Relational Calculus

1. Tuple Relational Calculus
2. Domain Relational Calculus

Tuple Relational Calculus

- A nonprocedural query language, where each query is of the form $\{t \mid P(t)\}$
- It is the set of all tuples t such that predicate P is true for t
- t is a *tuple variable*, $t[A]$ denotes the value of tuple t on attribute A
- $t \in r$ denotes that tuple t is in relation r
- P is a *formula* similar to that of the predicate calculus

Tuple Relational calculus

R

t	Relational DB	Predicate Calculus

- Tuple variable t ranges for all tuple of relation R
- T.A => column A of tuple t

- <u>Basic form</u>

$$\{t_1A_{i1}, t_2A_{i2}, t_3A_{i3}.... \mid \theta\}$$

Predicate calculus exp

condition

Relational Calculus

1. Set of attributes and constants
2. Set of comparison operators: (e.g., $<$, $\leq$, $=$, $\neq$, $>$, $\geq$)
3. Set of connectives: and ($\wedge$), or (v), not ($\neg$)
4. Implication ($\Rightarrow$): x $\Rightarrow$ y, if x if true, then y is true

$$x \Rightarrow y \equiv \neg x \vee y$$

5. Set of quantifiers:

   $\exists\, t \in r\, (Q\,(t\,)) \equiv$ "there exists" a tuple in t in relation r such that predicate $Q\,(t\,)$ is true

   $\forall t \in r\, (Q\,(t\,)) \equiv Q$ is true "for all" tuples t in relation r

Tuple Relational calculus

- Example

  Student(RollNo, Name, DepartmentNo, Sex)

  Query: Find RollNo and Name of student in Department No: 2

  $\{t.RollNo, t.Name \mid Student(t) \wedge t.DepartmentNo=2\}$

Quantifiers

- Student(rollno,name,deptno,sex)

  Query: Find rollno and name of all students in deptno:2

  {t.rollno,t.name | Student(t) ∧ t.deptno=2}

  ∃ : Existential Quantifiers

  ∀: Universal Quantifiers

  0 expresses the condition we can use quantifiers with tuple variables

  t……..|con(t1..)

Example for quantifiers

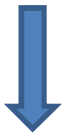
- Example

  Emp(Eid,Name,Add)

  Dependent(Did,Name,Eid)

  Query: Employee (Name) who has no dependent

e.Name | Emp(e) ∧ (true for emp having no dep)



free

False for e having some dep

True for e having some dep

∃d(dep(d) ∧ d.Eid = e.Erd)

Transformation Formula

$(\forall x)\ (p(x)) \equiv \neg(\exists x)\ (\neg P(x))$ $\neg\ \exists() \equiv \forall\ \neg()$

$\neg\ \neg\ \forall x\ (p(x)) \equiv \neg(\exists x)\ (\neg P(x))$ $\neg\ \forall() \equiv \exists\ \neg(u)$

$(\exists\ x)\ (p(x)) \equiv \neg(\forall\ x)\ (\neg P(x))$

$\neg\ \neg\ \exists\ x\ (p(x)) \equiv \neg(\forall\ x)\ \neg P(x)$

$(\forall x)\ (p(x) \wedge q(x)) \equiv \neg(\exists x)\ (\neg p(x)\ \vee \neg q(x)\ )$

$\neg\ \neg\ \forall x\ (p(x) \wedge q(x)) \equiv \neg(\exists x)\ \neg(p(x) \wedge q(x))$

$\overline{A.B} = \overline{A} + \overline{B}$

$\equiv \neg(\exists x)\ (\neg p(x)\ \vee \neg q(x))$

Transformation Formula

$$(\forall x)\,(p(x) \vee q(x)) \equiv \neg(\exists x)\,(\neg p(x) \wedge \neg q(x))$$

$$(\exists x)\,(p(x) \wedge q(x)) \equiv \neg(\forall x)\,(\neg p(x) \vee \neg q(x))$$

$$(\exists x)\,(p(x) \vee q(x)) \equiv \neg(\forall x)\,(\neg p(x) \wedge \neg q(x))$$

