UNIT 2

Lecture 23

Tuple Relational Calculus

Relational Languages

- Relational Algebra
  - It is an procedural language.
- Relational Calculus
  - It is an non procedural language (or Declarative Language)

Relational Calculus

1. Tuple Relational Calculus (TRC)
   - In this query language we work on tuple variable which ranges over relations.
2. Domain Relational Calculus (DRC)
   - In this query language we work on domain variables which ranges over relations.

Tuple Relational Calculus

- When we write a relational-algebra expression, we provide a sequence of procedures that generates the answer to our query.
- The tuple relational calculus, by contrast, is a **nonprocedural** query language. It describes the desired information without giving a specific procedure for obtaining that information.
- A query in the tuple relational calculus is expressed as

  $$\{t \mid P(t)\}$$

  that is, it is the set of all tuples *t* such that predicate *P* is true for *t*.
- we use $t[A]$ to denote the value of tuple *t* on attribute *A*, and
- we use $t \in r$ to denote that tuple *t* is in relation *r*.

TRC Example : 1

Display all the information of students whose branch is CSE.

RA : $\sigma_{\text{branch = “CSE”}}$ (STUDENT)

SQL > select * from student where br = ‘CSE’;

TRC : { t | t ϵ $STUDENT\ \wedge t[branch] = "CSE"$}

STUDENT

Rollno	Sname	Sem	Branch	Marks	Pno
1	RAM	3	CSE	40	121
2	SHYAM	5	CSE	50	122
3	MOHAN	7	CSE	55	123
4	GOPAL	5	IT	65	121
5	RINKI	3	MECH	40	122
6	PINKI	3	ETC	90	123

OUTPUT

Rollno	Sname	Sem	Branch	Marks	Pno
1	RAM	3	CSE	40	121
2	SHYAM	5	CSE	50	122
3	MOHAN	7	CSE	55	123

TRC Example : 2

Display all the information of students whose sem is 3.

RA : $\sigma_{sem = 3}$ (STUDENT)

SQL > select * from student where sem = 3;

TRC : { t | t ϵ $STUDENT \ \wedge t[sem] = 3\}$

STUDENT

Rollno	Sname	Sem	Branch	Marks	Pno
1	RAM	3	CSE	40	121
2	SHYAM	5	CSE	50	122
3	MOHAN	7	CSE	55	123
4	GOPAL	5	IT	65	121
5	RINKI	3	MECH	40	122
6	PINKI	3	ETC	90	123

OUTPUT

Rollno	Sname	Sem	Branch	Marks	Pno
1	RAM	3	CSE	40	121
5	RINKI	3	MECH	40	122
6	PINKI	3	ETC	90	123

TRC Example : 3

Display all the information of students whose marks is greater than 50.

RA : $\sigma_{\text{marks} > 50}$ (STUDENT)

SQL > select * from student where marks > 50;

TRC : { t | t ϵ $STUDENT \ \wedge t[marks] > 50\}$

STUDENT

Rollno	Sname	Sem	Branch	Marks	Pno
1	RAM	3	CSE	40	121
2	SHYAM	5	CSE	50	122
3	MOHAN	7	CSE	55	123
4	GOPAL	5	IT	65	121
5	RINKI	3	MECH	40	122
6	PINKI	3	ETC	90	123

OUTPUT

Rollno	Sname	Sem	Branch	Marks	Pno
3	MOHAN	7	CSE	55	123
4	GOPAL	5	IT	65	121
6	PINKI	3	ETC	90	123

TRC Example : 4

Display all the information of students whose branch is CSE or sem is 3.

RA : $\sigma_{\text{branch} = \text{"CSE"} \vee sem = 3}$ (STUDENT)

SQL > select * from student where branch = ‘CSE’ or sem = 3;

TRC : { t | t ϵ $STUDENT \wedge t[branch] = \text{"CSE"} \vee t[sem] = 3\}$

STUDENT

Rollno	Sname	Sem	Branch	Marks	Pno
1	RAM	3	CSE	40	121
2	SHYAM	5	CSE	50	122
3	MOHAN	7	CSE	55	123
4	GOPAL	5	IT	65	121
5	RINKI	3	MECH	40	122
6	PINKI	3	ETC	90	123

OUTPUT

Rollno	Sname	Sem	Branch	Marks	Pno
1	RAM	3	CSE	40	121
2	SHYAM	5	CSE	50	122
3	MOHAN	7	CSE	55	123
5	RINKI	3	MECH	40	122
6	PINKI	3	ETC	90	123

TRC Example : 5

Display all the information of students whose branch is CSE and sem is 3.

RA : $\sigma_{branch\ =\ \text{"CSE"}\ \wedge sem\ =\ 3}$ (STUDENT)

SQL > select * from student where branch = ‘CSE’ and sem = 3;

TRC : { t | t $\epsilon\ STUDENT\ \wedge t[branch] = \text{"CSE"}\ \wedge\ t[sem] = 3\}$

STUDENT

Rollno	Sname	Sem	Branch	Marks	Pno
1	RAM	3	CSE	40	121
2	SHYAM	5	CSE	50	122
3	MOHAN	7	CSE	55	123
4	GOPAL	5	IT	65	121
5	RINKI	3	MECH	40	122
6	PINKI	3	ETC	90	123

OUTPUT

Rollno	Sname	Sem	Branch	Marks	Pno
1	RAM	3	CSE	40	121

TRC Example : 6

Display the names of all the students.

RA : Π_{sname} (STUDENT)

SQL > select distinct sname from student;

TRC : { t | $\exists\ s \in STUDENT\ (t[sname] = s[sname])\}$

STUDENT

Rollno	Sname	Sem	Branch	Marks	Pno
1	RAM	3	CSE	40	121
2	SHYAM	5	CSE	50	122
3	MOHAN	7	CSE	55	123
4	GOPAL	5	IT	65	121
5	RINKI	3	MECH	40	122
6	PINKI	3	ETC	90	123

OUTPUT

Sname
RAM
SHYAM
MOHAN
GOPAL
RINKI
PINKI

TRC Example : 7

Display the name and semester value of all the students.

RA : $\Pi_{\text{sname, sem}}$ (STUDENT)

SQL > select distinct sname, sem from student;

TRC : { t | $\exists\, s \in STUDENT\ (t[sname] = s[sname]\ \wedge t[sem] = s[sem])$}

STUDENT

Rollno	Sname	Sem	Branch	Marks	Pno
1	RAM	3	CSE	40	121
2	SHYAM	5	CSE	50	122
3	MOHAN	7	CSE	55	123
4	GOPAL	5	IT	65	121
5	RINKI	3	MECH	40	122
6	PINKI	3	ETC	90	123

OUTPUT

Sname	Sem
RAM	3
SHYAM	5
MOHAN	7
GOPAL	5
RINKI	3
PINKI	3

TRC Example : 8

Display the name of all the students of CSE branch.

RA : $\Pi_{sname}(\sigma_{branch = \text{“CSE”}}(STUDENT))$

SQL > select distinct sname from (select * from student where branch = ‘CSE’);

TRC : TRC : { t | $\exists\, s \in STUDENT\ (t[sname] = s[sname]\ \wedge s[branch] = \text{"}CSE\text{"})\}$

STUDENT

Rollno	Sname	Sem	Branch	Marks	Pno
1	RAM	3	CSE	40	121
2	SHYAM	5	CSE	50	122
3	MOHAN	7	CSE	55	123
4	GOPAL	5	IT	65	121
5	RINKI	3	MECH	40	122
6	PINKI	3	ETC	90	123

OUTPUT

Sname
RAM
SHYAM
MOHAN

TRC Example : 9

Display the name of branches in which project 121 or 122 or both are running.

