12/3/2015 Relational Algebra

Lecture Topics

I. Background
II. Operators
- Projection
- Selection
- Cartesian Product
- Union
- Difference
- Intersection

III. Computability

I. Background

- Foundation for database query language
- SQL is based on relational algebra with some extensions

I.1 Sets and Operations

If A, B, and C are sets

$\cup$: union, $A \cup B = \{x \mid x \in A \vee x \in B\}$

$\cap$: intersection, $A \cap B = \{x \mid x \in A \wedge x \in B\}$

$-$: difference, $A - B = \{x \mid x \in A \wedge x \notin B\}$

$\times$: Cartesian Product, $A \times B = \{(x,y) \mid x \in A \wedge y \in B\}$

The above form an algebra, i.e., you can perform operations on results of operations, such as $(A \cap B) \times (C \times A)$

I.2 Relations

Relations are sets of tuples
tuples have a domain
assume domain is linearly ordered
i.e., $x < y$, $x = y$, or $y < x$

- order of rows does not matter
- does not matter if there are duplicate rows
- Note! in set theory, there is one empty set.
- We will assume a different empty relation for each relation schema

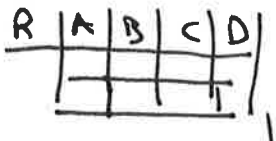
I.3 R.A. vs. SQL

- R.A. does not support: Primary keys, Foreign keys, inserts, deletes, updates, indexing, recovery, security, etc.
- only support for querying.
- does not care about efficiency

II Operations on Relations

- Any operation on relations produces a relation.

II.1 Projection



$\Pi_{A,B}(R)$

Select A, B
FROM R

column selection

12/3/2015 Relational Algebra Pg. 2

II.2 Selection

R	A	B	C	D
	1	a	x	g
	2	b	y	c
	3	a	c	s

$\sigma_{A<2}(R)$

SELECT *
FROM R
WHERE A<2

Row selection

R	A	B	C	D
	1	a	x	g

The condition (predicate) can be specified by a boolean formula
$\neg, \wedge, \vee$ on atomic conditions

atomic conditions are:
- a comparison between columns
- a comparison between column and constant

II.3 Cartesian Product

R	A	B
	1	10
	2	20

S	C	B	D
	40	10	10
	50	20	20

R×S

SELECT A, R.B, C, S.B, D
FROM R, S

	A	R.B	C	S.B	D
	1	10	40	10	10
	1	10	50	20	20
	2	20	40	10	10
	2	20	50	20	20

- useful to correlate information from two tables

II.4 Union

R	A	B
	1	10
	2	20

S	A	B
	1	10
	3	20

R ∪ S

SELECT *
FROM R
UNION
SELECT *
FROM S

A	B
1	10
2	20
3	20

MUST be union compatible
- require ~~sets~~ relations of same-arity
- columns should "probably" be from the same domains.

II.5 Difference

R	A	B
	1	10
	2	20

S	A	B
	1	10
	3	20

R − S

SELECT *
FROM R
MINUS
SELECT *
FROM S

A	B
2	20

- EXCEPT sometimes instead of MINUS
- MUST be union compatible

II.b Intersection

R	A	B
	1	10
	2	20

S	A	B
	1	10
	3	20

$R \cap S$

A	B
1	10

```
SELECT *
FROM R
INTERSECT
SELECT *
FROM S
```

- Union compatability required
- NOT FUNDAMENTAL

$R \cap S = R - (R - S)$

III Computability

- Given:
  - Person (Name, Sex, Age)
  - Birth (Parent, child)
- Produce:
  - Answer (Father, Daughter)
- Strategy:
  - Need two distinct copies of 'Person'

```
SELECT P.AS FATHER, C.AS DAUGHTER
FROM PERSON, BIRTH, PERSON AS PERSON1
 WHERE P=PERSON.N AND C=PERSON1.N
 AND PERSON.S='M' AND PERSON1.S='F'
```

- Produce: Answer (GRANDPARENT, GRANDCHILD)
- (GREAT-GRANDPARENT, GREAT-GRANDCHILD)
  - Cartesian product of (grandparent, gran... with (Parent, child), equality on the intermediate person.
- (GREAT* - parent, GREAT* -CHILD)?
- (Ancestor, Descendant)?

↳ not possible in relational algebra

Proof sketch: (by induction)

- Any R.A. query is limited in how many relations, or copy of relations, it can refer to
- Computing arbitrary (Ancestor, Descendant) pairs can't be done if the query is limited by the number of relations or copies of relations it can refer to

Another Example!

Can't compute transitive closure with R.A.

- you can with SQL 1999 standard

Example Schema: