Week - 1

Relational Calculus - 1

- Relational algebra is a procedural in nature
- procedural: How: what the sequence of steps req to obtain the desired Relation

Diff b/w Relation cal & Algebra

- Relational cal
  - -> language of defining new relation in a DB
  - Form the basis of a query language
  - Non-procedural in some sense
  - It uses the concepts of formal logic to express relational queries

Formulation of Relation cal

- Tuple Relation Cal
- Domain Relation Cal

Tuple Relation cal

- Relation is Expressed as a set

$\{ t \mid P(t) \}$ $t \rightarrow$ tuple variable

$P(t) \rightarrow$ predicate of tuple variable

predicate which must be true for every element of the set

$S = \{t \mid P(t)\}$ i.e., $P(t)$ must be true for each tuple in

DB Scheme

* Set of attributes

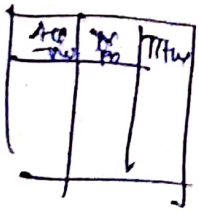
Acc. no, Yr-pub, title, Card-no, B-name, B-Add, S-name, S-Addr, DOI, Price, Dis

* Set of Relation schemes

Book-Scheme (ACC-NO, YR-pub, title)
User-scheme (Card-NO, B-name, B-add)
Supplier-scheme (S-name*, S-add*)
B-By-Scheme (Acc-no*, Card-no, DOI)
S-by-Scheme (ACC-NO, S-name, price, Dis)

* Set of Relations

book (book-Scheme)
user (user-scheme)
Supplier (Supplier-Schem)
barrow (~~barrow~~-scheme)
Supp (~~Supp~~ S-by-Scheme)



$\{t \mid t \in book \wedge t[YR\text{-}pub] = '1991'\}$

$\{t \mid t \in book\ (t[YR\text{-}pub) = '1991')\}$

Equivalent Relation Algebra

$\sigma_{YR\text{-}pub = '1991'}(book)$

- we dont want year
- How to select attribute of Subset

$$\{ t \mid \exists u \in book(u[YR\_pub] = '1991' \wedge t[ACC\_NO] = u[ACC\_NO] \wedge t[TITLE] = u[TITLE]) \}$$

(u → some tuple)

S

ACC_NO	TITLE

Book

ACC_NO	YR_PUB	TITLE

Supp

ACCNO	SNAME	PRICE	DOS

S

S_NAME

$$\{ t \mid \exists s \in Supp \wedge s[price] > 1000 \wedge s[sname] = t[s\_name] \}$$

. (1) get the names of supplier who has supplied atleast one book for 1000 (or) more

(4) Find the name of all Suppliers who have the same address as 'NAROSA'

$$\{t \mid \exists s \in Supplier(t[S\_name] = s[S\_name] \wedge \exists u \in Supplier(u[S\_Addr] = s[S\_Addr] \wedge u[S\_name] = \text{"NAROSA"}))\}$$

(5) Find the Suppliers who has supplied some title issued by "VIJAY" (may not be same book by Acc_No)

Supp, user, book, borrow)

$$\{t \mid \exists s \in Supp(t[S\_name] = s[SName] \wedge u[S\_name] = \text{"NAROSA"}$$

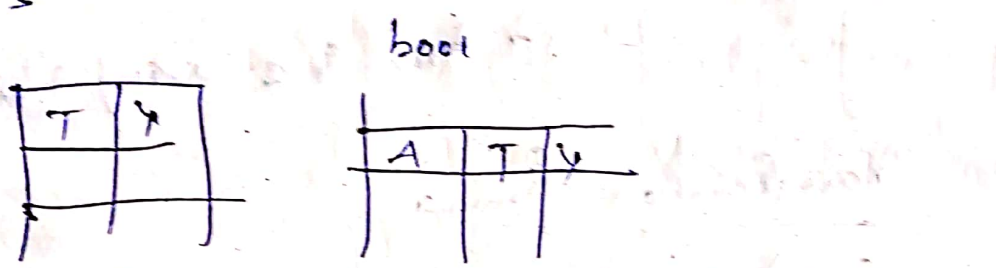
(6) Find the names of those suppliers who have supplied titles corresponding to all books issued by "vijay"

