POLITECHNIKA WROCŁAWSKA
WYDZIAŁ ELEKTRONIKI

PROJEKT Z BAZ DANYCH

Dziennik szkolny

AUTOR/AUTORZY:
Alicja Wróbel 238894
E-mail:
238894@student.pwr.edu.pl
Wrocław, 2018 r.

1. Wstęp

1.1. Cel projektu

Celem projektu jest baza danych szkolnego dziennika elektronicznego, która będzie wykorzystywana w Szkole Podstawowej w Pcimiu. Ma na celu usprawnić zarządzanie dokumentacją szkolną oraz optymalizacje pracy z dużą ilością danych, co pozwala na zwiększenie efektywności pracowników placówki oraz odciążenie ich od zarządzania dokumentacją.

1.2. Zakres projektu

System ma umożliwiać organizację zajęć szkolnych oraz nadzorowanie postępów ucznia w nauce przez nauczycieli oraz rodziców lub opiekunów prawnych. Zakres projektu będzie obejmował obsługę aplikacji dla nauczycieli i wychowawców.

2. Analiza wymagań

2.1. Opis działania i schemat logiczny systemu

W systemie zapisane są informacje o uczniach, klasach, pracownikach, realizowanych przedmiotach. Istnieje możliwość wpisywania ocen oraz nieobecności (ewentualnych usprawiedliwień) uczniów. Baza zawiera listę przedmiotów nauczanych w szkole wraz z jej opisem. Wykorzystywane do spisu zajęć odbywających się w określonym czasie, dniu tygodnia, sali oraz informacja dla której konkretnej klasy mają się odbyć. Dzięki temu jest możliwy widok planu lekcji. Dany przedmiot może odbywać się więcej niż jeden w tym samym czasie lecz w innej sali ponieważ może być prowadzony przez innego nauczyciela. Każdy nauczyciel ma możliwość nauczania więcej niż jednego przedmiotu. System ma również na celu przechowywania informacji o klasach uczących się w szkole. Każda z nich ma przypisanego wychowawcę, rok rozpoczęcia edukacji klasy oraz planowanego ukończenia szkoły. Baza oprócz przechowywania danych osobowych uczniów, będzie zawierać informacje o przynależności ucznia do klasy oraz postępów w nauce. Każdy z uczniów posiada swoją listę ocen oraz nieobecności. Informacje o ocenach zawierają datę jej wystawienia, wagę oraz przedmiot, z którego została uzyskana. Dzięki temu można uczniowie mają indywidualne dzienniczki. Co więcej lista nieobecności ma datę wystawienia i możliwość jej usprawiedliwienia. System ma również na celu zautomatyzowanie czynności takich jak obliczanie średnich ocen dla konkretnych uczniów jak i dla konkretnych klas, automatyczne generowanie danych gotowych do wystawienia świadectwa szkolnego.

2.2. Wymagania funkcjonalne

W ZAKRESIE FUNKCJONALNOŚCI PRZEZNACZONEJ DLA UCZNIÓW

- Sprawdzenie ocen cząstkowych, śródrocznych i końcowych
- Sprawdzenie listy uwag/pochwał
- Sprawdzenie frekwencji
- Sprawdzenie aktualnego planu lekcji
- Wysyłanie/odbieranie komunikatów nauczyciela/wychowawcy

W ZAKRESIE FUNKCJONALNOŚCI PRZEZNACZONEJ DLA OPIEKUNÓW:

- Sprawdzenie ocen cząstkowych, śródrocznych i końcowych
- Sprawdzenie listy uwag/pochwał
- Sprawdzenie frekwencji
- Sprawdzenie aktualnego planu lekcji
- Wysyłanie/odbieranie komunikatów nauczyciela/wychowawcy
- Możliwość usprawiedliwienia nieobecności podopiecznego

W ZAKRESIE FUNKCJONALNOŚCI PRZEZNACZONEJ DLA NAUCZYCIELI:

- Możliwość wprowadzania ocen uczniów
- Możliwość wprowadzania uwag i pochwał
- Możliwość rejestrowania frekwencji we wszystkich klasach
- Możliwość wpisania tematu lekcji

W ZAKRESIE FUNKCJONALNOŚCI PRZEZNACZONEJ DLA WYCHOWAWCÓW:

- Dostęp do wszystkich danych swoich wychowanków z możliwością modyfikacji
- Możliwość wydrukowania wyników nauczania ucznia
- Możliwość generowania świadectw szkolnych
- Możliwość wykonania zestawień statystycznych dotyczących wyników nauczania i frekwencji
- Możliwość wysyłania komunikatów do uczniów/opiekunów zarówno pojedynczo jak i do całego oddziału
- Szczegółowa analiza wyników nauczania swoich uczniów (wszystkie dane i zestawienia)
- Możliwość wprowadzania ocen z zachowania

