YILDIZ TEKNİK ÜNİVERSİTESİ

İlişkisel Cebir II ve SQL'e Giriş

Öğr. Gör. Dr. Yasemin Topuz

Yıldız Teknik Üniversitesi

BLM3041 – Veritabanı Yönetimi

17 Ekim 2025 | YTÜ Bilgisayar Mühendisliği

Neler konuşacağız?

- İlişkisel Cebir II
- Sorgu Ağaçları (Query Tree)
- SQL (Structured Query Language)
- Veri Tipleri ve Operatörler
- DDL (Data Defination Language)
- DML (Data Manupulation Language)
- SQL Veri ve Şema Tanımlamaları, Kısıtlar

İlişkisel Cebir – Division İşlemi

- İki bağıntıya uygulanan bir işlemdir.

(a)

SSN_PNOS

Essn	Pno
123456789	1
123456789	2
666884444	3
453453453	1
453453453	2
333445555	2
333445555	3
333445555	10
333445555	20
999887777	30
999887777	10
987987987	10
987987987	30
987654321	30
987654321	20
888665555	20

SMITH_PNOS

Pno
1
2

SSN_PNOS ÷ SMITH_PNOS

SSNS

Ssn
123456789
453453453

(b)

R

A	B
a1	b1
a2	b1
a3	b1
a4	b1
a1	b2
a3	b2
a2	b3
a3	b3
a4	b3
a1	b4
a2	b4
a3	b4

S

A
a1
a2
a3

R ÷ S

T

B
b1
b4

İlişkisel Cebir – Aggregate İşlemi

- Çeşitli fonksiyonların (matematiksel işlemlerin) kullanımı sonucu kümelenen değerlerin oluşturulması aşamasında uygulanır.

  - $\mathcal{F}_{MAX\ Salary}$ (Employee)
  - $\mathcal{F}_{MIN\ Salary}$ (Employee)
  - $\mathcal{F}_{AVG\ \ Salary}$ (Employee)
  - $\mathcal{F}_{COUNT\ SSN}$ (Employee)
  - $\mathcal{F}_{COUNT\ SSN,\ AVG\ Salary}$ (Employee)

İlişkisel Cebir – Group By İşlemi (with Aggregate)

- İki değere göre bir seçim gerçekleştirmek istiyoruz.

  - $DNO\ \mathcal{F}\ COUNT\ SSN, AVG\ Salary$(Employee)

Fname	Minit	Lname	Ssn	...	Salary	Super_ssn	Dno
John	B	Smith	123456789		30000	333445555	5
Franklin	T	Wong	333445555		40000	888665555	5
Ramesh	K	Narayan	666884444		38000	333445555	5
Joyce	A	English	453453453	...	25000	333445555	5
Alicia	J	Zelaya	999887777		25000	987654321	4
Jennifer	S	Wallace	987654321		43000	888665555	4
Ahmad	V	Jabbar	987987987		25000	987654321	4
James	E	Bong	888665555		55000	NULL	1

Grouping EMPLOYEE tuples by the value of Dno

Dno	Count_ssn	Average_salary
5	4	33250
4	3	31000
1	1	55000

Count_ssn	Average_salary
8	35125

İlişkisel Cebir – Örnekler

STUDENT (Sid, SName, GradYear, MajorId)

SELECT

- 2024 yılında mezun olan öğrencileri listele

$$\text{Q1: } \sigma_{GradYear=2024}\ (STUDENT)$$

- 2024 yılında bölüm numarası 10 veya 20 olan bölümlerden mezun olan öğrencileri listele

$$\text{Q2: } \sigma_{GradYear=2024\ \wedge\ (MajorId=10\ \vee\ MajorId=20)}\ (STUDENT)$$

$$\text{Q3: } \sigma_{MajorId=10\ or\ MajorId=20)}\ (\ \sigma_{GradYear=2024}\ (STUDENT))$$

İlişkisel Cebir – Örnekler

STUDENT (Sid, SName, GradYear, MajorId)

PROJECT

- Tüm öğrencilerin isimlerini ve mezuniyet yıllarını listele

$$\text{Q4: } \Pi_{SName,\, GradYear}\ (STUDENT)$$

- Bölüm numarası 10 olan tüm öğrencilerin isimlerini listele

$$\text{Q5: } \Pi_{SName}(\ \sigma_{MajorId=10}\ (STUDENT))$$

- Aşağıdaki sorgu için ne dersiniz?

$$\text{Q6: } \sigma_{MajorId=10}\ (\Pi_{SName}\ (STUDENT))$$

İlişkisel Cebir – Örnekler

STUDENT (Sid, SName, GradYear, MajorId)

SORT (S), RENAME (*P*), EXTEND (E)

- Tüm öğrencileri önce mezuniyet yılına sonra da isimlerini göre sırala

$$\text{Q7: } S_{GradYear,\, SName}\ (STUDENT)$$

- Bir önceki sorgudaki SName alanının (kolunun) adını **CSMajors** olarak değiştirme

$$\text{Q8: } P_{SName,CSMajors}\ (\text{Q7})$$

- Öğrenci tablosuna ilk mezun olan öğrenciden (2001) ilgili öğrencinin mezun olduğu yıla kadar kaç yıl geçtiğini hesaplayarak oluşturulan **GradClass** kolunu ekleme

$$\text{Q9: } E_{GradYear\ -2001,\, GradClass}\ (STUDENT)$$

İlişkisel Cebir – Örnekler

STUDENT (Sid, SName, GradYear, MajorId)

GROUP BY

- Bölümlere göre maksimum ve minimum mezuniyet yıllarını listeleme

Q10: $MajorId\ \mathcal{F}\ MIN(GradYear), MAX\ (GradYear)\ (STUDENT)$

SId	SName	GradYear	MajorId
1	joe	2004	10
2	amy	2004	20
3	max	2005	10
4	sue	2005	20
5	bob	2003	30
6	kim	2001	20
7	art	2004	30
8	pat	2001	20
9	lee	2004	10

MajorId	MinOfGradYear	MaxOfGradYear
10	2004	2005
20	2001	2005
30	2003	2004

İlişkisel Cebir – Örnekler

STUDENT (Sid, SName, GradYear, MajorId)

GROUP BY

- Mezuniyet yılına göre her bölümden kaç öğrencinin mezun olduğunu bulunuz.