RA : $\Pi_{branch}(\sigma_{pno = 121}(STUDENT)) \cup \Pi_{branch}(\sigma_{pno = 122}(STUDENT))$

SQL > select distinct branch from student where pno = 121 Union select distinct branch from student where pno = 122;

TRC : $\{ t \mid \exists\, s \in STUDENT\ (t[branch] = s[branch]\ \wedge s[pno] = 121)$
$\vee \exists\, u \in STUDENT\ (t[branch] = u[branch]\ \wedge u[pno] = 122)\}$

STUDENT

Rollno	Sname	Sem	Branch	Marks	Pno
1	RAM	3	CSE	40	121
2	SHYAM	5	CSE	50	122
3	MOHAN	7	CSE	55	123
4	GOPAL	5	IT	65	121
5	RINKI	3	MECH	40	122
6	PINKI	3	ETC	90	123

Branch
CSE
IT

Branch
CSE
MECH

OUTPUT

Branch
CSE
IT
MECH

TRC Example : 10

Display the name of branches in which project 121 and 122 are running.

RA : $\Pi_{branch}(\sigma_{pno = 121}(STUDENT)) \cap \Pi_{branch}(\sigma_{pno = 122}(STUDENT))$

SQL > select distinct branch from student where pno = 121 intersect select distinct branch from student where pno = 122;

TRC : $\{ t \mid \exists\, s \in STUDENT\ (t[branch] = s[branch]\ \wedge s[pno] = 121)$

$\wedge \exists\, u \in STUDENT\ (t[branch] = u[branch]\ \wedge u[pno] = 122)\}$

STUDENT

Rollno	Sname	Sem	Branch	Marks	Pno
1	RAM	3	CSE	40	121
2	SHYAM	5	CSE	50	122
3	MOHAN	7	CSE	55	123
4	GOPAL	5	IT	65	121
5	RINKI	3	MECH	40	122
6	PINKI	3	ETC	90	123

Branch
CSE
IT

Branch
CSE
MECH

OUTPUT

Branch
CSE

TRC Example : 11

Display the name of branches in which project 121 is running but 122 is not.

RA : $\Pi_{branch}(\sigma_{pno = 121}(STUDENT)) - \Pi_{branch}(\sigma_{pno = 122}(STUDENT))$

SQL > select distinct branch from student where pno = 121 minus / except select distinct branch from student where pno = 122;

TRC : $\{ t \mid \exists s \in STUDENT\ (t[branch] = s[branch] \land s[pno] = 121)$

$\land \neg \exists u \in STUDENT\ (t[branch] = u[branch] \land u[pno] = 122)\}$

STUDENT

Rollno	Sname	Sem	Branch	Marks	Pno
1	RAM	3	CSE	40	121
2	SHYAM	5	CSE	50	122
3	MOHAN	7	CSE	55	123
4	GOPAL	5	IT	65	121
5	RINKI	3	MECH	40	122
6	PINKI	3	ETC	90	123

Branch
CSE
IT

Branch
CSE
MECH

OUTPUT

Branch
IT

TRC Example : 12

STUDENT

Rollno	Sname	Sem	Branch	Marks	Pno
1	RAM	3	CSE	40	121
2	SHYAM	5	CSE	50	122
3	MOHAN	7	CSE	55	123
4	GOPAL	5	IT	65	121
5	RINKI	3	MECH	40	122
6	PINKI	3	ETC	90	123

PROJECT

Pno	Pname	Duration
121	P1	10
122	P2	20
123	P3	30

$\Pi_{sname, pname}$ (STUDENT $\bowtie_{student.pno = project.pno}$ PROJECT)

Sname	Pname
RAM	P1
GOPAL	P1
SHYAM	P2
RINKI	P2
MOHAN	P3
PINKI	P3

SQL > select distinct sname, pname from student inner join project on student.pno = project.pno; **[SQL 99 syntax]**

SQL > select distinct sname, pname from student, project where student.pno = project.pno;

TRC : $\{ t \mid \exists s \in STUDENT\ (t[sname] = s[sname]$
$\wedge\ \exists p \in PROJECT\ (t[pname] = p[pname]\ \wedge s[pno] = p[pno]))\}$

