Recursive Search

```
WITH trips (destination, route, nsegs, totalcost) AS
     ((SELECT destination, CAST(destination AS varchar(20)), 1, cost
       FROM flights                                          --- initial query
       WHERE origin='SFO')
     UNION ALL
      (SELECT f.destination,                                 --- recursive query
                 CAST(t.route||','||f.destination AS varchar(20)),
                 t.nsegs+1, t.totalcost+f.cost
       FROM trips t, flights f
       WHERE t.destination=f.origin
                 AND f.destination<>'SFO'                    --- stopping rule 1
                 AND f.origin<>'JFK'                         --- stopping rule 2
                 AND t.nsegs<=3))                            --- stopping rule 3
SELECT route,  totalcost                                     --- final query
FROM trips
WHERE destination='JFK' AND totalcost=                       --- lowest cost rule
                                  (SELECT min(totalcost)
                                   FROM trips
                                   WHERE destination='JFK') ;
```

Result

Trips

Destination	Route	Nsegs	Totalcost
DFW	DFW	1	300
ORD	ORD	1	275
LAX	LAX	1	50
JFK	DFW, JFK	2	525
LAX	DFW, LAX	2	500
ORD	DFW, ORD	2	400
DFW	LAX, DFW	2	250
DFW	ORD, DFW	2	375
JFK	ORD, JFK	2	525
DFW	DFW, LAX, DFW	3	700
DFW	DFW, ORD, DFW	3	500
JFK	DFW, ORD, JFK	3	650
LAX	LAX, DFW, LAX	3	450
JFK	LAX, DFW, JFK	3	475
ORD	LAX, DFW, ORD	3	350
LAX	ORD, DFW, LAX	3	575
JFK	ORD, DFW, JFK	3	600
ORD	ORD, DFW, ORD	3	475

Final result

route	totalcost
LAX, DFW, JFK	475

Recursive Search

- *Only change the final query slightly, the least transfer time routes can be found :*

… …

```
SELECT route,  totalcost                          --- final query
FROM trips
WHERE destination='JFK' AND nsegs=                --- least stop rule
                              (SELECT min(nsegs)
                               FROM trips
                               WHERE destination='JFK') ;
```

Final result

route	totalcost
DFW, JFK	525
ORD, JFK	525

Data Manipulation Language

- Insert
  - Insert a tuple into a table
    - *INSERT INTO EMPLOYEES VALUES ('Smith', 'John', '1980-06-10', 'Los Angles', 16, 45000);*
- Delete
  - Delete tuples fulfill qualifications
    - *DELETE FROM Person WHERE LastName = 'Rasmussen' ;*
- Update
  - Update the attributes' value of tuples fulfill qualifications
    - *UPDATE Person SET Address = 'Zhongshan 23', City = 'Nanjing' WHERE LastName = 'Wilson';*

View in SQL

- General view
  - Virtual tables derived from base tables
  - Logical data independence
  - Security of data
  - Update problems of view
- Temporary view and recursive query
  - WITH
  - RECURSIVE

Update problems of view

- CREATE VIEW YoungSailor AS

  SELECT sid, sname, rating

  FROM Sailors

  WHERE age<26;

- CREATE VIEW Ratingavg AS

  SELECT rating, AVG(age)

  FROM Sailors

  GROUP BY rating;