

Hubert Obarzanek

Bazy Danych Projekt "System do zapisywania się na przedmioty"

Zainstalowałem na komputerze serwer bazodanowy OracleSQL. Następnie skonfigurowałem wszystkie połączenia i nadałem uprawnienia użytkownikowi:

```
alter session set "_ORACLE_SCRIPT"=true;

create user new_user IDENTIFIED BY "123456789";
commit;

GRANT CREATE TABLE TO NEW_USER;
commit;

GRANT CREATE SESSION TO NEW_USER;
commit;

ALTER USER new_user quota unlimited on USERS;
commit;

grant CREATE SEQUENCE to NEW_USER;
grant CREATE PROCEDURE to NEW_USER;
grant CREATE TRIGGER to NEW_USER;
grant CREATE VIEW to NEW_USER;
commit;
```

następnie stworzyłem 5 tabel(kod wygenerowany przez Django): Nauczyciel, Uczeń, Przedmioty, Klasy, Zapisy na przedmioty:

```
--
-- Create model Classes
--
CREATE TABLE "DZINNIK_CLASSES"
(
    "ID"          NUMBER(19) GENERATED BY DEFAULT ON NULL AS IDENTITY NOT NULL PRIMARY KEY,
    "CLASS_NAME" NVARCHAR2(10)                                     NULL,
    "START_YEAR"  NUMBER(11)                                       NOT NULL,
    "END_YEAR"    NUMBER(11)                                       NOT NULL
);
--
-- Create model Teachers
--
CREATE TABLE "DZINNIK_TEACHERS"
(
    "ID"          NUMBER(19) GENERATED BY DEFAULT ON NULL AS IDENTITY NOT NULL PRIMARY KEY,
    "NAME"        NVARCHAR2(50)                                     NULL,
    "LASTNAME"    NVARCHAR2(50)                                     NULL,
    "BIRTHDATE"   TIMESTAMP                                         NOT NULL,
    "EMAIL"       NVARCHAR2(254)                                    NULL
);
```

