

CS589 Principles of DB Systems Lecture 1-1: Relational Model and Relational Algebra

David Maier (maier@cs.pdx.edu)

1



Administrivia

Class web page: piazza.com/pdx/spring2016/cs589/home

- Detailed class schedule
 - Topics
 - Reading assignments
 - Quizzes
 - Exam dates
- Lecture slides (.pdf)
 - posted before class begins
 - ink versions posted after lecture

Class text:

Levene and Loizou, *A Guided Tour of Relational Databases and Beyond,* Springer-Verlag, 1999.

Class discussion and questions will be on the Piazza page I will post scores on quizzes and assignments on D2L

CS 589 Principles of Database Systems, Spring 2016 © David Maier



Planned Activities

- Two exams [48%]
 - Dates per posted class schedule
 - In class, work by yourself, open book
 - Each over half of the class material
- Quizzes (8) [14%]
 - In class, work by yourself, closed book
 - One quiz every Tuesday, with some exceptions
 - Lowest quiz score will be dropped
- Homework Assignments (4) [36%]
 - May work with a partner, turn in 1 paper
 - Due on Thursdays
- Participation [2%]
 - Class worksheets
 - Activity on Piazza

CS 589 Principles of Database Systems, Spring 2016 © David Maier



Learning objectives

- Be familiar with the results and techniques presented here and be able to apply them in your own work.
- 2. Be able to read and study other DB results that have been formalized.
- 3. Be able to analyze and evaluate one or more particular formulations.
- 4. Be able to formalize aspects of your own research.
- 5. Understand the benefits and limitations that derive from formalizing aspects of DB work.

CS 589 Principles of Database Systems, Spring 2016 © David Maier



Unit 1: Relational Query Languages

- Relational model (per L & L)
- Relational algebra
- Relational calculus
 - Tuple calculus
 - Domain calculus
- Introduction to Datalog
 - Will return to Datalog in Unit 4
- Equivalence of languages

CS 589 Principles of Database Systems, Spring 2016 © David Maier

5



Relational Model & Relational Algebra

We assume you are familiar with the relational model and with relational algebra in some form.

- Introduce the definition of the relational model used in the text
- Introduce the definition of the relational algebra used in the text

Have a look at § 1.9.1 - 1.9.3 to see authors' notation for sets, orders, logic.

CS 589 Principles of Database Systems, Spring 2016 © David Maier



Relation schema

Relation schema – a relation symbol R with an associated similarity type, type(R). type(R) is a natural number that tells us the number of attributes in the relation schema

Discussion questions:

- What aspects of a relational schema are missing?
- Based on this definition of schema, how would you define union-compatibility?
- 3. Would type(R) = 0 make sense?

CS 589 Principles of Database Systems, Spring 2016 © David Maier

7



A relation schema with attribute n

For each relation schema, there is a 1-to-1 mapping called att from $\{1, 2, 3, ..., \text{type}(R)\}$ to u, where u is the universal set of names (to be used as names in this database).

Example: Relation symbol is Student with similarity type of 4

define the mapping att for this relation schema

att(1) = id

att(2) = last-name

att(3) = first-name

att (4) = major

Define $schema(R) = {att(1), att(2), ..., att(type(R))}$

Example: schema(Student) = {id, last-name, first-name, major}

CS 589 Principles of Database Systems, Spring 2016 © David Maier



A relation schema with attribute names

Attributes are *ordered* and *named* in this model.

Also assume each attribute A has an associated domain of values: DOM(A)

Discussion questions:

- 1. Is it possible for two attributes in one relation schema to have the same name?
- 2. Can a relation schema have an infinite number of attributes?
- 3. Can DOM(A) = DOM(B)?

CS 589 Principles of Database Systems, Spring 2016 © David Maier

9



A database schema

A *database* schema is a finite set $\mathcal{R} = \{R_1, R_2, ..., R_n\}$ such that each $R_i \in R$ is a relation schema.

The schema of ${\mathcal R}$ (the entire database) is defined as:

schema
$$(\mathcal{R}) = \bigcup_{i \in I}$$
 schema (R_i) ,
where $I = \{1, 2, ..., n\}$

CS 589 Principles of Database Systems, Spring 2016 © David Maier



First Normal Form assumption

- A relation schema is in First Normal Form (1NF) if all the domains of all attributes in schema(ℜ) are atomic
- A database schema is in 1NF if all its relation schemas are in 1NF
- Examples of attributes not in 1NF:
 - Set- or list-valued attribute
 - Attribute values that are complex objects
 - Attribute values that are relations: Nested Relations

CS 589 Principles of Database Systems, Spring 2016 © David Maier

11



Universal relation schema assumption

Notice – this is an assumption (not a definition).

