COMP9311 16s2 Database Systems

Exercises 01

Relational Design Theory, Relational Algebra

Notation: in the relational schemas below, primary key attributes are shown in **bold** font, foreign key attributes are shown in *italic* font, and primary key attributes that are also foreign keys are shown in ***bold italic*** font.

Example:

```
Student(id, name, degreeCode)
Degree(code, name, requirements)
Subject(code, name, syllabus)
Marks(studentId, subjectCode, session, mark)
```

In their respective relations, the student id, the degree code and the subject code are primary keys. In the Student relation, the degree code is a foreign key. In the Marks relation, the three attributes student id, subject code and session together form the primary key; the first two (student id and subject code) are also foreign keys.

Relational Design Theory

1. Consider the relation *R(A,B,C,D,E,F,G)* and the set of functional dependencies *F = { A → B, BC → F, BD → EG, AD → C, D → F, BEG → FA }* compute the following:

   a. A^+

   Answer:

   Derivation of A^+ ...
   given {A} ... using A → B gives {A,B} ... no further attributes can be added
   &impl; A^+ = {A,B}

   b. $ACEG^+$

   Answer:

   Derivation of $ACEG^+$
   given {A,C,E,G} ... using A → B gives {A,B,C,E,G} ... using BC → F gives {A,B,C,E,F,G} ... no more
   $ACEG^+$ = {A,B,C,E,F,G}

c. BD^+

Answer:

Derivation of BD^+
given {B,D} ... using BD → EG gives {B,D,E,G} ... using BEG → FA gives {A,B,D,E,F,G} ... using AD → C gives {A,B,C,D,E,F,G} ... no more
BD^+ = {A,B,C,D,E,F,G}

2. Consider the relation *R(A,B,C,D,E)* and the set set of functional dependencies $F = \{ A \rightarrow B, BC \rightarrow E, ED \rightarrow A \}$

   a. List all of the candidate keys for *R*.

   Answer:

   A candidate key is any set *X*, such that $X^+ = R$ and there is no *Y* subset of *X* such that $Y^+ = R$.

   In this case, the candidate keys are *CDE*, *ACD*, *BCD*.

   b. Is *R* in third normal form (3NF)?

   Answer:

   Yes, because the right hand sides of all dependencies (i.e. *B*, *E*, *A*) are parts of keys.

   c. Is *R* in Boyce-Codd normal form (BCNF)?

   Answer:

   No, because none of the left hand sides (i.e. *A*, *BC*, *ED*) contains a key.

3. Consider a relation *R(A,B,C,D)*. For each of the following sets of functional dependencies, assuming that those are the only dependencies that hold for *R*, do the following:

   a. List all of the candidate keys for *R*.

   b. Show whether *R* is in Boyce-Codd normal form (BCNF)?

   c. Show whether *R* is in third normal form (3NF)?

   i. $C \rightarrow D, \quad C \rightarrow A, \quad B \rightarrow C$

Answer:

a. Candidate keys: B
b. Not BCNF ... e.g. in $C \rightarrow A$, C does not contain a key
c. Not 3NF ... e.g. in $C \rightarrow A$, C does not contain a key, A is not part of a key

ii. $B \rightarrow C,\quad D \rightarrow A$

Answer:

a. Candidate keys: BD
b. Not 3NF ... neither right hand side is part of a key
c. Not BCNF ... neither left hand side contains a key

iii. $ABC \rightarrow D,\quad D \rightarrow A$

Answer:

a. Candidate keys: $ABC\quad BCD$
b. 3NF ... $ABC \rightarrow D$ is ok, and even $D \rightarrow A$ is ok, because A is a single attribute from the key
c. Not BCNF ... e.g. in $D \rightarrow A$, D does not contain a key

iv. $A \rightarrow B,\quad BC \rightarrow D,\quad A \rightarrow C$

Answer:

a. Candidate keys: A
b. Not 3NF ... e.g. in $A \rightarrow C$, C is not part of a key
c. Not BCNF ... e.g. in $BC \rightarrow D$, BC does not contain a key

v. $AB \rightarrow C,\quad AB \rightarrow D,\quad C \rightarrow A,\quad D \rightarrow B$

Answer:

a. Candidate keys: $AB\quad BC\quad CD\quad AD$
b. 3NF ... for AB case, first two fd's are ok, and the others are also ok because the RHS is a single attribute from the key
c. Not BCNF ... e.g. in $C \rightarrow A$, C does not contain a key

vi. $A \rightarrow BCD$

Answer:

a. Candidate keys: A
b. 3NF ... all left hand sides are superkeys
c. BCNF ... all left hand sides are superkeys

4. Consider a banking application that contains information about accounts, branches and customers. Each account is held at a specific branch, but a customer may hold more than one account and an account may have more than one associated customer.

Consider an unnormalised relation containing all of the attributes that are relevant to this application:

- *acct#* - unique account indentifier
- *branch#* - unique branch identifier
- *tfn* - unique customer identifier (**t**ax **f**ile **n**umber)
- *kind* - type of account (savings, cheque, ...)
- *balance* - amount of money in account
- *city* - city where branch is located
- *name* - customer's name

i.e. consider the relation *R(acct#, branch#, tfn, kind, balance, city, name)*

Based on the above description:

a. Devise a suitable set of functional dependencies among these attributes.

Answer:

acct# → kind, balance, branch#
branch# → city
tfn → name

These all come from the semantics of the problem (e.g. each account has exactly one type and balance, and is held at a specific branch).

b. Using these functional dependencies, decompose *R* into a set of BCNF relations.

Answer:

In this case, the 3NF decomposition from applying the above three *fd*s produces a BCNF decomposition as well. This is the simplest and most natural decomposition of the problem.

```
Account(acct#, kind, balance, branch)
Branch(branch#, city)
Customer(tfn, name)
CustAcc(customer, account)
```

The first three relations come directly from the dependencies; the last relation comes from the final check for a relation with a candidate key for *R* in the 3NF algorithm.

c. State whether the new relations are also in 3NF.

Answer:

Since we produced the new relations using the 3NF decomposition method, then they must be in 3NF.

5. Real estate agents conduct visits to rental properties

   - need to record which property, who went, when, results
   - each property is assigned a unique code (P#, e.g. PG4)
   - each staff member has a staff number (S#, e.g. SG43)
   - staff members use company cars to conduct visits
   - a visit occurs at a specific time on a given day
   - notes are made on the state of the property after each visit

   The company stores all of the associated data in a spreadsheet.

   Describe any functional dependencies that exist in this data. The table of sample data below below may give some ideas:

```
P#  | When        | Address    | Notes          | S#   | Name  | CarReg
----+-------------+------------+----------------+------+-------+-------
PG4 | 03/06 15:15 | 55 High St | Bathroom leak  | SG44 | Rob   | ABK754
PG1 | 04/06 11:10 | 47 High St | All ok         | SG44 | Rob   | ABK754
PG4 | 03/07 12:30 | 55 High St | All ok         | SG43 | Dave  | ATS123
PG1 | 05/07 15:00 | 47 High St | Broken window  | SG44 | Rob   | ABK754
PG2 | 13/07 12:00 | 12 High St | All ok         | SG42 | Peter | ATS123
PG1 | 10/08 09:00 | 47 High St | Window fixed   | SG42 | Peter | ATS123
PG3 | 11/08 14:00 | 99 High St | All ok         | SG41 | John  | AAA001
PG4 | 13/08 10:00 | 55 High St | All ok         | SG44 | Rob   | ABK754
PG3 | 05/09 11:15 | 99 High St | Bathroom leak  | SG42 | Peter | ATS123
```

   State assumptions used in determining the functional dependencies.

