Set Operation: Set-Difference

- R - S: returns a relation instance containing all tuples that occur in R but not in S.
- R and S must be union-compatible.
- Scheme of the result is the schema of R.

Set Operation: Set-Difference

S1

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

S2

sid	sname	rating	age
28	yuppy	9	35.0
31	lubber	8	55.5
44	guppy	5	35.0
58	rusty	10	35.0

S1 – S2

sid	sname	rating	age
22	dustin	7	45.0

28 *Division operation (÷) " for all "*

1. Find all customers who have an account at all branches located in Brooklyn.

deposit = (branch_name, account_number, customer_name, balance)
branch = (branch_name, branch_city, assets)

a. $\Pi_{\text{branch\_name}}(\sigma_{\text{branch\_city} = \text{"Brooklyn"}}(\text{branch}))$

b. $\Pi_{\text{customer\_name, branch\_name}}(\text{deposit})$

$$\Pi_{\text{customer\_name, branch\_name}}(\text{deposit}) \div \Pi_{\text{branch\_name}}(\sigma_{\text{branch\_city} = \text{"Brooklyn"}}(\text{branch}))$$

EXAMPLE

Customer_name Branch_name

C_n	B_n
S1	P1
S1	P2
S1	P3
S1	P4
S2	P1
S2	P2
S3	P2
S4	P2
S4	P4
S4	P5

X

Branch in Brooklyn city

B_n
P2
P4

Y

Customer_name

C_n
S1
S4

$X \div Y$

30

Outer Join (⟗)

Outer Join : Extension of join operation to deal with missing information.

1) Left Outer Join (⟕) : Takes all tuples from the left relation that did not match with any tuple in the right relation.
2) Right Outer Join (⟖) : Takes all tuples from the right relation that did not match with any tuple in the left relation.
3) Full Outer Join (⟗) : Takes all tuples from the both the relations that did not match with any tuple in the relations.

EXAMPLE

Outer Joins

DEPARTMENTS

DEPARTMENT_NAME	DEPARTMENT_ID
Administration	10
Marketing	20
Shipping	50
IT	60
Sales	80
Executive	90
Accounting	110
Contracting	190

8 rows selected.

EMPLOYEES

DEPARTMENT_ID	LAST_NAME
90	King
90	Kochhar
90	De Haan
60	Hunold
60	Ernst
60	Lorentz
50	Mourgos
50	Rajs
50	Davies
50	Matos
50	Vargas
80	Zlotkey

20 rows selected.

There are no employees in department 190.

32 *The Assignment operation (⟵)*

It is used to assign relational algebra expression to temporary relation variables.

temp1 ⟵ Query

temp2 ⟵ Query

Result = temp 1 – temp 2

Find all customers of the Perryridge branch who have an account there but not a loan.

$$temp\ 1 \longleftarrow \Pi_{customer\_name}(\sigma_{branch\_name = \text{“Perryridge”}}(deposite))$$

$$temp\ 2 \longleftarrow \Pi_{customer\_name}(\sigma_{branch\_name = \text{“Perryridge”}}(borrow))$$

Result = temp1 - temp2

33 *Natural Join (⋈)*

Perform Cartesian Product and selection forcing equality on those attributes that appear in both relations and finally removes duplicate rows.

Find all customers who have a loan at the bank and the cities in which they live.

borrow = (branch_name, loan_number, customer_name, amount)
customer = (customer_name, street, city)

$$\Pi_{\text{borrow.customer\_name, customer\_city}} (\sigma_{\text{borrow.customer\_name = customer.customer\_name,}} ($$

borrow X customer))

$$\Pi_{\text{borrow.customer\_name, customer\_city}} (\text{borrow} \bowtie \text{customer})$$

Borrow

b-n	Lo_no	C-n	amount
Dow	17	Jones	1000
Red	23	Smith	2000
Perr	15	Hayes	1500
Brig	10	Brooks	2200

Customer

C- n	street	City
Jones	M	H
Smith	N	R
Hayes	M	H
Brooks	S	B

Borrow X Customer

b-n	Lo_no	C-n		C-n	Street	City
	17	Jones	1000	Jones	M	H
...	...	...	...	Smith	N	R
...	...	...	...	Hayes	M	H
...	...	...	...	Brooks	S	B
Red	23	Smith	2000	Jones	M	H
Red	23	Smith	2000	Smith	N	R
...	...	...	...	Hayes	M	H
...	...	...	...	Brooks	S	B
Perr	15	Hayes	1500	Jones	M	H
...	...	...	...	Smith	N	R
Perr	15	Hayes	1500	Hayes	M	H
...	...	...	...	Brooks	S	B
Brig	10	Brooks	2200	Jones	M	H
...	...	...	...	Smith	N	R
...	...	...	...	Hayes	M	H
Brig	10	Brooks	2200	Brooks	S	B

35

Modifying the database

- Deletion → We may delete only whole tuples we cannot delete values of only particular attributes.

$r \longleftarrow r - E$

where r is a relation and E is a query.

Delete all of smiths accounts

$deposit - \sigma_{customer\_name} = \text{“Smith”}\ (deposit)$

Insertion → To insert data into a relation

$$r \leftarrow r \cup E$$

(relation) (query)

•Insert Smith has 1200 in account 9732 at the Perryridge branch

deposit = (branch_name, account_number, customer_name, balance)

Deposit ⟵ deposit U { (“Perryridge”, 9732, “Smith”, 1200)}

37 Updating

$$\text{Updating} \rightarrow S_{A \longleftarrow E}(r)$$

Where r is the name of a relation with attribute A, which is assigned the value of expression E.

Accounts with balances over 10,000 receive 6% interest, all other receive 5%.

a) $S_{\text{balance} \longleftarrow \text{balance} * 1.06}(\sigma_{\text{balance} > 10000}(\text{deposit}))$

b) $S_{\text{balance} \longleftarrow \text{balance} * 1.05}(\sigma_{\text{balance} \leq 10000}(\text{deposit}))$

Note :- If we changed the order, an account whose balance is just under 10,000 would receive 11.3% interest.

EXAMPLES

lives or employee (Person_name, street, city)

works (Person_name, company_name, salary)

located_in or company (company_name, city),

manages (Person_name, manager_name)

EXAMPLES

a) Find the names of all employees who work for First Bank Corporation

$$\Pi_{\text{person\_name}} (\sigma_{\text{company\_name} = \text{"FBC"}} (\text{works}))$$

b) Find the names and cities of residence of all employees who work for First Bank Corporation

$$\Pi_{\text{works.person\_name, lives. city}} (\sigma_{\text{company\_name} = \text{"FBC"}} (\text{works} \bowtie \text{lives}))$$

EXAMPLES

c) Find the names, street address and Cities of residence of all employees who work for First Bank Corporation and earn more than Rs 10000/- per annum.

$$\Pi_{\text{lives.person\_name, street, city}} (\sigma_{\text{works..company\_name} = \text{"FBC"} \wedge \text{salary} > 10000} (\text{works} \bowtie \text{lives}))$$

d) Find the names of all employees in this database who live in the same city as the Company for which they work.

$$\Pi_{\text{lives.person\_name}} (\sigma_{\text{lives.city} = \text{located\_in.city} \wedge \text{lives.person\_name} = \text{works.person\_name}} (\text{lives} \times (\text{works} \bowtie \text{located\_in}))$$

Or

$$\Pi_{\text{lives.person\_name}} (\text{lives} \bowtie (\text{works} \bowtie \text{located\_in}))$$

EXAMPLES

e) Find the names of all employees who live in the same City and on the same street as do their managers.

$$\Pi_{\text{lives.person\_name}} (\sigma_{\text{lives1.street} = \text{lives.street} \wedge \text{lives1.city} = \text{lives} \wedge \text{manages.person\_name}}$$

$$= \text{lives.person\_name} (\text{lives} \times (\Pi_{\text{street, city, managers.person\_name}} (\sigma_{\text{lives1.person\_name}}$$

$$= \text{manages.manager\_name} (\rho_{\text{lives1}} (\text{lives}) \times \text{manages})))))$$

f) Find the names of all employees in this database who do not work for First Bank Corporation.

$$\Pi_{\text{person\_name}} (\text{works}) - \Pi_{\text{person\_name}} (\sigma_{\text{company\_name} \neq \text{"FBC"}} (\text{works}))$$

EXAMPLES

g) Find the names of all employees in this database who earn more than every employee of Small Bank Corporation.

$$\Pi_{\text{person\_name}} (\sigma_{\text{works.salary} > \text{ALL W1.salary} \wedge \text{works.company\_name} \neq \text{"SBC"}} (\text{works} \times \Pi_{\text{salary}} (\sigma_{\text{company\_name} = \text{"SBC"}} (P_{w1}(\text{works}))))$$

h) Assume the Companies may be located in several cities. Find all Companies located in every city in which Small Bank Corporation is located.

$$\Pi_{\text{company\_name, city}} (\text{located\_in}) \div \Pi_{\text{city}} (\sigma_{\text{company\_name} = \text{"SBC"}} (\text{located\_in}))$$

EXAMPLES

Consider the following Relational database :-

Address (Player_name, city, state)

Plays (Player_name, club_name, salary)

Located_in (club_name, state)

Captaincy (Player_name, captain_name)

Q1. Find the name and state of all players playing for the club with club_name = "XYZ".

$\Pi_{\text{Address.P\_name, Address.state}} (\sigma_{\text{club\_name = "XYZ"}} (\text{Address} \bowtie \text{Plays}))$

Q2. Find all Players who live in the same state as the club for which they play.

$$\Pi_{\text{Address.Player\_name}} (\sigma_{\text{Address.player\_name} = \text{Plays.Player\_name} \wedge \text{Address.state} = \text{located\_in.state}} (\text{Address} \times (\text{Plays} \bowtie \text{located\_in})))$$

$$\Pi_{\text{Address.Player\_name}} (\text{Address} \bowtie \Pi_{\text{Player\_name, state}} (\text{Plays} \bowtie \text{located\_in}))$$

Q3. Find all Players who live in the same state as their captain

$$\Pi_{\text{Address.Player\_name}} (\sigma_{\text{Address1.state} = \text{Address.state} \wedge \text{captaincy.Player\_name} = \text{Address.Player\_name}} (\text{Address} \times (\Pi_{\text{state, captaincy.captain\_name}} (\sigma_{\text{Address1.player\_name} = \text{captaincy.captain\_name}} (\rho_{\text{Address1}} (\text{Address}) \times \text{captaincy})))))$$

$$\Pi_{\text{Address.Player\_name}} (\sigma_{\text{Address.state} = \text{Address1.state} \wedge \text{Address.Player\_name} = \text{Captaincy.Captain\_name}} (\text{Address} \times (\Pi_{\text{state, captaincy.captain\_name}} (\rho_{\text{Address1}} (\text{Address} \bowtie \text{captaincy})))))$$

Tuple Relational Calculus TRC

A query in T.R.C. is expressed as

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

The set of all tuples t such that predicate P is true for t.

t[A] ➔ denote the value of tuple t on attribute A.

Where P is a formula. Several tuple variables may appear in a formula.

46 Tuple Relational Calculus TRC

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.
- S [x] Θ U [y], where S and U are tuple variable x is an attribute on which S is defined, y is an attribute on which U is defined, and Θ is a comparison operator ($<, \leq, =, \neq, >, \geq$).
- 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.

EXAMPLES

1) Find the names of all employees who work for First Bank Corporation.

$\{ t \mid \exists S \in works (t [Person\_name] = S [Person\_name] \wedge S [company\_name] = \text{“ First Bank Corporation ”}) \}$

2) Find the names and cities of residence of all employees who work for First Bank Corporation.

$\{ t \mid \exists S \in works (t [Person\_name] = S [Person\_name] \wedge S [company\_name] = \text{“ First Bank Corporation ”} \wedge \exists U \in lives (U [Person\_name] = S [Person\_name] \wedge t [city] = U [city])) \}$

EXAMPLES

4) Find the names of all employees in this database who live in the same city as the Company for which they work.