A database schema \mathcal{R} satisfies the universal relational schema assumption if each attribute in database schema \mathcal{R} plays a unique role in \mathcal{R} .

Donain of an abhorte is the weight.

That is, all occurrences of an attribute in the

database schema are assumed to have the same meaning.

student(id, last_name, first_name, major)
course(id, dept, number, credits)

CS 589 Principles of Database Systems, Spring 2016 © David Maier



Universal relation schema assumption and union-compatibility

Two relation schemas, R and S, are union-compatible if they are identical (i.e., if their corresponding schemas have the same attribute set).

Discussion questions:

- How does this definition of union-compatibility (the one from the book) compare to an alternative definition of union compatibility: Two relation schemas have the same number of attributes and corresponding attributes have the same domain
- 2. Does the definition of union-compatibility in the book prevent us from taking the union of two relations that satisfy the above, alternative definition of union-compatibility?

CS 589 Principles of Database Systems, Spring 2016 © David Maier

13



And now for the data in a database

A tuple over a relation scheme R, with schema (R) = $\{A_1, A_2, ..., A_m\}$ where att(i) = A_i , for i = 1, 2, ..., m is a member of the Cartesian product

$$DOM(A_1) \times DOM(A_2) \times ... \times DOM(A_m)$$

A relation over R is a <u>finite</u> set of tuples over R.

14

CS 589 Principles of Database Systems, Spring 2016 © David Maier



Alternative definition of a tuple

A tuple t of relation scheme R over schema(R) is a total mapping from schema(R) to the union of the domains of the attributes of R such that $\forall A_i \in \text{schema}(R)$, $t(A_i) \in \text{DOM}(A_i)$

Example: Student(id, last-name, first-name, major)

using the first definition of tuple, an example is the sequence: <111 , Doe , John , CS>

using the second, alternative definition of tuple, t is a function: t(id) = 111, t(last-name) = Doe, t(first-name) = John, t(major) = CS.

What's the difference in these definitions?

CS 589 Principles of Database Systems, Spring 2016 © David Maier

15



A database (the data ...)

A database over $\mathcal{R} = \{R_1, R_2, ..., R_n\}$ is a set $d = \{r_1, r_2, ..., r_n\}$ such that each r_i is a relation over $R_i \in \mathcal{R}$

Discussion questions:

- Is is possible for a relation to be empty in a database?
- 2. Is it possible for two relations in a database to have exactly the same set of tuples?

CS 589 Principles of Database Systems, Spring 2016 © David Maier



Projection of a tuple onto one attribute

Projection of a tuple t in a relation r over schema R onto the attribute A_i in schema(R) is the i-th coordinate of t.

If a tuple t is defined as an element of the cross product of the domains, then t(i) is selecting the i-th component of this element of a cross product.

If a tuple t is defined as a mapping, then getting the value of attribute A_i is equal to applying the mapping to A_i : $t(A_i)$.

In different contexts, we might use positional [t(4)] or mapping [t(major)] notation.

CS 589 Principles of Database Systems, Spring 2016 © David Maier

17



Projection onto a set of attributes

We extend the notion of projection to a set of attributes,

Y = {att(
$$i_1$$
), att(i_2), ..., att(i_k)} \subseteq schema (R) with $i_1 < i_2 < ... < i_k$, as follows:

$$t[Y] = \langle t(i_1), t(i_2), ..., t(i_k) \rangle$$

Notes: Y is a set of attribute names.

Projection is defined for one tuple; the result of projection is one tuple.

t(4) or t(major) is selecting a value; t[major] is projecting the tuple t to produce a new tuple with one attribute.

CS 589 Principles of Database Systems, Spring 2016 © David Maier



Relational Algebra

- The relational algebra is a set of operators
 - Some unary, some binary
- Each operator takes in relation(s) and produces a relation
- A relational query is the composition of a set of operators
- Some binary operators require unioncompatibility, some do not.

CS 589 Principles of Database Systems, Spring 2016 © David Maier

10



Relational algebra: \cup , \cap , —

Union, intersection, and difference require that the two input relations are union-compatible.

