Çizge Renklendirme ile Ders Programı Hazırlama Proje Raporu

Helen Hacer Uzunçayır

Kocaeli Üniversitesi Teknoloji Fakültesiİzmit/Kocaeli  
211307061[@kocaeli.edu.tr](mailto:211307016@kocaeli.edu.tr)

Selin Avcı

Kocaeli Üniversitesi Teknoloji Fakültesiİzmit/Kocaeli  
211307026[@kocaeli.edu.tr](mailto:211307016@kocaeli.edu.tr)

Seher Köroğlu

Kocaeli Üniversitesi Teknoloji Fakültesiİzmit/Kocaeli  
[211307016@kocaeli.edu.tr](mailto:211307016@kocaeli.edu.tr)

*Abstract*—*This project is designed to manage and solve the constraints encountered in preparing schedules for teachers delivering courses. Database operations are carried out with a table containing information about the courses taught by teachers and constraints managed on a separate table. The main goal of the project is to create an optimized schedule, taking into account factors such as preventing course conflicts for teachers, assigning courses to desired days, and considering class numbers and names. Database operations are implemented using MySQL and provide the ability to track changes made in the database through a simple interface. During the course scheduling phase, graph coloring methods are used to produce solutions that comply with the specified constraints. These solutions are presented to the user in a simple tabular format.*

*Özet—Bu proje, ders veren hocaların ders programlarını hazırlarken karşılaşılan kısıtları yönetmek ve çözmek amacıyla tasarlanmıştır. Veritabanı işlemleri, öğretmenlere ait ders bilgilerini içeren bir tablo ile kısıtları ayrı bir tablo üzerinde gerçekleştirir. Projenin temel amacı, öğretmenlerin ders çakışmalarını önlemek, derslerini istedikleri günlere atamak, sınıf sayıları ve isimleri gibi faktörlere dikkat ederek optimize edilmiş bir ders programı oluşturmaktır. Veritabanı işlemleri, MySQL kullanılarak gerçekleştirilmiştir ve basit bir arayüz aracılığıyla veritabanında yapılan değişiklikleri takip etme imkanı sunmaktadır. Ders programı hazırlama aşamasında çizge renklendirme yöntemleri kullanılarak, belirlenen kısıtlara uygun çözümler üretilmekte ve bu çözümler basit bir tablo şeklinde kullanıcıya sunulmaktadır.*

1. **GİRİŞ**

Eğitim kurumlarındaki ders programlarını oluşturmak, hem öğretmenler hem de öğrenciler için karmaşık bir süreç olabilir. Bu süreçte karşılaşılan zorlukları ele alarak daha etkili bir çözüm sunan bu proje, ders programı hazırlama aşamasında ortaya çıkan kısıtları yönetmeyi ve optimize etmeyi amaçlamaktadır. Proje, öğretmenlere ait ders bilgilerini ve bu derslerin programlanmasında dikkate alınması gereken kısıtları içeren bir veritabanı yapısını kullanmaktadır. Öğretmenlerin ders çakışmalarını önlemek, derslerini istedikleri gün ve saatlere yerleştirmek, sınıf sayıları ve isimleri gibi faktörleri göz önünde bulundurarak daha dengeli bir ders programı oluşturmayı hedeflemektedir.

Veritabanı işlemleri, öğretmenlere ait ders bilgilerini düzenli bir şekilde kaydedip yönetmeyi sağlar. Kısıtlar ise öğretmenlerin ders programlarına ilişkin belirli kurallara uymalarını sağlar. Proje, basit bir arayüzle kullanıcılara veritabanındaki değişiklikleri takip etme olanağı sunar. Ayrıca, ders programını oluştururken çizge renklendirme yöntemlerini kullanarak, belirlenen kısıtlara uygun ve optimize edilmiş bir çözüm sunar. Bu sayede daha düzenli, çakışmasız ve dengeli ders programları oluşturulabilir.

1. **VERİTABANI KULLANIMI**

Bu projede, ders programlarını düzenlemek ve öğretmenlerin ders atamalarını etkin bir şekilde gerçekleştirmek amacıyla bir veritabanı tasarlanmış ve MySQL Workbench kullanılarak bu tasarım hayata geçirilmiştir. Veritabanı, öğretmenlerin dersleri, günleri, saatleri, kısıtları ve bu unsurlar arasındaki ilişkileri içerir. MySQL Workbench üzerinde yapılan tasarım, SQL tabanlı bir veritabanını ortaya çıkararak, öğretmenlere ders programlarını düzenleme, değiştirme ve sorgulama yeteneği sağlar.

**A)** **MySQL Workbench kullanımı ve SQL**

MySQL Workbench, MySQL veritabanı yönetimini kolaylaştıran bir araçtır. Bu araç, veritabanı tasarlama, uygulama geliştirme, SQL sorguları oluşturma, veritabanı yönetimi gibi bir dizi görevi gerçekleştirmek için kullanılır. Projenizde MySQL Workbench kullanılarak tasarlanan veritabanı, işlevselliği ve ilişkileri grafiksel bir arayüz üzerinde görsel olarak sunar.

SQL, ilişkisel veritabanı yönetim sistemlerinde (RDBMS) veri manipülasyonu, sorgulama ve yönetim için kullanılan bir programlama dilidir. Veritabanında tablolar, ilişkiler ve sorgular SQL kullanılarak oluşturulur ve yönetilir.

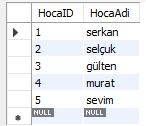
**B) Veritabanı Tasarımı**

MySQL Workbench üzerinde tasarlanan veritabanı, öğretmenlerin ders programlarını, kısıtları ve ilişkileri etkileşimli bir şekilde yönetebilmelerini sağlar. Tablolar arasındaki ilişkiler, referans anahtarlar ve benzersiz kimlikler gibi SQL konseptleri kullanılarak veritabanı tasarımı gerçekleştirilir.

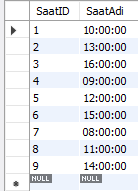
Bu projede MySQL Workbench kullanılarak tasarlanan SQL tabanlı veritabanı, öğretmenlerin ders programlarını oluştururken dikkate almaları gereken öğeleri içerir. Hocalar, dersler, günler, saatler ve kısıtlar gibi tablolar arasındaki ilişkiler, bu veritabanı üzerinde etkileşimli bir şekilde yönetilebilir. SQL sorguları, veritabanındaki bilgileri çekmek, güncellemek ve silmek için kullanılır.

**C) Veritabanı Modeli**

Veritabanı modelinde oluşturulan ilişkisel tablolar aşağıda verilmiştir:

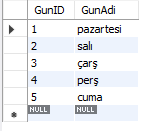
 *Resim C.1 Hoca Tablosu*

Resim C.1’ de görülen öğretmen tablosunda öğretmenler temsil edilmiştir. Tablo, her öğretmen için benzersiz bir kimlik (HocaID) ve öğretmenin adını tutan bir metin alanı (Text) içermektedir.

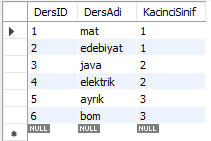


*Resim C.2 Saatler Tablosu*

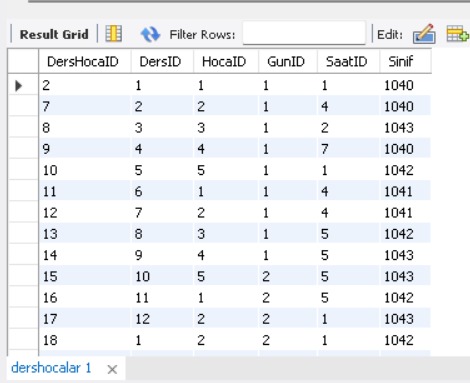
Resim C.2’de görülen Saat tablosunda derslerin hangi saatlerde verileceği belirtilmiştir. Saat ID, Saatin benzersiz kimliğini temsil eden bir tamsayı ve Saat kısmı ise dersin başlama saatini temsil eden bir TEXT alanıdır.

 *Resim C.3 Günler tablosu*

Resim C.3’te görülen Günler tablosunda GunID, Günün benzersiz kimliğini temsil eden bir tamsayı ve GunAdi da Günün adını temsil eden bir metin (TEXT) alanıdır.

 *Resim C.4 Dersler tablosu*

Resim C.4 tablosu ders programında öğretmenlerin vereceği dersleri, bunların benzersiz ID’lerini ve kaçıncı sınıfa verildiklerini içermektedir.



*Resim C.5 Ders Programı Tablosu*

DersHocalar Tablosu, derslerin hangi hocalar tarafından hangi gün ve saatlerde verileceğinin ilişkilendirildiği tablodur. Her ders-hoca ilişkisi için benzersiz bir kimlik (DersHocaID), dersin kimliğini temsil eden bir tamsayı alanı (DersID), öğretmenin kimliğini temsil eden bir tamsayı alanı (HocaID), günün kimliğini temsil eden bir tamsayı alanı (GunID) ve saatin kimliğini temsil eden bir tamsayı alanı (SaatID) içermektedir.

Bu projede kullanılan veritabanı modellemesinde, çeşitli tablolar arasında belirli ilişkiler kurularak öğretmenlerin ders programlarını düzenleme ve atama süreçlerini optimize etmek amaçlanmıştır. Hoca tablosu, öğretmenlerin bilgilerini içerir ve DersHocalar tablosu ile "Çoktan-Bire" ilişkisi kurulmuştur; bir öğretmenin birden fazla dersi olabilir, ancak her ders bir öğretmene aittir. Benzer şekilde, Ders tablosu, derslerin detaylarını içerir ve DersHocalar tablosu ile "Çoktan-Bire" ilişkisi ile bağlantılıdır; bir dersin birden fazla ataması olabilir, ancak her atama tek bir dersi temsil eder. Gun, Saat ve DersHocalar tabloları arasında da "Çoktan-Bire" ilişkileri vardır; her biri gün, saat ve atama bilgilerini içerir, ancak her atama tek bir gün ve saatle ilişkilidir.

Hoca tablosunda (*Resim C.1)*, her öğretmenin benzersiz kimliğini temsil eden HocaID, birincil anahtardır. Bu kimlik, öğretmenin adını temsil eden Adi alanıyla ilişkilidir.

Ders tablosunda (Resim C.4), her dersin benzersiz kimliğini temsil eden DersID, birincil anahtardır. DersAdi, dersin adını içerirken, KacinciSinif, dersin hangi sınıflara yönelik olduğunu belirtir.

Gun tablosunda (*Resim C.3)*, her günün benzersiz kimliğini temsil eden GunID, birincil anahtardır. GunAdi, günün adını içerir.

Saat tablosunda (*Resim C.2)*, her saatin benzersiz kimliğini temsil eden SaatID, birincil anahtardır. SaatAdi ise saati tutan bir TEXT alanıdır.

DersHocalar(*Resim C.5)* tablosunda, her dersin ve öğretmenin atamasını temsil eden DersHocaID, birincil anahtardır. DersID, HocaID, GunID ve SaatID, ilgili tablolarla ilişkili yabancı anahtarlar aracılığıyla bağlantı kurar. Bu tasarım, veritabanındaki verilerin benzersizliğini ve bütünlüğünü sağlamak için özenle düzenlenmiştir.

1. **KODLARIN AÇIKLANMASI**

**A) index.html in açıklanması:**

HTML dosyasında, bir öğretmen program takvimini düzenlemek için bir arayüz ve işlevsellik bulunmaktadır.

1. HTML Başlangıcı:

- `<!DOCTYPE html>`: HTML5 belgesi tanımlanıyor.

- `<html>`: HTML belgesinin başlangıcı.

- `<head>`: Belgenin başlık, karakter seti ve stil bilgilerini içerir.

- `<meta charset="UTF-8">`: Karakter seti UTF-8 olarak ayarlanıyor.

- `<title>`: Sayfa başlığı belirleniyor.

- `<meta>`: Mobil cihazlara uyumlu görünüm için viewport ayarları yapılıyor.

- `<style>`: Sayfaya özgü CSS kuralları burada tanımlanıyor.

2. Form ve Girdi Alanları:

- `<form>`: Ders ve hocaya ait bilgileri girmek için bir form oluşturuluyor.

- `id` ve `name` attribute'larıyla giriş alanları belirleniyor (`DersHocaID`, `DersID`, `HocaID`, `GunID`, `SaatID`, `Sinif`, `yeniDersID`, `yeniHocaID`, `yeniGunID`, `yeniSaatID`, `yeniSinif`).

- Butonlar: Ekle, Sil ve Güncelle işlemlerini yapacak butonlar eklenmiş (`addDersHoca()`, `deleteDersHoca()`, `updateDersHoca()`).

3. Fonksiyonlar:

3.1. addDersHoca() Fonksiyonu:

- Bu fonksiyon, kullanıcının girdiği ders ve hoca bilgilerini sunucuya eklemek için kullanılır.

- `fetch` API'si kullanılarak sunucuyla iletişim kurulur ve `/add-endpoint` endpoint'i çağrılır.

- HTTP POST isteği gönderilir ve isteğin başlık bilgileri "application/json" olarak belirlenir.

- Kullanıcının girdiği bilgiler JSON formatında bir objeye dönüştürülerek isteğin gövdesine eklenir.

- Sunucudan gelen cevap JSON olarak alınır ve başarılı veya başarısız olma durumuna göre kullanıcıya bir bildirim gösterilir.

3.2. deleteDersHoca() Fonksiyonu:

- Bu fonksiyon, kullanıcının girdiği DersHocaID'ye göre bir ders ve hocayı silmek için kullanılır.

- `fetch` API'si kullanılarak sunucuyla iletişim kurulur ve `/delete-endpoint` endpoint'i çağrılır.

- HTTP POST isteği gönderilir ve isteğin başlık bilgileri "application/json" olarak belirlenir.

- Kullanıcının girdiği DersHocaID bilgisi JSON formatında bir objeye dönüştürülerek isteğin gövdesine eklenir.

- Sunucudan gelen cevap JSON olarak alınır ve başarılı veya başarısız olma durumuna göre kullanıcıya bir bildirim gösterilir.

3.3. updateDersHoca() Fonksiyonu:

- Bu fonksiyon, kullanıcının girdiği bilgilerle belirli bir ders ve hocanın bilgilerini güncellemek için kullanılır.

- `fetch` API'si kullanılarak sunucuyla iletişim kurulur ve `/update-endpoint` endpoint'i çağrılır.

- HTTP POST isteği gönderilir ve isteğin başlık bilgileri "application/json" olarak belirlenir.

- Kullanıcının girdiği bilgiler JSON formatında bir objeye dönüştürülerek isteğin gövdesine eklenir.

- Sunucudan gelen cevap JSON olarak alınır ve başarılı veya başarısız olma durumuna göre kullanıcıya bir bildirim gösterilir.

3.4. DOMContentLoaded Event Listener:

- Bu event listener, sayfanın yüklenmesini bekler.

- Sayfa yüklendiğinde, sunucudan alınan program takvimini (`/get-schedule` endpoint'i) sayfada gösterir.

- Fetch API kullanılarak sunucudan gelen HTML içeriği, `haftaTablosu` id'li HTML elementine eklenir.

Bu fonksiyonları kullanabilmeniz için sunucu tarafında uygun endpoint'lerin ve işlemlerin tanımlanmış olması gerekmektedir. Ayrıca, sunucu tarafında veritabanı işlemleri yapacak kodların ve güvenlik önlemlerinin yer alması önemlidir.

4. Program Takvimi Tablosu:

- Bir HTML tablosu (`<table>`) oluşturulmuş.

- Tablonun başlık satırı (`<tr>`) günleri temsil eder.

- JavaScript kodu sayesinde, sunucudan alınan program takvimi (`/get-schedule` endpoint'i) bu tablonun içine yerleştirilir.

B**) app.py in açıklanması:**

Bu alandaki kodlar, bir MySQL veritabanındaki ders-hoca ilişkilerini graf oluşturan bir uygulamayı içeriyor. İlgili MySQL veritabanındaki tablolardan alınan verilerle bir NetworkX grafi oluşturuluyor. Daha sonra, özel bir renklendirme stratejisi kullanılarak, grafa renk ataması yapılıyor ve bu renklendirme Matplotlib kullanılarak görselleştiriliyor.

1. Graf Oluşturma:

- Daha önceki örnekte olduğu gibi, `networkx` kütüphanesi kullanılarak bir graf (`G`) oluşturulur.

- `DersHocalar` tablosundan alınan verilerle graf düğümleri oluşturulur.

- Kontrol fonksiyonu (`is\_valid\_connection`) kullanılarak, grafa yeni bir düğüm eklenirken belirli kısıtların sağlanıp sağlanmadığı kontrol edilir.

- Eğer kısıtlar sağlanıyorsa, düğüm ve kenar eklenir.

2. Renklendirme İşlemi:

- Önceki örnekteki gibi, düğümlere renk atama işlemi yapılır.

- Bu kez, renk atama işlemi için özel bir fonksiyon (`assign\_color`) kullanılır.

- Bu fonksiyon, her düğüm için mevcut komşu düğümlerin renklerini kontrol eder ve uygun bir renk atar.

3. Grafi Görselleştirme:

- Grafi Matplotlib ile görselleştirmek için `nx.draw` fonksiyonu kullanılır.

- Düğümler, etiketler ve renkler belirli bir düzende görselleştirilir.

Kodun ana mantığı, derslerin gün ve saatlere göre ilişkilendirildiği bir graf oluşturmak ve bu grafi görselleştirmektir. Renklendirme işlemi, derslerin birbirleriyle çakışmadan gün ve saatlere yerleştirilmesini sağlamak için yapılır.

Graf Teorisi Hakkında Kısa Bilgi:

Graf teorisi, düğümler (nodes) ve bunları birbirine bağlayan kenarlar (edges) aracılığıyla ilişkili yapıları inceleyen matematiksel bir disiplindir. Bir graf, bu düğüm ve kenarların bir araya gelmesiyle oluşur. Graf teorisi, ağlar, iletişim ağları, sosyal ağlar gibi birçok alanda uygulama bulur. Temel terimler arasında düğüm derecesi, bağlılık, yol, çevre, altgraf ve bağlı bileşen gibi kavramlar yer alır.

