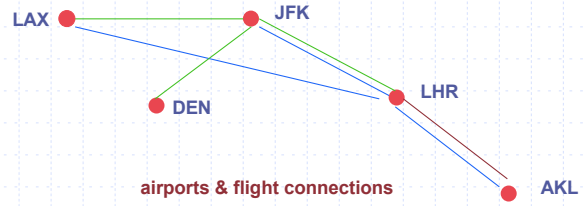
Case Study AIRLINE DB



airports & flight connections


```
-- airlines are abstract entities whose names are recorded

data Airline = BA | UA | NZ  deriving ( Eq, Show)
allAirlines  ::  [Airline]
allAirlines = [BA, UA, NZ]

type AirlineName  =  String
airlineName  ::  Airline -> AirlineName
airlineName  BA  =  "British Airways"
airlineName  UA  =  "United Airlines"
airlineName  NZ  =  "Air New Zealand"
```

```
-- airports are abstract entities, too

data  Aiport  = LHR | JFK  | DEN  | LAX |  AKL
                deriving ( Eq, Show)
allAirports  :: [Airport]
allAirports = [LHR,  JFK,   DEN,   LAX,   AKL]

type AirportName  =  String
type Country  =  String
type AirportInfo  =  ( AirportName, Country )

airportInfo  ::  Airport -> AirportInfo
airportInfo  LHR  =  ("London Heathrow", "England")
airportInfo  JFK  =  ("J F Kennedy", "United States")
airportInfo  DEN  =  ("Denver", "United States")
airportInfo  LAX  =  ("Los Angeles Int", "United States")
airportInfo  AKL  =  ("Auckland", "New Zealand")

airportName  ::  Airport -> AirportName
airportName   x  =  firstOf2 (airportInfo  x)

airportCountry  ::  Airport -> Country
airportCountry  x  = secondOf2  (airportInfo  x)
```



```
-- flights are abstract entities (airline, source, destination)

data  Flight  = BA1 | UA1 | UA123 | UA987 | UA234 | UA842 | NZ2
                deriving ( Eq, Show)
allFlights  :: [ Flight ]
allFlights = [BA1,  UA1,  UA123,  UA987,  UA234,  UA842,  NZ2 ]

flightInfo  ::  Flight -> (Airline, Airport, Airport)
flightInfo   BA1      =  (BA, LHR, JFK)
flightInfo   UA1      =  (UA, LHR, JFK)
flightInfo   UA123    =  (UA, JFK, DEN)
flightInfo   UA987    =  (UA, LHR, LAX)
flightInfo   UA234    =  (UA, DEN , LAX)
flightInfo   UA842    =  (UA, LAX, AKL)
flightInfo   NZ2      =  (NZ, LAX, AKL)

flightAirline   ::  Flight  -> Airline
flightAirline  f  =  firstOf3 (flightInfo  f)

flightSource   ::  Flight  -> Airport
flightSource  f  =  secondOf3 (flightInfo  f)

flightDest   ::  Flight  -> Airport
flightDest  f  =  thirdOf3 (flightInfo  f)
```

-- codes of the airports located in the United States

```
[ p | p <- allAirports, airportCountry p = "United States"]
```

-- all airports flown to/from by a given airline

```
serves  :: Airline  ->  [ Airport ]
serves   x =
            [flightSource  f  |  f <- allFlights, flightAirline  f == x]  ++
            [flightDest  f  |  f <- allFlights, flightAirline  f == x]
```


-- names of the airlines serving a given country

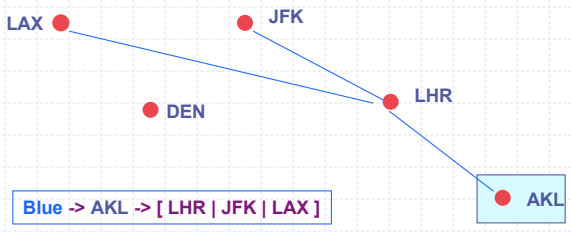
```
countryAirlines  ::  Country  ->  [ AirlineName ]
countryAirlines  y  = [airlineName  f  |
                                    f <- allAirlines,
                                    p <- serves  f,
                                    airportCountry  f == y]
```

-- all airports from where an airline flies to more than one destination

```
hubs :: Airline -> [ Airport ]
hubs x =
        [p | p <- allAirports,
         f1 <- allFlights,
        flightAirline f1 == x,
        flightSource f1 == p,
        f2 <- allFlights,
        flightAirline f2 == x,
        flightSource f2 == p,
        flightDest f1 /= flightDest f2]
```


-- all airports reachable from a given airport on a given airline

```
getthere :: Airline -> Airport -> [Airport]
getthere x y =
        dests ++ [y' | d <- dests, y' <- getthere x d]
        where dests = [ flightDest f | f <- allFlights,
                        flightAirline f == x, flightSource f == y]
```



Blue -> AKL -> [LHR | JFK | LAX]

Relational AIRLINE DB

LINE

ID	NAME
BA	British Airways

PORT

CODE	NAME	COUNTRY
LHR	Heathrow	England

CONNECT

No	L-ID	ORIG	DEST
704	BA	LHR	VIE

```
LINE (
  ID      char(3) primary key
  NAME varchar2(25))

PORT (
  CODE       char (3) primary key
  NAME       varchar2(25)
  COUNTRY  varchar2925))

CONNECT (
  No      number   primary key
  L-ID    char(3) ref   LINE(ID)
  ORIG   char(3) ref  PORT(CODE)
  DEST   char(3) ref  PORT(CODE))
```


-- airport codes located in the United States

```
[ p | p <- allAirports,
    airportCountry p = "United States"]
```

```
select ID from PORT
where COUNTRY = "United States"
```

```
[] ([] PORT (COUNTRY =' United States")) ID
```

-- airports served by a given airline

```
serves  x =
  [flightSource  f  |  f <- allFlights, flightAirline  f == x]
  ++ [flightDest  f  |  f <- allFlights, flightAirline  f == x]
```

```
select unique  LINE.NAME
from    LINE, PORT, CONNECT
where  LINE.ID = CONNECT.L-ID
and     (CODE = ORIG  or CODE = DEST)
and     LINE.NAME = x
```

```
□ (□ ((LINE ▸◂  PORT) ▸◂ CONNECT)
     (CODE = ORIG  or CODE = DEST ))
    NAME
```


-- airlines serving a given country

```
countryAirlines  y  = [airlineName  f  |
                f <- allAirlines, p <- serves  f,
                airportCountry  f == y]
```

```
select unique  LINE.NAME
from    LINE, PORT, CONNECT
where  LINE.ID = CONNECT.L-ID
and     (CODE = ORIG  or CODE = DEST)
and     PORT.COUNTRY = y
```

```
□ (□ ((LINE ▸◂  PORT) ▸◂ CONNECT)
     (CODE = ORIG  or CODE = DEST))
    COUNTRY
```

-- airports from where an airline flies to more than one destination

```
hubs :: Airline -> [ Airport ]
hubs  x =[p | p <- allAirports,
   f1 <- allFlights, flightAirline f1 == x, flightSource f1 == p,
   f2 <- allFlights, flightAirline f2 == x, flightSource f2 == p,
   flightDest f1 /= flightDest f2]
```

```
select ORIG from CONNECT
where L-ID = x
group by ORIG
      having count (*) > 1
```

[] ::= [] ([] (CONNECT (L-ID = x))) (ORIG, DEST)

returns all connection pairs for x - but R/Algebra does not provide for grouping, nor counting

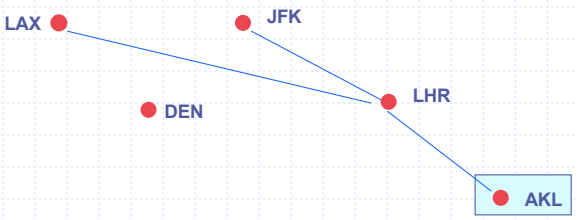
if [] :: Relation [] List *(i.e. with repetitions) existed, than*

[] A (ORIG) - [] A (ORIG)

would give the answer

-- all airports reachable from a given airport on a given airline

```
getthere  x  y =
   dests  ++  [y'  | d <-  dests,  y' <- getthere  x  d]
   where dests = [ flightDest  f | f <- allFlights,
                       flightAirline  f == x,  flightSource  f == y]
```

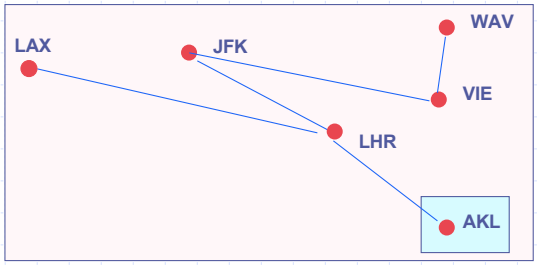



```
select DEST from CONNECT
where L-ID = x
and ORIG = y
```

→ (Blue, AKL) -> LHR

```
SQL> select * from GRAPH;

ORIG DEST
---  ---
AKL  LHR
LHR  JFK
LHR  LAX
JFK  VIE
VIE  WAW

SQL> get q1
  1  select level, dest
  2  from graph
  3  connect by prior dest = orig
  4* start with orig = 'AKL'
SQL> /

   LEVEL  DEST
  ------  ----
  1        LHR
    2      JFK
      3    VIE
        4  WAW
    2      LAX

SQL>
```

```
getthere  x  y =
    dests  ++  [y'  | d <-  dests,  y' <- getthere  x  d]
    where dests = [ flightDest  f | f <- allFlights,
                          flightAirline  f == x,  flightSource  f == y]
```




```
select LEVEL, ORIG, DEST
from CONNECT where L-ID = x
        connect by prior DEST = ORIG
        start with ORIG = y
```