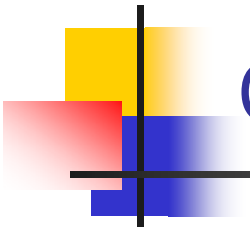
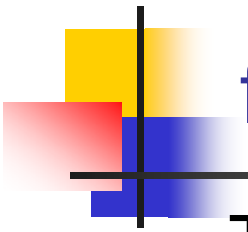
CS589 Principles of DB Systems
Fall 2008
Lecture 4b: Domain Independence and Safety

Lois Delcambre
lmd@cs.pdx.edu
503 725-2405



Goals for today

- Discuss the use of mathematical concepts to define query languages
- Introduce the problems that can occur in tuple/domain calculus queries and Datalog .
- Define syntactic restrictions to avoid the problems
  - Domain independence (for calculus)
  - Safety (for Datalog)



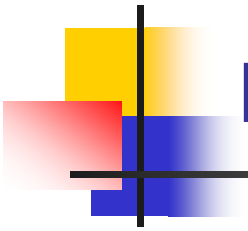
Introduction – formalizing query languages

- To formalize the relational model, we choose:
  - mathematical constructs to represent relations and
  - mathematical expressions to represent query expressions.

Two different formal QLs – both relationally complete

- Relational algebra
  - relations are sets of tuples (defined over domains); a relation is a subset of the cross product of the domains.
  - a query expression is a well-formed combination of relational algebra operators (adapted from set theory). Selection predicates include equality. (simplifying assumption)
- Relational calculus (both tuple and domain)
  - relation schemas are predicate symbols; relation instances are interpretations. (If the tuple <101, “Bob”, “CS”, null> is in the Student table, then Student(101, “Bob”, “CS”, null> is true)
  - a query expression is a set definition, where the condition that must be met is a 1st order predicate calculus well-formed formula (wff). Predicates are relation schemas plus equality (=). (simplifying assumption)

Relational Algebra



- There's no problem ... regarding safety. There's no need to define domain independence.
- That is, any well-formed relational algebra expression is allowed.
- The semantics of a query (what constitutes the query answer) is well-defined; the query answer is finite.

What's the problem with calculus?

Student(name, age)
Teacher(name, age)

$\{n{:}name, a{:}age \mid \neg Student(n, a)\}$

$\{n{:}name, a{:}age \mid \exists n_1(\exists a_1(Student(n,a_1) \vee Teacher(n_1,a)\}$

Are these valid well-formed formulas?
Do we want to consider these to be queries?
What would the query answer be?

What's the problem with calculus?

Student(name, age)
Teacher(name, age)

{n:name, a:age | ¬Student(n, a) }

{n:name, a:age | $\exists n_1(\exists a_1$(Student(n,a_1) v Teacher (n_1,a)}

Are these valid well-formed formulas? Yes
Do we want to consider them to be queries? Maybe not
What would the query answer be? Hmmmm...

What's the query answer?

Student(name, age)

Teacher(name, age)

{n:name, a:age | ¬Student(n, a) }

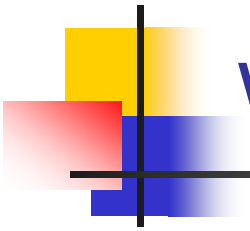
Query answer depends on the domain for n and a.

DOM_j(name) is the set of possible names (at time $_j$)

DOM_j(age) is the set of possible ages (at time $_j$)

If domains are infinite, query answer is not a relation because relations must be finite.

If domains are finite, then changing the domain changes the query answer.



What's the query answer?

$\{n{:}name, a{:}age \mid \exists n_1(\exists a_1(Student(n,a_1) \vee Teacher(n_1,a)\}$

What's wrong here?

If $Student(n, a_1)$ is true and $Teacher(n_1,a)$ is false, the formula is true. But what are the values for a?

And similarly, if $Teacher(n_1,a)$ is true and $Student(n, a_1)$ is false, what are the values for n?

In both cases, values will depend on the domains.

Domain Independence

(informal definition) A query is domain independent if for one database instance, the answer to the query does not change if the domain changes.

The two main problems that cause domain dependence:

- ¬F
  - where a variable in F does not occur positively in some atomic formula elsewhere in the query
- F1 v F2
  - such that they have different free variables

Determining whether a query is domain independent is *undecidable*.

So…we first define “positive” and “negative” occurrences of a variable

Positive occurrences of a variable	Negative occurrences of a variable
$R(y_1, y_2, \ldots, x, \ldots, y_k)$	x does not appear in formula F at all
$x=v$ where v is a constant	$x=y$ where y is a variable, y occurs negatively
$\neg F$ if x occurs negatively in F	$\neg F$ if x occurs positively in F
$F_1 \wedge F_2$ if x occurs positively in F_1 or positively in F_2	$F_1 \wedge F_2$ if x occurs negatively in F_1 and negatively in F_2
$F_1 \vee F_2$ if x occurs positively in F_1 and positively in F_2	$F_1 \vee F_2$ if x occurs negatively in F_1 or negatively in F_2
$F_1 \rightarrow F_2$ if x occurs negatively in F_1 and positively in F_2	$F_1 \rightarrow F_2$ if x is positive in F_1 and negative in F_2
$\exists y{:}A$ (F) if $x \neq y$, x occurs pos. in F	$\exists y{:}A$ (F) if $x \neq y$, x occurs pos. in F

