Chapter 3

Exercise 3.1

Study the database schema containing the relations:

FILMS (FilmNumber,Title, Director, Year, ProductionCost)
ARTISTS (ActorNumber, Surname, FirstName, Sex, BirthDate, Nationality)
ROLES (FilmNumber, ActorNumber, Character)

1. Produce a database on this schema for which the joins between the various relation are all complete
2. Assuming two referential constraints between the relation ROLES and the other two, discuss possible cases of incomplete join.
3. Show a cartesian product that involves relations in this database.
4. Show a database for which one (or more) of the joins is (are) empty.

Solution:

FILMS

FilmNumber	Title	Director	Year	ProductionCost
145684	Armageddon	15434	1997	5000000
457343	Life is Beautiful	67532	1998	1000000
563822	Ronin	34573	1997	2000000

ARTISTS

ActorNumber	Surname	FirstName	Sex	Birthday	Nationality
67532	Benigni	Roberto	Male	14/03/1950	Italian
12456	DeNiro	Robert	Male	22/04/1951	American
45673	Braschi	Nicoletta	Female	1/05/1954	Italian
67777	Willis	Bruce	Male	3/2/1959	American
12345	Tyler	Liv	Female	18/02/1962	American

ROLES

FilmNumber	ActorNumber	Character
457343	67532	Guido
457343	45673	Dora
145684	67777	Harry
145684	12345	Grace
563822	12456	Sam

In this instance of database, the two (natural) joins between ROLES and ARTISTS, and between ROLES and FILMS are complete. In this schema there exists also a reference between “Director” and “ActorNumber”, when an actor is also director (such as Benigni).
If we had included all directors in table ARTISTS, then the join with ROLES would not be complete.

The referential constraints in this schema are on “ActorNumber” and “FilmNumber”. If a value on these attributes in table ROLES has not a corresponding value in tables ARTISTS and FILMS, the information in ROLES have not sense.
Yet is possible to admit values in table ARTISTS without corresponding values in ROLES (for example because they are directors).

This situation involves incomplete joins. The same holds for attribute “FilmNumber”.
An example of cartesian product between ARTIST and FILMS on the database above is:

FilmNum	Title	Dir	Year	ProdCost	ActorNum	Surname	FirstName	Sex	Birthday	Nationality
145684	Armageddon	15434	1997	5000000	67532	Benigni	Roberto	Male	14/03/50	Italian
457343	Life is Beautiful	67532	1998	1000000	67532	Benigni	Roberto	Male	14/03/50	Italian
563822	Ronin	34573	1997	2000000	67532	Benigni	Roberto	Male	14/03/50	Italian
145684	Armageddon	15434	1997	5000000	12456	DeNiro	Robert	Male	22/04/51	American
457343	Life is Beautiful	67532	1998	1000000	12456	DeNiro	Robert	Male	22/04/51	American
563822	Ronin	34573	1997	2000000	12456	DeNiro	Robert	Male	22/04/51	American
145684	Armageddon	15434	1997	5000000	45673	Braschi	Nicoletta	Female	1/05/54	Italian
457343	Life is Beautiful	67532	1998	1000000	45673	Braschi	Nicoletta	Female	1/05/54	Italian
563822	Ronin	34573	1997	2000000	45673	Braschi	Nicoletta	Female	1/05/54	Italian
145684	Armageddon	15434	1997	5000000	67777	Willis	Bruce	Male	3/2/59	American
457343	Life is Beautiful	67532	1998	1000000	67777	Willis	Bruce	Male	3/2/59	American
563822	Ronin	34573	1997	2000000	67777	Willis	Bruce	Male	3/2/59	American
145684	Armageddon	15434	1997	5000000	12345	Tyler	Liv	Female	18/02/62	American
457343	Life is Beautiful	67532	1998	1000000	12345	Tyler	Liv	Female	18/02/62	American
563822	Ronin	34573	1997	2000000	12345	Tyler	Liv	Female	18/02/62	American

Another database, with empty joins, may be the following one: values in table ROLES have not reference in the other tables:

FILMS

FilmNumber	Title	Director	Year	ProductionCost
145684	Armageddon	15434	1997	5000000
457343	Life is Beautiful	67532	1998	1000000
563822	Ronin	34573	1997	2000000

ARTISTS

ActorNumber	Surname	FirstName	Sex	Birthday	Nationality
67532	Benigni	Roberto	Male	14/03/1950	Italian
12456	DeNiro	Robert	Male	22/04/1951	American
45673	Braschi	Nicoletta	Female	1/05/1954	Italian
67777	Willis	Bruce	Male	3/2/1959	American
12345	Tyler	Liv	Female	18/02/1962	American

ROLES

FilmNumber	ActorNumber	Character
478384	67500	Peter
467343	42223	Dora
185682	67754	Harry
945684	99845	John
963822	12000	Mark

Exercise 3.2

With reference to the schema in Exercise 3.1, express the following queries in relational algebra, in domain calculus, in tuple calculus and in Datalog:

1. The titles of the films starring Henry Fonda;
2. The titles of the films in which the director is also an actor;
3. The actors who have played two characters in the same film; show the title of the films, first name and surname of the actor and the two characters;
4. The titles of the films in which the actors are all of the same sex;

Solution:

1.