W ZAKRESIE FUNKCJONALNOŚCI PRZEZNACZONEJ DLA DYREKTORÓW I ADMINISTRATORÓW:

- Dostęp do wszystkich danych uczniów i możliwość ich modyfikacji
- Analiza uwag wpisanych wszystkim uczniom
- Analiza wyników nauczania, w szczególności ocen końcowych (jak stawiają stopnie nauczyciele, jak wyglądają stopnie z poszczególnych przedmiotów)
- Analiza frekwencji

2.3. Wymagania niefunkcjonalne

2.3.1. Wykorzystywane technologie i narzędzia

Jako system zarządzania bazą danych zostanie wykorzystany MySQL, natomiast oprogramowanie końcowe zostanie wykonane w technologii Java. Aplikacja zostanie wykonana w wersji desktopowej na system operacyjny Windows.

2.3.2. Wymagania dotyczące rozmiaru bazy danych

Rozmiar bazy będzie zależał od ilości uczniów w szkole, których przewidywanych jest 300. Do każdego dziecka przypisany jest opiekun. Różnych klas jest 15, przedmiotów 10, nauczycieli 20. W przeciągu semestru uczeń otrzyma ok. 80 ocen, co daje 24000 ocen w bazie dla wszystkich uczniów. W przeciągu tygodnia odbędzie się 525 lekcji w całej szkole.

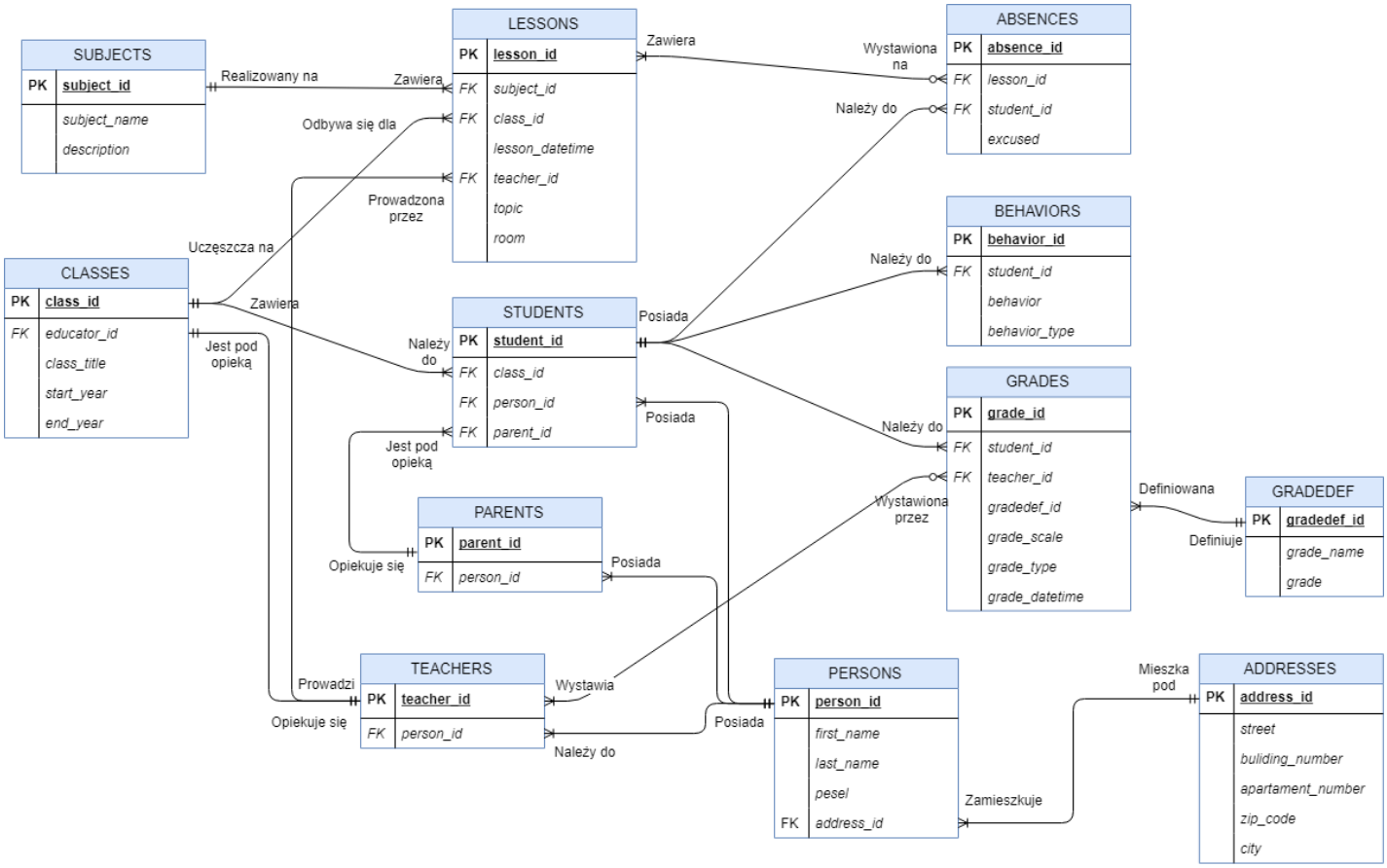
2.3.3. Wymagania dotyczące bezpieczeństwa systemu