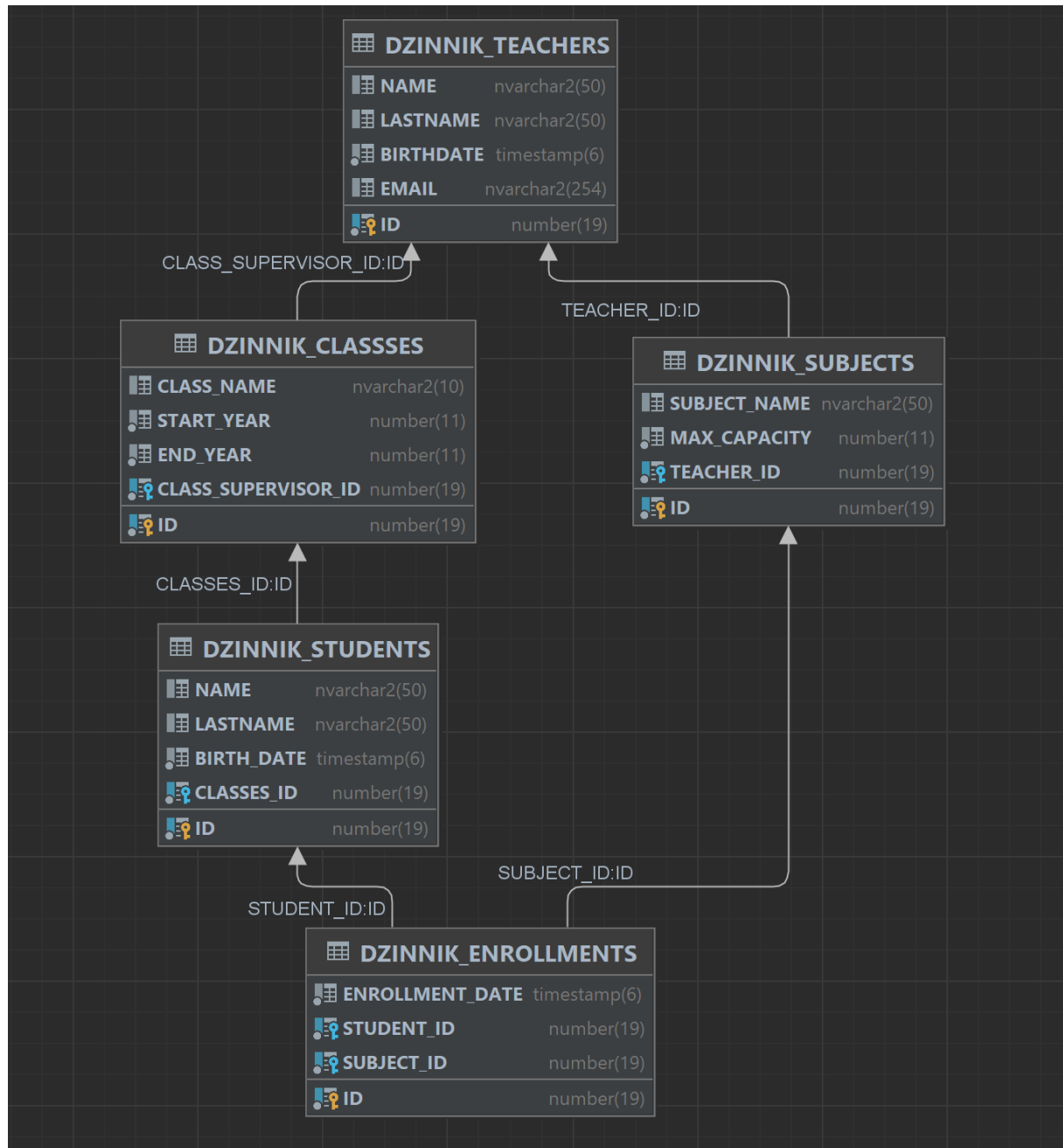
```
--
-- Create model Subjects
--
CREATE TABLE "DZINNIK_SUBJECTS"
(
    "ID" NUMBER(19) GENERATED BY DEFAULT ON NULL AS IDENTITY NOT NULL PRIMARY KEY,
    "SUBJECT_NAME"
-- Create model Students
--
    CREATE TABLE "DZINNIK_STUDENTS"
    ("ID" NUMBER (19) GENERATED BY DEFAULT ON NULL AS IDENTITY NOT NULL PRIMARY KEY,
    "NAME" NVARCHAR--
-- Create model Enrollments
--
    CREATE TABLE "DZINNIK_ENROLLMENTS"
    ("ID" NUMBER (19) GENERATED BY DEFAULT ON NULL AS IDENTITY NOT NULL PRIMARY KEY,
    "ENROLLMENT_DATE" TIMESTAMP NOT NULL,
    "STUDENT_ID" NUMBER (19) NOT NULL,
    "SUBJECT_ID" NUMBER (19) NOT NULL);
--
```

```

-- Add field class_supervisor to classes
--
ALTER TABLE "DZINNIK_CLASSESSES"
  ADD "CLASS_SUPERVISOR_ID" NUMBER(19) NOT NULL
  CONSTRAINT "DZINNIK_C_CLASS_SUP_C8959F6A_F" REFERENCES "DZINNIK_TEACHERS" ("ID") DEFERRABLE INITIALLY DEFERRED;
ALTER TABLE "DZINNIK_SUBJECTS"
  ADD CONSTRAINT "DZINNIK_S_TEACHER_I_3FAAA196_F" FOREIGN KEY ("TEACHER_ID") REFERENCES
    "DZINNIK_TEACHERS" ("ID") DEFERRABLE INITIALLY DEFERRED;
CREATE INDEX "DZINNIK_SU_TEACHER_ID_3FAAA196" ON "DZINNIK_SUBJECTS" ("TEACHER_ID");
ALTER TABLE "DZINNIK_STUDENTS"
  ADD CONSTRAINT "DZINNIK_S_CLASSES_I_8E345D65_F" FOREIGN KEY ("CLASSES_ID") REFERENCES
    "DZINNIK_CLASSESSES" ("ID") DEFERRABLE INITIALLY DEFERRED;
CREATE INDEX "DZINNIK_ST_CLASSES_ID_8E345D65" ON "DZINNIK_STUDENTS" ("CLASSES_ID");
ALTER TABLE "DZINNIK_ENROLLMENTS"
  ADD CONSTRAINT "DZINNIK_E_STUDENT_I_8D8F63E3_F" FOREIGN KEY ("STUDENT_ID") REFERENCES
    "DZINNIK_STUDENTS" ("ID") DEFERRABLE INITIALLY DEFERRED;
ALTER TABLE "DZINNIK_ENROLLMENTS"
  ADD CONSTRAINT "DZINNIK_E_SUBJECT_I_040ED708_F" FOREIGN KEY ("SUBJECT_ID") REFERENCES
    "DZINNIK_SUBJECTS" ("ID") DEFERRABLE INITIALLY DEFERRED;
CREATE INDEX "DZINNIK_EN_STUDENT_ID_8D8F63E3" ON "DZINNIK_ENROLLMENTS" ("STUDENT_ID");
CREATE INDEX "DZINNIK_EN_SUBJECT_ID_040ED708" ON "DZINNIK_ENROLLMENTS" ("SUBJECT_ID");
CREATE INDEX "DZINNIK_CL_CLASS_SUPE_C8959F6A" ON "DZINNIK_CLASSESSES" ("CLASS_SUPERVISOR_ID");