   Answer:

   - $P \rightarrow A$ ... the property code identifies a particular address
   - $PW \rightarrow N$ ... notes are based on a particular visit to a property
   - $PW \rightarrow S$ ... one staff member carries out each property visit
   - $S \rightarrow M$ ... the staff number determines the staff member's name
   - $S \rightarrow C$ ... each staff member uses a particular car (from table)

   Other dependencies are possible (e.g. $M \rightarrow S$), but they appear less plausible, given the semantics of the application.

6. Consider a company supplying temporary employees to hotels:

- the company has contracts with different hotels
- it may have several contracts with a given hotel
- contracts are identified by a code (e.g. C12345)
- staff work at different hotels as needed
- staff have tax file #'s (TFN, e.g. T123)
- hotels have Aus business #'s (ABN, e.g. H234)

Describe any functional dependencies that exist in this data. The table of sample data below below may give some ideas:

Contract	**T**FN	**N**ame	H**r**s	**A**BN	**H**otel
C12345	T311	John Smith	12	H765	Four Seasons
C18765	T255	Brad Green	12	H234	Crown Plaza
C12346	T311	John Smith	15	H765	Four Seasons
C12345	T255	Brad Green	10	H765	Four Seasons
C14422	T311	John Smith	6	H222	Sheraton
C14422	T888	Will Smith	9	H222	Sheraton
C18477	T123	Clair Bell	15	H222	Sheraton

State assumptions used in determining the functional dependencies.

Answer:

Describe any functional dependencies that exist in this data.

- $C \rightarrow A$ ... a contract is with a particular ABN
- $A \rightarrow H$ ... each hotel has a unique ABN
- $T \rightarrow N$ ... each employee has a unique TFN
- $CT \rightarrow R$ ... each employee works specified hours on a contract

Other potential dependencies that **do not** apply:

- $C \rightarrow T$ ... because many employees can work on a contract
- $H \rightarrow A$ ... two different hotels may have the same name (e.g. Hilton)

7. Consider the schema *R* and set of fds *F*

   $R = ABCDEFGH$
   $F = \{ ABH \rightarrow C,\ A \rightarrow DE,\ BGH \rightarrow F,\ F \rightarrow ADH,\ BH \rightarrow GE \}$

   Produce a BCNF decomposition of *R*.

   Answer:

This is just one of many possible decompositions. Variations occur because of choice of candidate key (although not in this example) and choice of non-BCNF FD to resolve at each step.

- We start from a schema: *ABCDEFGH*, with key *BH* (work it out from FDs).
- The FD $A \rightarrow DE$ violates BCNF (FD with non key on LHS).
- To fix, we need to decompose into tables: *ADE* and *ABCFGH*.
- FDs for *ADE* are $\{ A \rightarrow DE \}$, therefore key is *A*, therefore BCNF.
- FDs for *ABCFGH* are $\{ ABH \rightarrow C,\ BGH \rightarrow F,\ F \rightarrow AH,\ BH \rightarrow G \}$
- Key for *ABCFGH* is *BH*, and FD $F \rightarrow AH$ violates BCNF (FD with non key on LHS).
- To fix, we need to dcompose into tables: *AFH* and *BCFG*.
- FDs for *ADE* are $\{ F \rightarrow AH \}$, therefore key is *F*, therefore BCNF.
- FDs for *BCFG* are $\{ \}$, so key is *BCFG* and table is BCNF.
- Final schema (with keys boldened): ***A**DE*, ***F**AH*, ***BCFG***

8. Consider the schema *R* and set of fds *F*

$R = ABCDEFGH$
$F = \{ ABH \rightarrow C,\ A \rightarrow D,\ C \rightarrow E,\ BGH \rightarrow F,\ F \rightarrow AD,\ E \rightarrow F,\ BH \rightarrow E \}$
$F_c = \{ BH \rightarrow C,\ A \rightarrow D,\ C \rightarrow E,\ F \rightarrow A,\ E \rightarrow F,\ BH \rightarrow E \}$

Produce a 3NF decomposition of *R*.

Answer:

3NF is constructed directly from the minimal cover, after combinining dependencies with a common right hand side where possible.

$F_c = BH \rightarrow C,\ \ A \rightarrow D,\ \ C \rightarrow E,\ \ F \rightarrow A,\ \ E \rightarrow F$

Gives the following tables (with table keys in **bold**):

***BH**C* ***A**D* ***C**E* ***F**A* ***E**F*

A key for *R* is *BHG*; *G* must be included because it appears in no function dependency.

Since no table contains the whole key for *R*, we must add a table containing just the key, giving:

3NF = ***BH**C* ***A**D* ***C**E* ***F**A* ***E**F* ***BGH***

Relational Algebra

9. Consider the following two relations:

R

A	B	C
a1	b1	c1
a1	b2	c2
a2	b1	c1

S

B	C
b1	c1
b2	c2

Give the relation that results from each of the following relational algebra expressions on these relations:

a. *R* Div *S*

Answer:

R Div *S* =

A
a1

b. *R* Div (Sel[B != b1](*S*))

Answer:

Tmp1 = Sel[B != b1](*S*) =

B	C
b2	c2

R Div *Tmp1* =

A
a1

c. *R* Div (Sel[B != b2](*S*))

Answer:

Tmp1 = Sel[B != b2](*S*) =

B	C
b1	c1

R Div *Tmp1* =

A
a1
a2

d. *R* × *S*) - (Sel[R.C=S.C](*R* Join[B=B] *S*)

Answer:

Tmp1 = *R* Join[R.B=S.B] *S* =

R.A	R.B	R.C	S.B	S.C
a1	b1	c1	b1	c1
a1	b2	c2	b2	c2
a2	b1	c1	b1	c1

Tmp2 = Sel[R.C=S.C](*Tmp1*) =

R.A	R.B	R.C	S.B	S.C
a1	b1	c1	b1	c1
a1	b2	c2	b2	c2

a2	b1	c1	b1	c1

Tmp3 = R × S =

R.A	R.B	R.C	S.B	S.C
a1	b1	c1	b1	c1
a1	b1	c1	b2	c2
a1	b2	c2	b1	c1
a1	b2	c2	b2	c2
a2	b1	c1	b1	c1
a2	b1	c1	b2	c2

Tmp3 - Tmp2 =

R.A	R.B	R.C	S.B	S.C
a1	b1	c1	b2	c2
a1	b2	c2	b1	c1
a2	b1	c1	b2	c2

10. Consider two relations *R1* and *R2*, where *R1* contains *N1* tuples and *R2* contains *N2* tuples, and *N1* > *N2* > 0. Give the minimum and maximum possible sizes (in tuples) for the result relation produced by each of the following relational algebra expressions. In each case state any assumptions about the schemas of *R1* and *R2* that are needed to make the expression meaningful.

    a. *R1* Union *R2*
    b. *R1* Intersect *R2*
    c. *R1* - *R2*
    d. *R1* × *R2*
    e. Sel$_{[a=5]}$(*R1*)
    f. Proj$_{[a]}$(*R1*)

g. *R1* Div *R2*

Answer:

Expression	**Assumptions**	**Min**	**Max**
R1 Union *R2*	*R1* and *R2* are union-compatible	*N1* (when $R2 \subset R1$)	*N1+N2* (when $R1 \cap R2 = \varnothing$)
R1 Intersect *R2*	*R1* and *R2* are union-compatible	*0* (when $R1 \cap R2 = \varnothing$)	*N2* *(when $R2 \subset R1$)*
R1 - R2	*R1* and *R2* are union-compatible	*N1-N2* (when $R2 \cap R1 = R2$)	*N1* (when $R1 \cap R2 = \varnothing$)
R1 × R2	None	*N1*N2*	*N1*N2*
$\text{Sel}_{[a=5]}(R1)$	*R1* has an attribute *a*	*0* (no matches)	*N1* (all match)
$\text{Proj}_{[a]}(R1)$	*R1* has an attribute *a*, *N1 > 0*	*1*	*N1*
R1 Div *R2*	The set of *R2*'s attributes is a subset of *R1*'s attributes	*0*	*floor(N1/N2)*

11. Consider the following relational schema:

```
Suppliers(sid, sname, address)
Parts(pid, pname, colour)
```

```
    Catalog(sid, pid, cost)
```

Assume that the ids are integers, that `cost` is a real number, that all other attributes are strings, that the ***`sid`*** field in Catalog is a foreign key containing a supplier id, and that the ***pid*** field in Catalog is a foreign key containing a part id. Write a relational algebra expression to answer each of the following queries:

a. Find the *names* of suppliers who supply some red part.

Answer:

```
RedPartIds = Proj[pid](Sel[colour='red'](Parts))
RedPartSupplierIds = Proj[sid](RedPartIds Join Catalog)
Answer = Proj[sname](RedPartSupplierIds Join Suppliers)
```

b. Find the *sids* of suppliers who supply some red or green part.

Answer:

```
RedGreenPartIds = Proj[pid](Sel[colour='red' OR colour='green'](Parts))
Answer = Proj[sid](RedGreenPartIds Join Catalog)
```

c. Find the *sids* of suppliers who supply some red part or whose address is 221 Packer Street.

Answer:

```
RedPartIds = Proj[pid](Sel[colour='red'](Parts))
RedPartSupplierIds = Proj[sid](RedPartIds Join Catalog)
PackerStSupplierIds = Proj[sid](Sel[address='221 Packer Street'](Suppliers))
Answer = RedPartSupplierIds Union PackerStSupplierIds
```

d. Find the *sids* of suppliers who supply some red part and some green part.

Answer:

```
RedPartIds = Proj[pid](Sel[colour='red'](Parts))
RedPartSupplierIds = Proj[sid](RedPartIds Join Catalog)
GreenPartIds = Proj[pid](Sel[colour='green'](Parts))
GreenPartSupplierIds = Proj[sid](GreenPartIds Join Catalog)
Answer = RedPartSupplierIds   Intersect   GreenPartSupplierIds
```

e. Find the *sids* of suppliers who supply every part.

Answer:

```
AllPartIds = Proj[pid](Parts)
PartSuppliers = Proj[sid,pid](Catalog)
Answer = PartSuppliers Div AllPartIds
```

1. Proj 投影，取出pid的列
2.Proj 投影，取出sid，pid的列
3.Div：包含了AllpartIds里边的所有sid

f. Find the *sids* of suppliers who supply every red part.

Answer:

```
RedPartIds = Proj[pid](Sel[colour='red'](Parts))
PartSuppliers = Proj[sid,pid](Catalog)
Answer = PartSuppliers Div RedPartIds
```

g. Find the *sids* of suppliers who supply every red or green part.

Answer:

```
RedGreenPartIds = Proj[pid](Sel[colour='red' OR colour='green'](Parts))
Answer = PartSuppliers Div RedGreenPartIds
```

h. Find the *sids* of suppliers who supply every red part or supply every green part.

Answer:

```
RedPartIds = Proj[pid](Sel[colour='red'](Parts))
GreenPartIds = Proj[pid](Sel[colour='green'](Parts))
PartSuppliers = Proj[sid,pid](Catalog)
AllRedPartSuppliers = PartSuppliers Div RedPartIds
AllGreenPartSuppliers = PartSuppliers Div GreenPartIds
Answer = AllRedPartSuppliers Union AllGreenPartSuppliers
```

i. Find the *pids* of parts that are supplied by at least two different suppliers.

Answer:

```
C1 = Catalog
```

```
C2 = Catalog
SupplierPartPairs = Sel[C1.sid!=C2.sid](C1 Join[pid] C2)
Answer = Proj[C1.pid](SupplierPartPairs)
```

j. Find pairs of *sids* such that the supplier with the first *sid* charges more for some part than the supplier with the second sid.

Answer:

```
C1 = Catalog
C2 = Catalog
SupplierPartPairs = Sel[C1.sid!=C2.sid AND C1.cost>C2.cost](C1 Join[pid] C2)
Answer = Proj[C1.sid,C2.sid](SupplierPartPairs)
```

k. Find the *pids* of the most expensive part(s) supplied by suppliers named "Yosemite Sham".

Answer:

```
R1 = Proj[sid,pid,cost](Sel[name='Yosemite Sham'](Suppliers Join Catalog))
R2 = R1
R3 = Rename[1->sid,2->pid,3->cost](Sel[R2.cost<R1.cost](R1 × R2))
Answer = Proj[pid](R2 Minus Proj[sid,pid,cost](R3))
```

l. Find the *pids* of parts supplied by every supplier at a price less than 200 dollars (if any supplier either does not supply the part or charges more than 200 dollars for it, the part should not be selected).

Answer:

```
CheapParts = Sel[cost<200](Catalog)
AllSuppliers = Proj[sid](Suppliers)
Answer = Proj[part,supplier](CheapParts Div AllSuppliers)
```

12. Consider the following relational schema containing flight, aircraft and pilot information for an airline:

```
Flights(flno, from, to, distance, departs, arrives)
Aircraft(aid, aname, cruisingRange)
Certified(employee, aircraft)
Employees(eid, ename, salary)
```

The `Flights` relation describes a particular timetabled fight from a source (city) to a destination (city) at a particular time each week. Note that this schema doesn't model which specific aircraft makes the flight. The `Aircraft` relation describes individual planes, with the `aname` field containing values like `'Boeing 747'`, `'Airbus A300'`, etc. The `Certified` relation indicates which pilots (who are employees) are qualified to fly which aircraft. The ***`employee`*** field contains the *eid* of the pilot, while the ***`aircraft`*** field contains Finally, the `Employees` relation contains information about all of the employees of the airline, including the pilots.

Write, where possible, a relational algebra expression to answer each of the following queries. If it is not possible to express any query in relational algebra, suggest what extra mechanisms might make it possible.

a. Find the *ids* of pilots certified for 'Boeing 747' aircraft.

Answer:

Approach: collect together information about aicraft and certification and project out just the employee ids for tuples that involve 747's.

```
Answer = Proj[eid](Sel[aname='Boeing 747'](Aircraft Join Certified))
```

b. Find the *names* of pilots certified for 'Boeing 747' aircraft.

Answer:

Approach: collect together information about pilots and what planes they are certified on and project out just the name (since we need the pilot's name this time, we need to include `Employees`).

```
Answer = Proj[ename](Sel[aname='Boeing 747'](Aircraft Join Certified Join Employees))
```

c. Find the *ids* of all aircraft that can be used on non-stop flights from New York to Los Angeles.

Answer:

Approach: get all of the flights from New York to Los Angeles (to get the distance), join these with aircraft information and select only those combinations where the plane is capable of cruising that far. If you were fussy, you could also include flights from Los Angeles to New York with a disjunction in the inner `Sel` (in case the distance was different for some reason e.g. slightly different route).

```
Answer =
        Proj[aid](
                Aircraft
                Join[cruisingRange>distance]
                Sel[from='New York' AND to='Los Angeles'](Flights)
        )
```

d. Identify the flights that can be piloted by a pilot whose salary is more than $100,000.

Answer:

Approach: collect together information about which planes can make which flights and pilot certification, then select just those for which the pilot is certified and where the pilot earns more than $100000.

```
Temp   =
        Proj[flno,aid](
                Flights
                Join[distance<cruisingRange]
                Aircraft
        )
Answer =
        Proj[flno](
                Sel[salary>100000](
                        Temp Join Certified Join Employees
                )
        )
```

e. Find the names of pilots who can operate planes with a range greater than 3000 miles, but are not certified on 'Boeing 747' aircraft.

Answer:

Approach: get a list of employee ids who are certified for aircraft with a crusing range more than 3000 miles, then subtract the employee ids for those who *are* certified on a 747, leaving just those who are *not* certified on a 747; to get their names, join with the employees relation and project out the names.

```
LongRangeAircraftCert = Proj[eid](Sel[cruisingRange>3000](Aircraft Join Certfied))
Boeing747Cert = Proj[eid](Sel[aname='Boeing 747'](Aircraft Join Certfied))
Answer =
        Proj[ename](
                Employees Join (LongRangeAircraftCert Minus Boeing747Cert)
        )
```

f. Find the total amount paid to employees as salaries.

Answer:

```
Answer = Sum[salary](Employees)
```

g. Find the *ids* of employees who make the highest salary.

Answer:

Approach: find employees who do not have the highest salary (via the join), and then subtract them from the set of all employees, thus leaving just the highest paid employees.

```
E1 = Employees
E2 = Employees
Employees = Proj[eid](Employees)
LowerPaidEmployees = Proj[E2.eid](E1 Join[E1.salary>E2.salary] E2)
Answer = Employees Minus LowerPaidEmployees
```

h. Find the *ids* of employees who make the second highest salary.

Answer:

Approach: use the idea from the previous question; first find highest paid employees, then remove them from list of employees, then find highest paid employees in what's left; these will be the second highest-paid employees.

```
E1 = Employees
E2 = Employees
HighestPaid =
        Proj[eid,salary](Employees)
        Minus
        Proj[E2.eid,E2.salary](E1 Join[E1.salary>E2.salary] E2)
NotHighestPaid =
        Proj[eid,salary](Employees)  Minus  HighestPaid
E3 = NotHighestPaid
E4 = NotHighestPaid
SecondHighestPaid =
        NotHighestPaid
        Minus
        Proj[E4.eid,E4.salary](E3 Join[E3.salary>E4.salary] E4)
```

i. Find the *ids* of employees who are certified for the largest number of aircraft.

Answer:

Approach: XXX

```
R1 = GroupBy[employee,Count[employee]](Certified)
R2 = Rename[1->employee,2->ncertified](R1)
R3 = Max[ncertified](R2)
Answer = Sel[ncertified=R3](R1)
```

j. Find the *ids* of employees who are certified for exactly three aircraft.

Answer:

Approach: find pilots who are certified for at least three aircraft; then find pilots who are certified for at least four aircraft; then subtract the first set from the latter and you'll pilots who are certified for exactly three aircraft; the "counting" is built into the join condition.

```
R1 = R2 = R3 = R4 = Certified
CertifiedForAtLeast3 =
        Proj[eid](
                Sel[R1.eid=R2.eid=R3.eid AND R1.aid!=R2.aid!=R3.aid](
                        R1 × R2 × R3
                )
        )
CertifiedForAtLeast4 =
        Proj[eid](
                Sel[R1.eid=R2.eid=R3.eid=R4.eid AND R1.aid!=R2.aid!=R3.aid!=R4.aid](
                        R1 × R2 × R3 × R4
                )
        )
Answer = CertifiedForAtLeast3 Minus CertifiedForAtLeast4
```

k. Find if there is any non-stop or single-stop way to travel by air from Sydney to New York.

Answer:

Approach: find any non-stop flights from Sydney to New York; find any single-stop flights from Sydney to New York; take the union of these two sets.

```
F1 = F2 = Flights
SydToNYNonStop = Proj[flno](Sel[from='Sydney' AND to='New York'](Flights))
SydToNYSingleStop = Proj[flno](Sel[F1.from='Sydney' AND F2.to='New York'](F1 Join[F1.to=F2.from] F2))
```

```
Answer = SydToNYNonStop Union SydToNYSingleStop
```

l. Is there a sequence of flights from Sydney to Timbuktu (somewhere in the middle of Africa)? Each flight in the sequence is required to depart from the destination of the previous flight; the first flight must leave Sydney, and the last flight must arrive in Timbuktu, but there is no restriction on the number of intermediate flights.

   Answer:

   This is a generalisation of the previous question to routes with an arbitrary number of stops. This cannot be expressed in standard relational algebra. It requires some kind of iterative or recursive querying mechanism before it will work. If we're willing to put an upper bound on the number of intermediate stops, we can generalise the previous query to do it.