The Tuple Relational Calculus (TRC)

* TRC is another formal query language for the relational model.
* TRC is nonprocedural, as it does not provide a description of how to (in what order) to evaluate a query, and only specifies what to retrieve.
* It has the same expressive power as relational algebra, i.e. any retrieval that can be specified in relational algebra can also be specified in TRC.
* Relational calculus is important for two reasons:
  1. It has a firm basis in mathematical logic.
  2. SQL has some of its foundation in TRC.

How to write TRC expressions:

* TRC is based on specifying a number of tuple variables.
* A TRC expression has the form:

  $\{t \mid \text{COND}(t)\}$

  where t is the tuple variable and COND(t) is a boolean expression, which evaluates to TRUE or FALSE.
* The general form of a TRC expression is:

  $\{t_1.A_j, t_2.A_k, \ldots, t_n.A_m \mid \text{COND}(t_1, t_2, t_3, \ldots, t_n, t_{n+1}, \ldots, t_{n+m})\}$

  where $t_1, t_2, \ldots t_n$ are tuple variables and COND is a condition involving those tuple variables. You can think of tuple variables as tuple instances in a table (relation) and you access individual attribute values for those tuples using the ".attribute_name" notation. The first part of the TRC expression specifies which attribute values you would like to list in the query result (this is like a projection in relational algebra) and the COND part specifies the filters to apply when running your query (this is like a selection, join etc. in relational algebra). If you do not specify a list of attributes for a tuple variable on the first part of the expression, all attribute values for that tuple are printed.

* Every formula (COND) consists of predicate calculus atoms, which can be one of the following:

1. An atom of the form $R(t_i)$, where R is a relation name & t_i is a tuple variable. $R(t_i)$ means t_i is a tuple in the relation R (e.g. EMPLOYEE (t) means t is a tuple in the relation EMPLOYEE).

2. An atom of the form "$t_i.A$ OP $t_j.B$", where t_i and t_j are tuple variables, A is an attribute of t_i, B is an attribute of t_j and OP is an operator in the set $\{=, \neq, <, >, \leq, \geq\}$

3. An atom of the form "$t_i.A$ OP c", where c is a constant.

A formula is made up of one or more atoms connected via the logical operators AND, OR, and NOT.

The Existential and Universal Quantifiers:

* Universal quantifier: $\forall$ => $(\forall t)$ F is TRUE if <u>every</u> possible tuple that can be assigned to t in F is substituted for t, F is true for every such substitution.

* Existential quantifier: $\exists$ => $(\exists t)$ F is TRUE if there exists <u>any</u> tuple t that makes F true.

* A tuple variable t is <u>bound</u> in a formula F if it appears in an $(\forall t)$ or $(\exists t)$ clause, otherwise it is <u>free</u>.

Transforming Universal & Existential Quantifiers:

Rules: (P and Q are predicates on variable x)

- $(\forall x)(P(x)) \equiv$ NOT $(\exists x)$ (NOT (P(x)))
- $(\exists x)(P(x)) \equiv$ NOT $(\forall x)$ (NOT (P(x)))
- $(\forall x)$ (P(x) AND Q(x)) $\equiv$ NOT $(\exists x)$ (NOT (P(x)) OR NOT (Q(x)))
- $(\forall x)$ (P(x) OR Q(x)) $\equiv$ NOT $(\exists x)$ (NOT (P(x)) AND NOT (Q(x)))
- $(\exists x)$ (P(x) OR Q(x)) $\equiv$ NOT $(\forall x)$ (NOT (P(x)) AND NOT (Q(x)))
- $(\exists x)$ (P(x) AND Q(x)) $\equiv$ NOT $(\forall x)$ (NOT (P(x)) OR NOT (Q(x)))

Tuple Relational Calculus Examples:

(a) Retrieve the names of employees in department 5 who work more than 10 hours per week on the 'ProductX' project.

TRC expression:

$$\{e.Fname, e.Minit, e.Lname \mid EMPLOYEE(e) \text{ AND } e.Dno = 5 \text{ AND } (\exists p)(\exists w)(WORKS\_ON(w) \text{ AND } PROJECT(p) \text{ AND } w.Pno = p.Pnumber \text{ AND } p.Pname = 'ProductX' \text{ AND } w.Hours > 10)\}$$

(b) List the names of employees who have a dependent with the same first name as themselves.

TRC expression:

$$\{e.Fname, e.Minit, e.Lname \mid EMPLOYEE(e) \text{ AND } (\exists d)(DEPENDENT(d) \text{ AND } d.Essn = e.Ssn \text{ AND } e.Fname = d.Dependent\_name)\}$$

(c) Find the names of employees that are directly supervised by 'Franklin Wong'.

TRC expression:

$$\{e.Fname, e.Minit, e.Lname \mid EMPLOYEE(e) \text{ AND } (\exists s)(EMPLOYEE(s) \text{ AND } s.Fname = 'Franklin' \text{ AND } s.Lname = 'Wong' \text{ AND } e.Superssn = s.Ssn)\}$$

(d) Retrieve the names and salaries of employees who work on every project.

TRC expression:

{e.Fname, e.Minit, e.Lname, e.Salary | EMPLOYEE(e) AND (∀p)
(NOT(PROJECT(p)) OR
(∃w)(WORKS_ON(w) AND
p.Pnumber = w.Pno AND
w.Essn = e.Ssn))}

(e) Retrieve the names of employees who do not work on any project.

TRC expression:

{e.Fname, e.Minit, e.Lname | EMPLOYEE(e) AND NOT (∃w)(WORKS_ON(w)
AND w.Essn = e.Ssn)}

(f) Find the names and addresses of employees who work on at least one project located in Houston but whose department has no location in Houston.

TRC expression:

{e.Fname, e.Minit, e.Lname, e.Address | EMPLOYEE(e) AND (∃p)(∃w)(
PROJECT(p) AND WORKS_ON(w)
AND e.Ssn = w.Essn AND
w.Pno = P.Pnumber AND p.Location = 'Houston'
AND NOT (∃l)(DEPT_LOCATIONS(l) AND
e.Dno = l.Dnumber AND l.Dlocation = 'Houston
))}

(g) List the names of department managers who have no dependents.

TRC expression:

{e.Fname, e.Minit, e.Lname | EMPLOYEE(e) AND (∃d)(DEPARTMENT(d) AND
e.Ssn = d.Mgrssn AND NOT (∃x)(DEPENDENT(x)
AND x.Essn = e.Ssn))}