Pracownicy jak i administrator systemu biorą odpowiedzialność za przechowywanie i bezpieczeństwo danych. Do aplikacji będzie wykorzystany proces uwierzytelniania przy logowaniu oraz konieczność podania hasła. Przy wpisywaniu ocen końcowych zostaną wykorzystane tokeny pozwalające zatwierdzić wpisaną ocenę.

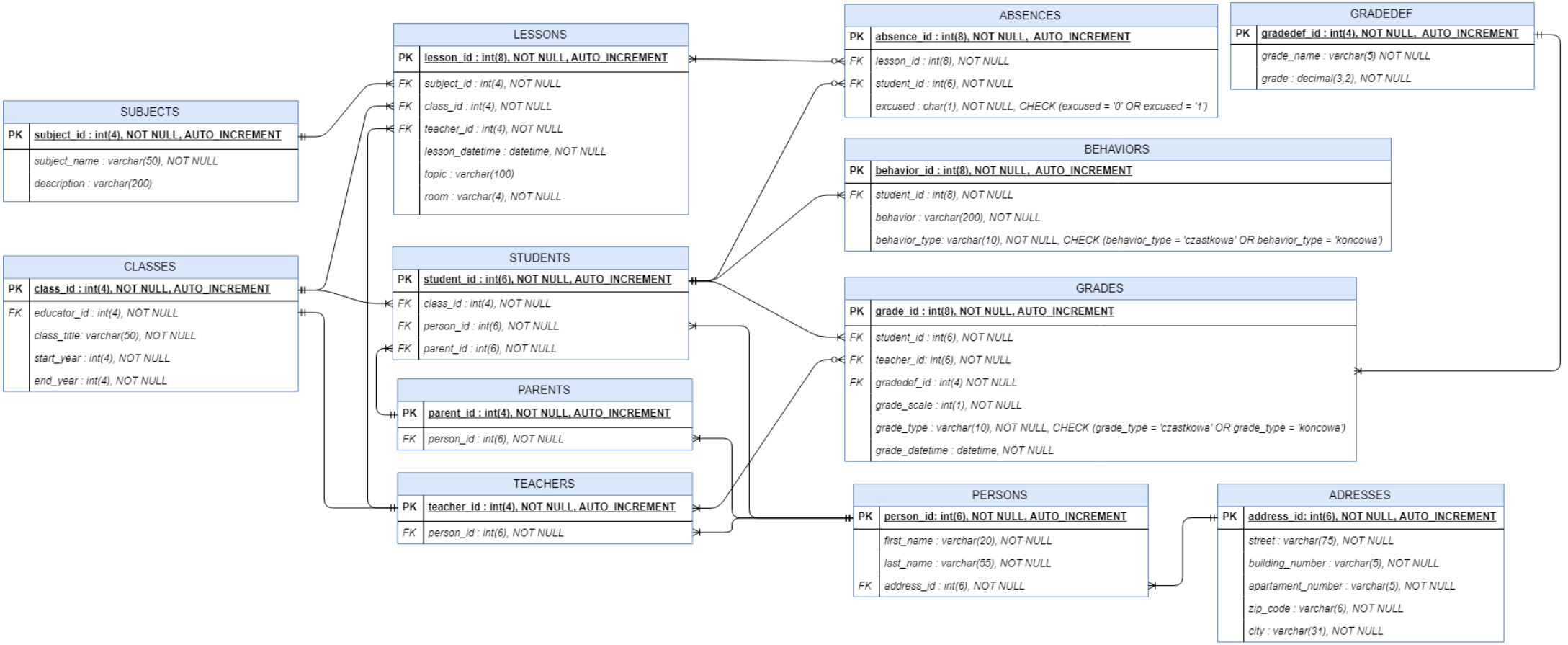
3.1. Projekt systemu

3.1.1. Analiza rzeczywistości, model logiczny i konceptualny

Szkoła w Pcimiu ma udostępniać uczniom ich własne postępy w nauce oraz kontrole nad nimi przez opiekunów. Nauczyciele i wychowawcy wprowadzają dane postępów podopiecznych, tematu lekcji oraz komunikatów. W systemie po uwierzytelnieniu zostaną wyświetlone funkcje dostępne dla danego użytkownika. W bazie będą znajdować się wszystkie dane dotyczące uczniów, ocen, frekwencji, uwag, planów zajęć, tematów. Dostęp do wglądu i modyfikacji poszczególnych danych zależy od udostępnionych funkcji dla użytkownika.