Q11: $_{MajorId,GradYear}\mathcal{F}_{COUNT\ (SId)}\ (STUDENT)$

SId	SName	GradYear	MajorId
1	joe	2004	10
2	amy	2004	20
3	max	2005	10
4	sue	2005	20
5	bob	2003	30
6	kim	2001	20
7	art	2004	30
8	pat	2001	20
9	lee	2004	10

MajorId	GradYear	CountOfSId
10	2004	2
10	2005	1
20	2001	2
20	2004	1
20	2005	1
30	2003	1
30	2004	1

İlişkisel Cebir – Örnekler

STUDENT (Sid, SName, GradYear, MajorId)

GROUP BY

- Herhangi bir öğrencinin minimum mezuniyet yılını bulunuz.

$$\text{Q12: } \mathcal{F}_{MIN\ (GradYear)}\ (STUDENT)$$

SId	SName	GradYear	MajorId
1	joe	2004	10
2	amy	2004	20
3	max	2005	10
4	sue	2005	20
5	bob	2003	30
6	kim	2001	20
7	art	2004	30
8	pat	2001	20
9	lee	2004	10

- Bir bölümü olan tüm öğrencilerin sayısını bulunuz.

$$\text{Q13: } \mathcal{F}_{COUNT\ (MajorId)}\ (STUDENT)$$

- Öğrencilerin okuduğu kaç farklı bölüm olduğunu bulunuz.

$$\text{Q14: } \mathcal{F}_{COUNTDISTINCT\ (MajorId)}\ (STUDENT)$$

İlişkisel Cebir – Örnekler

GROUP BY

ENROLL (EId, StudentId, SectionId, grade)

EId	StudentId	SectionId	Grade
14	1	13	A
24	1	43	C
34	2	43	B+
44	4	33	B
54	4	53	A
64	6	53	A

- En çok «A» notu alınan section'da kaç tane «A» notu alındığını bulunuz.

Q15: $\sigma_{Grade='A'}(ENROLL)$

EId	StudentId	SectionId	Grade
14	1	13	A
54	4	53	A
64	6	53	A

Q16: ${}_{SectionId}\mathcal{F}_{COUNT\,(EId)}\,(Q16)$

SectionId	CountOfEId
13	1
53	2

Q17: $\mathcal{F}_{MAX\,(CountOfEId)}$ (Q17)

MaxOfCountOfEId
2

İlişkisel Cebir – Örnekler

STUDENT (Sid, SName, GradYear, MajorId)
SECTION (SectId, CourseId, Prof, YearOfRead)

UNION

- Tüm öğrencilerin ve hocaların isimlerini bulunuz.

Q18: $P_{SName,Person}\ (\Pi_{SName}\ (STUDENT))$

Q19: $P_{Prof,\ Person}\ (\Pi_{Prof}\ (SECTION))$

Q20 = Q18 ∪ Q19

İlişkisel Cebir – Örnekler

STUDENT	SId	SName	GradYear	MajorId
	1	joe	2004	10
	2	amy	2004	20
	3	max	2005	10
	4	sue	2005	20
	5	bob	2003	30
	6	kim	2001	20
	7	art	2004	30
	8	pat	2001	20
	9	lee	2004	10

DEPT	DId	DName
	10	compsci
	20	math
	30	drama

PRODUCT

- Öğrencilerin ve bölümlerin tüm olası çarpımlarını bulunuz.

  Q21: $STUDENT \times DEPT$

- Tüm öğrencileri ve onların okuduğu bölüm isimlerini bulunuz

  Q22: $\sigma_{MajorId=DId}\ (STUDENT \times DEPT)$

SId	SName	MajorId	GradYear	DId	DName
1	joe	10	2004	10	compsci
2	amy	20	2004	10	compsci
3	max	10	2005	10	compsci
4	sue	20	2005	10	compsci
5	bob	30	2003	10	compsci
6	kim	20	2001	10	compsci
7	art	30	2004	10	compsci
8	pat	20	2001	10	compsci
9	lee	10	2004	10	compsci
1	joe	10	2004	20	math
2	amy	20	2004	20	math
3	max	10	2005	20	math
4	sue	20	2005	20	math
5	bob	30	2003	20	math
6	kim	20	2001	20	math
7	art	30	2004	20	math
8	pat	20	2001	20	math
9	lee	10	2004	20	math
1	joe	10	2004	30	drama
2	amy	20	2004	30	drama
3	max	10	2005	30	drama
4	sue	20	2005	30	drama
5	bob	30	2003	30	drama
6	kim	20	2001	30	drama
7	art	30	2004	30	drama
8	pat	20	2001	30	drama
9	lee	10	2004	30	drama

İlişkisel Cebir – Örnekler

STUDENT (Sid, SName, GradYear, MajorId)
DEPT (DId, DName)
SECTION (SectId, CourseId, Prof, YearOfRead)
ENROLL (EId, StudentId, SectionId, grade)

JOIN

- Tüm öğrencileri ve onların bölüm isimlerini bulunuz.

Q23: $STUDENT \bowtie_{MajorId=DId} DEPT)$

- 2004 yılında «Joe» isimli öğrencilerin aldığı notları bulunuz.

Q24: $\sigma_{SName=«Joe»}\ (STUDENT\ )$

Q25: $(Q24 \bowtie_{SId=StudentId} ENROLL)$

Q26: $\sigma_{YearOfRead=2004}\ (SECTION\ )$

Q27: $(Q25 \bowtie_{SectionId=SectId} Q26)$

Q28: $\Pi_{Grade}\,(Q27)$

İlişkisel Cebir – Örnekler

STUDENT (Sid, SName, GradYear, MajorId)
SECTION (SectId, CourseId, Prof, YearOfRead)
ENROLL (EId, StudentId, SectionId, grade)

JOIN

- En çok «A» notunun verildiği Section'ı bulalım.

  Q29: $(Q17 \bowtie_{MaxCountOfEId=CountOfEId} Q16)$

  Q30: $(Q29 \bowtie_{SectionId=SectId} SECTION)$

- «Joe» ile aynı bölümde okuyan öğrencileri bulunuz.

