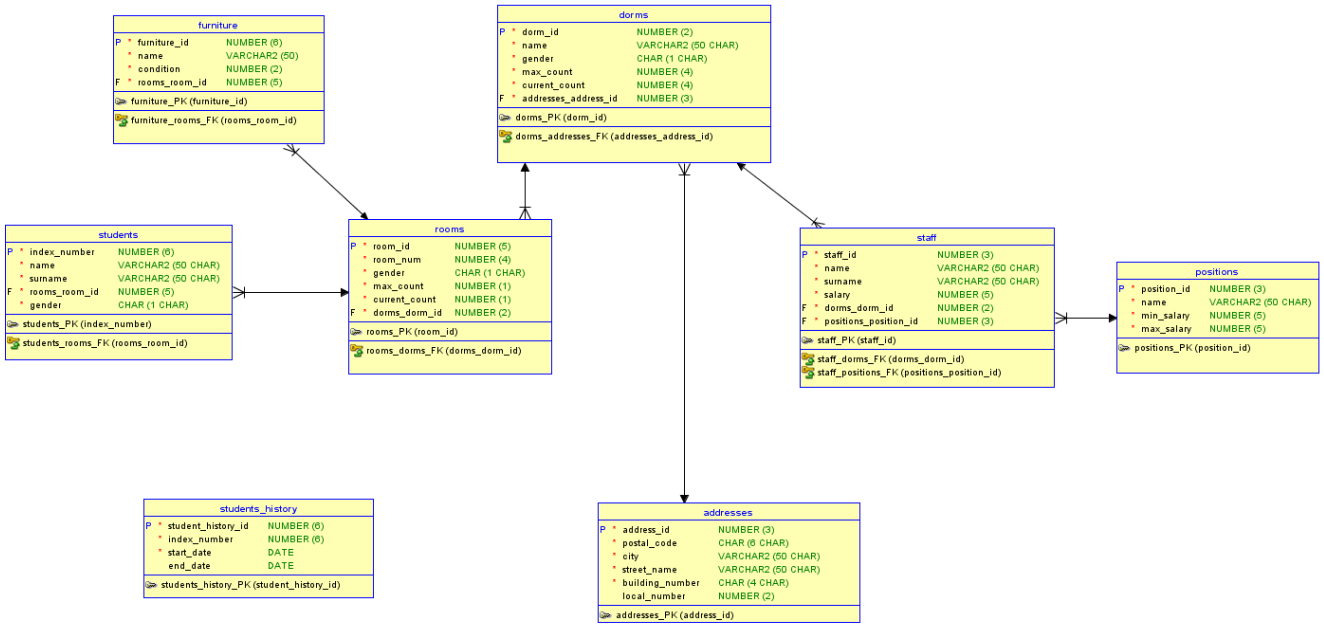
- **STUDENT_HISTORY_ID – klucz główny,**
- **INDEX_NUMBER – numer indeksu studenta,**
- **START_DATE – data kwaterunku,**
- **END_DATE – data zakończenia meldunku.**

Kolumny CURRENT_COUNT zarówno tablicy ROOMS jak i DORMS nie były konieczne do poprawnego działania systemu, natomiast uznaliśmy, że zaoszczędziłyby one zbędnych obliczeń: za każdym razem, gdy próbujemy zameldować studenta **nie trzeba sumować wszystkich mieszkańców zarówno akademika jak i pokoju**, aby się upewnić, czy nie zostaną przekroczone limity**.**

**Model logiczny:**




**Model relacyjny:**




# 2. Procedury wraz z kursorami

**Na potrzeby projektu zostały stworzone procedury:**

- **ADD_STUDENT() –** procedura dodaje studenta do wskazanego pokoju, uprzednio sprawdzając dostępność miejsc oraz jego przeznaczenie (GENDER). Inkrementuje odpowiednie liczniki CURRENT_COUNT.
- **DELETE_STUDENT() – procedura usuwa mieszkańca o wskazanym nr indeksu, sprawdza również czy takowy istnieje.**
- **ALOCATE_STUDENT_IN_DORM() –** procedura znajduje wolne miejsce we wskazanym akademiku, rejestruje w nim studenta przy użyciu procedury ADD_STUDENT(). Znajdowanie miejsca odbywa się przy użyciu kursora niejawnego: iteruje po wszystkich pokojach z wolnymi miejscami, wybiera z nich ten o największej ich ilości.

## 3. Funkcje

- **CALCULATE_AVG_POSITION_SALARY() –** oblicza średnie zarobki na stanowisku o podanej nazwie.
- **CALCULATE_FURNITURE_IN_DORM() –** zlicza meble zarejestrowane w akademiku o podanym ID.

## 4. Wyzwalacze

- **CHECK_SALARY_POSITIONS** – sprawdza, czy pensja (MIN lub MAX) jest dodatnia.
- **CHECK_SALARY_STAFF** – sprawdza, czy pensja pracownika mieści się w danym przedziale.
- **DELETE_STUDENTS** – aktualizuje wpis w tabeli STUDENTS_HISTORY, gdy student zostanie wymeldowany.
- **INSERT_STUDENTS_HISTORY** – dodaje wpis w tabeli STUDENTS_HISTORY, gdy student się zamelduje.

**Oprócz powyższych wyzwalaczy, każda tablica posiada dodatkowy wykorzystujący sekwencję generującą odpowiedni klucz główny.**

## 5. Aplikacja w JAVIE

Aplikacja w języku programowania JAVA umożliwia połączenie się z naszą bazą danych. Ponadto znajdują się w niej 3 funkcje, które opisane zostały poniżej:

• **getColumnCount()** – przyjmuje stringa oznaczającego nazwę tabeli i zwraca liczbę kolumn w owej tabeli. Jest to funkcja pomocnicza używana w pozostałych funkcjach,

• **showTable()** – przyjmuje stringa oznaczającego nazwę tabeli i wyświetla wszystkie znajdujące się w niej dane w przyjaznym dla oka formacie ,

• **showStudentsByRooms()** – wyświetla listę studentów wraz z ich pokojami oraz akademikami.