```

oto schemat relacyjnej bazy danych:



Następnie utworzyłem przykładowe dane:

```
insert into DZINNIK_TEACHERS(NAME, LASTNAME, BIRTHDATE, EMAIL)
VALUES ('Jan', 'Kowalski', '2000-05-19', 'jankowalski@agh.pl');
insert into DZINNIK_TEACHERS(NAME, LASTNAME, BIRTHDATE, EMAIL)
VALUES ('Michał', 'Nowak', '1900-01-18', 'michalnowak@uj.pl');
insert into DZINNIK_TEACHERS(NAME, LASTNAME, BIRTHDATE, EMAIL)
VALUES ('Anna', 'Wisniewska', '1985-12-25', 'anka@uw.pl');
```

```
insert into DZINNIK_CLASSES (class_name, start_year, end_year, class_supervisor_id)
VALUES ('5a', 2023, 2026, 1);
insert into DZINNIK_CLASSES (class_name, start_year, end_year, class_supervisor_id)
VALUES ('3b', 2022, 2025, 2);
```

```
insert into DZINNIK_STUDENTS (NAME, LASTNAME, BIRTH_DATE, CLASSES_ID)
VALUES ('Karol', 'Kowalski', '2009-12-15', 1);
insert into DZINNIK_STUDENTS (NAME, LASTNAME, BIRTH_DATE, CLASSES_ID)
VALUES ('Katarzyna', 'Nowak', '2006-10-12', 1);
insert into DZINNIK_STUDENTS (NAME, LASTNAME, BIRTH_DATE, CLASSES_ID)
VALUES ('Wojciech', 'Nowakowski', '2013-09-01', 2);
insert into DZINNIK_STUDENTS (NAME, LASTNAME, BIRTH_DATE, CLASSES_ID)
VALUES ('Karolina', 'Jakas', '2014-08-03', 2);
```

```
insert into DZINNIK_SUBJECTS (SUBJECT_NAME, MAX_CAPACITY, TEACHER_ID)
values ('Matematyka', 20, 1);
insert into DZINNIK_SUBJECTS (SUBJECT_NAME, MAX_CAPACITY, TEACHER_ID)
values ('J.Polski', 5, 2);
insert into DZINNIK_SUBJECTS (SUBJECT_NAME, MAX_CAPACITY, TEACHER_ID)
values ('Geografia', 2, 4);
```

stworzyłem widok który pokazuje ilość zajętych miejsc na dany przedmiot:

```
--ilość zajętych miejsc na dany przedmiot
create or replace view occupied_seats as
select SUBJECT_ID, count(*) as occupied
from DZINNIK_ENROLLMENTS
group by SUBJECT_ID;
```

widok który pokazuje ilość wolnych miejsc na dany przedmiot:

```
--ilość wolnych miejsc na dany przedmiot
CREATE OR REPLACE VIEW available_seats as
select SUBJECT_ID, DZINNIK_SUBJECTS.MAX_CAPACITY - COALESCE(os.occupied, 0) as available_seats
from DZINNIK_SUBJECTS
left join occupied_seats os on DZINNIK_SUBJECTS.ID = os.SUBJECT_ID;
```

stworzyłem procedurę która umożliwia zapisywanie się na zajęcia. Sprawdza ona czy podane numery id studenta i przedmiotu istnieją w bazie, następnie sprawdza czy student nie jest już zapisany na te zajęcia a następnie sprawdza czy jest wolne miejsce na tych zajęciach:

```
--zapisywanie studenta na zajęcia
create or replace procedure enroll_student(student_id in number, subject_id in number) is
v_available_seats number;
begin
if not exists(select * from DZINNIK_STUDENTS where ID = student_id) then
raise_application_error(-20000, 'student o podanym id nie istnieje');
end if;
if not exists(select * from DZINNIK_SUBJECTS where ID = subject_id) then
raise_application_error(-20000, 'przedmiot o podanym id nie istnieje');
end if;
--sprawdza czy student jest już zapisany na zajęcia
if exists(select * from DZINNIK_ENROLLMENTS where SUBJECT_ID = subject_id and STUDENT_ID = student_id) then
raise_application_error(-20000, 'student jest już zapisany na zajęcia');
end if;
--sprawdza czy są dostępne miejsca
select available_seats into v_available_seats from available_seats where available_seats.SUBJECT_ID = subject_id;
if v_available_seats <= 0 then
raise_application_error(-20000, 'brak wolnych miejsc');
end if;

insert into DZINNIK_ENROLLMENTS (ENROLLMENT_DATE, STUDENT_ID, SUBJECT_ID) VALUES (SYSDATE, student_id, subject_id);
end;
```

