  - WT = Result of above evaluation.
  - Remove duplicates from WT.
  - Remove from WT any tuples that are also present in RT.

Note that the recursive union algorithm to build the ReachableFrom table require the use of the CYCLE clause, and is thus more complex and is inefficient.

3 Naive Algorithm

An even simpler algorithm, the naive algorithm, can be used to compute the ReachableFrom query.

- Define one result table RT, and a working table WT.
- Initialize $RT = WT = \phi$.
- WT = result of evaluating initialization and recursive part using value of RT for ReachableFrom.
- while (WT NOT EQUAL RT)
  - Remove duplicates from WT.
  - $RT = WT$.
  - WT = result of evaluating initialization and recursive part using value of RT for ReachableFrom.

The naive algorithm is simpler to understand. You simply evaluate the query repeatedly making better and better estimates for the final result table. This is the meaning of a fixpoint. However, the naive algorithm is inefficient as compared to the semi-naive algorithm, as it repeats computation done in earlier iterations.

Semi-Naive Algorithm for Recursive Queries

Nelson Mattos
IBM
mattos@VNET.IBM.COM

Inderpal Singh Mumick
Bell Laboratories
mumick@research.att.com

Hamid Pirahesh
IBM
pirahesh@almaden.ibm.com

March 11, 1996

1 Introduction

We will illustrate the semi-naive algorithm on recursive queries for recursion where duplicates must be eliminated. We will then give an even simpler algorithm, the naive algorithm, to evaluate such queries.

EXAMPLE 1.1

```
(R1):  WITH RECURSIVE ReachableFrom(Source, Dest) AS
       (SELECT Source, Dest
        FROM Flights
        UNION
        SELECT in.Source, out.Dest
        FROM ReachableFrom in, Flights out
        WHERE in.Dest = out.Source
       )
       SELECT * FROM ReachableFrom ;
```

The query above lists all pairs of cities such that one can fly from one to the other. Duplicates are not important, and the UNION operator specifies that duplicates be eliminated. The expression (SELECT Source, Dest FROM Flights) is called the initialization part. The expression (SELECT in.Source, out.Dest FROM ReachableFrom in, Flights out WHERE in.Dest = out.Source) is called the recursive part.

We assume the the Flight table has cycles.

■

2 Semi-Naive Algorithm

The semi-naive algorithm works as follows:

- Define one work table WT, and one result table RT.
- Initialize $RT = WT = \phi$.
- WT = result of evaluating initialization part.
- Remove duplicates from WT.
- while (WT is not empty)
  - $RT = RT \cup WT$.
  - Evaluate recursive part using value of WT for ReachableFrom.