  Q31: $\Pi_{MajorId}(\sigma_{SName=«Joe»}(STUDENT))$

  Q32: $P_{MajorId,JoesMajor}(Q31)$

  Q33: $(Q32 \bowtie_{JoesMajor=MajorId} STUDENT)$

Q15: $\sigma_{Grade='A'}(ENROLL)$

Q16: ${}_{SectionId}\mathcal{F}_{COUNT(EId)}(Q15)$

Q17: $\mathcal{F}_{MAX(CountOfEId)}$ (Q16)

İlişkisel Cebir – Örnekler

STUDENT (Sid, SName, GradYear, MajorId)
DEPT (DId, DName)
SECTION (SectId, CourseId, Prof, YearOfRead)
ENROLL (EId, StudentId, SectionId, grade)

SEMI-JOIN (EXISTS)

- En az 1 öğrencisi olan departmanları bulunuz.

  Q34: $(DEPT \ltimes_{DId=MajorId} STUDENT)$

- Benzer çözüm

  Q35: $(DEPT \bowtie_{DId=MajorId} STUDENT)$ Q36: $_{DId,DName}\mathcal{F}\ (Q35)$

- «Einstein» dan ders alan öğrencileri bulalım.

  Q37: $\sigma_{Prof=«Einstein»}\ (SECTION)$

  Q38: $(ENROLL \ltimes_{SectionId=SectId}\ Q37)$

  Q39: $(STUDENT \ltimes_{SId=StudentId}\ Q38)$

İlişkisel Cebir – Örnekler

ANTI-JOIN (NOT EXISTS)

STUDENT (Sid, SName, GradYear, MajorId)
DEPT (DId, DName)
SECTION (SectId, CourseId, Prof, YearOfRead)
ENROLL (EId, StudentId, SectionId, grade)

- Hiç öğrencisi olmayan departmanları listeleyiniz.

  Q40: $(DEPT \triangleright_{DId=MajorId} STUDENT)$

- Hiç «F» notu alınmayan dersleri bulunuz.

  Q41: $\sigma_{grade=«F»}(ENROLL)$

  Q42: $(SECTION \triangleright_{SecId=SectionId} Q41)$

- Üstteki soru için aşağıdaki çözüm olur mu?

  Q43: $(\sigma_{grade<>«F»}(ENROLL) \bowtie_{SectionId=SecId} SECTION)$

İlişkisel Cebir – Örnekler

SECTION (SectId, CourseId, Prof, YearOfRead)
ENROLL (EId, StudentId, SectionId, grade)

ANTI-JOIN (NOT EXISTS)

- Hiç «F» notu vermeyen hocaları bulunuz.

Q44: $(\sigma_{grade=«F»}\ (ENROLL\ )$

Q45: $(SECTION \ltimes_{SecId=SectionId}\ Q44)$

Q46: $P_{Prof,BadProf}\ (Q45)$

Q47: $(SECTION \triangleright_{Prof=BadProf}\ Q46)$

Q48: $_{Prof}\mathcal{F}\ (Q47)$

İlişkisel Cebir – Örnekler

STUDENT (Sid, SName, GradYear, MajorId)
ENROLL (EId, StudentId, SectionId, grade)

OUTER-JOIN

- Tüm öğrenciler için ayrı ayrı aldığı farklı notların sayısını bulunuz.

Q50: $(STUDENT \bowtie_{SId=StudentId} ENROLL)$

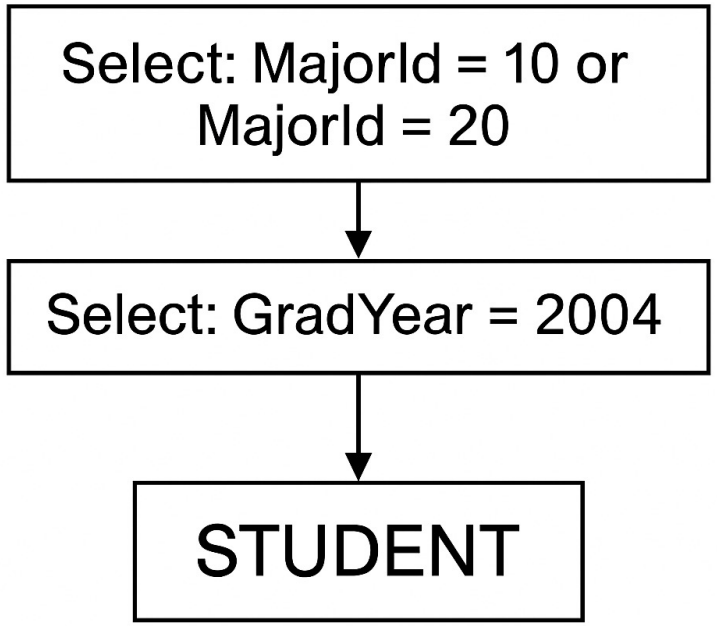
Q51: ${}_{SId,\,Grade}\mathcal{F}_{COUNT\,(EId)}\,(Q50)$

Query Tree (Sorgu Ağaçları)

- Sorgular veritabanında gerçeklenmeden önce sorgu ağacına dönüştürülür.
- Bir sorgunun içsel veri yapısını temsil eder.
- Avantajlar;
  - Hangi tabloları kullanıyoruz,
  - Hangi tabloları hangi özellikler üzerinden birbirlerine bağlıyoruz.
- Ağaç aşağıya doğru ne kadar uzarsa o kadar büyüklükte JOIN işlemi uygulayacağımızı gösterir.
- Sorgunun çalıştırılması için gerekli olan işlemlerin tespit edilmesi
  - Ara sonuçların üretilmesi
  - Optimizasyonların gerçeklenmesi
- Sorgu ağacındaki her bir node (düğüm); Select, project, join, rename, division vb.
- Yapraklar (lead) temel ilişkileri (tabloları) temsil eder.
- Bir sorgu ağacı sorgunun karmaşıklığının ve yapılan işlemlerin anlaşılabilmesi için güzel bir temsildir.