Reminder: Definition of tuple calculus syntax (Ramakrishnan & Gehrke)

Rel is a relation name, *R* and *S* are tuple variables, *a* is an attribute of *R*, *b* is an attribute of *S*, *op* is one of {<, >, =, ≤, ≥, ≠}

An atomic formula is one of the following:

R ∈ *Rel, R.a op S.b, R.a op constant* (or *constant op R.a*)

A formula is:

any atomic formula

¬F, (F), F1^F2, F1vF2, F1⇒F2 (if F, F1, and F2 are formula)

∃R(F(R)) where F is formula with R a free variable

Focus on this → ∀R(F(R)) where F is a formula with R a free variable

A tuple calculus query is an expression of the form:

{ T | F(T) } where T is the only free variable in F

Definition of tuple calculus syntax (Ramakrishnan & Gehrke)

Consider the formula:

$\forall R(F(R))$ where F is a formula with R a free variable

IF F is the formula $\forall x{:}A\ (G(x_1, x_2, ..., x, ..., x_n)$, then $<v_1, v_2, ..., v_{i-1}, v_{i+1}, ...v_n>$ satisifes F if for all constants $v_i \in DOM(A)$, when v_i is substituted for x, $<v_1, v_2, ..., v_n>$ satisifes G.

Can we write relational algebra divide queries?

Student(s-id, name) Taken (s-id, c-id) Core(c-id)

$\{s \mid s \in Student(\forall c \in Core(\exists t \in Taken(t.c\text{-}id = c.d\text{-}id \wedge t.s\text{-}id = s.s\text{-}id)))\}$

But $\forall r \in R(F)$ is actually shorthand for $\forall r(r \in R \rightarrow F)$

$\{s \mid s \in Student(\forall c(c \in Core \rightarrow (\exists t \in Taken(t.c\text{-}id = c.d\text{-}id \wedge t.s\text{-}id = s.s\text{-}id)))\}$

But $x \rightarrow y$ is equivalent to $\neg x \vee Y$

$\{s \mid s \in Student(\forall c(\neg(c \in Core) \vee (\exists t \in Taken(t.c\text{-}id = c.d\text{-}id \wedge t.s\text{-}id = s.s\text{-}id)))\}$

previous slide is repeated here...

Positive occurrences of a variable	Negative occurrences of a variable
$R(y_1, y_2, \ldots, x, \ldots, y_k)$	x does not appear in formula F
$x=v$ where v is a constant	$x=y$ where y is a variable
	F_1 and
$F_1 \vee F_2$ if x is positive in F_1 and positive in F_2	$F_1 \vee F_2$ if x is negative in F_1 or negative in F_2
$F_1 \rightarrow F_2$ if x is negative in F_1 and positive in F_2	$F_1 \rightarrow F_2$ if x is positive in F_1 and negative in F_2
$\exists y:A$ (F) if $x \neq y$, x pos. in F	$\exists y:A$ (F) if $x \neq y$, x pos. in F

I think there is one detail missing in this. If x=y and y is a variable and y occurs positively in F, then I think x occurs positively.

Syntatic restriction: "Allowed" domain calculus queries

- A domain calculus query is allowed if its formula is allowed.
- A domain calculus formula F is allowed if all of these conditions hold:
  - Every free variable in F occurs positively in F (What are the free variables?)
  - For every subformula $\exists x{:}A\ (G)$ of F, x is positive in G
  - For every subformula $\forall x{:}A\ (G)$ of F, x is negative in G

Every allowed domain calculus formula is domain-independent

- left as an exercise ...

The problems with Datalog

Consider the following queries:

Result(x) :- Course(x_1, x_2).

Answer(x_1, x_2) :- $\neg$Course(x_1, x_2).

Ans(x_1) :- Course(x_1, x_2), x_5=x_6.

These Datalog queries either have infinite answers or the evaluation procedure would run into problems. We say that these queries are not *safe*.

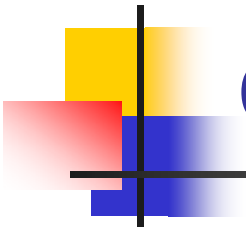
So, we define positive and negative variables and safe Datalog rules

A Datalog rule is safe if all the variables appearing in the literals of C (both head and body) occur positively in C.

A Datalog program is safe if all the rules are safe.

A variable x occurs positively if x appears in:

- $R(y_1, y_2, \ldots, x, \ldots, y_k)$ where this is a positive literal in the body of the rule
- x=v where v is a constant
- x=y or y=x where y is a variable that occurs positively in C



Comments

- In a safe Datalog rule, all the variables in the head of the rule must occur in one or more literals in the body.
- All the variables that occur in a negative literal in the body of a safe Datalog rule, must occur positively in one or more atomic formulae in its body.
- Safe Datalog query $\rightarrow$ query answer is always finite.
- Allowable domain calculus query $\rightarrow$ query answer is always finite.