RA-Relational Algebra

Sorgu Dilleri (Query Languages)

- Veritabanından bilgi almak için kullanılan dil.
- Dillerin katagorileri
  - Yordamsal (Procedural)
  - Yordamsal olmayan (Non-procedural), veya bildirimsel (declarative)
- Pür Diller (“Pure” languages):
  - Relational algebra (İlişkisel Cebir)
  - Relational calculus(ilişkisel hesap)
    - Tuple relational calculus
    - Domain relational calculus

RA:relational algebra

- has a similiar power with SQL.
- RA has NO formal syntax while SQL is industry standard language.
- RA is task oriented query language(procedural) while SQL is result-oriented (declarative).
- DBMS translate SQL to RA in order to execute it.
- RA query can be expressed as a *query tree* that contains nodes for tables and operators within the query. *query tree* shows execution order and is used in execution&optimization.

İlişkisel Cebir (Relational Algebra)

- Altı temel operatör
  - Seçim (select): σ
  - İzdüşüm (project): Π
  - Küme Kesişim($\cap$), Küme Birleşim(union): $\cup$
  - Küme Fark (set difference): –
  - Kartezyen Çarpım (Cartesian product): x
  - Değiştirme (rename): ρ
- Operatörler tek yada iki **ilişkiyi** girdi olarak alıp çıktı olarak yeni bir **ilişki** üretir.

operators

- Single-table operators:
  - Select
  - Project
- 2-table operators
  - Union, intersection, set difference
  - product
  - division*
  - join*
  - semijoin*
  - antijoin*
  - outerjoin*

Seçim İşlemi (Select Operation) – Örnek

- R ilişkisi

A	B	C	D
α	α	1	7
α	β	5	7
β	β	12	3
β	β	23	10

- $\sigma_{A=B \wedge D > 5}(r)$

A	B	C	D
α	α	1	7
β	β	23	10

Seçim İşlemi (Select Operation)

- Notasyon: $\sigma_p(r)$
- *p* **secim kriteri (selection predicate)**
- Tanım:

$$\sigma_p(\boldsymbol{r}) = \{t \mid t \in r \textbf{ and } p(t)\}$$

p burada propositional calculustaki bir formüldür ve

$\wedge$ (**and**), $\vee$ (**or**), $\neg$ (**not**)
terimlerini bağlanmış şekilde içermektedir.

Her terim

<attribute> *op* <attribute> veya <constant>

ki *op:* **=**, $\neq$, **>**, $\geq$. **<**. $\leq$ olmaktadır.

İzdüşüm yada Atma İşlemi (Project Operation) – Örnek

- *r ilişkisi*:

A	B	C
α	10	1
α	20	1
β	30	1
β	40	2

$\Pi_{A,C}(r)$

A	C
α	1
α	1
β	1
β	2

=

A	C
α	1
β	1
β	2

İzdüşüm-Atma işlemi (Project Operation)

- Notasyon: $\Pi_{A_1, A_2, \ldots, A_k}(r)$
  A_1, A_2 öznitelik adları ve r ilişki adı.
- Çıktı, listelenmemiş olan kolonların atılmasıyla ortaya çıkan k kolonlu ilişkidir.
- Tekrar eden satırlar sonuçtan silinir. İlişkiler küme olarak tanımlı olduğundan dolayı.

Birleşim İşlemi (Union Operation) – Örnek

- *r, s ilişkileri:*

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

- r ∪ s:

A	B
α	1
α	2
β	1
β	3

Birleşim İşlemi (Union Operation)

- Notasyon: $r \cup s$
- Tanım:

  $r \cup s = \{t \mid t \in r \text{ or } t \in s\}$

- $r \cup s$ nun geçerli olması için.

  1. *r, s* aynı sayıda özniteliğe sahip olmalıdır (**arity)**

  2. öznitelik alanları **uyumlu (compatible)** olmalıdır (örnek: r'nin 2. kolonu snin 2. kolonu ile aynı tip değerlere sahip olması)

Küme Kesişim İşlemi (Set-Intersection Operation)

- Notasyon: $r \cap s$
- Tanım:
- $r \cap s = \{ t \mid t \in r \textbf{ and } t \in s \}$
- Varsayım:
  - r ve s aynı arity'e sahip
  - r ve s öznitellikleri uyumlu
- Not: $r \cap s = r - (r - s)$

Küme Kesişimi İşlemi– Örnek

- *r, s ilişkisi*:

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

- $r \cap s$

A	B
α	2

Fark İşlemi (Set Difference Operation) – Örnek

- *r*, *s ilişkisi*:

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

- *r – s:*

A	B
α	1
β	1

Fark İşlemi (Set Difference Operation)

- Notasyon $r - s$
- tanım:

$$r - s = \{t \mid t \in r \textbf{ and } t \notin s\}$$

Fark İşlemi uyumlu (**compatible**) ilişkiler içersinde yapılır.