Relational Algebra:

Π_{Title}(FILMS $\bowtie$ ($\sigma_{(FirstName=\text{"Henry"}) \wedge (Surname=\text{"Fonda"})}$ (ARTISTS) $\bowtie$ ROLES))

Domain Calculus:

{ Title : t | FILMS (FilmNumber: fn, Title: t,Director: d, Year: y,ProductionCost: pc) $\wedge$
ARTISTS (ActorNumber: an, Surname:sur , FirstName: n, Sex: s,
BirthDate:b, Nationality:nat) $\wedge$
ROLES (FilmNumber: fn, ActorNumber: an, Character: ch) $\wedge$
(sur= "Fonda") $\wedge$ (n= "Henry") }

Tuple Calculus:

{ F.title | F(FILMS), A(ARTIST), R(ROLES) |
F.FilmNumber=R.FilmNumber $\wedge$ A.ArtistNumber=R.ArtistNumber $\wedge$
A.Surname= "Fonda" $\wedge$ A.FirstName= "Henry" }

Datalog:

FILMSWITHFONDA (Title: t) $\leftarrow$
FILMS (FilmNumber: fn, Title: t,Director: d, Year: y,ProductionCost: pc),
ARTISTS (ActorNumber: an, Surname: "Fonda", FirstName: "Henry", Sex: s,
BirthDate: b, Nationality: nat),
ROLES (FilmNumber: fn, ActorNumber: an, Character: ch)

2.

Relational Algebra:

Π_{Title} ($\sigma_{Director=ArtistNumber)}$(ROLES $\bowtie$ FILMS))

Domain Calculus:

{ Title: t | FILMS (FilmNumber: fn, Title: t,Director: d, Year: y,ProductionCost: pc) $\wedge$
ROLES (FilmNumber: fn, ActorNumber: d, Character: ch) }

Tuple Calculus:

{ F.Title | F(FILMS), R(RULES) |
F.FilmNumber=R.FilmNumber $\wedge$ F.Director=R.ActorNumber }

Datalog:

FILMSWITHDIRECTORARTIST(Title: t) ←

FILMS (FilmNumber: fn, Title: t,Director: d, Year: y,ProductionCost: pc)
ROLES (FilmNumber: fn, ActorNumber: d, Character: ch)

3.

Relational Algebra:

$\Pi_{\text{Title,FirstName,Surname,Character1,Character}}$
($\rho_{\text{FilmNumber1, ActorNumber1,Character1}\leftarrow \text{FilmNumber,ActorNumber,Character}}$ (ROLES)
$\bowtie_{(\text{FilmNumber1=FilmNumber}) \wedge (\text{ActorNumber1=ActorNumber}) \wedge (\text{Character1} \neq \text{Character})}$
ROLES)
$\bowtie$ ARTISTS $\bowtie$ FILMS)

Domain Calculus:

{ Title: t, FirstName: fn, Surname: sur, Character: ch, Character1: ch1 |
FILMS (FilmNumber: fn, Title: t,Director: d, Year: y,ProductionCost: pc) $\wedge$
ARTISTS (ActorNumber: an, Surname:sur , FirstName: n, Sex: s, BirthDate:b, Nationality: nat)$\wedge$
ROLES (FilmNumber: fn, ActorNumber: an, Character: ch) $\wedge$
ROLES (FilmNumber: fn, ActorNumber: an, Character: ch1)$\wedge$
(ch $\neq$ ch1) }

Tuple Calculus :

{ F.Title, A.Surname, A.FirstName, R1.Character, R2.Character |
F (FILMS), A (ARTISTS), R1(ROLES), R2(ROLES) |
F.FilmNumber=R1.FilmNumber $\wedge$ A.ActorNumber=R1.ActorNumber $\wedge$
R2.FilmNumber=R1.FilmNumber $\wedge$ R2.ActorNumber=R1.ActorNumber $\wedge$
R2.Character $\neq$ R1.Character }

Datalog:

TWOCHARACTERS(Title: t, FirstName: fn, Surname: sur, Character: ch, Character1: ch1) ←
FILMS (FilmNumber: fn, Title: t,Director: d, Year: y,ProductionCost: pc) ,
ARTISTS (ActorNumber: an, Surname:sur , FirstName: n, Sex: s, BirthDate:b, Nationality:nat),
ROLES (FilmNumber: fn, ActorNumber: an, Character: ch) ,
ROLES (FilmNumber: fn, ActorNumber: an, Character: ch1),
(ch $\neq$ ch1).

4.

Relational Algebra (with intuitive abbreviations for attribute names):

Π_{Title} (FILMS) -
Π_{Title} (FILMS) $\bowtie$ $\sigma_{\text{Sex}\neq\text{Sex1}}$((ARTISTS $\bowtie$ ROLES) $\bowtie$
$\rho_{\text{FN1,AN1,Ch1,S1,FN1,Sex1,BD1,N1}\leftarrow\text{FN,AN,Ch,S,FN,Sex,BD,N}}$ (ARTISTS $\bowtie$ ROLES))

Domain Calculus:

{ Title: t | FILMS (FilmNumber: fn, Title: t,Director: d, Year: y,PCost: pc) ∧
¬∃t1(∃d1(∃y1(∃pd1(FILMS(FilmNumber: fn, Title: t1,Director: d1, Year: y1,PCost: pd1) ∧
ARTISTS (ActorN:an1, Surname:sur1, FirstName: n1, Sex: s1,
BirthDate:b1, Nationality:nat1) ∧
ARTISTS (ActorNumber: an2, Surname:sur2 , FirstName: n2, Sex: s2,
BirthDate:b2, Nationality:nat2) ∧
ROLES (FilmNumber: fn, ActorNumber: an1, Character: ch1) ∧
ROLES (FilmNumber: fn, ActorNumber: an2, Character: ch2) ∧
(s1 ≠ s2)
)))) }

Tuple Calculus:

{ F.Title | F(FILMS) |
¬(∃F1(FILMS)(∃A1(ARTISTS) (∃A2(ARTISTS) (∃R1(ROLES) (∃R2(ROLES) ∧
A1.ArtistNumber=R1.ArtistNumber∧ F1.FilmNumber=R1.FilmNumber ∧
A2.ArtistNUmber=R2.ArtistNumber ∧ R1.FilmNumber=R2.FilmNumber ∧
A1.Sex≠ A2.sex))))) }