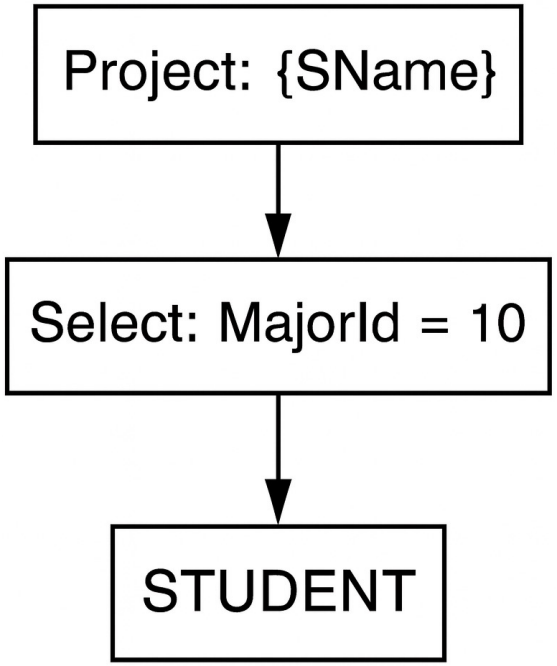
Query Tree (Sorgu Ağaçları)

- 2024 yılında bölüm numarası 10 veya 20 olan bölümlerden mezun olan öğrencileri listele
- .



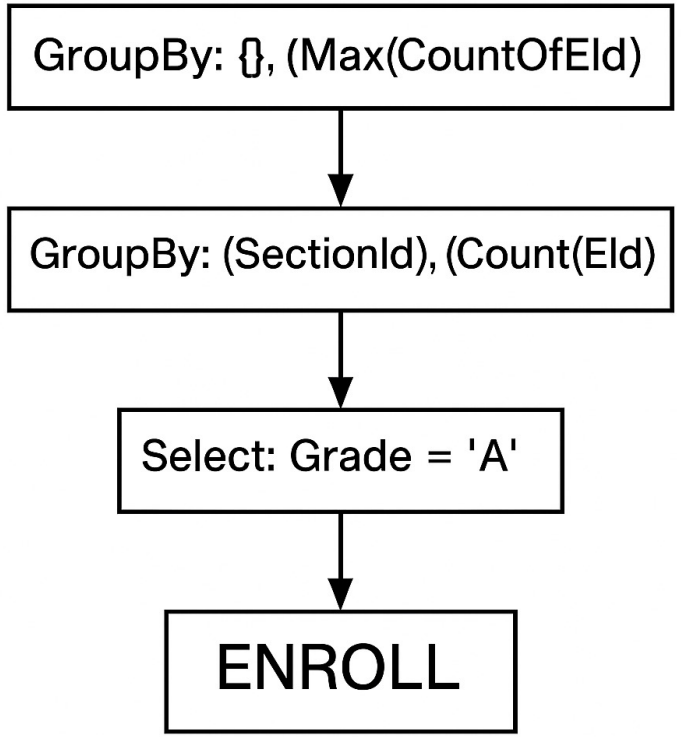
Query Tree (Sorgu Ağaçları)

- Bölüm numarası 10 olan tüm öğrencilerin isimlerini listele



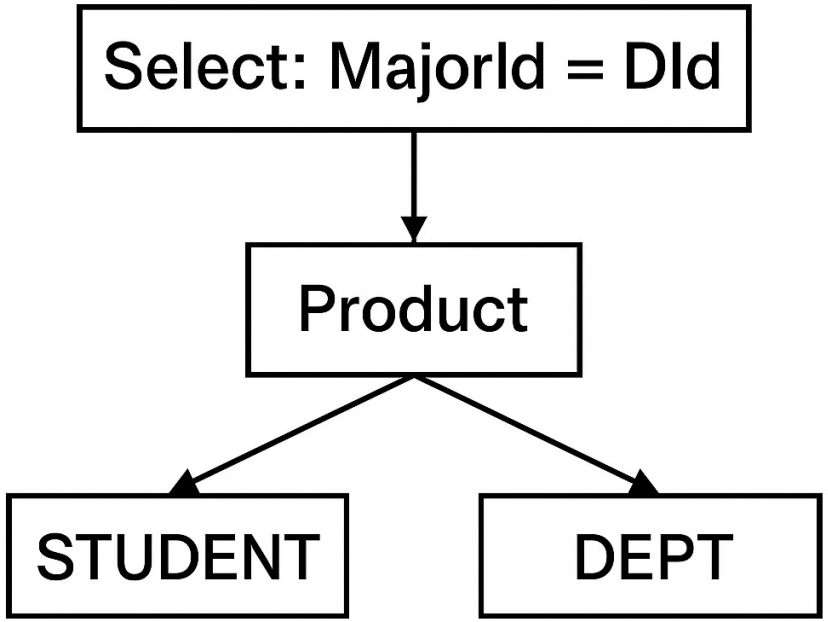
Query Tree (Sorgu Ağaçları)

- En çok «A» notu alınan derste kaç tane «A» notu alındığını bulunuz.



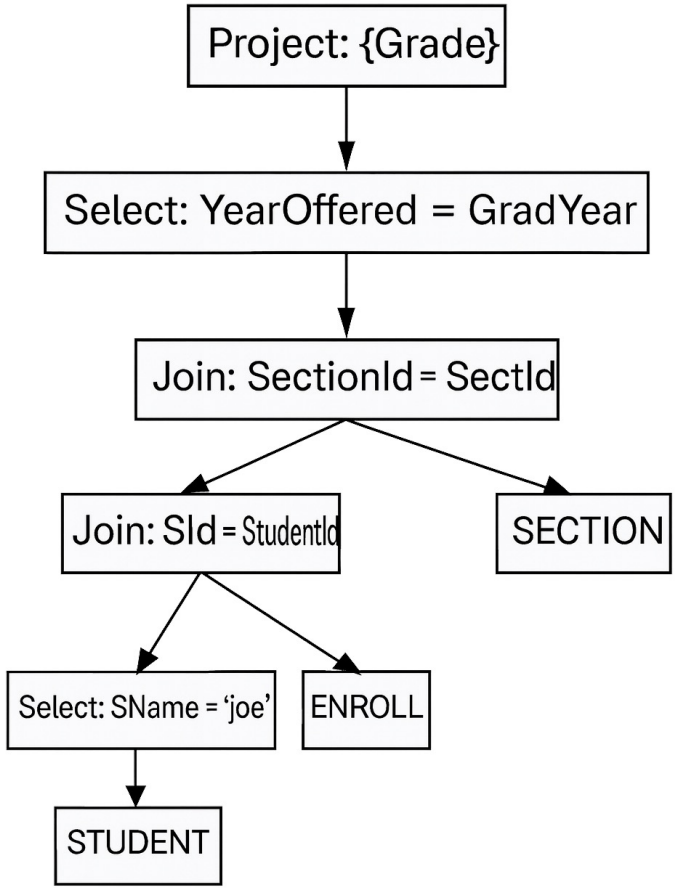
Query Tree (Sorgu Ağaçları)

- Tüm öğrencileri ve onların bölüm isimlerini bulunuz.