- *r ve s* aynı arity ye sahip olmalı
- r ve s nin öznitelik alanları uyumlu olmalı.

Kartezyen Çarpım İşlemi (Cartesian-Product Operation) – Örnek

- *r, s işlemi:*

A	B
α	1
β	2

r

C	D	E
α	10	a
β	10	a
β	20	b
γ	10	b

s

- *r x s:*

A	B	C	D	E
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

Kartezyen Çarpım İşlemi (Cartesian-Product Operation)

- Notasyon r x s
- Tanım:

$$r \text{ x } s = \{t\, q \mid t \in r \textbf{ and } q \in s\}$$

- r(R) ve s(S) nin öznitellikleri ayrık (disjoint) olduğu kabul edilir. ($R \cap S = \varnothing$).
- Eğer *r(R)* ve *s(S*) nin öznitellikleri ayrık değil ise o zaman değiştirme kullanılmalıdır.

Additional Operations

- *join*, division, renaming:
- Not essential, but very useful.
- Since each operation returns a relation, operations can be *composed*! (Algebra is "closed".)

JOIN

- İki ilişkinin kartezyen çarpımının bir alt kümesi (P koşulunu sağlayan satırlar) seçilir.
- join (T1,T2,P) ≡ select(product(T1,T2),P)
- Çeşitli türleri vardır:
  - Theta-join (yukarda tanımlandığı gibi, koşul üzerinden birleştirme)
  - natural join(doğal birleştirme),
  - outer join (dış birleştirme)
  - anti join
  - semi join

Joined Relations

- **Join operations** take two relations and return as a result another relation.
- A join operation is a Cartesian product which requires that tuples in the two relations match (under some condition). It also specifies the attributes that are present in the result of the join.
- The join operations are typically used as subquery expressions in the **from** clause.

- Theta-Join or Condition –join(c:cond)
  - Join(R,S,c) = Select (product(R,S), c)
  - *Result schema* same as that of cross-product.
  - (sid) sname rating age (sid) bid day

Student record db state

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

COURSE	CId	Title	DeptId
	12	db systems	10
	22	compilers	10
	32	calculus	20
	42	algebra	20
	52	acting	30
	62	elocution	30

SECTION	SectId	CourseId	Prof	YearOffered
	13	12	turing	2004
	23	12	turing	2005
	33	32	newton	2000
	43	32	einstein	2001
	53	62	brando	2001

ENROLL	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

Figure 1-1
Some records for a university database

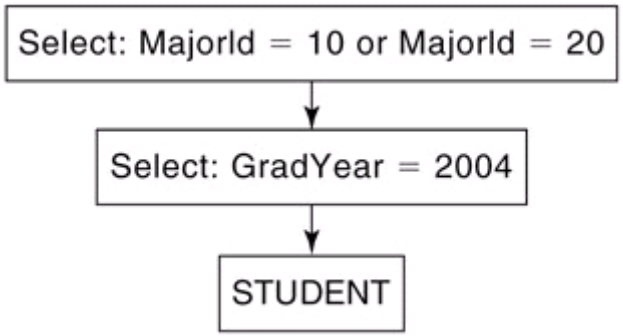
select

```
STUDENT(SId, SName, GradYear, MajorId)
DEPT(DId, DName)
COURSE(CId, Title, DeptId)
SECTION(SectId, CourseId, Prof, YearOffered)
ENROLL(EId, StudentId, SectionId, Grade)
```

- list students who graduated in 2004:
  - Q1: select(STUDENT,GradYear=2004)
- list students who graduated in 2004 with a major of 10 or 20:
  - Q2: select (STUDENT,GradYear=2004 and (MajorId=10 or MajorId=20))

    Q3: select (select(STUDENT,GradYear=2004), MajorId=10 or MajorId=20)

    Q4: select (Q1, MajorId=10 or MajorId=20)
- query tree for Q3:



project

```
STUDENT(SId, SName, GradYear, MajorId)
DEPT(DId, DName)
COURSE(CId, Title, DeptId)
SECTION(SectId, CourseId, Prof, YearOffered)
ENROLL(EId, StudentId, SectionId, Grade)
```

- List the name and graduation year of all students:
  - Q5: project (STUDENT, {SName,GradYear})
- List the name of all students having major 10
  - Q6: project (select(STUDENT, MajorId=10), {SName})
- What about the following query?
  - Q7: select (project (STUDENT, {SName}), MajorId=10)
- Query tree for Q6: