Database Management System 14

Relational Calculus

Chittaranjan Pradhan
School of Computer Engineering,
KIIT University

Relational Calculus

Relational Calculus

Relational calculus is non-procedural

In relational calculus, a query is solved by defining a solution relation in a single step

Relational calculus is mainly based on the well-known propositional calculus, which is a method of calculating with sentences or declarations

Various types of relational calculus are:

- Tuple Relational Calculus (TRC)
- Domain Relational Calculus (DRC)

Tuple Relational Calculus (TRC)

Tuple Relational Calculus (TRC)

A tuple variable is a variable that takes on tuples of a particular relation schema as values

A tuple relational calculus query has the form:
{T/ P(T)}
The result of this query is the set of all tuples t for which the formula P(T) evaluates to TRUE with T = t
Sailors (sid, sname, rating, age)

Query: Find all the sailors with a rating above 4
$\{S/S \in \text{Sailors} \wedge S.\text{rating} > 4\}$

Tuple Relational Calculus (TRC)...

Tuple Relational Calculus (TRC)

Let Rel $\rightarrow$ be a relation name, R, S $\rightarrow$ be the tuple variables, a $\rightarrow$ an attribute of R, b $\rightarrow$ an attribute of S, op $\rightarrow$ operator in the set {<, $\leq$, >, $\geq$, =, $\neq$}. An Atomic formula is one of the following:

- R $\in$ Rel
- R.a op S.b
- R.a op Constant or Constant op R.a

*To represent the join and division of relational algebra by relational calculus, we need quantifiers such as: **existential for join** and **universal for division***

A quantifier quantifies or indicates the quantity of something

The existential quantifier (**$\exists$**) states that at least one instance of a particular type of thing exist
Similarly, the universal quantifier(**$\forall$**) states that some condition applies to all or to every row of some type

Tuple Relational Calculus (TRC)...

Tuple Relational Calculus (TRC)

A formula is recursively defined by using the following rules:

- Any atomic formula
- If p and q are formulae, then $\neg$ p, p $\wedge$ q, p $\vee$ q, or p $\Rightarrow$ q are also formulae
- If p is a formula that contains T as a variable, then $\exists$ T(p) and $\forall$ T(p) are also formulae

The quantifiers $\exists$ and $\forall$ are said to **bind** the tuple variable R; whereas a variable is said to be **free** in a formula if the formula does not contain an occurrence of a quantifier that binds it

In most of the queries, the output is shown by using the free variables

Safe Expressions

Safe Expressions

Whenever we use universal quantifiers or existential quantifiers in a calculus expression, we must make sure that the resulting expression makes sense

A safe expression in relational calculus is one that is guaranteed to yield a finite number of tuples as its result; otherwise, the expression is called unsafe

That means, an expression is said to be safe if all values in its result are from the domain of the expression

Queries

Sailor Database

Sailors(<u>sid</u>, sname, rating, age)

Boats(<u>bid</u>, bname, color)

Reserves(<u>sid</u>, <u>bid</u>, <u>day</u>)

Query: Find the names & ages of sailors with a rating above 4

{T/∃S ∈ Sailors (S.rating >4 ∧ T.sname=S.sname ∧ T.age= S.age)}

Query: Find the sailor name, boat id & reservation date for each reservation

{T/∃R ∈ Reserves ∃S ∈ Sailors (R.sid = S.sid ∧ T.sname=S.sname ∧ T.bid=R.bid ∧ T.day=R.day)}

Query: Find the names of sailors who have reserved boat 111

{T/∃R ∈ Reserves ∃S ∈ Sailors (R.sid = S.sid ∧ R.bid=111 ∧ T.sname=S.sname)}

Queries...

Query: Find the names of sailors who have reserved a green boat

{T/∃S ∈ Sailors ∃R ∈ Reserves(R.sid = S.sid ∧ T.sname=S.sname ∧ ∃B ∈ Boats (B.bid=R.bid ∧ B.color='green'))}

This query can also be written as:
{T/∃S ∈ Sailors ∃R ∈ Reserves ∃B ∈ Boats(R.sid = S.sid ∧ B.bid=R.bid ∧ B.color='green' ∧ T.sname=S.sname)}

Query: Find the names of sailors who have reserved at least 2 boats

{T/∃S ∈ Sailors ∃R1 ∈ Reserves ∃R2 ∈ Reserves(S.sid=R1.sid ∧ R1.sid= R2.sid ∧ R1.bid ≠ R2.bid ∧ T.sname=S.sname)}

Queries...

