# Database design

## Tables

Room(rName, level, prevRoom, coords, description)

Cords refers the location of the item on the floor plan

Description is a general description of the room’s funcion

Staff(tutorID, lastName, firstName, room)

Room is the office for the member of staff

Route(from, to, route)

Route is the text description of the route

## Route finding algorithm

### Skeleton and descriptions:

**Route(from, to, stepFree): List of String**

1. Backtrack to height 0 node for both from and to storing list of visited nodes
2. A)If different level then...

B)If same level then get earliest common node ‘ecn’

1. Return routeNav(btcFrom, ecn, LR) + routeNav(btcTo, ecn, RL)

**Backtrack(from): List of Room**

// using prevRoom value in table create an list of rooms backtracking from the parameter until no prevRoom found. This is known as a height 0 node.

**ECN(btcFrom, btcTo): Room**

// work from the last item in the backtracked lists finding out the leftmost room that the lists have in common.

**RouteNav(backtrack, ecn, direction): List of String directions**

// use backtrack list to call route table up to ecn. Direction is either left to right or right to left when using backtrack.

### Pseudo code

**Route(from, to, stepFree): List of String**

btcFrom = backtrack(from)

btcTo = backtrack(to)

if(from.level == to.level) then

ecn = ECN(btcFrom, btcTo)

return routeNav(from, ecn) + routeNav(ecn, to)

end if

**Backtrack(from): List of Room**

List btc

btc.add(from)

while (prevRoom = *SELECT prevRoom FROM Room WHERE rName = from.rName* != ‘’) do

btc.addprevRoom

od

**ECN(btcFrom, btcTo): Room**

fromPoint = btcFrom.length – 1

toPoint = btcTo.length -1

while( btcFrom.get(fromPointer) = btcFrom.get(toPointer) ) do

fromPointer = fromPointer -1

toPointer = toPointer – 1

od

return btcFrom.get(fromPointer + 1)

**RouteNav(backtrack, ecn, direction): List of String**