Composite and Non-equi Joins

**USF\_ID:U72254994(Venkata Satya Sai Lakshmi Lavanya)**

**Subject\_ID:ISM6218(Advanced Database Management System)**

1. **Run an experiment comparing an equi-join and a non-equi join**

The below screenshot displays a query that has both equi-join and non-equi join in the on clause. The equi

Join Part is used to make sure that dog\_IDs are displayed even if they are same, under the condition that

they received different services. The non-equi join part makes sure that only **varied** services received by a

particular dog are displayed.

Graphical user interface, application, Word

Description automatically generated



1. **Support a user story demonstrating the difference.**

In real life scenario, this data can be used for making service recommendations. For example,

Let us say that a dog received the same service most of the times it was brought to the dog salon

but occasionally let us assume that the Dog received a different service. In this scenario, even if

the dog received the same service most of the times there is still a slight probability that the customer

picks a different service for his dog. Hence, it can be useful to show up these occasional services

used, in the recommendations as it can prevent repetitive recommendations about the same service and

it can remind them of a different service that they opted for long ago. If we want to include results of dogs that

receive only one type of service too then instead of <> we can use = and this becomes an equi join.

1. **Run an experiment using a composite join.**

In the hospital database the nurses are assigned to a particular ward i.e., most of the days they are assigned to a particular Ward for that date. However, on Particular dates they rotate their assignment with other wards i.e., on particular days

they work in multiple wards with different time slots. The below mentioned composite join will return the results

of nurses who are assigned to multiple ward at different time slots on a particular day. Further a simple join is also used

in the below query to join the Ward ID with it’s location and Phone no.

Graphical user interface, text, application, Word

Description automatically generated



There is no way to obtain these results using only a simple join. In simple join each row in first table

(Nurse\_assignment) is compared to each row in the second table and every time the on clause condition is

matched results are returned.Simple joins are usually used when there is a single column in first table that matches

with the column values of second table. However, in case of self-join every column in first table matches

with every other column of second table hence it is difficult to use a simple join. When multiple conditions

are to be satisfied before returning rows and if these multiple Conditions use same tables repetitively then we use

composite join. Further as we can see the ward and ward\_assignment Table have only one column with matching

values i..e, the Ward\_Id column in Ward\_assignment table refers to the Ward\_Id column in ward table. Therefore,

here simple join is used to join the Ward\_assignment table and ward table i.e., the on clause doesn’t have multiple

conditions using ‘and’ keyword.

1. Support a user story demonstrating the difference.

The difference has been demonstrated above using a single example that has both composite and simple join.

In real life scenario, this data can be used to track deviations in regular shift assignments.

**Stored Procedures For above Queries(Sprocs)**

**Graphical user interface, application, Word

Description automatically generated**

**Graphical user interface, text, application, Word, email

Description automatically generated**

Graphical user interface

Description automatically generated with medium confidence

A picture containing graphical user interface

Description automatically generated