$$\{t \mid \forall s \in User(s[S\_NAME] = \text{"VIJAY"} \Rightarrow \exists p \in borrow(p[Card\_no] = s[Card\_no] \wedge \exists v \in book(v[Acc\_No] = p[Acc\_No]) \wedge \exists r \in bod(v[Title] = r[Title]) \wedge \exists m \in Supp(r[Acc\_no] = m[Acc\_no] \wedge$$

For correct of all

$$\{t \mid \forall u \in book\ (u[YR\_PUB] = '1984' \Rightarrow t[TITLE] = u[TITLE])\}$$

$$\{t \mid \forall u \in book\ (t[YR\_PUB] = u[PR\_PUB] \Rightarrow t[TITLE] = u[TITLE])\}$$



Ex! Student (Roll NO Name, DeptNO, Sex)

Query: Find Roll.NO and name of Student in dept no 2

$$\{t.\ Roll.NO,\ t.\ Name \mid Student(t) \wedge t.\ Dept = 2 \wedge t.\ Sex = Male\}$$

$$\{t_1.A_j, t_2.A_k, \ldots \mid Condition(t_1, t_2 \ldots)\}$$

A_j: Attribute

t_1: tuple variable

Using Quantifiers

$\exists$: Existential Quantifier

$\forall$: Universal Quantifier

Ex: Emp (Eid, Name, Add)

Dept (Did, Name, Eid)

Query: Employee (Name) who has no dependents

$e.Name \mid Emp(e) \wedge \neg(\exists d(dep(d) \wedge dEid = e.Eid))$

$\neg$ (True for e having some dept)

Transformation formula

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

$\neg\exists \equiv \forall\neg$

$\neg\forall \equiv \exists\neg$

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

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

$\neg(\exists x)(p(x)) \Rightarrow (\forall x)\neg(p(x))$

Query: List the name of employee who have no dependent

$$\{e.Name \mid Employee(e) \wedge (\neg \exists d (Depndt(d) \wedge \wedge (e.EId = d.EId)))\}$$

$$\{e.Name \mid employee(e) \wedge (\forall d \neg (Depends(d) \wedge (e.EId = d.EId)))\}$$

If we apply demorgans law

$$\{e.Name \mid employee(e) \wedge (\forall d (\neg Dep(d) \vee \neg (e.EId = d.EId)))\}$$

Ex:

depositor (cust_name, acc_no)

borrower (cust_name, loan_no)

loan (load_no, branch_name, amount)

Customer (cust_name, city, Street)

Account (Acc_no branch_name, balance

Branch (Branch_name, branch_city, assets)

(1) Find the loan details above 1200

$\{t \mid loan(t) \wedge t.[amount] > 1200\}$

(2) Find the name of all customers who have a loan from branch 'x';

$b.name \mid borrower(b) \wedge \exists L (loan(l) \wedge$

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

$l.branch\_name = 'x'$

(or)

$\{ b.name \mid \exists b \in borrower \wedge \exists l \in loan$

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

$\wedge l.branch\_name = 'x'$

(5) cust who have account or loan or both

$$\{ t \mid cust(t) \wedge (\exists b (borrower(b) \wedge b.custname = t.cust\_name) \vee \exists b (depositor(d) \wedge d.cust\_name = t.custname)) \}$$

Domain Relation Query

List the name of employee who have no dept to manager

Employee (First name, Lastname, Eid, DOB, Add, Sex, Salary, Dno)
a b c d e f g h

Dept (Dno, Dname, mid)
x y z

$$\{ a\ b \mid \exists c (Employee(a\,b\,c\,d\,e\,f\,g\,h) \wedge \neg \exists z (Dept(x,y,z) \wedge z = c)) \}$$