3.1.2 Model fizyczny i ograniczenia integralności danych



3.1.3 Inne elementy schematu – mechanizmy przetwarzania danych

Triggery

Triggery zostaną utworzone w odniesieniu do trzech tabel ABSENCES, GRADES oraz BEHAVIORS. Do każdej z nich zostanie przypisany trigger aktywowany po modyfikacji tabeli, który będzie zawierał informacje o typie zdarzenia, dacie oraz osobie modyfikującej.

- T_GRADE_LOGS – trigger wyzwalany przy dodawaniu/edycji/usunięciu wiersza w tabeli GRADES.
- T_ABSENCES_LOGS – trigger wyzwalany przy dodawaniu/edycji/usunięciu wiersza w tabeli ABSENCES
- T_BEHAVIORS_LOGS – trigger wyzwalany przy dodawaniu/edycji/usunięciu wiersza w tabeli ABSENCES

Widoki

Widoki będą utworzone na podstawie łączenia różnych tabel, poniżej w nawiasach zostały zapisane dokładnie jakich.

TIMETABLE – widok zawierający siatkę zajęć dla danej klasy(CLASSES), o konkretnej godzinie, dniu tygodnia(LESSONS) i z konkretnym nauczycielem(TEACHERS).

STUDENTS ABSENCES – lista nieobecności(ABSENCES) uczniów(STUDENTS) danej klasy(CLASSES).

REPORT CARD – świadectwo z uzyskanymi dotychczas średnimi ocen(GRADES) z danych przedmiotów(LESSONS), zachowaniem(BEHAVIORS) oraz dane osobowe ucznia(STUDENTS).

W celu uzyskania dokładnych danych nauczycieli oraz uczniów tabele te zostaną połączone z tabelą PERSONS.

Indeksy

Zostanie utworzony indeks do tabeli STUDENTS na kolumnach first_name oraz last_name pobranych z PERSONS.

3.1.4 Bezpieczeństwo na poziomie baz danych

Podział ze względu na dostęp do danych. Tabele, które nie zostały wymienione są wyłącznie do odczytu dla poniżej wymienionych użytkowników.

WYCHOWACA			NAUCZYCIEL		
ADD	UPADE	DELETE	ADD	UPADE	DELETE
GRADES	GRADES	GRADES	GRADES	GRADES	-
ABSENCES	ABSENCES	ABSENCES	ABSENCES	-	-
BEHAVORS	BEHAVORS	BEHAVORS	BEHAVORS (behavour_type='czastkowa')	BEHAVORS	-
-	LESSONS (topic)	-	-	LESSONS (topic)	-
-	PERSONS	-	-	-	-

