

# CS 4604: Introduction to Database Management Systems

B. Aditya Prakash

Lecture #3: SQL and Relational Algebra



### Reminder

NO office hours today!

Extended (10am-12noon) hours on 1/30 and2/4



## **Formal Query Languages**

- How do we collect information?
- E.g. Find SSNs of people in 4604 (recall everything is a set!)



## What is SQL

- SQL = Structured Query Language (pronounced "sequel").
- Language for defining as well as querying data in an RDBMS.
- Primary mechanism for querying and modifying the data in an RDBMS.
- SQL is declarative:
  - Say what you want to accomplish, without specifying how.
  - One of the main reasons for the commercial success of RDMBSs.
- SQL has many standards and implementations:
  - ANSI SQL
  - SQL-92/SQL2 (null operations, outerjoins)
  - SQL-99/SQL3 (recursion, triggers, objects)
  - Vendor-specific variations.

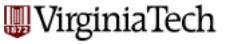


## Relational Algebra

 Relational algebra is a notation for specifying queries about the contents of relations

 Notation of relational algebra eases the task of reasoning about queries

 Operations in relational algebra have counterparts in SQL

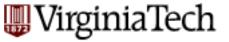


## What is an Algebra?

- An algebra is a set of operators and operands
  - Arithmetic: operands are variables and constants, operators are  $+,-,\times,\div$ , /, etc.
  - Set algebra: operands are sets and operators are ∩, U, -

- An algebra allows us to
  - construct expressions by combining operands and expression using operators
  - has rules for reasoning about expressions

$$a^2 + 2 \times a \times b + 2b$$
,  $(a + b)^2$   
R - (R - S), R \cap S



## **Basics of Relational Algebra**

- Operands are relations, thought of as sets of tuples.
- Think of operands as variables, whose tuples are unknown.
- Results of operations are also sets of tuples.
- Think of expressions in relational algebra as queries, which construct new relations from given relations.
- Four types of operators:
  - Select/Show parts of a single relation: projection and selection.
  - Usual set operations (union, intersection, difference).
  - Combine the tuples of two relations, such as cartesian product and joins.
  - Renaming.



## **Projection**

- The projection operator produces from a relation R a new relation containing only some of R's columns
- "Delete" (i.e. not show) attributes not in projection list
- Duplicates eliminated (sets vs multisets)
- To obtain a relation containing only the columns A<sub>1</sub>,A<sub>2</sub>,...A<sub>n</sub> of R

RA:  $\pi$  A<sub>1</sub>,A<sub>2</sub>,...A<sub>n</sub> (R)

**SQL: SELECT** A1,A2,... An **FROM** R;



# **Projection Example**

*S2* 

<u>sid</u>	sname	rating	age	
28	уирру	9	35.0	
31	lubber	8	55.5	
44	guppy	5	35.0	
58	rusty	10	35.0	

 $\pi_{sname,rating}(S2)$ 

sname	rating
уирру	9
lubber	8
guppy	5
rusty	10

 $\pi_{age}(S2)$ 

age	
35.0	
55.5	



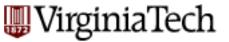
## Selection

- The selection operator applied to a relation R produces a new relation with a subset of R's tuples
- The tuples in the resulting relation satisfy some condition C that involves the attributes of R
  - with duplicate removal

 $RA: \sigma_{C}(R)$ 

**SQL: SELECT** \***FROM** R **WHERE** C;

 The WHERE clause of a SQL command corresponds to σ()



## **Selection: Syntax of Conditional**

- Syntax of conditional (C): similar to conditionals in programming languages.
- Values compared are constants and attributes of the relations mentioned in the FROM clause.

- We may apply usual arithmetic operators to numeric values before comparing them.
  - RA Compare values using standard arithmetic operators.

11

**SQL** Compare values using =, <>, <, >, <=, >=.

Prakash 2013 VT CS 4604



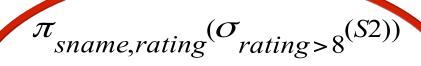
# **Selection Example**

*S2* 

<u>sid</u>	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

$$\sigma_{rating>8}(S2)$$

sid	sname	rating	age
28	yuppy	9	35.0
58	rusty	10	35.0



sname	rating
yuppy	9
rusty	10

**Combining Operators** 



## **Set Operations: Union**

- Standard definition: The union of two relations R and S is the set of tuples that are in R, or S or in both.
- When is it valid?
  - R and S must have identical sets of attributes and the types of the attributes must be the same.
  - The attributes of R and S must occur in the same order.



## **Set Operations: Union**

- RA RUS
- SQL (SELECT \* FROM R)UNION(SELECT \* FROM S);



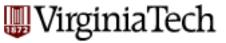
## **Set Operations: Intersection**

- The intersection of R and S is the set of tuples in both R and S
- Same conditions hold on R and S as for the union operator
- $\blacksquare$  RA R  $\cap$  S
- SQL (SELECT \* FROM R) INTERSECT (SELECT \* FROM S);



## **Set Operations: Difference**

- Set of tuples in R but NOT in S
- Same conditions on R and S as union
- $\blacksquare$  RA R  $\cap$  S
- SQL (SELECT \* FROM R)EXCEPT(SELECT \* FROM S);
- $\blacksquare$  R (R S) = R \cap S



## **Difference**

S1 S2

<u>sid</u>	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

<u>sid</u>	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

$$S1-S2$$

sid	sname	rating	age
22	dustin	7	45.0

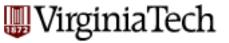


#### **Cartesian Product**

- The Cartesian product (or cross-product or product) of two relations R and S is a the set of pairs that can be formed by pairing each tuple of R with each tuple of S.
  - The result is a relation whose schema is the schema for R followed by the schema for S.

RA: RXS

SQL: SELECT \* FROM R, S;



#### **Cartesian Product**

*S*1

<u>sid</u>	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

R1

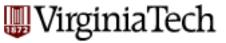
sid	<u>bid</u>	<u>day</u>
22	101	10/10/96
58	103	11/12/96

S1 X R1

(sid)	sname	rating	age	(sid)	bid	day
22	dustin	7	45.0	22	101	10/10/96
22	dustin	7	45.0	58	103	11/12/96
31	lubber	8	55.5	22	101	10/10/96
31	lubber	8	55.5	58	103	11/12/96
58	rusty	10	35.0	22	101	10/10/96
58	rusty	10	35.0	58	103	11/12/96

We rename attributes to avoid ambiguity or we prefix attribute with the name of the relation it belongs to.

?

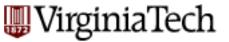


## Theta-Join

The theta-join of two relations R and S is the set of tuples in the Cartesian product of R and S that satisfy some condition C.

RA: R 
$$\bowtie_C$$
 S

$$R \bowtie_C S = \sigma_C(R \times S)$$



#### **Theta-Join**

*S*1

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

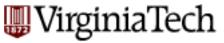
R1

sid	<u>bid</u>	<u>day</u>
22	101	10/10/96
58	103	11/12/96

$$S1 \bowtie S1.sid < R1.sid$$

(sid)	sname	rating	age	(sid)	bid	day
22	dustin	7	45.0	58	103	11/12/96
31	lubber	8	55.5	58	103	11/12/96

$$R\bowtie_{c} S = \sigma_{c}(R \times S)$$



### **Natural Join**

- The natural join of two relations R and S is a set of pairs of tuples, one from R and one from S, that agree on whatever attributes are common to the schemas of R and S.
- The schema for the result contains the union of the attributes of R and S.
- Assume the schemas R(A,B, C) and S(B, C,D)

RA: R S

**SQL: SELECT \*** 

FROM R, S

WHERE R.B = S.B AND R.C = S.C;



## **Operators so far**

#### Remove parts of single relations

- Projection:  $\pi_{(A,B)}(R)$  and SELECT A, B FROM R
- Selection:  $\sigma_C(R)$  and SELECT \* FROM R WHERE C
- Combining Projection and Selection:
- $\pi_{(A,B)}(\sigma_C(R))$
- SELECT A, B FROM R WHERE C



## **Operations so far**

#### Set operations

 R and S must have the same attributes, same attribute types, and same order of attributes

- Union: R U S and (R) UNION (S)
- Intersection:  $R \cap S$  and (R) INTERSECT (S)
- Difference: R S and (R) EXCEPT (S)



## **Operations so far**

- Combine the tuples of two relations
  - Cartesian Product: R X S, .... FROM R, S .....
  - Theta Join: R  $\bowtie_C$  S, .... FROM R, S WHERE C
  - Natural Join: R ➤ S



- If two relations have the same attribute, disambiguate the attributes by prefixing the attribute with the name of the relation it belongs to.
- How do we answer the query "Name pairs of students who live at the same address"? Students (Name, Address)
  - We need to take the cross-product of Students with itself?
  - How do we refer to the two "copies" of Students?
  - Use the rename operator.



```
RA: give R the name S;
R has n attributes,
which are \rho_{S (A_1,A_2,...A_n)} (R)
called A1, A2, ..., An in S
```

**SQL:** Use the **AS** keyword in the **FROM** clause: Students AS Students1 renames Students to Students1.

**SQL**: Use the **AS** keyword in the **SELECT** clause to rename attributes.



Name pairs of students who live at the same address:

```
RA: \pi_{S1.Name,S2.Name} (
\sigma_{S1.Address=S2.Address} (
\rho_{S1}(Students) \times \rho_{S2}(Students)))
```

SQL: SELECT S1.name, S2.name
FROM Students AS S1, Students AS S2
WHERE S1.address = S2.address



Name pairs of students who live at the same address:

SQL: SELECT S1.name, S2.name
FROM Students AS S1, Students AS S2
WHERE S1.address = S2.address

- Are these correct?
- No !!! the result includes tuples where a student is paired with himself/herself
- Solution: Add the condition S1.name <> S2.name.



## Other Details in SQL

Read Chapters 6.1.3-6.1.8 of the textbook for string comparison, pattern matching, NULL and UNKNOWN values, dates and times, and ordering the output.



# **Independence of Operators**

The operators we have covered so far are:

$$\pi_{A,B}(R), \sigma_C(R), \rho_{S(A_1,A_2)}(R)$$
  
 $R \cup S, R \cap S, R - S, R \times S, R \bowtie S, R \bowtie_C S$ 

- Are all of them needed?
- NO!



## **Independence of Operators**

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

$$R \bowtie_C = \sigma_C(R \times S)$$

$$R \bowtie S = ??$$



# **Independence of Operators**

$$R \bowtie S$$

- Suppose R and S share the attributes A1,A2,..An
- Let L be the list of attributes in R followed by the list of attributes in S
- Let C be the condition

R.A1 = S.A1 AND R.A2 = S.A2 AND ..... R.An = S.An

$$R \bowtie S = \pi_L(\sigma_C(R \times S))$$