Query Tree (Sorgu Ağaçları)

- «Joe» isimli öğrencinin mezuniyet yılında aldığı derslerin notlarını bulunuz.



SQL (Structured Query Language)

- Tarihçe:
  - SQL1 @1989
  - SQL2 @1992
  - SQL3 @1999
- SQL İlişkisel Cebir'e ek olarak;
  - Ekleme (Insert), Silme (Delete), Güncelleme (Update)
  - Aritmetik İşlemler
  - Veri Tasarımı
  - Görev Atamaları
  - Kümeleme Fonksiyonları

SQL (Structured Query Language)

- **DML** — Veritabanından bilgi sorgulama; veritabanına kayıt (tuple/satır) ekleme, silme ve güncelleme olanağı sağlar.
- **Bütünlük (Integrity)** — DDL, bütünlük kısıtlarını (integrity constraints) belirtmek için komutlar içerir.
- **Görünüm Tanımı (View definition)** — DDL, görünümleri (views) tanımlamak için komutlar içerir.
- **İşlem Denetimi (Transaction control)** — işlemlerin başlangıç ve bitişini belirtmek için komutlar içerir.
- **Gömülü SQL ve Dinamik SQL** — SQL deyimlerinin genel amaçlı programlama dillerine nasıl gömüleceğini ve çalışma anında dinamik olarak nasıl oluşturulacağını tanımlar.
- **Yetkilendirme (Authorization)** — ilişkiler (tablolar) ve görünümler için erişim haklarını belirtmeye yarayan komutları içerir.

SQL Veri ve Şema Tanımlamaları, Kısıtlar

- **DDL (Data Definition Language)**
  - CREATE
  - DROP
  - ALTER
- **DML (Data Manipulation Language)**
  - INSERT
  - UPDATE
  - DELETE
- **Gelişmiş veri tanımları**
  - NOT NULL
  - PRIMARY KEY
  - UNIQUE
  - FOREIGN KEY

SQL Veri ve Şema Tanımlamaları, Kısıtlar

```
CREATE TABLE DEPARTMENT (
    Dname                   VARCHAR(15)             NOT NULL,
    Dnumber                 INTEGER                 NOT NULL,
    Mgr_ssn                 CHAR(9)                 NOT NULL DEFAULT '888665555',
    Mgr_start_date          DATE,

    CONSTRAINT DEPTPK
        PRIMARY KEY(Dnumber),
    CONSTRAINT DEPTSK
        UNIQUE(Dname),
    CONSTRAINT DEPTMGRFK
        FOREIGN KEY(Mgr_ssn) REFERENCES EMPLOYEE(Ssn)
                        ON DELETE SET DEFAULT
                        ON UPDATE CASCADE
);
```

SQL Veri ve Şema Tanımlamaları, Kısıtlar

```
CREATE TABLE EMPLOYEE (
    Fname           VARCHAR(15)         NOT NULL,
    Minit           CHAR,
    Lname           VARCHAR(15)         NOT NULL,
    Ssn             CHAR(9)             NOT NULL,
    Bdate           DATE,
    Address         VARCHAR(30),
    Sex             CHAR,
    Salary          DECIMAL(10,2),
    Super_ssn       CHAR(9),
    Dno             INTEGER             NOT NULL DEFAULT 1,

    CONSTRAINT EMPPK
        PRIMARY KEY(Ssn),
    CONSTRAINT EMPSUPERFK
        FOREIGN KEY(Super_ssn) REFERENCES EMPLOYEE(Ssn)
                ON DELETE SET NULL
                ON UPDATE CASCADE,
    CONSTRAINT EMPDEPTFK
        FOREIGN KEY(Dno) REFERENCES DEPARTMENT(Dnumber)
                ON DELETE SET DEFAULT
                ON UPDATE CASCADE
);
```

SQL Veri ve Şema Tanımlamaları, Kısıtlar

```
CREATE TABLE DEPT_LOCATIONS (
    Dnumber             INTEGER                 NOT NULL,
    Dlocation           VARCHAR(15)             NOT NULL,

    PRIMARY KEY(Dnumber, Dlocation),
    FOREIGN KEY(Dnumber) REFERENCES DEPARTMENT(Dnumber)
                        ON DELETE CASCADE
                        ON UPDATE CASCADE
);
```

SQL Veri ve Şema Tanımlamaları, Kısıtlar

```
CREATE TABLE DEPENDENT (
    Essn                CHAR(9)             NOT NULL,
    Dependent_name      VARCHAR(15)         NOT NULL,
    Sex                 CHAR,
    Bdate               DATE,
    Relationship        VARCHAR(8),

    PRIMARY KEY(Essn, Dependent_name),
    CONSTRAINT DEPENDEMPFK FOREIGN KEY(Essn) REFERENCES EMPLOYEE(Ssn)
        ON DELETE CASCADE ON UPDATE CASCADE
);
```

SQL Veri ve Şema Tanımlamaları, Kısıtlar

```
CREATE TABLE PROJECT (
    Pname                   VARCHAR(15)                 NOT NULL,
    Pnumber                 INTEGER                     NOT NULL,
    Plocation               VARCHAR(15),
    Dnum                    INTEGER                     NOT NULL,

    CONSTRAINT PROJPK
        PRIMARY KEY(Pnumber),
    CONSTRAINT PROJSK
        UNIQUE(Pname),
    CONSTRAINT PROJDEPTFK
        FOREIGN KEY(Dnum) REFERENCES DEPARTMENT(Dnumber)
                    ON DELETE CASCADE
                    ON UPDATE CASCADE
);
```

SQL Veri ve Şema Tanımlamaları, Kısıtlar

```
CREATE TABLE WORKS_ON (
    Essn                CHAR(9)             NOT NULL,
    Pno                 INTEGER             NOT NULL,
    Hours               DECIMAL(3,1),

    PRIMARY KEY(Essn, Pno),
    CONSTRAINT WORKSEMPFK
        FOREIGN KEY(Essn) REFERENCES EMPLOYEE(Ssn)
                ON DELETE CASCADE
                ON UPDATE CASCADE,
    CONSTRAINT WORKSPROJFK
        FOREIGN KEY(Pno) REFERENCES PROJECT(Pnumber)
                ON DELETE CASCADE
                ON UPDATE CASCADE
);
```

Tablolarda Güncellemeler

- **Insert (Ekle)**

```
INSERT INTO EMPLOYEE (Fname, Minit, Lname, Ssn, Bdate, Address, Sex, Salary, Super_ssn, Dno)
VALUES  ('James', 'E', 'Borg',   '888665555', '1937-11-10', '450 Stone, Houston, TX', 'M', 55000.00, NULL,        1),
        ('Franklin', 'T', 'Wong','333445555', '1955-12-08', '638 Voss, Houston, TX',   'M', 40000.00, '888665555', 5);
```

- **Delete (Sil)**
  - Çalışan ilişkisindeki tüm kayıtları (tuples) kaldır

```
DELETE FROM EMPLOYEE;
```

- **Update (Güncelleme)**

```
UPDATE EMPLOYEE SET dno = 1;
```

- **Drop Table (Tabloyu Kaldır)**

```
DROP TABLE EMPLOYEE;
```

Tablolarda Güncellemeler

- **Alter (Yapıyı Değiştir)**

```
ALTER TABLE [TABLO_ADI] (ADD/RENAME/ALTER/DROP COLUMN(opsiyonel)) [KOLON_ADI]
```

```
ALTER TABLE EMPLOYEE ADD COLUMN Email VARCHAR(50);
```

- İlişkideki mevcut tüm kayıtlar için yeni özniteliğin değeri NULL olarak atanır.

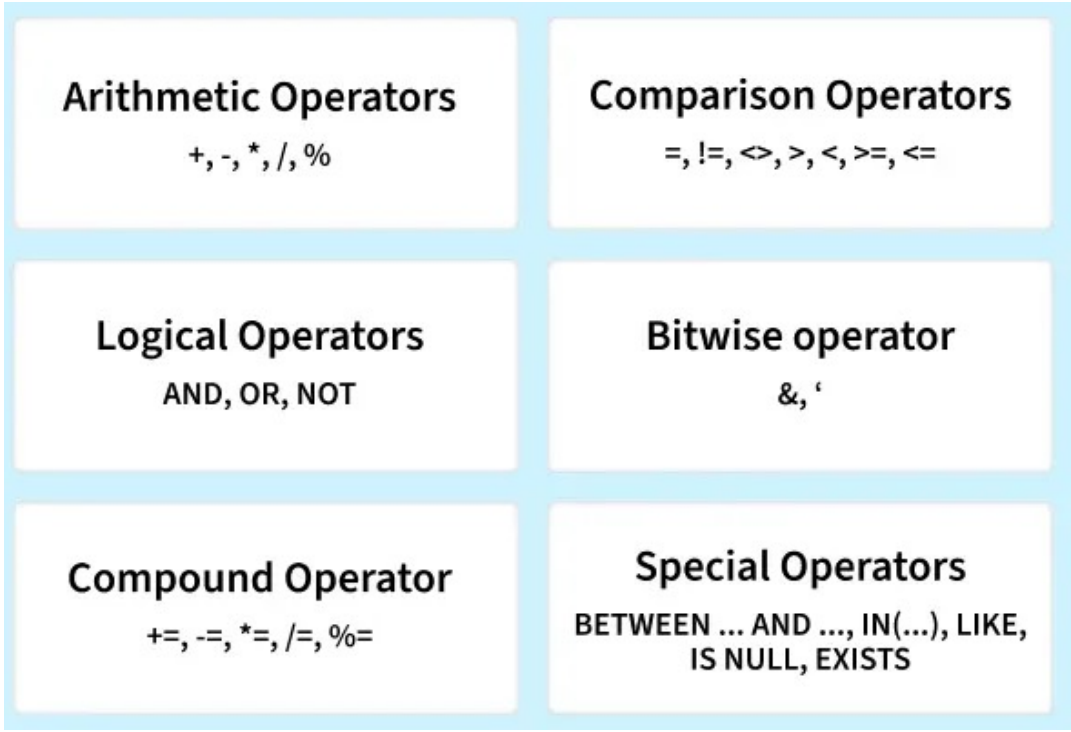
```
ALTER TABLE EMPLOYEE RENAME COLUMN Email TO EmailAdress;
```

```
ALTER TABLE EMPLOYEE ALTER COLUMN EmailAdress TYPE VARCHAR(100);
```

```
ALTER TABLE EMPLOYEE DROP COLUMN EmailAdress;
```

SQL (Structured Query Language) – Operatörler

- SQL operatörleri, SQL sorgularında veriler üzerinde işlem yapmak için kullanılan semboller veya anahtar sözcüklerdir.
- Hesaplamalar, karşılaştırmalar ve mantıksal kontroller gibi işlemleri gerçekleştir.
- Veritabanlarında verilerin filtrelenmesini, hesaplanmasını ve güncellenmesini sağlar.
- Sorgu optimizasyonu ve doğru veri yönetimi için gereklidir.



SQL – SELECT (Seçme) Yapısı

- Tipik bir SQL sorgusunun biçimi:

  SELECT A1, A2, ..., An

  FROM r1, r2, ..., rm

  WHERE P

- Ai bir özniteliği (sütunu) ifade eder.
- ri bir ilişkiyi (tabloyu) ifade eder.
- P bir koşuldur (önermedir/predicate).

- Bir SQL sorgusunun sonucu bir **ilişki (tablo)**dur.

SQL – SELECT (Seçme) Yapısı

- Select bölümü, sorgu sonucunda istenen öznitelikleri (sütunları) listeler.
- **Örnek:** Tüm çalışanların isimlerini bul:

```
SELECT Fname, Lname from EMPLOYEE;
```

- **Not:** SQL adları büyük/küçük harfe duyarsızdır (ör. büyük veya küçük harf kullanabilirsiniz).
  - Fname ≡ FNAME ≡ fname aynı kabul edilir.