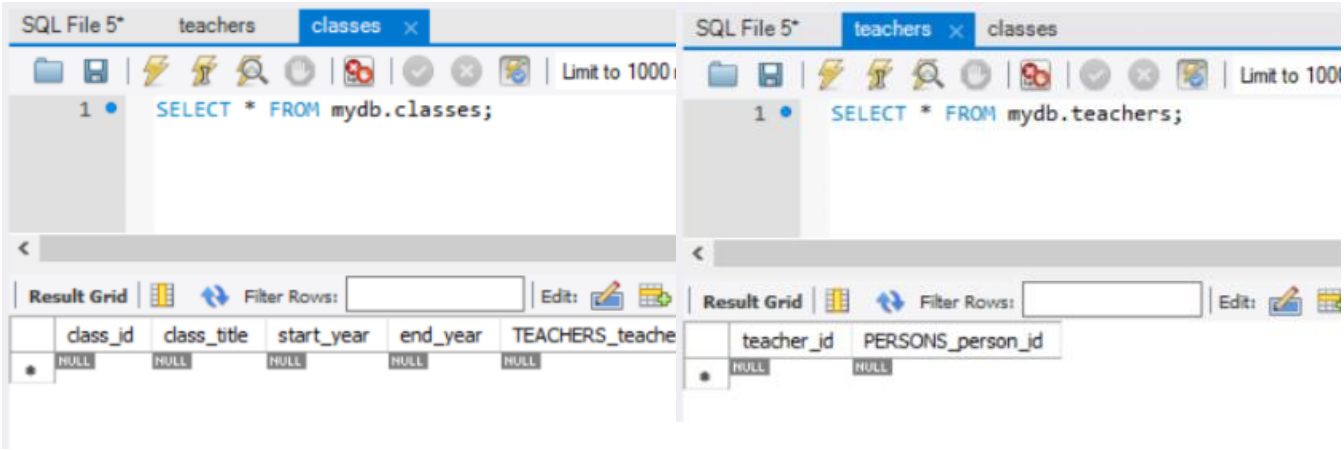
4. Implementacja sytemu baz danych

4.1 Tworzenie tabel i definiowanie ograniczeń

Tabele, relacje oraz ograniczenia zostały wygenerowane automatycznie na podstawie stworzonego modelu fizycznego. Przykład dla tabeli TEACHERS oraz CLASSES.

```
64  -- -----------------------------------------------------
65  -- Table `mydb`.`TEACHERS`
66  -- -----------------------------------------------------
67  CREATE TABLE IF NOT EXISTS `mydb`.`TEACHERS` (
68    `teacher_id` INT NOT NULL AUTO_INCREMENT,
69    `PERSONS_person_id` INT NOT NULL,
70    PRIMARY KEY (`teacher_id`, `PERSONS_person_id`),
71    INDEX `fk_TEACHERS_PERSONS1_idx` (`PERSONS_person_id` ASC),
72    CONSTRAINT `fk_TEACHERS_PERSONS1`
73      FOREIGN KEY (`PERSONS_person_id`)
74      REFERENCES `mydb`.`PERSONS` (`person_id`)
75      ON DELETE NO ACTION
76      ON UPDATE NO ACTION)
77  ENGINE = InnoDB;
78
79
80  -- -----------------------------------------------------
81  -- Table `mydb`.`CLASSES`
82  -- -----------------------------------------------------
83  CREATE TABLE IF NOT EXISTS `mydb`.`CLASSES` (
84    `class_id` INT NOT NULL AUTO_INCREMENT,
85    `class_title` VARCHAR(50) NOT NULL,
86    `start_year` INT NOT NULL,
87    `end_year` INT NOT NULL,
88    `TEACHERS_teacher_id` INT NOT NULL,
89    PRIMARY KEY (`class_id`, `TEACHERS_teacher_id`),
90    INDEX `fk_CLASSES_TEACHERS1_idx` (`TEACHERS_teacher_id` ASC),
91    CONSTRAINT `fk_CLASSES_TEACHERS1`
92      FOREIGN KEY (`TEACHERS_teacher_id`)
93      REFERENCES `mydb`.`TEACHERS` (`teacher_id`)
94      ON DELETE NO ACTION
95      ON UPDATE NO ACTION)
96  ENGINE = InnoDB;
```

Efekt implementacji kodu



4.2 Przykładowe dane oraz ich test

Dodanie przykładowych danych do tabeli PERSONS oraz STUDENTS

```
INSERT INTO `PERSONS` (`person_id`,`first_name`,`last_name`,`pesel`,`ADDRESSES_address_id`) VALUES
(1,"Paweł","Kwiecień","1208971234",7),
(2,"Adam","Zieliński","1236452184",8),
(3,"Łukasz","Bednarek","1245435341",9),
(4,"Gabriel","Nowak","2312342451",10),
(5,"Paweł","Kołodziej","1234574313",11),
(6,"Jakub","Pietrzak","1235432311",12),
(7,"Alicja","Wróbel","1223331222",13),
(8,"Katarzyna","Dybowska","1222223432",1),
(9,"Anna","Nowak","1231241246",2),
(10,"Julia","Nowakowska","1223434221",3),
(11,"Maria","Dybowska","1235435342",4),
(12,"Agnieszka","Wróbel","2345423432",5),
(13,"Monika","Pietrzak","5346624524",6),
```

```
INSERT INTO `STUDENTS` (`student_id`,`PERSONS_person_id`,`PARENTS_parent_id`,`CLASSES_class_id`) VALUES
(1,1,6,1),
(2,4,5,1),
(3,5,4,1),
(4,6,3,1),
(5,7,2,2),
(6,8,1,2),
(7,10,7,2);
```

Efekt implementacji

```
SELECT * FROM mydb.persons;
```

Result Grid | Filter Rows: | Edit: | Export/Import:

person_id	first_name	last_name	pesel	ADDRESSES_address_id
1	Paweł	Kwiecień	1208971234	7
2	Adam	Zieliński	1236452184	8
3	Łukasz	Bednarek	1245435341	9
4	Gabriel	Nowak	2312342451	10
5	Paweł	Kołodziej	1234574313	11
6	Jakub	Pietrzak	1235432311	12
7	Alicja	Wróbel	1223331222	13
8	Katarzyna	Dybowska	1222223432	1
9	Anna	Nowak	1231241246	2
10	Julia	Nowakowska	1223434221	3
11	Maria	Dybowska	1235435342	4
12	Agnieszka	Wróbel	2345423432	5
13	Monika	Pietrzak	5346624524	6

