IUC, Bilgisayar Müh. Böl., Veritabanı Sistemleri, Güz 2020, Atakan Kurt
Ödev 1: İlişkisel Cebir ve SQL
Teslim tarihi: 9.12.2020 24:00
Teslim yeri: AKSIS
Konu: İlişkisel Cebir ve SQL

KURALLAR:

1. Ödevde 10 soru olup, her soru 1 puandır. Bu ödevin dönem notuna etkisi %10 dur. Yani alınan her puan dönem notuna 1 puan olarak yansır.
2. Ödevde kopya çekmek disiplin suçudur. Ödev de bir sorunun dahi kopya olması durumunda ödevin tamamı (kopya çeken ve kopya veren ayırımı yapılmaksızın) kopya sayılacak olup, ayrıca not olarak -10 puan verilecektir. Yani dönem sonu toplam notunuzdan 10 puan silinecektir. Disiplin soruşturmasında ceza almanız durumunda dersten doğrudan kalacaksınız.
3. Ödevin teslim tarihi. Eposta ile ödev sadece kabul edilmeyecektir.
4. Geç ödev teslimi yapılmayacaktır. Ödevden haberi olmamak mazeret sayılmaz.
5. Bu ödevi PostgreSQL veya MySQL veritabanında yapabilirsiniz. Başka veritabanında yapılan ödevler kabul edilmez.
6. Veritabanını oluşturmak için aşağıdaki CREATE TABLE komutlarını kullanınız. Tabloların içine kendi INSERT komutlarınızla her sorguda en az bir kayıt döndürecek şekilde rastgele kayıtlar ekleyiniz.
7. Sorguları gereksiz yere karmaşık yazmayınız. İhtiyaç yoksa yada soruda talep edilmiyorsa cevapta alt sorgu kullanmayınız. İhtiyaç duyulmayan tabloları sorgularda gereksiz yere kullanmayınız. Yani: o tablo olmadan sorgu doğru çalışıyorsa, sorguda o tabloya ihtiyaç yok demektir. Bazı sorgularda özellikle kullanılması istenen komut ve operatörler varsa onları kullanarak soruları cevaplayınız. Aykırı durumlarda not kırılacaktır.
8. Ara/geçici tablolara, viewlere (sanal tablolara) ve sütun/alan adlarına anlamlı isimler veriniz.
9. **Ödevler PDF olarak teslim edilecektir.**
10. **İlişkisel cebir sorularını ya elle yazıp, resmini çekip dosyaya resim olarak ekleyiniz ve MSWORDde düzgün bir şekilde (yani doğru Yunanca semboller ve subcript/superscript kullanarak, yada matematiksel formul seçeneğini kullanarak)**
11. **SQL içinHer sorguyu ve sorgunun sonucunu (döndürülen kayıtları) kopyala-yapıştır ile veya başka bir şekilde bir metin veya MS Word dosyasına aktarınız.** Bu dosyada
    a. **Her soru için soru numarası, soru metni, sorgu ve VTYSnin döndürdüğü kayıtlar yani sorgu sonuçu bulunmalıdır. Sorgunuzu ve alt sorgularınızı yazarken kolay okunabilmesi için indentation / hızalama kullanınız. SELECT/FROM/WHERE/GROUP BY/HAVING/ORDER BY/… cümleciklerini farklı satırlarda ve alt alta gelecek şekilde düzgün olarak yazınız. Sorgularda anahtar kelimeler büyük harfle, tablo ve alan adları küçük harfle yazılı olmalıdır.**
    b. Dosya: VT-NNNNNNNN-HWX. şeklinde isimlendirilmelidir. NNNNNNNN: öğrenci-no, X: ödev-no, ext: dosya formatı (doc, docx, txt veya pdf) Dosyanın en başında dersin kodu, adı, öğrenci numaranız, adınız ve ödev numarası (1. ödev), hangi veritabanı sistemi ile yapıldığı yazılı olmalıdır. Eğer grup olarak yaptıysanız gruptakilerin numara ve adları olmalıdır. Ayrıca ‘gündüz öğretimi’, ‘akşam öğretimi’ diye belirtiniz.

Student (sid, name, did, avgGrade) // ogrenci(ogrenci-no, adi, bolum-no, ogrenciNotOrtalaması)
Take (sid, cid, grade) // ders-al(ogrenci-no, ders-kodu, notu)
Course (cid, title, credits, did, avgGrade)// ders(ders-kodu, adi, kredisi, bolum-no, dersNotOrtalaması)
Department (did, name, avgGrade) // bolum(bolum-no, adi, bolumNotOrtalaması)
Teacher (tid, name, did, avgGrade) // hoca(hoca-no, adi, bolum-no, dersSayisi)
Teach (tid, cid) // ders-ver(hoca-no, ders-kodu)

Yukarıdaki veritabanı şemasını kullanarak aşağıdaki soruları ilişkisel cebir ve SQL ile yazınız. Veritabanını oluşturmak için en aşağıda verilen CREATE TABLE komutlarını kullanınız. SQL sorgu ve cevaplarını MySQL veya PostgreSQLde çalıştırarak KURALLARda açıklandığı şekilde veriniz.