Transforming Quantification -2

Employee(Eid,Name,Add)
Dependent(Dis,Name,Eid)
Query: List the name if employee who have NO dependent

{e.Name | Employee(e) ∧ (¬ ∃d(Dependent(d) ∧ ∧ (e.Eid = d.Eid)))}

{e.Name | Employee(e) ∧ (∀d ¬ (Dep(d) ∧ (e.Eid = d.Eid)))}

{e.Name | Employee(e) ∧(∀d (¬ Dep(d) v ¬ (e.Eid = d.Eid)))}

Safe Expression

- It is possible to write tuple calculus expressions that generate infinite relations.
- For example, $\{ t \mid \neg\, t \in r \}$ results in an infinite relation if the domain of any attribute of relation *r* is infinite
- To guard against the problem, we restrict the set of allowable expressions to safe expressions.
- An expression $\{t \mid P(t)\}$ in the tuple relational calculus is *safe* if every component of *t* appears in one of the relations, tuples, or constants that appear in *P*
  - NOTE: this is more than just a syntax condition.
    - E.g. $\{ t \mid t[A] = 5 \vee$ **true** $\}$ is not safe --- it defines an infinite set with attribute values that do not appear in any relation or tuples or constants in *P*.

SAFE EXPRESSION

{t | Employee(t)} → Safe expression

{t | ¬ Employee(t)} → Unsafe expression

Example

depositor(cust_name,acc_no)

borrower(cust_name,loan_no)

loan(loan-no,branch_name,amount)

customer(cust_name,city,street)

account(acc_no,brach_name,balance)

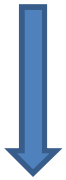
branch(branch_name,branch_city,accets)

Q1: Find the loan details of loan above 1200

{t | loan(t) ∧ t.amount > 1200}

Q1: Find the names of all customers who have a loan from branch ‘x’

$\{ b.name \mid borrower(b) \wedge \exists_l(loan(l) \wedge l.loan\_no = b.loan\_no \wedge l.branch\_name = \text{‘x’})\}$



OR

$\{b.name \mid \exists_b \in borrower \wedge \exists_l \in loan$

$\wedge l.loan\_no = b.loan\_no$

$\wedge l.branch\_name = \text{‘x’}\}$

Query: Customer who have account or loan or both

$\{t \mid cutomer(t) \wedge$

$\rightarrow$ if borrower 1

or

if depositor 2

1$\rightarrow$ $\exists_b(borrower(b) \wedge b.cust\_name = t.cust\_name)$

2$\rightarrow$ $\exists_d(depositor(d) \wedge d.cust\_name = t.cust\_name)$

$\{t|customer(t) \wedge (\exists_b(borrower(b) \wedge b.cust\_name = t.cust\_name)$

$\vee \exists d(depositor(d) \wedge d.cust\_name = t.cust\_name))\}$

Domain relational calculus

SQL→ tuple relation calculus

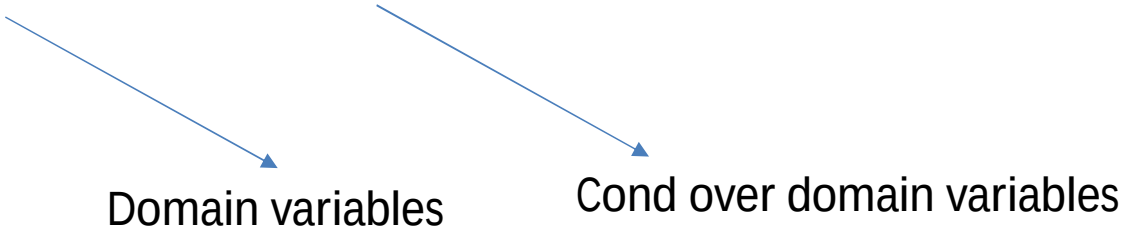
QBE→ domain relational calculus

(query by example)