SQL File 5* | students | persons | students | Limit to 1000 rows

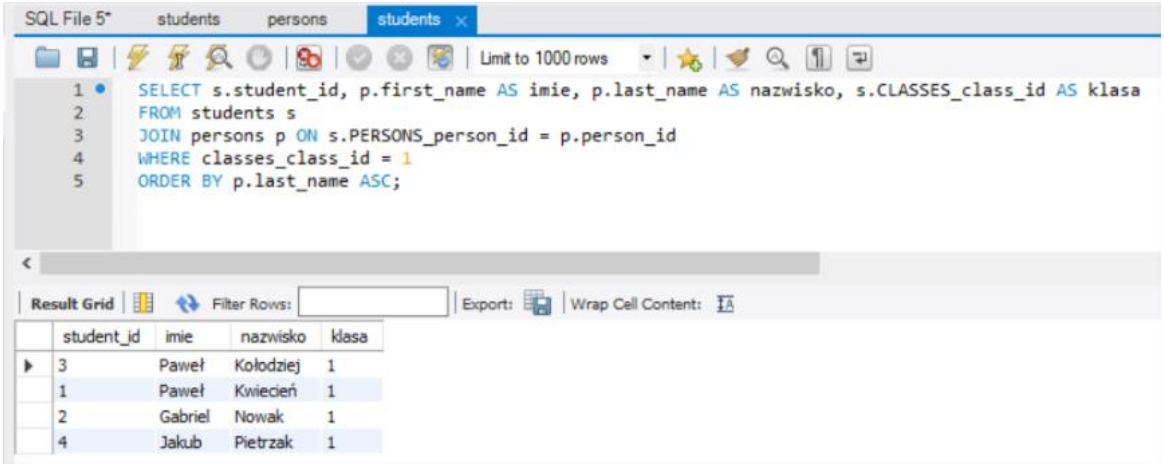
```
SELECT * FROM mydb.students;
```

Result Grid | Filter Rows: | Edit: | Export/Import:

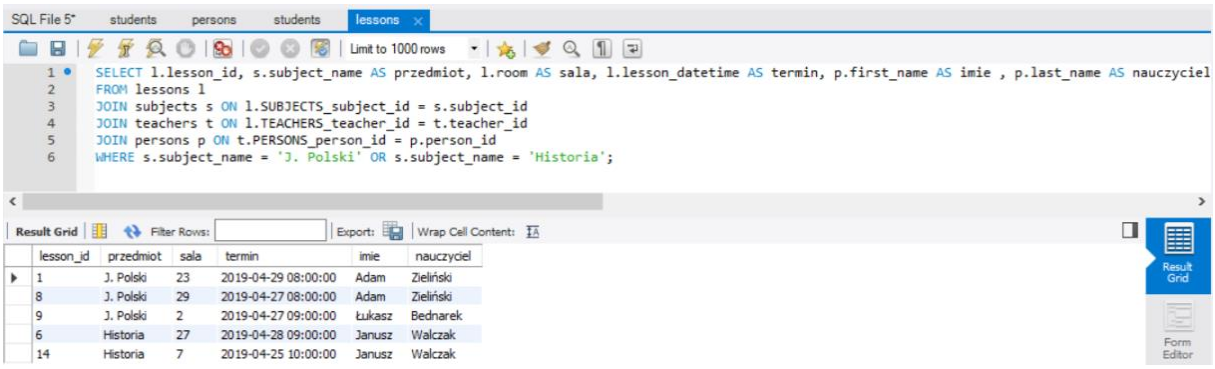
student_id	PERSONS_person_id	PARENTS_parent_id	CLASSES_class_id
1	1	6	1
2	4	5	1
3	5	4	1
4	6	3	1
5	7	2	2
6	8	1	2
7	10	7	2
NULL	NULL	NULL	NULL

