Relational Algebra and Relational Calculus :
Theoretical Basis of Database

Relational Calculus [basis for QBE]
Calculus is higher order; declarative; no order of operations; defines RESULT.

Tuple relational calculus
{t | COND(t)} all tuples for which COND(t) is true
AND, OR, NOT

Quantifiers:

Such that	|	SuchThat
for every	∀	ForEvery
there exits	∃	ThereExists
member of	ε	IN

Free and bound:
A BOUND tuple is within a QUANTIFIER.
Only free tuples (that are not bound) should only appear to the left of **| (such that)**
For example, {t| (t IN EMPLOYEE) AND (t.name='Fred')}
Variables should normally be BOUND.
Free tuples are usually what you want as a result of your query.
Free tuples should only appear to the left of | in { (x,y)| (description of x and y)}
{t|(NOT(EMPLOYEE))} ***BAD****

Formulas:
(well formed formula, or wff)
A FORMULA is either true or false.
R(t), (R is a relation, t is a tuple,) is an ATOM that is either true or false.
Such an ATOM is a FORMULA.
If t1 and t2 are tuple variables, A and B are attributes:
T1.A =t2.B (or < <= > >= <>) is a formula.
If F1 and F2 are formulas, then:
F1 AND F2, F1 OR F2, NOT(F1) are formulas.
If F is a formula, then so is (∀ t)(F) and (∃ t)(F) (t is a tuple variable)

Technically, Tuple Relational Calculus allows only tuple variables;
We will focus on Tuple Relational Calculus.
(Domain relational Calculus allows values of attributes.)

[Allowed to say ForEvery, ThereExists, SuchThat]
{b,t|(∀ b)(∃ t)(formula)}

(∃t)(F) is TRUE IFF(if and only if) it is true for at least ONE tuple in F.
(∀t)(F) is TRUE IFF it is true for ALL tuples in F.

(∀t)(P(t)) is defined to be equal to: NOT((∃t)(NOT(P(t)))
((∃t)(NOT(P(t)) is defined to be equal to: NOT((∀t)(P(t)))

Rules of logic, such as DeMorgan's Laws, apply:
(Negating OR gets you an AND; negating AND gets you an OR)

IT IS (tautologically) TRUE THAT:
NOT(A and B) == NOT(A) or NOT(B)
NOT(A or B) == NOT(A) and NOT(B)

(∀t)(P(t)AND Q(t)) is defined to be equal to: NOT((∃t)(NOT(P(t) OR NOT(Q(t))))
(∀t)(P(t)OR Q(t)) is defined to be equal to: NOT((∃t)(NOT(P(t) AND NOT(Q(t))))
((∃t)(NOT(P(t) OR Q(t)) is defined to be equal to: NOT((∀t)(P(t)) AND Q(t))
((∃t)(NOT(P(t) AND Q(t)) is defined to be equal to: NOT((∀t)(P(t)) OR Q(t))

P=>Q means implied; is defined to be equal to NOT(P) OR Q

BE CAREFUL with UNIVERSAL QUANTIFIERS:
{t|NOT(EMPLOYEE(t))} is not SAFE
It yields an infinite number of tuples!
(You should always specify what t IS)

This is, technically, an assertion that might be true or might be false:
(∀t)(employee(t)->(salary(t)>0))

Queries are in the form of a set.
{e |(EMPLOY(e) }
(t.name, t.address | EMPLOY(t)}
(t.name, t.address | EMPLOY(t) AND
(∃d) (DEPARTMENT(d)
AND (d.deptname='Research')AND (d.deptnumber=t.deptno))}

'b' for boss 'e' for employee
{b.name,e.name|(EMPLOY(e) AND EMPLOY(b) AND (e<>b) AND SUPER(b,e)) }
{b.name|((∀e)((EMPLOY(b) AND EMPLOY(e) → NOT(SUPER((e,b)))))))

Star Trek example
s for series a for actor(the star table) p person
{a.role,s| (STAR(a) AND Series(s) AND (∀s)(∃p)((p.series=s.series) AND (p.role=a.role))}

Practice:

EMPLOY (name, id, dept)
DEPT (name, number, ...)
WORKSON (empnum, projnum)
PROJECT (name, num, dept)
e employer
d department
p project

List the name and address of all employees who work for the research department.

List names of employees who work on all projects controlled by department 5

Relational Calculus Examples

List the name and address of all employees who work for the Research Department.

{t.Fname, t.Lname, t.Address | EMPLOYEE(t) AND
(ThereExists d}(DEPARTMENT(d) AND (d.Dname=’Research’) AND (d.Dnumber = t.Dno))
}

For every project in Stafford, list project number, department, manager’s name
{p.Pnumber, p.Dnum, m.Lname |
PROJECT(p) AND EMPLOYEE(m) AND (p.Plocation=’Stafford’) AND
(ThereExists d)(DEPARTMENT(d) AND (p.Dnumber = d.Dnumber) AND
(d.Mgr_ssn = m.Ssn))}

Each employee who works for SOME project controlled by department 5
{e.Lname, e.Fname | EMPLOYEE(e) AND
(ThereExists p) (ThereExists w) (PROJECT(p) (WorksOn w) AND (p.dNum=5) AND
(w.Essn=e.SSn) AND (p.Pnumber = w.Pno))
}

Stark Trek:
List all roles that appear in all series
{r.roleName) | (role® AND (ForEvery s) Series(s) ->
(ThereExists a) (APPEAR(a) AND
(a.role = r.roleName) AND a.series = s.seriesName) })
}
ForEvery corresponds to NOT ThereExists NOT
ThereExits corresponds to NOT ForEvery NOT