{ t | ∃ S ∈ works (t [Person_name] = S [Person_name] ∧ ∃ U ∈ located_in (U [company_name] = S [company_name] ∧ ∃ V ∈ lives (V [Person_name] = S [Person_name] ∧ V [city] = U [city]))) }

EXAMPLES

5) Find the names of all employees who live in the same City and on the same street as do their managers.

{ t | ∃ S ∈ lives (t [Person_name] = S [Person_name] ∃ U ∈ manages (S[person_name] = U [Person_name] ∃ V ∈ lives (U [manager_name] = V [Person_name] ∧ S [street] = V [street] ∧ S [city] = V [city])))}

6) Find the names of all employees in this database who do not work for First Bank Corporation.

{ t | ∃ S ∈ works (t [Person_name] = S [Person_name] ∧ S [company_name] ≠ " First Bank Corporation ") }

EXAMPLES

8) Assume the Companies may be located in several cities. Find all Companies located in every city in which Small Bank Corporation is located.

{ t | ∀ S ∈ located_in (S [company_name] = " Small Bank Corporation "
=> ∃ U ∈ located_in (t [company_name] = U [company_name] ∧ U [company_ name] ∧ S [branch_city] = U [branch_city])) }

The set of all company name such that for all tuples S in the located_in relation, if the value of S on attribute company_name is small bank corporation then the company located in the city whose name appears in the branch city attribute of S .

53

EXAMPLES

8) Assume the Companies may be located in several cities. Find all Companies located in every city in which Small Bank Corporation is located.

{ t | ∀ S ∈ located_in (S [company_name] = " Small Bank Corporation "
=> ∃ U ∈ located_in (t [company_name] = U [company_name] ∧ U [company_ name] ∧ S [branch_city] = U [branch_city])) }

The set of all company name such that for all tuples S in the located_in relation, if the value of S on attribute company_name is small bank corporation then the company located in the city whose name appears in the branch city attribute of S .

Domain Relational Calculus (DRC)

An expression in the domain relational calculus is of the form

$\{ < x_1, x_2, -----, x_n > | P(x_1, x_2, -----, x_n) \}$

Where $x_1, x_2, -----, x_n$ represent domain variables P represent a formula composed of atoms. An atom in the domain relational calculus has one of the following forms.

1. $< x_1, x_2, -----, x_n > \in r$, where r is a relation on n attributes and $x_1, x_2, -----, x_n$ are domain variables or constant.
2. $X \Theta Y$, where X and Y are domain variables and Θ is a comparison operator ($<, \leq, =, \neq, >, \geq$)
3. $X \Theta c$, where X is a domain variables and Θ is a comparison operator and c is a constant.

EXAMPLES

3) Find all customers having a loan from the Perryridge branch and the city in which they live

$\{ < c\ (name),\ x\ (city) > \mid \exists\ b, l, a\ (< b, l, c, a > \in borrow \wedge b = \text{“Perryridge”} \wedge \exists\ y\ (< c, y, x \in customer\ ))\}$

4) Find all customers having a loan, an account or both at the Perryridge branch

$\{ < c > \mid \exists\ b, l, a\ (< b, l, c, a > \in borrow \wedge b = \text{“Perryridge”}\ ) \vee \exists\ b, A, n\ (< b, A, c, n > \in deposit \wedge b = \text{“Perryridge”}\ )\}$

Difference between T.R.C. and D.R.C.

1. In tuple relational calculus variable is assigned to a relation where as in domain calculus a variable is defined for a given set of domain.
2. The domain calculus formation is comparatively easier then T.R.C. the reason is when there is a value which is occurring repeatedly , then domain relational calculus is defined several occurrence of same variable, where in T.R.C. these occurrence are defined with several distinct variable.
3. The domain calculus is complete relational calculus as compare to tuple relational calculus.

EXAMPLES

7) Find the names of all employees in this database who earn more than every employee of Small Bank Corporation.

$\{ t \mid \exists S \in works (t [Person\_name] = S [Person\_name] \wedge \exists U \in works (U [company\_name] = \text{“ Small Bank Corporation ”} \wedge S [salary] > U [salary])) \}$