Testy bazy

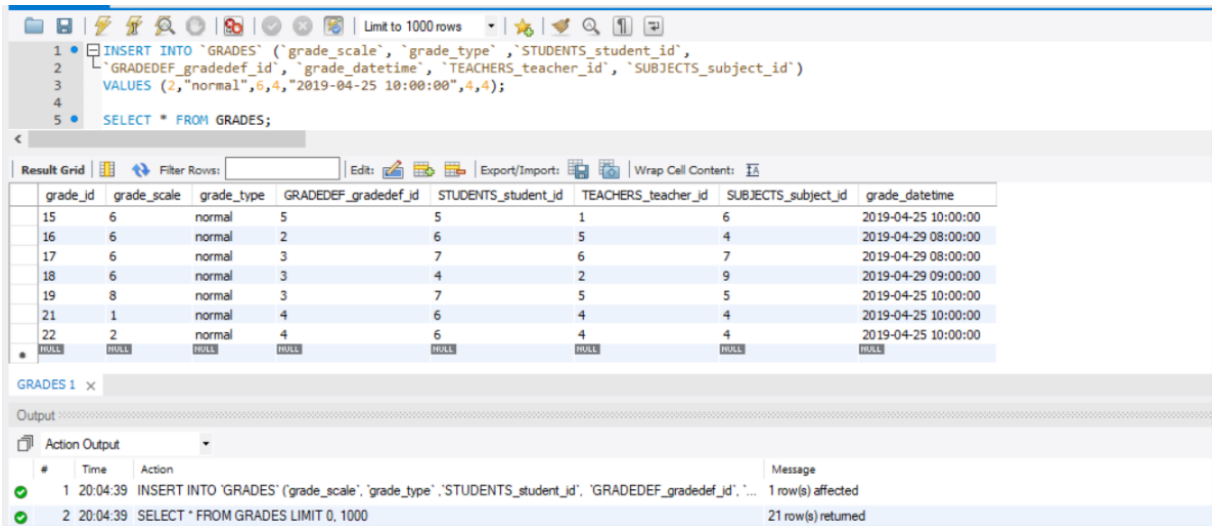
Wyszukanie wszystkich uczniów klasy o numerze id 1



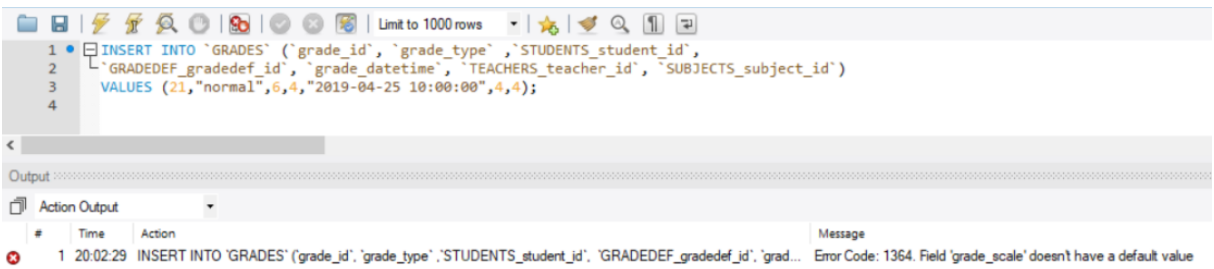
Wyszukanie wszystkich zajęć z języka polskiego oraz historii



Auto increment



Not null



Unique

```
INSERT INTO `GRADES` (`grade_id`,`grade_scale`, `grade_type` ,`STUDENTS_student_id`,
`GRADEDEF_gradedef_id`, `grade_datetime`, `TEACHERS_teacher_id`, `SUBJECTS_subject_id`)
 VALUES (20,8,"normal",6,4,"2019-04-25 10:00:00",4,4),
 (20,8,"normal",6,4,"2019-04-25 10:00:00",4,4);
```

Output

Action Output

#	Time	Action	Message
1	20:01:16	INSERT INTO 'GRADES' ('grade_id','grade_scale', 'grade_type' ,'STUDENTS_student_id', 'GRADEDEF_grad...	Error Code: 1062. Duplicate entry '20-4-6-4-4' for key 'PRIMARY'

4.3. Implementacja mechanizmów przetwarzania danych

Trigger dodający informacje do tabeli logów, wyzwalany przy dodaniu wiersza w tabeli ABSENCES

```
CREATE TRIGGER T_ABSENCES_I
AFTER INSERT
ON ABSENCES
FOR EACH ROW
INSERT INTO ABSENCES_LOGS(log_date,who,change_type) VALUES(now(),'TEACHER','INSERT');
```

Tabela logów przed i po poleceniu INSERT na tabeli ABSENCES

```
SELECT * FROM mydb.absences_logs;
```

log_date	who	change_type
2019-01-09 18:33:27	TEACHER	INSERT

```
SELECT * FROM mydb.absences_logs;
```

log_date	who	change_type
2019-01-09 18:33:27	TEACHER	INSERT
2019-01-09 18:38:38	TEACHER	INSERT