SQL – SELECT (Seçme) Yapısı

- SQL, hem tablolarda hem de sorgu sonuçlarında yinelenen (duplicate) satırlara izin verir.
- Yinelenenleri kaldırmak için SELECT'ten sonra **DISTINCT** anahtar sözcüğünü kullanılır.
- **Örnek:** Tüm çalışanların departman numaralarını getir ve tekrarları kaldır:

```
SELECT DISTINCT (dno) from EMPLOYEE;
```

	dno integer
1	1
2	5
3	4

- ALL anahtar sözcüğü, tekrarların kaldırılmamasını belirtir (varsayılan davranıştır).

```
SELECT ALL (dno) from EMPLOYEE;
```

	dno integer
1	1
2	5
3	5
4	5
5	5
6	4
7	4
8	4

SQL – SELECT (Seçme) Yapısı

- Yıldız **(*)** kullanmak "tüm öznitelikler" anlamına gelir.

```
SELECT * from EMPLOYEE;
```

- Satır (tuple) öznitelikleri üzerinde **+, –, *, /** gibi aritmetik işlemler kullanılabilir.
  - Çalışanların maaşlarını 2 katına çıkaralım.

```
SELECT Fname, Salary, Salary*2 AS "Yeni Maaş" from EMPLOYEE;
```

- İfadeyi yeniden adlandırmak için **AS** kullanılabilir.

	fname character varying (15)	lname character varying (15)	salary numeric (10,2)	Yeni Maaş numeric
1	John	Smith	30000.00	60000.00
2	Franklin	Wong	40000.00	80000.00
3	Joyce	English	25000.00	50000.00
4	Ramesh	Narayan	38000.00	76000.00
5	James	Borg	55000.00	110000.00
6	Jennifer	Wallace	43000.00	86000.00
7	Ahmad	Jabbar	25000.00	50000.00
8	Alicia	Zelaya	25000.00	50000.00

SQL – WHERE (Koşul) Yapısı

- WHERE bölümü, sonuçtaki satırların sağlaması gereken koşulları belirtir. İlişkisel cebirdeki seçim (selection) önermesine karşılık gelir.
  - Departman numarası 5 olan çalışanları bulalım.

```
SELECT Fname, ssn, salary, dno from EMPLOYEE WHERE dno=5;
```

	fname character varying (15)	ssn [PK] character (9)	salary numeric (10,2)	dno integer
1	Franklin	333445555	40000.00	5
2	Joyce	453453453	25000.00	5
3	Ramesh	666884444	38000.00	5
4	James	888665555	55000.00	5

- SQL'de mantıksal bağlaçlar: AND, OR, NOT kullanılabilir. Karşılaştırma operatörleri: <, <=, >, >=, =, <> (eşit değil). Bu karşılaştırmalar aritmetik ifadelerin sonuçlarına da uygulanabilir.
  - Departman numarası 5 olup maaş 40.000'den fazla olan çalışanları bulalım.

```
SELECT Fname, ssn, salary, dno from EMPLOYEE WHERE dno=5 AND salary>40000;
```

SQL – FROM Yapısı

- **FROM** bölümü, sorguda yer alan ilişkileri (tabloları) listeler.
  İlişkisel cebirdeki kartezyen çarpıma karşılık gelir.
  - DEPARTMENT × DEPT_LOCATIONS kartezyen çarpımı:

```
SELECT * from DEPARTMENT, DEPT_LOCATIONS;
```

- Bu sorgu, her olası department satırı ile her olası dept_locations satırını eşleştirir; sonuçta iki tablodaki tüm sütunlar yer alır.

	dname character varying (15)	dnumber integer	mgr_ssn character (9)	mgr_start_date date	dnumber integer	dlocation character varying (15)
1	Headquarters	1	888665555	1981-06-19	1	Houston
2	Administration	4	987654321	1995-01-01	1	Houston
3	Research	5	333445555	1988-05-22	1	Houston
4	Headquarters	1	888665555	1981-06-19	4	Stratford
5	Administration	4	987654321	1995-01-01	4	Stratford
6	Research	5	333445555	1988-05-22	4	Stratford
7	Headquarters	1	888665555	1981-06-19	5	Bellaire
8	Administration	4	987654321	1995-01-01	5	Bellaire
9	Research	5	333445555	1988-05-22	5	Bellaire
10	Headquarters	1	888665555	1981-06-19	5	Sugarland
11	Administration	4	987654321	1995-01-01	5	Sugarland
12	Research	5	333445555	1988-05-22	5	Sugarland
13	Headquarters	1	888665555	1981-06-19	5	Houston
14	Administration	4	987654321	1995-01-01	5	Houston
15	Research	5	333445555	1988-05-22	5	Houston

SQL – FROM Yapısı

- Kartezyen çarpım tek başına genelde faydalı değildir; asıl iş, **WHERE** ile seçim koşulunu ekleyince olur (ilişkisel cebirde "selection").
  Tipik kullanım — doğal birleştirme koşulu eklemek:

```
SELECT dep.dnumber, dep.dname, deploc.dlocation
from DEPARTMENT AS dep, DEPT_LOCATIONS AS deploc
WHERE dep.dnumber=deploc.dnumber;
```

	dnumber integer	dname character varying (15)	dlocation character varying (1
1	1	Headquarters	Houston
2	4	Administration	Stratford
3	5	Research	Bellaire
4	5	Research	Sugarland
5	5	Research	Houston

- Aynısını **JOIN** sözdizimiyle de yazabilirsiniz:

```
SELECT dep.dnumber, dep.dname, deploc.dlocation
from DEPARTMENT AS dep JOIN DEPT_LOCATIONS AS deploc
ON dep.dnumber=deploc.dnumber;
```

- **AS** kelime olarak isteğe bağlıdır ve atlanabilir.