Domain variable:- ranges over domain of attributes

Query Format

$\{x_1,x_2,x_3.....| \text{ cond}(x_1,x_2,x_3,.....)\} = \{ < x_1, x_2, ..., x_n > | P(x_1, x_2, ..., x_n)\}$



Domain relational calculus

Example:

Employee(firstname,lastname,eid,dob,add,sex,salary,dno)

Department(dno,dname,mid)

(a=firstname,b=lastname,c=eid,d=dob,e=add,f=sex,g=salary,h=dno)

(x=dno,y=dname,z=mid)

Query:- List the name and address of the employee whose name is Foo Bar

a,b,c,….,x,y,z → domain variables

$\{abe \mid \exists_c \exists_d \exists_f \exists_g \exists_h (Employee(abcdefgh) \wedge (a='Foo') \wedge (b='Bar'))\}$

$\{abe \mid Employee(abcdefgh) \wedge (a='Foo') \wedge (b='Bar')\}$

{abc | Employee('Foo','Bar',c,d,e,f,g,h)}

Domain relational calculus

Query:- List the name of the employee who have no department to manage

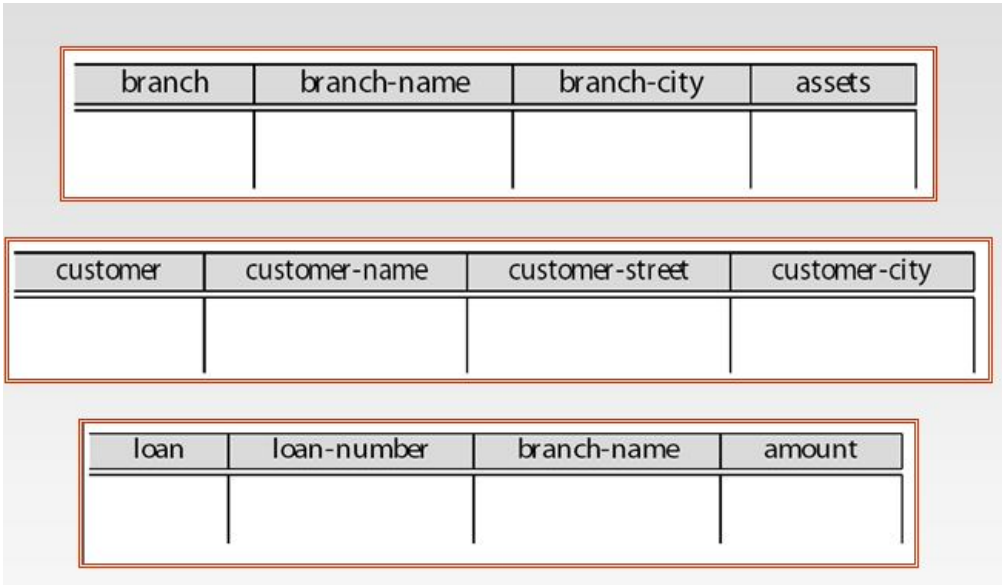
Employee(firstname,lastname,eid,dob,add,sex,salary,dno)
Department(dno,dname,mid)

Relation

$\{ab \mid \exists_c (Employee(abcdefgh) \wedge \neg \exists_z (Dep(xyz) \wedge z=c)$

QBE — Basic Structure

- A graphical query language which is based (roughly) on the domain relational calculus
- **Two dimensional syntax** – system creates templates of relations that are requested by users
- Queries are expressed “by example”



Queries on One Relation

- Find all loan numbers at the Perryridge branch.

loan	*loan_number*	*branch_name*	*amount*
	P._*x*	Perryridge	

- _x is a variable (optional; can be omitted in above query)
- P. means print (display)
- duplicates are removed by default
- To retain duplicates use P.ALL

loan	*loan_number*	*branch_name*	*amount*
	P.ALL.	Perryridge	

Queries on One Relation (Cont.)

- Display full details of all loans
  - Method 1:

loan	*loan-number*	*branch-name*	*amount*
	P._x	P._y	P._z

  - Method 2: Shorthand notation

loan	*loan_number*	*branch_name*	*amount*
P.			

QBE — Basic Structure