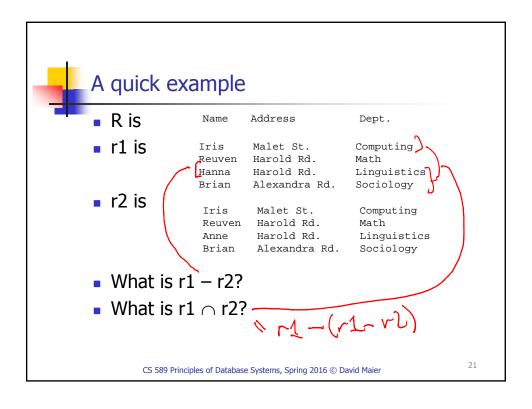
Union:
$$r_1 \cup r_2 = \{t \mid t \in r_1 \text{ or } t \in r_2\}$$

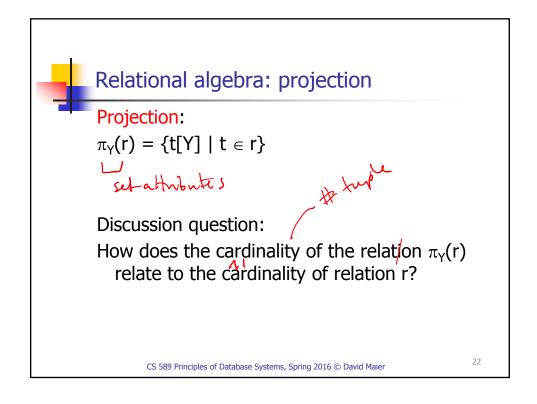
Intersection: $r_1 \cap r_2 = \{t \mid t \in r_1 \text{ and } t \in r_2\}$

Difference:
$$r_1 - r_2 = \{t \mid t \in r_1 \text{ and } t \notin r_2\}$$

Note: each operator is defined by the set of tuples it produces (based on tuples in the input relations).

CS 589 Principles of Database Systems, Spring 2016 © David Maier







Relational Algebra: Selection

Suppose we have one tuple in our hand. How do we translate that into something that is true or false, to drive a conditional selection process?

∠Logical implication: Let r be a relation over relation schema R, t a tuple in r, F, F_1 , and F_2 are selection formulae, then t logically implies (\models) F is defined as:

```
t \models A=a, if the expression t(A)=a evaluates to true
t \models A=B, if the expression t(A)=t(B) evaluates to true
t \models F_1 \land F_2, if t \models F_1 and t \models F_2
t \models F_1 \lor F_2, if t \models F_1 or t \models F_2
```

$$t \models \neg F, \text{ if } t \text{ does not } \models F$$
 $t \models (F), \text{ if } t \models F$
 $t \models (F), \text{ if } t \models F$
 $t \models (F), \text{ if } t \models F$

$$t(id)=150 \lor \neg(t(major)=CS)$$

CS 589 Principles of Database Systems, Spring 2016 © David Maier



Relational algebra: selection, natural join

Selection:

$$\sigma_{\mathsf{F}}(\mathsf{r}) = \{ \mathsf{t} \mid \mathsf{t} \in \mathsf{r} \text{ and } \mathsf{t} \not\models \mathsf{F} \}$$

Natural join: $\nearrow \sim \mathcal{R}$

 $r_1 \bowtie r_2 = \{ t \mid \exists t_1 \in r_1 \text{ and } \exists t_2 \in r_2 \text{ such that } S_1 \text{ (AB)} \text{$ $t[schema(R_2)] = t_2$

Where schema(R) = schema(R₁) \cup schema(R₂)

Discussion questions:

- Which attributes are we joining on? Scheme (R) (Scheme (R))
- What happens if there are no attributes to join on?

CS 589 Principles of Database Systems, Spring 2016 © David Maier



Discussion Questions

- What are the equivalent relational algebra operations for
- $\sigma_{F1 \wedge F2}(r) = \sigma_{F1}(r) \cap \sigma_{F2}(r)$
- $\sigma_{F1\vee F2}(r) = \sigma_{F1}(r) \cup \sigma_{F2}(r)$ $\sigma_{-F}(r) = r \sigma_{F1}(r)$

CS 589 Principles of Database Systems, Spring 2016 © David Maier



Natural join example

Student	S-Id	Name	F-Id	
	1	John	101	
	2	Maria	101	
	3	Wei	102	

Faculty	F-Id	F-Name	Rank
	101	Dave	Prof
	102	Tim	Prof
	103	Niru	Assoc Prof

One of the tuples in the answer: t

based on these two existing tuples:



 $r_1\bowtie r_2$ = { $t~|~\exists t_1\in r_1~\text{and}~\exists t_2\in r_2~\text{such that}~$ $t[schema(R_1)] = t_1$ and $t[schema(R_2)] = t_2$

Where schema(R) = schema(R₁) \cup schema(R₂)

The natural join is ALL such tuples that can be constructed.

CS 589 Principles of Database Systems, Spring 2016 © David Maier



Renaming

Let r be a relation over relation schema R, A be an attribute of schema(R) and B an attribute in u which is not in schema(R).