  DEPARTMENT **AS** dep ≡ DEPARTMENT dep

SQL – String Operasyonları

- Dize eşleme (string-matching) — LIKE operatörü
  - % → herhangi bir alt diziyi (0+ karakter) eşler.
  - _ → tek bir karakteri eşler.
- İsminde « a » harfi geçen tüm çalışanları listeleyiniz.

	fname character varying (15)	lname character varying (15)
1	Franklin	Wong
2	Ramesh	Narayan
3	James	Borg
4	Ahmad	Jabbar
5	Alicia	Zelaya

```
SELECT FName, Lname from EMPLOYEE WHERE Fname LIKE '%a%';
```

- 5 harfli ve ortadaki harfi «m» olan çalışanları listeleyiniz.

```
SELECT FName, Lname from EMPLOYEE WHERE Fname LIKE '__m__';
```

- Yüzde işaretini (%) harf olarak eşlemek (escape): "100%" metnini tam olarak

```
SELECT FName, Lname from EMPLOYEE WHERE Fname LIKE '100\%' ESCAPE '\';
```

- Burada \ kaçış karakteri olarak tanımlandı; \% artık yüzde karakterinin kendisini ifade eder.

SQL – String Operasyonları

- Büyük/küçük harf duyarlılığı veritabanına göre değişir. PostgreSQL'de **LIKE** büyük/küçük harfe duyarlıdır. Duyarsız eşleme için **ILIKE** kullanabilirsiniz:
- **Eşleme örnekleri**
  - 'Intro%' → "Intro" ile başlayan tüm ifadeler
  - '%Comp%' → içinde "Comp" geçen tüm ifadeler
  - '_ _ _' → tam üç karakter
  - '_ _ _%' → en az üç karakter
- **Dize işlemleri (SQL)**
  - Birleştirme (concatenation): `SELECT FName || ' ' || Lname AS "Name And Surname" from EMPLOYEE`
  - Büyüt/küçültme: UPPER(name), LOWER(name) `SELECT LOWER(FName), UPPER(Lname) from EMPLOYEE`
  - Uzunluk: LENGTH(name) (bazı sistemlerde CHAR_LENGTH) `SELECT LENGTH(fname) FROM EMPLOYEE`
  - Alt dize: SUBSTRING(name, 2, 3) `SELECT SUBSTRING(fname,1,4) FROM EMPLOYEE`
  - Trim: TRIM('A ' FROM name) `SELECT TRIM(' ' from fname) FROM EMPLOYEE`

SQL – Order (Sıralama) İşlemleri

- Çalışanların isimlerini A'dan Z'ye sıralayalım.

```
SELECT Fname, LName FROM EMPLOYEE ORDER BY FNAME ASC;
```

- Sıralama yönü:
  - **ASC** (artan) varsayılandır.
  - **DESC** (azalan) için açıkça belirtin:
- Birden çok alana göre sıralama:
  - ORDER BY fname, lname; -- önce ada, sonra soyada göre
  - (Gerekirse yönleri değiştirebilirsiniz: ORDER BY fname ASC, lname DESC)

	fname character var	lname character va
1	Ahmad	Jabbar
2	Alicia	Zelaya
3	Franklin	Wong
4	James	Borg
5	Jennifer	Wallace
6	John	Smith
7	Joyce	English
8	Ramesh	Narayan

- Not: Bazı veritabanlarında NULL değerlerin sıralamadaki yeri farklı olabilir; PostgreSQL'de kontrol için NULLS FIRST/NULLS LAST kullanılabilir:
  - ORDER BY name ASC NULLS LAST;

SQL – Between Karşılaştırma Operatörü

- Maaşı 30.000 ile 50.000 (dahil) arasında olan çalışanları listeleyelim.

```
SELECT Fname, LName, Salary FROM EMPLOYEE WHERE salary BETWEEN 30000 and 50000
```

- Aralık uçları dahildir (≥ alt sınır, ≤ üst sınır).
- **Tersi için:** NOT BETWEEN 30.000 AND 50.000
- **Eşdeğeri:** WHERE salary >= 30000 AND salary <= 50000

	fname character var	lname character va	salary numeric (10,2)
1	John	Smith	30000.00
2	Franklin	Wong	40000.00
3	Joyce	English	25000.00
4	Ramesh	Narayan	38000.00
5	James	Borg	55000.00
6	Jennifer	Wallace	43000.00
7	Ahmad	Jabbar	25000.00
8	Alicia	Zelaya	25000.00

SQL – SET (Küme) Operasyonları

- 2017 Sonbahar veya 2018 İlkbahar döneminde verilen dersleri bulunuz.

```
(SELECT course_id  FROM SECTION WHERE sem = 'Fall' AND year = 2017)
UNION
(SELECT course_id  FROM SECTION WHERE sem = 'Spring' AND year = 2018);
```

- 2017 Sonbahar ve 2018 İlkbahar döneminde verilen dersleri bulunuz.

```
(SELECT course_id  FROM SECTION WHERE sem = 'Fall' AND year = 2017)
INTERSECT
(SELECT course_id  FROM SECTION WHERE sem = 'Spring' AND year = 2018);
```

- 2017 Sonbahar döneminde verilen ancak 2018 İlkbahar döneminde verilmeyen dersleri bulunuz.

```
(SELECT course_id  from SECTION WHERE sem = 'Fall' AND year = 2017)
EXCEPT
(SELECT course_id  FROM SECTION WHERE sem = 'Spring' AND year = 2018);
```

- Yukarıdaki işlemlerin her biri, yinelenenleri otomatik olarak ortadan kaldırır. Yinelenenleri korumak için; **UNION ALL, INTERSECT ALL, EXCEPT ALL**

SQL – NULL Değeri

- Tuplelar bazı öznitelikler için **NULL** ile gösterilen boş bir değere sahip olması mümkündür.
- Null, bilinmeyen bir değeri veya bir değerin var olmadığını belirtir.
- Null içeren herhangi bir aritmetik ifadenin sonucu NULL'dur.
  - Örnek: 5 + **NULL**, **NULL** döndürür.

- Null, boş değerleri kontrol etmek için kullanılabilir.
- Örnek: Yöneticisi olmayan çalışanları bulunuz.

```
SELECT fname, ssn, super_ssn from EMPLOYEE WHERE super_ssn is NULL;
```

- **NOT NULL** ise boş olmayan değerleri döndürür.

YILDIZ TEKNİK ÜNİVERSİTESİ

Yasemin Topuz

Yıldız Teknik Üniversitesi

ytopuz@yildiz.edu.tr