C**) index.py’ ın fonksiyonlarının açıklanması:**

1. connect:

- Bu fonksiyon, MySQL veritabanına bağlanmayı sağlar.

- `mysql.connector` kütüphanesini kullanarak belirtilen parametrelerle bir bağlantı oluşturur.

- Eğer bağlantı başarılı olursa, bağlantı nesnesini döner.

- Hata durumunda `mysql.connector.Error` sınıfından bir hata döner.

2. find\_id\_by\_name:

- Veritabanında bir tablodan bir öğenin ID'sini bulmak için kullanılır.

- `table\_name` parametresi tablonun adını, `name` parametresi ise aranan öğenin adını temsil eder.

- Örneğin, `find\_id\_by\_name("ders", "Matematik")` çağrısı ile "Matematik" dersinin ID'sini bulabilirsiniz.

- Eğer öğe bulunursa, ilgili ID'yi döner. Bulunamazsa `None` döner.

- Hata durumunda `mysql.connector.Error` sınıfından bir hata döner.

3. add:

- Yeni bir ders-hoca ilişkisi eklemek için kullanılır.

- Verilen ders adı, hoca adı, gün adı, saat adı ve sınıf bilgilerini kullanarak ilişki ekler.

- Kısıtlara uygunluk kontrolü yapar. Eğer kısıtlar sağlanıyorsa ilişki eklenir, aksi halde eklenmez.

- Hata durumunda `mysql.connector.Error` sınıfından bir hata döner.

4. delete:

- Belirli bir ders-hoca ilişkisini silmek için kullanılır.

- Verilen `DersHocaID` parametresi ile ilgili ilişkiyi siler.

- Hata durumunda `mysql.connector.Error` sınıfından bir hata döner.

5. update:

- Varolan bir ders-hoca ilişkisini güncellemek için kullanılır.

- İlgili kısıtların sağlanıp sağlanmadığını kontrol eder ve güncelleme işlemini gerçekleştirir.

- Hata durumunda `mysql.connector.Error` sınıfından bir hata döner.

6. check\_constraints:

- Belirli kısıtların sağlanıp sağlanmadığını kontrol eder.

- Eklemek ya da güncellemek istenilen ders-hoca ilişkisinin belirli gün ve saatte başka bir sınıfta dersi olup olmadığını kontrol eder.

- Hata durumunda `mysql.connector.Error` sınıfından bir hata döner.

7. get\_schedule\_from\_database:

- Veritabanından ders programını çekmek için kullanılır.

- Ders programını sınıf bazında gün ve saat bilgileriyle bir sözlük olarak döndürür.

- Hata durumunda `mysql.connector.Error` sınıfından bir hata döner.

8. Flask Uygulamasının Endpoint'leri:

- \*\*index:\*\*

- Ana sayfa. `index.html` sayfasını render eder.

- \*\*add\_endpoint:\*\*

- POST isteğiyle çağrıldığında, veritabanına yeni bir ders-hoca ilişkisi ekler.

- \*\*delete\_endpoint:\*\*

- POST isteğiyle çağrıldığında, belirtilen `DersHocaID`'ye sahip ders-hoca ilişkisini siler.

- \*\*update\_endpoint:

- POST isteğiyle çağrıldığında, belirtilen verilerle ders-hoca ilişkisini günceller.

- get\_schedule:

- GET isteğiyle çağrıldığında, ders programını HTML tabloları şeklinde döndürür.

Bu fonksiyonlar, Flask uygulamasının temel işlevselliğini sağlar. Ders-hoca ilişkilerini eklemek, silmek, güncellemek ve ders programını görüntülemek için kullanılır. Flask uygulaması, web arayüzü üzerinden bu işlemleri gerçekleştirmek üzere tasarlanmıştır.

Kaynakça

1. <https://medium.com/kodlayan-nesil/flask-nedir-9364c1bb5f41>
2. [https://feyzaozen.medium.com/graf-teorisi-2-graf-renklendirme-welsh-powell-algoritmas%C4%B1-205087fa8aa](https://feyzaozen.medium.com/graf-teorisi-2-graf-renklendirme-welsh-powell-algoritmas%2525C4%2525B1-205087fa8aa)
3. <https://en.wikipedia.org/wiki/SQL>
4. <https://python.sitesi.web.tr/python-fonksiyonlar.html>