Datalog:

DIFFERENTSEX (Title: t) ←
FILMS (FilmNumber: fn, Title: t,Director: d, Year: y,ProductionCost: pc),
ARTISTS (ActorNumber: an1, Surname:sur1 , FirstName: n1, Sex: s1, BirthDate:b1,
Nationality:nat1),
ARTISTS (ActorNumber: an2, Surname:sur2 , FirstName: n2, Sex: s2, BirthDate:b2,
Nationality:nat2),
ROLES (FilmNumber: fn, ActorNumber: an1, Character: ch1),
ROLES (FilmNumber: fn, ActorNumber: an2, Character: ch2),
(s1 ≠ s2)

SAMESEX(Title: t) ←
FILMS (FilmNumber: fn, Title: t,Director: d, Year: y,ProductionCost: pc),
NOT DIFFERENTSEX(Title: t)

Exercise 3.3

Consider the database containing the following relation:

REPRESENTATIVE(Number, Surname, FirstName,Committee,County,Constituency)
CONSTITUENCIES(County,Number, Name)
COUNTIES(Code,Name,Region)
COMMITTEES(Number,Name, President)

Formulate the following queries in relational algebra, in domain calculus and in tuple calculus:

1. find the name and surname of the presidents of the committees in which there is at least one representative from the county of Borsetshire;
2. find the name and surname of the members of the finance committee;
3. find the name, surname and constituency of the members of the finance committee;
4. find the name, surname, county and region of election of the delegates of the finance committee;
5. find the regions in which representatives having the same surname have been elected.

Solution:

1.

Relational Algebra:

$$\Pi_{FnP,Surp}((\ \sigma_{Name=\text{“Borsetshire”}}\ (COUNTIES) \rtimes_{Code=County} (REPRESENTATIVE \rtimes_{Committee=NumberC} \rho_{NumberC,NameC \leftarrow Number,Name}(COMMITTEE)\ ) \rtimes_{President=NumberP} \rho_{NumberP,SurP,FnP,CommitteeP,CountyP,ConstituencyP \leftarrow Number,Firstname,Surname,Committee,County,Constituency}(REPRESENTATIVE)))$$

Domain Calculus:

$$\{\ Surname: surp, FirstName: fnp \mid REPRESENTATIVE\ (Number: np, Surname: surp, FirstName: fnp, Committee: cmp, County: cop, Constituency: cstp) \wedge REPRESENTATIVE\ (Number: n, Surname: sur, FirstName: fn, Committee: cm, County:co, Constituency: cst) \wedge COMMITTEES(\ Number: cm, Name: nameC, President: np) \wedge COUNTIES(Code:co, Name: CountyName, Region: r) \wedge (CountyName= \text{“Borsetshire”})\ \}$$

Tuple Calculus:

$$\{\ P.(Surname, FirstName) \mid P\ (REPRESENTATIVE), R\ (REPRESENTATIVE), C(COMMITTEES), CO(COUNTY) \mid (P.Number=C.President) \wedge (R.Committee=C.Number) \wedge (C.Number=CO.Code) \wedge (CO.Name=\text{”Borsetshire”})\ \}$$

2.

Relational Algebra:

$$\Pi_{FirstName,Surname}\ (\sigma_{Name=\text{“Finance”}}(\ REPRESENTATIVE) \rtimes_{Committee=NumberC} \rho_{NumberC \leftarrow Number}(COMMITTEE)))$$

Domain Calculus:

$$\{FirstName: fn, Surname: sur \mid REPRESENTATIVE\ (Number: n, Surname: sur, FirstName: fn, Committee: cm, County:co, Constituency: cst) \wedge COMMITTEES(\ Number: cm, Name: nameC, President: np) \wedge (NameC= \text{“Finance”})\ \}$$

Tuple Calculus:

$$\{R.(FirstName,Surname) \mid R(REPRESENTATIVE), C(COMMITTEES) \mid (R.Committee=C.Number) \wedge (C.Name= \text{“Finance”})\ \}$$

3.

Relational Algebra:

$$\Pi_{FirstName,Surname,NameCon}((\sigma_{Name=\text{“Finance”}}(\text{ REPRESENTATIVE}) \bowtie_{Committee=NumberC} \rho_{NumberC \leftarrow Number}(\text{COMMITEE})) \bowtie_{(Constituency=NumberCon) \wedge (County=CountyCon)} \rho_{CountyCon,NumberCon,NameCon \leftarrow County,Number,Name}(\text{ CONSTITUENCIES}))$$

Domain Calculus:

{FirstName: fn, Surname: sur, Costituencies:Namecon |
REPRESENTATIVE (Number: n, Surname: sur, FirstName: fn, Committee: cm,
County:co,Constituency: cst) ∧
COMMITTEES(Number: cm, Name: nameC, President: np) ∧
CONSTITUENCIES(County: co, Number: cst, Name: Namecon)∧
(NameC= “Finance”) }

Tuple Calculus:

{R.(FirstName,Surname),CON.Name | R(REPRESENTATIVE), C(COMMITTEES),
CON(CONSTITUENCIES) |
(R.Committee=C.Number) ∧ (C.Name= “Finance”) ∧
(R.County=CON.County) ∧
(R.Constituency=CON.Number) }

4.

Relational Algebra:

$$\Pi_{FirstName,Surname,CountyName,RegionName}((\sigma_{Name=\text{“Finance”}}(\text{ REPRESENTATIVE}) \bowtie_{Committee=NumberC} \rho_{NumberC \leftarrow Number}(\text{COMMITEE})) \bowtie_{County=CountyCode} \rho_{CountyCode,CountyName \leftarrow Code,Name}(\text{COUNTIES}) \bowtie_{Region=RegionCode} \rho_{RegionCode,RegionName \leftarrow Code.Name}(\text{REGIONS}))$$

Domain Calculus:

{FirstName: fn, Surname: sur, County: CountyName,Region: RegionName |
REPRESENTATIVE (Number: n, Surname: sur, FirstName: fn, Committee: cm,
County:co,Constituency: cst) ∧
COMMITTEES(Number: cm, Name: nameC, President: np) ∧
COUNTIES(Number:co, Name: CountyName, Region: r) ∧
REGIONS (Code: r, Name: RegionName)∧
(NameC= “Finance”) }

Tuple Calculus:

{R.(FirstName,Surname),CO.Name,RG.Name | R(REPRESENTATIVE), C(COMMITTEES),
CO(COUNTIES), RG(REGIONS) |
(R.Committee=C.Number) ∧(R.County=CO.Code) ∧
(CO.Region=RG.Code) ∧ (C.Name= “Finance”) }

5.

Relational Algebra:

$$\Pi_{RegionName}(\sigma_{(Surname=Surname1)\wedge(Number\neq Number1)}$$
$$((\Pi_{Number,Surname,Region}\ (REPRESENTATIVE \bowtie_{County=Code} COUNTIES)$$
$$\bowtie_{Region=Region1}$$
$$\rho_{Number1,Surname1,Region1\leftarrow Number,Surname,Region}$$
$$(\Pi_{Number,Surname,Region}\ (REPRESENTATIVE \bowtie_{County=Code} COUNTIES))$$
$$\bowtie_{Region=RegionCode}$$
$$\rho_{RegionCode,RegionName\leftarrow Code,Name}(REGIONS))$$

Domain Calculus:

$$\{RegionName: rn \mid REPRESENTATIVE\ (Number: n, Surname: sur, FirstName: fn, Committee: cm,$$
$$County:co,Constituency: cst) \wedge$$
$$COUNTIES(\ Number:co, Name: CountyName, Region: r) \wedge$$
$$REPRESENTATIVE\ (Number: n1, Surname: sur1, FirstName: fn1, Committee: cm1,$$
$$County:co1,Constituency: cst1) \wedge$$
$$COUNTIES(\ Number:co1, Name: CountyName1, Region: r) \wedge$$
$$REGIONS(Code:r, Name: rn) \wedge$$
$$(sur=sur1) \wedge (n\neq n1)\ \ \}$$

Tuple Calculus:

$$\{R.Name \mid R(REGIONS), RP1(REPRESENTATIVE), RP2(REPRESENTATIVE),$$
$$C1(COUNTIES), C2(COUNTIES) \mid$$
$$(RP1.County=C1.Code) \wedge (RP2.County=C2.Code) \wedge$$
$$(C1.Region=C2.Region) \wedge (R.Code=C1.Region) \wedge$$
$$(RP1.Surname=RP2.Surname) \wedge (RP1.Number \neq RP2.Number))\ \ \}$$

Exercise 3.4

Show how the formulation of the queries in Exercise 3.3 could be facilitated by the definitions of views.

Solution:

In Exercise 3.3, in queries 2,3,4 we need to know Name and Surname of members of the finance committee. So it is reasonable to make this query only once, writing a view:

$$FINANCEMEMBERS = \sigma_{Name=\text{“Finance”}}(\ REPRESENTATIVE \bowtie_{Committee=NumberC}$$
$$\rho_{NumberC\leftarrow Number}(COMMITEE)\ )$$

This query, with the projection on Surname and FirstName, answers to query n° 2; it can be used to simplify the queries 2 and 3 as follow:

$$\Pi_{FirstName,Surname,NameCon}(FINANCEMEMBERS \bowtie_{(Constituency=NumberCon)\wedge(County=CountyCon)}$$
$$\rho_{CountyCon,NumberCon,NameCon\leftarrow County,Number,Name}(\ CONSTITUENCIES))$$

$$\Pi_{FirstName,Surname,CountyName,RegionName}$$
$$(FINANCEMEMBERS \bowtie_{County=CountyCode} \rho_{CountyCode,CountyName \leftarrow Code,Name} (COUNTIES)$$
$$\bowtie_{Region=RegionCode} \rho_{RegionCode,RegionName \leftarrow Code.Name}(REGIONS))$$

In query 5, the join between REPRESENTATIVE and COUNTIES is made two times; so it is possible to define a view:

$$REPRWITHCOUNTY = REPRESENTATIVE \bowtie_{County=Code} COUNTIES$$

The query becomes:

$$\Pi_{RegionName}(\sigma_{(Surname=Surname1) \wedge (Number \neq Number1)}$$
$$(\Pi_{Number,Surname,Region} (REPRWITHCOUNTY) \bowtie_{Region=Region1}$$
$$\rho_{Number1,Surname1,Region1 \leftarrow Number,Surname,Region}(\Pi_{Number,Surname,Region} (REPRWITHCOUNTY))$$
$$\bowtie_{Region=RegionCode} \rho_{RegionCode,RegionName \leftarrow Code,Name}(REGIONS))$$

Exercise 3.5

Consider the database schema on the relations:

COURSES(Number, Faculty, CourseTitle,Tutor)
STUDENTS (Number, Surname, FirstName, Faculty)
TUTORS (Number, Surname, FirstName)
EXAMS(Student, Course, Grade, Date)
STUDYPLAN (Student, Course, Year)

Formulate in relational algebra, in domain calculus, in tuple calculus and in Datalog, the queries that produce:

1. The students who have gained ‘A’ in at least one exam, showing, for each of them, the first name, surname and the date of the first of such occasions;
2. For every course in the engineering faculty, the students who passed the exam during the last session;
3. The students who passed all the exams required by their respective study plan;
4. For every course in the literature faculty, the student (or students) who passed the exam with the highest grade;
5. The students whose study plan require them to attend lectures only in their own faculty;
6. First name and surname of the students who haven taken an exam with a tutor having the same surname as the students.

Solution:

1)

Relational Algebra:

$$\Pi_{FirstName,Surname,\ Date}$$
$$(\ (\Pi_{Student,Date}\ (\sigma_{Grade=\ 'A'}(EXAMS)) -$$
$$\Pi_{Student,Date}(\sigma_{Grade=\ 'A'}(EXAMS \bowtie_{(Student=Student1) \wedge (Date \geq Date1)}$$
$$\rho_{Student1,Course1,Grade1,Date1 \leftarrow\ Student,Course,Grade,Date}(EXAMS))))$$
$$\bowtie_{Student=Number}(STUDENTS)\ )$$

Domain Calculus:

$$\{\text{FirstName: fn, Surname: sur, Date: d} \mid$$
$$\text{STUDENTS(Number: n, FirstName: fn, Surname: sur, Faculty: f)} \wedge$$
$$\text{EXAMS(student: n, Course: c, Grade: g, Date: d)} \wedge$$
$$(g = \text{‘A’}) \wedge$$
$$\neg(\exists c1\ (\exists\ d1\ (\text{EXAMS (Student: n, Course: c1, Grade: g, Data: d1)} \wedge (d1<d)))) \}$$

Tuple Calculus:

$$\{\ \text{S.(FirstName,Surname), E.Date} \mid \text{S (STUDENTS), E(EXAMS)} \mid$$
$$(\text{S.Number=E.Student}) \wedge (\text{E.Grade= ‘A’}) \wedge$$
$$\neg(\exists\ \text{E1(EXAMS)}\ ((\text{E1.Student=S.Number}) \wedge (\text{E1.Grade= ‘A’}) \wedge (\text{E1.Date < E.Date}))) \}$$

Datalog:

OTHERA (Student: n, Date: d) ←
EXAMS(Student: n, Course: c, Grade: ‘A’, Date: d),
EXAMS(Student: n1, Course: c1, Grade: ‘A’, Date: d1),
(d ≥ d1)

FIRSTA (FirstName: fn, Surname: sur, Date: d) ←
STUDENTS(Number: n, FirstName: fn, Surname: sur, Faculty: f),
EXAMS(student: n, Course: c, Grade: ‘A’, Date: d),
NOT OtherA(Student: n, Data: d)

2)

Relational Algebra:

$$\Pi_{\text{Surname, FirstName, StudentNumber}}$$
$$(\sigma_{(\text{Faculty= “Engineering”}) \wedge (\text{Date} \geq \text{StartSession}) \wedge (\text{Date} \leq \text{EndSession})}\ (\text{COURSES} \bowtie_{\text{Number=Course}} \text{EXAMS})$$
$$\bowtie_{\text{Student=StudentNumber}} \rho_{\text{StudentNumber,StudentFaculty} \leftarrow \text{Number,Faculty}}(\text{STUDENTS}))$$

Domain Calculus

$$\{\text{FirstName: fn, Surname: sur, StudentNumber: sn} \mid$$
$$\text{STUDENTS(Number: sn, FirstName: fn, Surname: sur, Faculty: sf)} \wedge$$
$$\text{EXAMS(Student: sn, Course: c, Grade: g, Date: d)} \wedge$$
$$\text{COURSES(Number: c, Faculty: f, CourseTitle: ct, Tutor: t)} \wedge$$
$$(f = \text{“Engineering”}) \wedge (d \leq \text{EndSession}) \wedge (d \geq \text{StartSession}) \}$$

Tuple Calculus:

$$\{\ \text{S.(FirstName,Surname, StudentNumber)} \mid \text{S (STUDENTS), E(EXAMS),C(COURSES)} \mid$$
$$(\text{S.Number=E.Student}) \wedge (\text{E.Course= C.Number}) \wedge (\text{C.Faculty=“Engineering”}) \wedge$$
$$(\text{E.Date} \leq \text{EndSession}) \wedge (\text{E.Date} \geq \text{StartSession}) \}$$

Datalog:

GOODSTUDENTS(FirstName: fn, Surname: sur, StudentNumber: sn) ←
STUDENTS(Number: sn, FirstName: fn, Surname: sur, Faculty: sf) ,
EXAMS(Student: sn, Course: c, Grade: g, Date: d),
COURSES(Number: c, Faculty: “Engineering”, CourseTitle: ct, Tutor: t),
(d ≤ EndSession),(d ≥ StartSession)

3)

Relational Algebra:

$$\Pi_{Student}(STUDYPLAN) - \Pi_{Student}(\Pi_{Student,Course}(STUDYPLAN) - \Pi_{Student,Course}(EXAMS))$$

Domain Calculus:

$$\{ Student: n \mid STUDYPLAN(Student:n, Course:c, Year: y) \wedge (\forall c1(\forall y1(\forall g1,\forall d1(\neg(STUDYPLAN(Student: n, Course: c1, Year: y1) \vee (EXAMS(Student: n, Course: c1, Grade: g1, Date: d1)))))) \}$$

Tuple Calculus:

$$\{ S.Student \mid S(STUDYPLAN) \mid (\forall S1(STUDYPLAN) (\forall E(EXAMS) (\neg(S.Student=S1.Student) \vee ((S1.Student=E.Student)\wedge(S1.Course=E.Course)))) \}$$

Datalog:

NOTALLEXAMS(Student: n) ←
STUDYPLAN(Student:n, Course :c , Year: y),
NOT EXAMS(Student: n, Course: c, Grade: g, Date :d)

ALLEXAMS(Student: n) ← STUDYPLAN(Student:n, Course :c , Year: y)
NOT NOTALLEXAMS(Student: n)

4)

Relational Algebra:

$$\Pi_{Students}(EXAMS \rtimes_{Course=Number} \sigma_{Faculty= \text{“Literature”}} (COURSES)) - \Pi_{Student}\sigma_{(Grade> Grade1)\wedge(Faculty= \text{“Literature”})}(COURSES \rtimes_{Number=Course} EXAMS \rtimes_{Course=Course1} \rho_{Student1,Course1, Grade1, Date1 \leftarrow Student,Course, Grade, Date}(EXAMS)))$$

Domain Calculus:

$$\{ Student: sn \mid EXAMS(Student: sn, Course: c, Grade: g, Date: d) \wedge COURSES(Number: c, Faculty: f, CourseTitle: ct, Tutor: t) \wedge (f= \text{“Literature”}) \wedge \neg(\exists sn1(\exists d1 (\exists g1 (EXAMS(Student: sn1, Course: c, Date: d1, Grade: g1) \wedge (g1< g))))) \}$$

Tuple Calculus:

$$\{ E.Student \mid E(EXAMS), C(COURSES) \mid (E.Course=C.Number) \wedge (C.Faculty= \text{“Literature”}) \wedge (\forall E1(EXAMS)\ \neg((E1.Course=E.Course) \vee (E1.Grade > E.Grade)) \}$$

Datalog:

STUDENTNOTBEST(Student: sn) ←
EXAMS(Student: sn, Course: c, Grade: g, Date: d), COURSES(Number: c,
Faculty: “Literature”, CourseTitle: ct, Tutor: t),
EXAMS(Student: sn1, Course: c, Grade: g1, Date: d1),
(g > g1)

STUDENTBEST(Student: sn) ←
EXAMS(Student: sn, Course: c, Grade: g, Date: d),
COURSES(Number: c, Faculty: “Literature”, CourseTitle: ct, Tutor: t),
NOT STUDENTNOTBEST(Student: sn)

5)

Relational Algebra:

$$\Pi_{Student}(STUDYPLAN) - \Pi_{Student}(\sigma_{Sfaculty \neq Faculty}(\rho_{Snumber,Sfaculty \leftarrow Number,Faculty}(STUDENTS) \bowtie_{Snumber=Student} STUDYPLAN \bowtie_{Course=Number} COURSES))$$

Domain Calculus:

$$\{ Student: n \mid STUDYPLAN(Student:n, Course :c , Year: y) \wedge \neg(\exists fn(\exists sur,\exists sf(\exists f,(\exists ct,\exists t(STUDENTS(Number: n, FirstName: fn, Surname: sur, Faculty: sf) \wedge COURSES(Number: c, Faculty: f, CourseTitle: ct, Tutor: t) \wedge (sf \neq f)))))) \}$$

Tuple Calculus:

$$\{ SP.Student \mid SP(STUDYPLAN) \mid \neg(\exists S(STUDENTS)\ (\exists C(COURSES) ((SP.Course=C.Number) \wedge (S.Number=SP.Student) \wedge (C.Faculty \neq S.Faculty)))) \}$$

Datalog:

DIFFERENTFACULTY(Student: n) ←
STUDYPLAN(Student:n, Course :c , Year: y),
STUDENTS(Number: n, FirstName: fn, Surname: sur, Faculty: sf) ,
COURSES(Number: c, Faculty: f, CourseTitle: ct, Tutor: t),
(sf≠f)

SAMEFACULTY(Student: n) ←
STUDYPLAN(Student:n, Course :c , Year: y),
NOT DIFFERENTFACULTY(Student: n)

6)

Relational Algebra:

$$\Pi_{Surname,FirstName}\ (\sigma_{Surname=Tsur}(\ \rho_{Snumber,SFaculty \leftarrow Number,Faculty}(STUDENTS) \bowtie_{Snumber=Student} EXAMS \bowtie_{Tutor=Tnumber} \rho_{Tnumber,Tfn,Tsur \leftarrow Number,FirstName,Surname}\ (TUTORS)\ ))$$

Domain Calculus:

$$\{\ FirstName: fn, Surname: sur \mid STUDENTS(Number: n, FirstName: fn, Surname: sur, Faculty: sf) \wedge EXAMS(student: n, Course: c, Grade: g, Date: d\ ) \wedge COURSES(\ Number: c, Faculty: f, CourseTitle: ct, Tutor: t) \wedge TUTORS(\ Number: t, Surname:sur, FirstName: tfn)\ \}$$

Tuple Calculus:

$$\{\ S.(FirstName,Surname) \mid S(STUDENTS), E\ (EXAMS), C(COURSES), T(TUTORS) \mid (S.Number=E.Student) \wedge (E.Course=C.Number) \wedge (C.Tutor=T.Number) \wedge (T.Surname=S.Surname)\ \}$$

Datalog:

SAMESURNAME (FirstName: fn, Surname: sur) ←
STUDENTS(Number: n, FirstName: fn, Surname: sur, Faculty: sf) ,
EXAMS(student: n, Course: c, Grade: g, Date: d) ,
COURSES(Number: c, Faculty: f, CourseTitle: ct, Tutor: t),
TUTORS(Number: t, Surname:sur, FirstName: tfn)

Exercise 3.6

With reference to the following database schema:

CITIES(<u>Name</u>, Region, Population)
CROSSING (<u>City</u>, <u>River</u>)
RIVERS (<u>River</u>, Length)

Formulate the following queries in relational algebra, domain calculus, tuple calculus and Datalog:

1. Find the names, regions and population for the cities that (i) have more than 50 thousand in habitants and (ii) are crossed by the Thames or the Mersey
2. Find the cities that are crossed by (at least) two rivers, giving the name of the city and that of the longest of the rivers.

