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

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

Learning objectives

1. 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.

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

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.

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:

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

A relation schema with attribute names

no duplicate attributes

For each relation schema, there is a 1-to-1 **mapping** called att from {1, 2, 3, …, type(R)} to $\mathcal{U}$, where $\mathcal{U}$ 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}

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)?

A database schema

A *database* schema is a finite set

$\mathcal{R} = \{R_1, R_2, \ldots, R_n\}$

such that each $R_i \in$ R is a relation schema.

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

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

First Normal Form assumption

- A relation schema is in First Normal Form (1NF) if all the domains of all attributes in schema($\mathcal{R}$) 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

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 $\mathcal{R}$.

Domain of an attribute is the same everywhere

That is, all occurrences of an attribute in the database schema are assumed to have the same meaning.

```
student(sid, last_name, first_name, major)
course(cid, dept, number, credits)
```

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:

1. 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?

And now for the data in a database

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

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

A relation over R is a finite *set* of tuples over R.

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 $\forall A_i \in$ schema(R), $t(A_i) \in DOM(A_i)$

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?

A database (the data …)

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

Discussion questions:

1. 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?

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.

Projection onto a set of attributes

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

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

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

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.

t = <111, John, Doe, CS>
t[{last_name, first_name}] = <John, Doe>

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 union-compatibility, some do not.

Relational algebra: $\cup$, $\cap$, —

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

Union: $r_1 \cup r_2 = \{t \mid t \in r_1 \text{ or } t \in r_2\}$ no dups

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).

A quick example

- R is

Name	Address	Dept.

- r1 is

Iris	Malet St.	Computing
Reuven	Harold Rd.	Math
Hanna	Harold Rd.	Linguistics
Brian	Alexandra Rd.	Sociology

- r2 is

Iris	Malet St.	Computing
Reuven	Harold Rd.	Math
Anne	Harold Rd.	Linguistics
Brian	Alexandra Rd.	Sociology

- What is r1 – r2?
- What is r1 ∩ r2?

= r1 − (r1 − r2)

Relational algebra: projection

Projection:

$\underbrace{\pi_Y}_{\text{set attributes}}(r) = \{t[Y] \mid t \in r\}$

Discussion question:

How does the cardinality of the relation $\pi_Y(r)$ relate to the cardinality of relation r?

tuple

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:

constant

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 $\models$ $F_1 \wedge F_2$, if t $\models F_1$ and t $\models F_2$

t $\models$ $F_1 \vee F_2$, if t $\models F_1$ or t $\models F_2$

t $\models$ $\neg$F, if t does not $\models$ F

t $\models$ (F), if t $\models$ F

$\sigma_{id=150 \vee \neg(major=CS)}$

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

CS 589 Principles of Database Systems, Spring 2016 © David Maier 23

Relational algebra: selection, natural join

Selection:

$\sigma_F(r) = \{ t \mid t \in r \text{ and } t \models F \}$

$r_1(ABC) \bowtie r_2(BCD)$

$r(ABCD)$

Natural join: on R

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

Where schema(R) = schema(R_1) $\cup$ schema(R_2)

$S_1(AB) \bowtie S_2(CD)$

$S(ABCD)$

Discussion questions:

1. Which attributes are we joining on? Schema(R_1) $\cap$ Schema(R_2)
2. What happens if there are no attributes to join on?

CS 589 Principles of Database Systems, Spring 2016 © David Maier 24

Discussion Questions

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

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

1	John	101	Dave	Prof

based on these two existing tuples:

t_1

1	John	101

t_2

101	Dave	Prof

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

Where $\text{schema}(R) = \text{schema}(R_1) \cup \text{schema}(R_2)$

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

Renaming

Let r be a relation over relation schema R, A be an attribute of schema(R) and B an attribute in $\mathcal{U}$ which is not in schema(R).

rho

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

$\rho_{A \to B}(r) = \{ t \mid \exists u \in r$ such that
$t[schema(S) - \{B\}] = u[schema(R) - \{A\}]$
and
$t[B] = u[A]\}$

Can anyone say this in simple English?

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.

can be expressed with other operations

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

$r \div s = \{ t[X] \mid t \in r$ and $s \subseteq \pi_Y(\sigma_F(r))$ where
$X = \{A_1, A_2, \ldots, A_q\}$ and
F is the formula $A_1 = t[A_1] \wedge \ldots \wedge A_q = t[A_q]\}$

Division

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

s:

TOPIC
databases
software-engineering
distributed-computing

r:

Lecturer	TOPIC
Jack	databases
Jack	software-engineering
Jack	distributed-computing
Jeffrey	databases
Jeffrey	distributed-computing
Jeffrey	automata theory
Udai	expert-systems
Udai	software-engineering
Jin	databases
Jin	software-engineering
Jin	distributed-computing
Jin	algorithms

Which lecturers can cover all these topics?

Y = {TOPIC} X = {Lecturer}

Relational algebra queries

A relational algebra expression (i.e., query) is a well-formed expression consisting of a finite number of relational algebra operators ($\cup$, $\cap$, $-$, 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.

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

Aggregate Functions

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

If X = ∅, we apply F over the entire relation

NAME	DEPT	SALARY	DAY
Abdu	Computing	2000	Monday
~~Abdu~~	~~Computing~~	~~2000~~	Tuesday
~~Abdu~~	~~Computing~~	~~2000~~	Thursday
Hanna	Computing	1400	Wednesday
~~Hanna~~	~~Computing~~	~~1400~~	Friday
Richard	Computing	1000	Friday
Martine	Maths	1600	Tuesday
Martine	Philosophy	1600	Friday
Reuven	Maths	1500	Wednesday
~~Reuven~~	~~Maths~~	~~1500~~	Thursday
Dan	Linguistics	1000	Tuesday
Ruth	Linguistics	1100	Monday

What is the answer to: COUNT

$COUNT(\pi_{NAME}(r))$ = 7

$COUNT^{DEPT}(\pi_{NAME,DEPT}(r))$

$SUM_{SALARY}{}^{DEPT}(\pi_{NAME,DEPT,SALARY}(r))$

DEPT	SUM
computing	4400
maths	3100
philosophy	1600
linguistics	2100

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

Operator Sets

Some operators are redundant

$r \cap s =$ $r - (r - s)$

also, division

There are other equivalent sets

$\times$ for $\bowtie$

Some things not expressible: transitive closure