TRC Example : 13

STUDENT

Rollno	Sname	Sem	Branch	Marks	Pno
1	RAM	3	CSE	40	121
2	SHYAM	5	CSE	50	122
3	MOHAN	7	CSE	55	123
4	GOPAL	5	IT	65	121
5	RINKI	3	MECH	40	122
6	PINKI	3	ETC	90	123

PROJECT

Pno	Pname	Duration
121	P1	10
122	P2	20
123	P3	30

$\Pi_{\text{branch, pno}}$ (STUDENT) $\div$ Π_{pno} (PROJECT)

OUTPUT

Branch
CSE

Equivalent SQL Query

SQL > select distinct branch from student s1
where not exists (select pno from project p
where not exists (select branch from student s2
where s1.branch = s2.branch and p.pno = s2.pno));

TRC : { t | $\exists\, s \in STUDENT\ (t[branch] = s[branch]$
$\wedge\ \forall\, p \in PROJECT\ \rightarrow\ (s[pno] = p[pno])\}$

Formal Definition of Tuple Relational Calculus

- A tuple-relational-calculus expression is of the form

  $\{t \mid P(t)\}$

  where *P* is a *formula*.
- Several tuple variables may appear in a formula.
- A tuple variable is said to be a *free variable* unless it is quantified by a $\exists$ or $\forall$.
- Thus, in

TRC : $\{ t \in STUDENT \ \wedge \ \exists\, p \in PROJECT\ (t[pno] = p[pname])\}$

t is a free variable. Tuple variable *p* is said to be a *bound* variable.

Formal Definition of Tuple Relational Calculus

- A tuple-relational-calculus formula is built up out of *atoms*.
- An atom has one of the following forms:
  - *s* ∈ *r*, where *s* is a tuple variable and *r* is a relation (we do not allow use of the ∉ operator)
  - *s*[*x*] Θ *u*[*y*], where *s* and *u* are tuple variables, *x* is an attribute on which *s* is defined, *y* is an attribute on which *u* is defined, and Θ is a comparison operator (<, ≤, =, =, >, ≥); we require that attributes *x* and *y* have domains whose members can be compared by Θ
  - *s*[*x*] Θ *c*, where *s* is a tuple variable, *x* is an attribute on which *s* is defined, Θ is a comparison operator, and *c* is a constant in the domain of attribute *x*

Formal Definition of Tuple Relational Calculus

- We build up formulae from atoms by using the following rules:
  - An atom is a formula.
  - If $P1$ is a formula, then so are $\neg P1$ and $(P1)$.
  - If $P1$ and $P2$ are formulae, then so are $P1 \vee P2$, $P1 \wedge P2$, and $P1 \Rightarrow P2$.
  - If $P1(s)$ is a formula containing a free tuple variable s, and r is a relation, then

    $\exists\, s \in r\ (P1(s))$ and $\forall\, s \in r\ (P1(s))$

    are also formulae.

Formal Definition of Tuple Relational Calculus

- As we could for the relational algebra, we can write equivalent expressions that are not identical in appearance.
- In the tuple relational calculus, these equivalences include the following three rules:
  1. $P1 \land P2$ is equivalent to $\neg(\neg(P1) \lor \neg(P2))$.
  2. $\forall t \in r\ (P1(t))$ is equivalent to $\neg \exists t \in r\ (\neg P1(t))$.
  3. $P1 \Rightarrow P2$ is equivalent to $\neg(P1) \lor P2$.

Safety of Expressions

- There is one final issue to be addressed. A tuple-relational-calculus expression may generate an infinite relation.
- Suppose that we write the expression

  $\{t \mid \neg (t \in STUDENT)\}$

- There are infinitely many tuples that are not in STUDENT.
- Most of these tuples contain values that do not even appear in the database! Clearly, we do not wish to allow such expressions.
- This query is known as unsafe query in tuple relational calculus.

For Video lecture on this topic please subscribe to my youtube channel.

The link for my youtube channel is

https://www.youtube.com/channel/UCRWGtE76JlTp1iim6aOTRuw?sub_confirmation=1