1. (10) 'Ali KURT' (name) adlı öğrencinin sid'sini ve notlarını (grade) listeleyiniz.
2. (10) 'Ayşe KURT' (name) adlı öğrencinin aldığı, fakat 'Ali KURT' adlı öğrencinin almadığı derslerin kayıtlarını (yani course tablosunun tüm sütunlarını) listeleyiniz. (EXCEPT kullanınız gerekiyor)
3. (10) Öğrencilerin sid'lerini ve aldıkları derslerin sayısını, not ortalamasını, en yüksek ve en düşük notlarını listeleyiniz.
4. Bölümlerin did'leri, öğrenci sayılarını öğrenci syılarına göre azalan sırada listeleyiniz (İlişkisel cebirle yazmayınız)
5. (10) 2'den fazla ders veren hocaların sid'leri, verdikleri ders sayısı ve derslerini alan öğrencilerin sayılarını listeleyiniz.
6. (10) 'Bilgisayar Müh' (department.name) adlı bölümdeki öğrencilerden 'Elektrik Müh' (department.name) adlı bölümdeki derslerden *alan*larının (take tablosunu kullan) kayıtlarını (student tablosundaki tüm alanları listele) listeleyiniz.
7. Her dersteki öğrenci sayılarının ortalamalarını (take tablosundan her dersi kaç öğrencinin aldığı bulunacak, sonra da bu sayıların ortalamaları bulunacak) bulup, bu ortalamadan daha fazla öğrencisi olan derslerin kayıtlarını listeleyiniz. (önce ortalamadan daha fazla öğrencisi olan derslerin cid'leri bulunacak, sonra bu cid'lerden yola çıkarak course tablosundaki ders kayıtları bulunacak)
8. GROUP BY kullanmadan iki farklı ders alan öğrencilerin kayıtlarını listeleyiniz. (2 farklı ders alan dendiği için take tablosunun 2 defa kullanılması gerekiyor! Sınıfta örnek yapmıştık. Slidelarda da örnek var. NOT: önce bu öğrencilerin sid'leri bir (alt) sorgu ile bulunacak sonra bu sid'ler üzerinden student tablosundaki kayıtlara yani tüm sütunlara ulaşılacak)
9. Hiç ders vermeyen (take tablosunda bu hocaya ait kayıt yok demektir) hocaları listeleyiniz.
10. Ders veren hocaların kayıtlarını (teacher tablosundaki tüm sütunları) listeleyiniz. (Yani take tablosunda tid geçen tüm hocalar)

MYSQL için:

```
create table Department
 (did integer(5) not null,
 dname varchar(30) not null,
 comments varchar(50),
 email varchar(30),
 primary key(did));

create table Student
 (sid integer(5) not null,
 fname varchar(30) not null,
```

```
 lname varchar(30),
 birthdate date,
 birthplace varchar(50),
 did integer(5),
 foreign key (did) references Department(did),
 primary key(sid));

create table Course
 (cid integer(5) not null,
 title varchar(30) not null,
 description varchar(50),
 credits integer(2),
 did  integer(5),
 foreign key (did) references Department(did),
 primary key(cid));

create table Teacher
 (tid integer(5) not null,
 fname varchar(30) not null,
 lname varchar(30),
 birthdate date,
 birthplace varchar(50),
 did integer(5),
 foreign key (did) references Department(did),
 primary key(tid));

create table Take
 (sid integer(5) not null,
 cid integer(5) not null,
 grade float,
 foreign key (sid) references Student(sid),
 foreign key (cid) references Course(cid),
 primary key (sid,cid));

create table Teach
 (tid integer(5) not null,
 cid integer(5) not null,
 foreign key (tid) references Teacher(tid),
 foreign key (cid) references Course(cid),
 primary key (tid,cid));
```

PostgreSQL için:

```
create table Student
 (sid integer not null,
 fname varchar(30) not null,
 lname varchar(30),
 birthdate date,
 birthplace varchar(50),
 did integer,
 foreign key (did) references Department(did),
```

```
 primary key(sid));

create table Course
 (cid integer not null,
 title varchar(30) not null,
 description varchar(50),
 credits integer,
 did  integer,
 foreign key (did) references Department(did),
 primary key(cid));

create table Teacher
 (tid integer not null,
 fname varchar(30) not null,
 lname varchar(30),
 birthdate date,
 birthplace varchar(50),
 did integer,
 foreign key (did) references Department(did),
 primary key(tid));

create table Take
 (sid integer not null,
 cid integer not null,
 grade float,
 foreign key (sid) references Student(sid),
 foreign key (cid) references Course(cid),
 primary key (sid,cid));

create table Teach
 (tid integer not null,
 cid integer not null,
 foreign key (tid) references Teacher(tid),
 foreign key (cid) references Course(cid),
 primary key (tid,cid));
```