Query: Find the names of sailors who have reserved all boats

$\{T / \exists S \in Sailors\ \forall B \in Boats(\exists R \in Reserves\ (S.sid{=}R.sid \wedge R.bid{=}\ B.bid \wedge T.sname{=}\ S.sname))\}$

Query: Find sailors who have reserved all green boats

$\{S / S \in Sailors \wedge \forall B \in Boats\ (B.color{=}'green' \Rightarrow (\exists\ R \in Reserves\ (S.sid{=}R.sid \wedge R.bid{=}\ B.bid))\}$

Domain Relational Calculus (DRC)

Domain Relational Calculus (DRC)

In tuple relational calculus, the variables range over the tuples whereas in domain relational calculus, the variables range over the domains

The domain variables are the ones which range over the underlying domains instead of over the relations

The domain relational calculus query has the form:
$\{<x_1, x_2, \ldots x_n> \mid P(x_1, x_2, \ldots x_n)\}$
where x_i is either a domain variable or a constant and $P(x_1, x_2, \ldots x_n)$ is the domain relational calculus formula whose only free variables are the variables among the xi, $1 \leq i \leq n$

The result of this query is the set of all tuples $<x_1, x_2, \ldots x_n>$ for which the formula evaluates to TRUE

Domain Relational Calculus (DRC)...

Domain Relational Calculus (DRC)

Let Rel $\rightarrow$ be a relation name, X, Y $\rightarrow$ be the domain variables, op $\rightarrow$ an operator in the set {<, $\leq$, >, $\geq$, =, $\neq$}. An Atomic formula in domain relational calculus is one of the following:

- $<x_1, x_2, \ldots x_n> \in$ Rel
- X op Y
- X op Constant or Constant op X

A formula is recursively defined by using the following rules:

- Any atomic formula
- If p and q are formulae, then $\neg$p, p $\wedge$ q, p $\vee$ q, or p $\Rightarrow$ q are also formulae
- If p is a formula that contains X as a domain variable, then $\exists$X(p) and $\forall$ X(p) are also formulae

The quantifiers $\exists$ & $\forall$ are said to bind the domain variable X. Whereas a variable is said to be free in a formula if the formula does not contain an occurrence of a quantifier that binds it

Queries

Sailor Database

Sailors(sid, sname, rating, age)

Boats(bid, bname, color)

Reserves(sid, bid, day)

Query: Find all sailors with a rating above 7

$\{<I, N, T, A> / <I, N, T, A> \in \text{Sailors} \wedge T > 7\}$

Query: Find the names of sailors who reserved boat 111

$\{<N> / \exists I, T, A (<I, N, T, A> \in \text{Sailors} \wedge \exists <Ir, Br, D> \in \text{Reserves} (Ir=I \wedge Br=111))\}$

This query can also be written as:

$\{<N> / \exists I, T, A (<I, N, T, A> \in \text{Sailors} \wedge \exists <Ir, Br, D> \in \text{Reserves} (Ir=I \wedge Br=111))\}$

or

$\{<N> / \exists I, T, A (<I, N, T, A> \in \text{Sailors} \wedge \exists D(<I, 111, D> \in \text{Reserves}))\}$

Queries...

Query: Find the names of sailors who have reserved a green boat

$$\{<N>/\exists I, T, A\ (<I, N, T, A> \in \text{Sailors} \wedge <I, Br, D> \in \text{Reserves} \wedge \exists <Br, Bn, \text{'green'}> \in \text{Boats})\}$$

Query: Find the names of sailors who have reserved at least 2 boats

$$\{<N>/\exists I, T, A\ (<I, N, T, A> \in \text{Sailors} \wedge \exists Br1, Br2, D1, D2(<I, Br1, D1> \in \text{Reserves} \wedge <I, Br2, D2> \in \text{Reserves} \wedge Br1 \neq Br2))\}$$

Query: Find the names of sailors who have reserved all boats

$$\{<N>/\exists I, T, A(<I, N, T, A> \in \text{Sailors} \wedge \forall <B, Bn, C> \in \text{Boats}\ (\exists <Ir, Br, D> \in \text{Reserves}\ (\ I{=}Ir \wedge Br{=}B)))\}$$

Query: Find sailors who have reserved all green boats

$$\{<I,N,T,A>/<I,N,T,A> \in \text{Sailors} \wedge \forall <B, Bn, C> \in \text{Boats}\ (C{=}\text{'green'} \Rightarrow \exists <Ir, Br, D> \in \text{Reserves}\ (\ I{=}Ir \wedge Br{=}B)))\}$$