

Formula defines relation

- Free variables in a formula take on the values of tuples
- A tuple is in the defined relation if and only if when substituted for a free variable, it satisfies (makes true) the formula

Free variable:

 $\exists x, \forall x \text{ bind } x - \text{truth or falsehood no longer depends on a specific value of x lf x is not bound it is free$

Quantifiers

There exists: $\exists x (f(x))$ for formula f with free variable x

• Is true if there is *some tuple* which when substituted for x makes f true

For all: $\forall x (f(x))$ for formula f with free variable x

• Is true if *any tuple* substituted for x makes f true i.e. all tuples when substituted for x make f true

Example

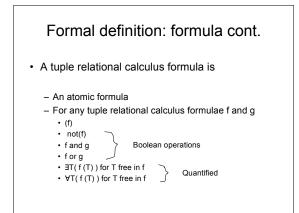
{T | $\exists A \exists B$ (A ε Winners and B ε Winners and

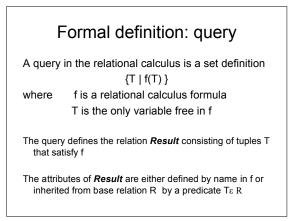
- $\begin{aligned} A[name] = T[name] \text{ and } A[tournament] = T[tournament] \text{ and } \\ B[tournament] = T[tournament] \text{ and } T[name2] = B[name]) \ \ \} \end{aligned}$
- T not constrained to be element of a named relation
- Result has attributes defined by naming them in the formula: T[name], T[tournament], T[name2]
 so scheme for result. (name tournament, name2)
 - so schema for result: (name, tournament, name2) unordered
- Tuples T in result have values for (name, tournament, name2) that satisfy the formula
- What is the resulting relation?

Formal definition: formula

· A tuple relational calculus formula is

- An atomic formula (uses predicate and constants):
 - $T \in R$ where
 - T is a variable ranging over tuples
 - R is a named relation in the database a *base relation*
 - T[a] op W[b] where
 - a and b are names of attributes of T and W, respectively, - op is one of $< > = \neq \le \ge$
 - T[a] **op** constant
 - constant op T[a]

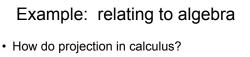




Some abbreviations for logic

- (p => q) equivalent to ((not p) or q)
- $\forall x(f(x)) \text{ equiv. to } not(\exists x(not f(x)))$
- $\exists x(f(x)) equiv. to not(\forall x(not f(x)))$
- $\forall x \in S(f)$ equiv. to $\forall x ((x \in S) \Rightarrow f)$
- $\exists x \in S (f)$ equiv. to $\exists x ((x \in S) and f)$

note departure from Silberschatz et. al.



}

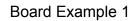
 How do projection in calculus π_{name,year} (Winners)

becomes

```
{ T | \exists W (W \epsilon Winners \land
T[name] = W[name] \land
T[year] = W[year] )
```

∧ denotes AND

Board examples



students: (<u>SS#</u>, name, PUaddr, homeAddr, classYr) employees: (<u>SS#</u>, name, addr, startYr) assignment: (<u>position</u>, division, SS#, managerSS#) study: (<u>SS#</u>, academic_dept., adviser)

find SS#, name, and classYr of all student employees

Board Example 2

students: (<u>SS#</u>, name, PUaddr, homeAddr, classYr) employees: (<u>SS#</u>, name, addr, startYr) assignment: (<u>position</u>, division, SS#, managerSS#) study: (<u>SS#</u>, academic_dept., adviser)

find (student, manager) pairs where both are students - report SS#s

Board Example 3

students: (<u>SS#</u>, name, PUaddr, homeAddr, classYr) employees: (<u>SS#</u>, name, addr, startYr) assignment: (<u>position</u>, division, SS#, managerSS#) study: (<u>SS#</u>, academic_dept., adviser)

find names of all CS students working for the library (library a division)

Board Example 4

students: (<u>SS#</u>, name, PUaddr, homeAddr, classYr) employees: (<u>SS#</u>, name, addr, startYr) assignment: (<u>position</u>, division, SS#, managerSS\$)

division foreign key referencing PUdivision

study: (<u>SS#</u>, academic_dept., adviser) SS# foreign key referencing students

PUdivision: (division_name, address, director)

Find academic departments that have students working in all divisions

Evaluating query in calculus

Declarative – how build new relation {x|f(x)}?

- · Go through each candidate tuple value for x
- Is f(x) true when substitute candidate value for free variable x?
- · If yes, candidate tuple is in new relation
- If no, candiate tuple is out

What are candidates?

- Do we know domain of x?
- Is domain finite?

Problem

- Consider {T | not (T ε Winners) }
 Wide open what is schema for Result?
- Consider {T | ∀S ((S ε Winners) => (not (T[name] = S[name] and T[year] = S[year]))) }
 Now Result:(name, year) but universe is infinite

Don't want to consider infinite set of values

Constants of a database and query

Want consider only finite set of values - What are constants in database and guery?

Define:

- Let I be an instance of a database
- A specific set of tuples (relation) for each base relational schema
- · Let Q be a relational calculus query
- Domain (I,Q) is the set of all constants in Q or I
- Let Q(I) denote the relation resulting from applying Q to I

Safe query

A query Q on a relational database with base schemas $\{R_i\}$ is safe if and only if:

1. for all instances I of {R_i} , any tuple in Q(I) contains only values in Domain(I, Q)

Means at worst candidates are all tuples can form from finite set of values in Domain(I, Q)

Safe query: need more

Require testing quantifiers has finite universe:

- For each ∃T(p(T)) in the formula of Q, if p(t) is true for tuple t, then attributes of t are in Domain(I, Q)
- 3. For each ∀T(p(T)) in the formula of Q, if *t* is a tuple containing a constant not in Domain(I,Q), then p(*t*) is true
- => Only need to test tuples in Domain(I,Q)

Safe query: all conditions

A query Q on a relational database with base schemas $\{R_i\}$ is safe if and only if:

- 1. for all instances I of $\{R_i\}$, any tuple in Q(I) contains only values in $\text{Domain}(I,\,Q)$
- 2. For each $\exists T(p(T))$ in the formula of Q, if p(t) is true for tuple *t*, then attributes of *t* are in Domain(I, Q)
- For each ∀T(p(T)) in the formula of Q, if t is a tuple containing a constant not in Domain(I,Q), then p(t) is true

Equivalence Algebra and Calculus

The relational algebra and the tuple relational calculus **over safe queries** are equivalent in expressiveness

Domain relational calculus

- Similar but variables range over domain values (i.e. attribute values) not tuples
- Is equivalent to tuple relational calculus when both restricted to safe expressions

Example:

N, M range over Winners.name

- K ranges over Winners.tournament
- Y, Z range over Winners.year

Summary

- The relational calculus provides an alternate way to express queries
- A formal model based on logical formulae and set theory
- Equivalence with algebra means can use either or both – but only one for formal proofs
- Next we will see that SQL borrows from both