Widok lekcji dla klasy o numerze id 1

```
CREATE VIEW TIMETABLE AS
SELECT s.subject_name, l.lesson_datetime, l.room, p.first_name, p.last_name
FROM LESSONS l
JOIN SUBJECTS s ON l.SUBJECTS_subject_id = s.subject_id
JOIN TEACHERS t ON l.TEACHERS_teacher_id = t.teacher_id
JOIN PERSONS p ON t.PERSONS_person_id = p.person_id
WHERE CLASSES_class_id =1
ORDER BY `lesson_datetime` ASC;
```

```
SELECT * FROM mydb.timetable;
```

subject_name	lesson_datetime	room	first_name	last_name
Informatyka	2019-04-25 09:00:00	5	Jan	Mazur
Matematyka	2019-04-26 08:00:00	3	Roman	Kaczmarek
J. Polski	2019-04-27 08:00:00	29	Adam	Zieliński
J. Polski	2019-04-27 09:00:00	2	Łukasz	Bednarek
W-F	2019-04-28 08:00:00	26	Marian	Majewski
Historia	2019-04-28 09:00:00	27	Janusz	Walczak
J. Polski	2019-04-29 08:00:00	23	Adam	Zieliński
Matematyka	2019-04-29 09:00:00	25	Łukasz	Bednarek
J. Angielski	2019-04-29 10:00:00	26	Roman	Kaczmarek

Widok nieobecności klasy o numerze id 1

```
1  CREATE VIEW ABSENCES1 AS
2  SELECT a.excused, l.lesson_datetime, p.first_name, p.last_name
3  FROM ABSENCES a
4  JOIN LESSONS l ON a.LESSONS_lesson_id = l.lesson_id
5  JOIN STUDENTS s ON a.STUDENTS_student_id = s.student_id
6  JOIN PERSONS p ON s.PERSONS_person_id = p.person_id
7  WHERE l.CLASSES_class_id=1
8  ORDER BY `lesson_datetime` ASC;
```

```
SELECT * FROM mydb.absences1;
```

excused	lesson_datetime	first_name	last_name
0	2019-04-28 08:00:00	Jakub	Pietrzak
0	2019-04-28 09:00:00	Paweł	Kołodziej
1	2019-04-29 08:00:00	Gabriel	Nowak
1	2019-04-29 09:00:00	Paweł	Kwiecień

Indeksy

Indeksy na kluczach obcych zostały wygenerowane automatycznie. Przykłady indeksów dla tabeli GRADES, PERSONS oraz ABSENCES

```
205  INDEX `fk_GRADES_GRADEDEF1_idx` (`GRADEDEF_gradedef_id` ASC),
206  INDEX `fk_GRADES_STUDENTS1_idx` (`STUDENTS_student_id` ASC),
207  INDEX `fk_GRADES_TEACHERS1_idx` (`TEACHERS_teacher_id` ASC),
208  INDEX `fk_GRADES_SUBJECTS1_idx` (`SUBJECTS_subject_id` ASC),
209  INDEX `idx_grades_time` (`grade_datetime` DESC),
210  CONSTRAINT `fk_GRADES_GRADEDEF1`
```

```
41  INDEX `fk_PERSONS_ADDRESSES1_idx` (`ADDRESSES_address_id` ASC),
42  INDEX `inx_name` (`first_name` ASC, `last_name` ASC),
```

Czas działania kodu bez zastosowania indeksów oraz z indeksami dla tabeli PERSONS

```
sql> SELECT first_name, last_name FROM PERSONS
     WHERE last_name = 'Russam'
[2019-01-15 21:54:11] 1 row retrieved starting from 1 in 179 ms (execution: 14 ms, fetching: 165 ms)
sql> SELECT first_name, last_name FROM PERSONS
     WHERE last_name = 'Russam'
[2019-01-15 22:04:36] 1 row retrieved starting from 1 in 55 ms (execution: 20 ms, fetching: 35 ms)
```

Uprawnienia użytkowników nauczyciel oraz wychowawca

```
4  CREATE USER 'educator'@'localhost' IDENTIFIED BY 'educator';
5  CREATE USER 'teacher'@'localhost' IDENTIFIED BY 'teacher';
6
7  GRANT all privileges ON GRADES TO 'educator'@'localhost';
8  GRANT all privileges ON ABSENCES TO 'educator'@'localhost';
9  GRANT all privileges ON BEHAVIORS TO 'educator'@'localhost';
10 GRANT SELECT, UPDATE ON LESSONS TO 'educator'@'localhost';
11 GRANT SELECT, UPDATE ON PERSONS TO 'educator'@'localhost';
12
13 GRANT SELECT, INSERT, UPDATE ON GRADES TO 'teacher'@'localhost';
14 GRANT SELECT, INSERT ON ABSENCES TO 'teacher'@'localhost';
15 GRANT SELECT, INSERT, UPDATE ON BEHAVIORS TO 'teacher'@'localhost';
16 GRANT SELECT, UPDATE ON LESSONS TO 'teacher'@'localhost';
17 GRANT SELECT ON PERSONS TO 'teacher'@'localhost';
```

Zgody i odmowy dla poleceń wykonanych przez użytkownika educator i teacher