Solution:

1) Relational Algebra:

$$\Pi_{\text{Name, Region, Population}} (\sigma_{(\text{River="Thames"})\vee(\text{River="Mersey"})}(\text{CROSSING}) \rtimes_{\text{City=Name}} \sigma_{\text{Population}>50000}(\text{CITIES}))$$

Domain Calculus:

$$\{ \text{Name: n, Region: r, Population: p} \mid \text{CITIES (Name: n, Region: reg, Population : p)} \wedge \text{CROSSING(City: n, River: r)} \wedge (p > 50000) \wedge ((\text{River= “Thames”}) \vee (\text{River= “Mersey”})) \}$$

Tuple Calculus:

$$\{ \text{C.(Name,Region,Population)} \mid \text{C(CITIES), R (CROSSING)} \mid (\text{C.Name= R.City}) \wedge (\text{C.Population} > 50000) \wedge ((\text{R.River= “Thames”}) \vee (\text{R.River= “Mersey”})) \}$$

Datalog:

$$\text{NOTHAMESNOMERSEY (City: c)} \leftarrow \text{CROSSING (City: c, River: r)}, (r \neq \text{“Thames”}), (r \neq \text{“Mersey”})$$

$$\text{THAMESORMERSEY (City: c, Region : reg, Population p)} \leftarrow \text{CITIES (Name: c Region: reg, Population : p)}, \text{CROSSING(City: n, River: r)}, (p > 50000), \text{NOT NOTHAMESNOMERSEY (City: c)}$$

2) Relational Algebra:

$$\Pi_{\text{City,River}} (\sigma_{\text{River} \neq \text{River1}} (\text{CROSSING} \rtimes_{\text{City=City1}} \rho_{\text{City1,River1}\leftarrow\text{City,River}} (\text{CROSSING}))) -$$

$$\Pi_{\text{City,River}} (\sigma_{(\text{River}\neq\text{River1})\wedge(\text{Length} < \text{Length1})}((\text{RIVERS} \rtimes \text{CROSSING}) \rtimes_{\text{City=City1}} \rho_{\text{City1,River1, Length1}\leftarrow \text{City,River, Length}}(\text{RIVERS} \rtimes \text{CROSSING})))$$

Domain Calculus:

$$\{ \text{City: n, River: r} \mid \text{CROSSING (City: n, River: r)} \wedge \text{CROSSING (City: n, River: r1)} \wedge (r \neq r1) \wedge \text{RIVERS(River: r, Length: l)} \wedge (\forall\, r2\ (\forall\, l2\ \neg((\text{CROSSING (City: n, River: r2)} \wedge \text{RIVERS(River: r2, Length: l2)}) \vee (l2 < l)))) \}$$

Tuple Calculus:

$$\{\text{C.(City, River)} \mid \text{C(CROSSING), C1(CROSSING), R(RIVERS)} \mid (\text{C.City=C1.City}) \wedge (\text{C.River} \neq \text{C1.River}) \wedge (\text{C.River=R.River}) \wedge (\forall \text{C2(CROSSING)}\ (\forall \text{R2(RIVERS)}\ (\text{C2.City} \neq \text{C.City}) \vee (\text{R2.River=R.River}) \wedge (\text{R2.Length} < \text{R.Length})))) \}$$

Datalog:

SHORTRIVERS (City: c, River: r) ←
CROSSING (City: c, River: r),
RIVERS(River: r, Length: l),
CROSSING (City: c, River: r1),
RIVERS(River: r1, Length: l1),
(l < l1)

LONGRIVERS (City : c, River: r) ←
CROSSING (City: c, River: r),
CROSSING (City: c, River: r1),
(r ≠r1),
NOT ShortRiver (City: c, River: r)

Exercise 3.7

With reference to the following database schema:

TRIBUTARIES (Tributary, River)
RIVERS (River, Length)

Formulate in Datalog the query that finds all the tributaries, direct and indirect, of the Mississippi.

Solution:

ALLTRIBUTARIES (Tributary: t) ←
TRIBUTARIES (Tributary: t, River: “Mississippi”)

ALLTRIBUTARIES (Tributary: t) ←
TRIBUTARIES (Tributary: t, River:r),
ALLTRIBUTARIES (Tributary: r)

Exercise 3.8

Consider the relational schema consisting of the following relations:

TUTORS (Number, Surname, FirstName)
COURSES(Number, CourseName, Tutor)
STUDENTS(Number, Surname, FirstName)
EXAMS (Student, Course, Date, Grade)

With reference to this schema, formulate the expressions of algebra, tuple relational calculus and Datalog that produce:

1. The exams passed by the student Detrouvelan-Delaney (supposing him to be the only one with such a surname), indicating, for each exam, the name of the course, the grade achieved and the name of the tutor;

2. The tutors who teach two courses (and not more than two), indicating the surname and first name of the tutors and the names of the two courses.

Solution:

1) Relational Algebra:

$$\Pi_{CourseName,Grade,Tsurname,Tname}(\sigma_{Surname= \text{“Detrouvelan-Delaney”}} (STUDENTS \bowtie_{Number=Student} EXAMS \bowtie_{Course=Cnumber} \rho_{Cnumber \leftarrow Number}(COURSES) \bowtie_{Tutor=Tnumber} \rho_{Tnumber,Tsurname,Tname \leftarrow Number,Surname,FirstName} (TUTORS)))$$

Tuple Calculus:

$$\{ C.CourseName, E.Grade, T.(Surname,FirstName) \mid C(COURSES), E(EXAMS), T(TUTORS), S(STUDENTS) \mid (S.Number=E.Student) \wedge (E.Course=C.Number) \wedge (C.Tutor= T.Number) \wedge (S.Surname= \text{“Detrouvelan-Delaney”}) \}$$

Datalog:

```
DETROUVELANEXAMS( CName: cn, Grade: g, TSurname: tsur, TName: tname ) ←
    STUDENTS (Number: n, Surname: “Detrouvelan-Delaney”, FirstName: fn),
    EXAMS( Student: n, Course: c, Grade: g, Date: d),
    COURSES (Number: c, CourseName:cn, Tutor: t),
    TUTORS(Number: t, Surname: tsur, FirstName: tname )
```