Renaming, ρ , of A to B in r, is a relation over schema(S) = (schema(R) – {A}) \cup {B}, defined by:

$$\rho_{A\to B}(r)=\{\ t\ |\ \exists u\in r\ \text{such that}$$

$$t[\text{schema}(S)-\{B\}]=u[\text{schema}(R)-\{A\}]$$
 and
$$t[B]=u[A]\}$$

Can anyone say this in simple English?

CS 589 Principles of Database Systems, Spring 2016 © David Maier

27



Division

Let r be a relation over relation schema R, with schema(R) = XY, and s be a relation over relation schema S, with schema(S) = Y.

The superscript with other schema S, with schema S, wit

The division of r by s is a relation over relation schema R1 where schema(R1) = X is defined as:

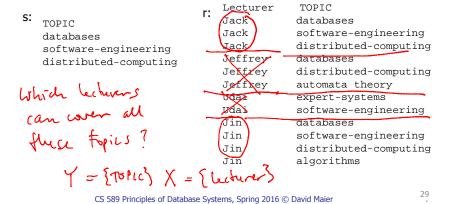
$$\begin{array}{c} r \div s = \{\; t[X] \mid t \in r \; \text{and} \; s \subseteq \pi_Y(\sigma_F(r)) \; \text{where} \\ X = \{A_1,\,A_2,\,...,\,A_q\} \; \text{and} \\ F \; \text{is the formula} \; A_1 {=} t[A_1] \; \wedge \; ... \; \wedge \; A_q {=} t[A_q]\} \end{array}$$

CS 589 Principles of Database Systems, Spring 2016 © David Maier



Division

- What does the division operator have to do with universal quantification?
- What is r ÷ s for these relations?





Relational algebra queries

A relational algebra expression (i.e., query) is a well-formed expression consisting of a finite number of relational algebra operators (\cup , \cap , \neg , σ , π , \bowtie , ρ , \div) whose operands are relation schemas which can be treated as input variables to the query.

An answer to a relational algebra query is obtained by replacing every occurrence of R_i in the query by a relation over R_i and computing the results by invoking the relational algebra operators in the query.

A query language is relationally complete if it is at least as expressive as the relational algebra.

CS 589 Principles of Database Systems, Spring 2016 © David Maier



Aggregate Functions

- Need answers for "summary" queries
 - How many?
 - Overall average?
 - Maximum, minimum
 - Sum
- Other relational algebra compositions cannot answer these, because we lack computations that iterate over tuples
- Aggregate: a function over an attribute, which given a finite set of tuples returns a natural number
 - Book is in error here...may not be a natural number
 - Common aggregates: COUNT, MIN, MAX, SUM, AVG

CS 589 Principles of Database Systems, Spring 2016 © David Maier

31



Aggregate Functions

 $F_A^X(r)$ means the result of applying F to attribute A, partitioned into distinct groups by X

If $X = \emptyset$, we apply F over the entire relation

NAME Abdu	DEPT Computing	SALARY 2000	DAY Monday	What is the an	swer to:		
Abdu	Computing	2000	Tuesday	COLINT/ /			
Abdu	Computing	2000	Thursday	$COUNT(\pi_{NAME}($	r)) 🗲 🕂		
Hanna Hanna	Computing Computing	1400 1400	Wednesday Friday	COLUNITEDEDT!			
	d Computing	1000	Friday	$COUNT^{DEPT}(\pi_{NAME,DEPT}(r))$			
(Martine		1600	Tuesday	DEDT.			
Martine	e Philosophy	1600	Friday	SUM _{SALARY} DEPT ($\pi_{NAME,DEPT,SALARY}(r)$)		
Reuven	Maths	1500	Wednesday	DERT	SUM		
Reuven	Maths	1500	Thursday		44.7 = -9		
(Dan	Linguistics	1000	Tuesday	computing	4400		
Ruth	Linguistics	1100	Monday	maths	3100		
			J	phieosophy	1600		
CS 589 Principles of Database Systems, Spring 2016 © David Maier 2100 32							



Relational Completeness

- The set of queries expressible in relational algebra is widely considered the minimal set of queries for any reasonable relational query language
- A query language is said to be relationally complete if it is at least as expressive as the relational algebra

CS 589 Principles of Database Systems, Spring 2016 © David Maier

33



Operator Sets

Some operators are redundant

$$r \cap s = r - (r - s)$$
 also, division

There are other equivalent sets

Some things not expressible: transitive closure

CS 589 Principles of Database Systems, Spring 2016 © David Maier