procedura która usuwa studenta z zapisanego przedmiotu:
przed usunięciem go sprawdza czy student rzeczywiście był zapisany na ten przedmiot:

```
--usuwanie studenta z zapisanego przedmiotu
create or replace procedure remove_student_from_enrollment(
    student_id in number,
    subject_id in number
) is
begin
    if not exists(select * from DZINNIK_ENROLLMENTS where SUBJECT_ID = subject_id and STUDENT_ID = student_id) then
        raise_application_error(-20000, 'student nie jest zapisany do tego przedmiotu');
    end if;
    delete from DZINNIK_ENROLLMENTS where STUDENT_ID = student_id and SUBJECT_ID = subject_id;
end;
```

stworzyłem również trigger który przed usunięciem przedmiotu sprawdza czy ktoś jest zapisany na dany przedmiot i jeśli tak to nie usuwa danego przedmiotu.

```
--trigger przed usunięciem przedmiotu:
CREATE OR REPLACE TRIGGER before_delete_subject
BEFORE DELETE ON DZINNIK_SUBJECTS
FOR EACH ROW
DECLARE
    subject_id NUMBER;
BEGIN
    subject_id := :OLD.ID;

    -- Sprawdź, czy istnieją zapisy na ten przedmiot
    IF EXISTS(SELECT 1 FROM DZINNIK_ENROLLMENTS WHERE SUBJECT_ID = subject_id) THEN
        -- Jeśli istnieją zapisy, zatrzymaj usuwanie rekordu
        RAISE_APPLICATION_ERROR(-20001, 'Nie można usunąć przedmiotu. Istnieją zapisy uczniów na ten przedmiot.');
```

Powyższe widoki, procedury i trigger nie zostały wykorzystane w projekcie. Frontend i backend zostały zrobione w frameworku Django.

Na początku tworzenia projektu w django musiałem skonfigurować z jakiego typu bazy będziemy korzystać oraz połączenie do bazy danych:

fragment pliku settings.py:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.oracle',
        'NAME': "XE",
        "USER": "NEW_USER",
        "PASSWORD": "123456789",
        "HOST": "127.0.0.1",
        "PORT": "1521",
    }
}
```

Następnie w pliku models.py stworzyłem 5 klas z polami które odzwierciedlają odpowiednio tabele oraz kolumny w bazie danych.

plik models.py:

```
from django.db import models

# Create your models here.

class Teachers(models.Model):
    name=models.CharField(max_length=50)
    lastname=models.CharField(max_length=50)
    birthdate=models.DateTimeField()
    email=models.EmailField()

    def __str__(self):
        return self.name+" "+self.lastname

class Classes(models.Model):
    class_name=models.CharField(max_length=10)
    start_year=models.IntegerField()
    end_year=models.IntegerField()
    class_supervisor=models.ForeignKey(Teachers,on_delete=models.CASCADE)

class Students(models.Model):
    name=models.CharField(max_length=50)
    lastname=models.CharField(max_length=50)
    birth_date=models.DateTimeField()
    classes=models.ForeignKey(Classes,on_delete=models.CASCADE)

    def __str__(self):
        return self.name+" "+self.lastname

class Subjects(models.Model):
    subject_name=models.CharField(max_length=50)
    teacher=models.ForeignKey(Teachers,on_delete=models.CASCADE)
    max_capacity=models.IntegerField()

    def __str__(self):
        return self.subject_name
```



```
class Enrollments(models.Model):
    student=models.ForeignKey(Students,on_delete=models.CASCADE)
    subject=models.ForeignKey(Subjects,on_delete=models.CASCADE)
    enrollment_date=models.DateTimeField(auto_now_add=True)
```

Później stworzyłem widoki z szablonami html które odpowiadają za generowanie danej podstrony:

widok index:

```
def index(request):
    subjects_list=Subjects.objects.all()
    template=loader.get_template("dzinnik/index.html")
    context={
        "subjects_list": subjects_list
    }
    return HttpResponse(template.render(context,request))
```

szablon index.html:

```
{% if subjects_list %}
    <table>
        <tr>
            <th>Nazwa przedmiotu</th>
            <th>Prowadzący</th>
            <th>Liczba miejsc</th>
        </tr>
        {% for subject in subjects_list %}
            <tr>
                <th>{{subject.subject_name}}</th>
                <th>{{subject.teacher.name}} {{subject.teacher.lastname}}</th>
                <th>{{subject.max_capacity}}</th>
            </tr>
        {% endfor %}
    </table>
{% else %}
    <p>No subjects.</p>
{% endif %}

<a href="{% url 'add_subject' %}">Dodaj przedmiot</a>
<a href="{% url 'enroll_to_subject' %}">Zapisz się na przedmiot</a>
```

widok add_subject:

```
def add_subject(request):
    if request.method != 'POST':
        form = AddSubjectForm()
    else:
        form = AddSubjectForm(data=request.POST)
        if form.is_valid():
            new_subject = form.save(commit=True)
            new_subject.save()
            return redirect('index')
    context = {'form': form}
    return render(request, 'dzinnik/add_subject.html', context)
```

w tym widoku musiałem skorzystać z formularza który pobierze dane od użytkownika i na ich podstawie doda odpowiedni rekord w bazie danych:

formularz AddSubjectForm (plik forms.py):

```
class AddSubjectForm(forms.ModelForm):
    class Meta:
        model=Subjects
        fields=['subject_name','teacher','max_capacity']
        labels={'subject_name':'Nazwa przedmiotu','teacher':'Nauczyciel','max_capacity':'Maksymalna ilość'}
```

szablon add_subject.html:

```
<form method="post" action="{% url 'add_subject' %}">
    {% csrf_token %}
    {{form.as_p}}

    <button name="submit">Dodaj przedmiot</button>
    <input type="hidden" name="next" value="{% url 'index' %}" />
</form>
```

widok enroll_to_subject:

```
def enroll_to_subject(request):
    if request.method!='POST':
        form=EnrollToSubject()
    else:
        form=EnrollToSubject(data=request.POST)
        if form.is_valid():
            new_enrollment=form.save(commit=True)
            new_enrollment.save()
            return redirect('index')
    context={'form':form}
    return render(request,'dzinnik/enroll_to_subject.html',context)
```

w tym widoku korzystam z formularza `EnrollToSubject`, który doda rekord do tabeli `ENROLLMENTS` :

```
class EnrollToSubject(forms.ModelForm):
    class Meta:
        model=Enrollments
        fields=['student','subject']
        labels={'student':'Uczeń','subject':'Przedmiot'}
```

szablon `enroll_to_subject.html`:

```
<form method="post" action="{% url 'enroll_to_subject' %}">
    {% csrf_token %}
    {{form.as_p}}

    <button name="submit">Zapisz się!</button>
    <input type="hidden" name="next" value="{% url 'index' %}" />
</form>
```

aby wszystkie te widoki działały dodałem odpowiednie przekierowania url w pliku `urls.py`:

```
urlpatterns = [
    path("", views.index, name="index"),
    path("add_subject/", views.add_subject, name="add_subject"),
    path("enroll_to_subject/", views.enroll_to_subject, name="enroll_to_subject"),
]
```

Finalnie program prezentował się następująco:

Strona główna wygląda następująco:


Nazwa przedmiotu	Prowadzący	Liczba miejsc
Matematyka	Jan Kowalski	20
J.Polski	Michał Nowak	5
Geografia	Anna Wisniewska	2
Informatyka	Jan Kowalski	10

[Dodaj przedmiot](#) [Zapisz się na przedmiot](#)

Znajdują się tutaj 2 odnośniki dzięki którym możemy Dodać do bazy przedmiot lub zapisać się na dany przedmiot.

widok na dodawanie przedmiotu:

Nazwa przedmiotu:

Nauczyciel: 

Maksymalna

Możemy wpisać tutaj nazwę przedmiotu, wybrać z listy nauczyciela prowadzącego i ustalić ilość dostępnych miejsc.

widok na zapisywanie się na przedmioty:

Uczeń: ▼

Przedmiot: ▼

Zapisz się!

W tym widoku mamy 2 rozwijane listy, w których wybieramy nazwisko ucznia i przedmiot na który dany uczeń chce się zapisać.

Uczeń: ▼

Przedm: ▼


Zapisz

Karol Kowalski

Katarzyna Nowak

Wojciech Nowakowski

Karolina Jakas

Uczeń: 

Przedmiot: 

Zapisz się!

Matematyka
J.Polski
Geografia
Informatyka
J angielski