2) Relational Algebra

$$\Pi_{Surname,FirstName,CourseName,CourseName1} (\Pi_{Tutor,CourseName,CourseName1} (COURSES \bowtie_{(Tutor=Tutor1) \wedge (Number \neq (Number1)} \rho_{Number1,CourseName1,Tutor1 \leftarrow Number,CourseName,Tutor} (COURSES)) - \Pi_{Tutor,CourseName,CourseName1} (\sigma_{Number \neq Number2}(COURSES \bowtie_{(Tutor=Tutor1) \wedge (Number \neq Number1)} \rho_{Number1,CourseName1,Tutor1 \leftarrow Number,CourseName,Tutor} (COURSES) \bowtie_{(Tutor1=Tutor2) \wedge (Number1 \neq Number2)} \rho_{Number2,CourseName2,Tutor2 \leftarrow Number,CourseName,Tutor} (COURSES))) \bowtie_{Tutor=Tnumber} \rho_{Tnumber \leftarrow Number} (TUTORS))$$

Tuple Calculus:

$$\{ T.(FirstName,Surname), C.CourseName, C1.CourseName \mid T(TUTORS), C (COURSES), C1(COURSES) \mid (T.Number)=(C.Tutor) \wedge (T.Number)=(C1.Tutor) \wedge (C.Number \neq C1.Number) \wedge \neg(\exists C2(COURSES) ((C2.Tutor=T.Number) \wedge (C2.Number \neq C.Number) \wedge (C2.Number \neq C1.Number))) \}$$

Datalog:

MORETHANTWO (Tutor : t) ←
COURSES (Number: n, CourseName: cn, Tutor: t),
COURSES (Number: n1, CourseName: cn1, Tutor: t),
COURSES (Number: n1, CourseName: cn1, Tutor: t),
(n≠n1),(n≠n2),(n1≠n2)

EXACTLYTWO (FirstName: fn, Surname: sn, Course1: cn1, Course2: cn2) ←
TUTORS (Number : t, FirstName: fn, Surname: sn),
COURSES (Number: n1, CourseName: cn1, Tutor: t),
COURSES (Number: n2, CourseName: cn2, Tutor: t),
(n1≠n2),
NOT MoreThanTwo(Tutor: t)

Exercise 3.9

Consider a relational schema containing the relations:

$$R_1\,(ABC),\ R_2\,(DG),\ R_3(EF)$$

Formulate in tuple and domain relational calculus, the query formulated in relational algebra with the following expression:

$$(R_3 \bowtie_{G=E} R_2) \cup (\rho_{DG \leftarrow AC} (\Pi_{ACEF} (R_1 \bowtie_{B=F} R_3)))$$

Solution:

This expression cannot be formulated in tuple calculus because of the presence of the union between different tables.
In domain calculus, the expression becomes:

$$\{ D: d, G: g, E: e, F: f \mid R_3(E:e, F:f) \wedge ((R_2(D: d, G: g) \wedge (g=e)) \vee (R_1(A: d, B: b, C: g) \wedge (b=f))) \}$$

Exercise 3.10

With reference to the schema in Exercise 3.9, formulate in relational algebra the queries specified in domain calculus by means of the following expression:

$$\{ H: g, B: b \mid R_1(A: a, B: b, C: c) \wedge R_2(D: c, G: g) \}$$
$$\{ A: a, B: b \mid R_2(D:a , G: b) \wedge R_3 (E: a, F: b) \}$$
$$\{ A: a, B: b \mid R_1(A:a , B: b, C: c) \wedge \exists a' (R_1(A: a' , B: b, C: c) \wedge a \neq a') \}$$
$$\{ A: a, B: b \mid R_1(A:a , B: b, C: c) \wedge \forall a' (\neg R_1(A: a' , B: b, C: c) \vee a=a') \}$$
$$\{ A: a, B: b \mid R_1(A:a , B: b, C: c) \wedge \neg\exists a'(R_1(A: a' , B: b, C: c) \wedge a \neq a') \}$$

Solution:

$$\rho_{H \leftarrow G} (\Pi_{BG} (R_1 \rtimes_{C=D} R_2))$$
$$\rho_{AB \leftarrow DG} (\Pi_{DG} (R_2 \rtimes_{(D=E) \wedge (G=F)} R_3))$$
$$\Pi_{AB}(\sigma_{A \neq A1} (R_1 \rtimes_{(B=B1) \wedge (C=C1)} \rho_{A1,B1,C1 \leftarrow ABC}(R_1)))$$
$$\Pi_{AB} (R_1) - \Pi_{AB}(\sigma_{A \neq A1} (R_1 \rtimes_{(B=B1) \wedge (C=C1)} \rho_{A1,B1,C1 \leftarrow ABC}(R_1)))$$
$$\Pi_{AB} (R_1) - \Pi_{AB}(\sigma_{A \neq A1} (R_1 \rtimes_{(B=B1) \wedge (C=C1)} \rho_{A1,B1,C1 \leftarrow ABC}(R_1)))$$

Exercise 3.11

Consider the following algebraic expression:

$$\Pi_{ADH} (\sigma_{(B=C) \wedge (E=F) \wedge (A>20) \wedge (G=10)} ((R_1 \rtimes R_3) \rtimes R_2)$$

which refers to the schema

$$R_1(AB), R_2 (CDE), R_3(FGH)$$

and transform it, with the goal of reducing the size of the intermediate results.

Solution:

$$\Pi_{ADH} (\sigma_{A>20}(R_1) \rtimes_{B=C} \Pi_{CDH} (R_2 \rtimes_{E=F} \Pi_{FH} (\sigma_{G=10}(R_3))))$$