**Query Optimization Technique-Converted “NOT EXISTS” or “EXISTS” function to “INNER JOIN”**

Prepared by

**Infosys**

**INFOSYS LIMITED**

**COPYRIGHT NOTICE**

©2021 Infosys Limited, Bengaluru, India. All rights reserved. Infosys believes the information in this document is accurate as of its publication date; such information is subject to change without notice. Infosys acknowledges the proprietary rights of other companies to the trademarks, product names and such other intellectual property rights mentioned in this document. Except as expressly permitted, neither this document nor any part of it may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, printing, photocopying, recording or otherwise, without the prior permission of Infosys Limited and/or any named intellectual property rights holders under this document.

Infosys Limited

Hosur Road

Electronic City, 3rd Cross

Bangalore 560 100

India.

Telephone: (91) (80)28520 261-270

Fax: (91) (80) 8520 362

Website: <http://www.infosys.com>

Table of Contents

Overview………………………………………………………………………………………………………………………………………………………….3

Benefits………………………………………………………………………………………………………………………………………………………….3

Features…………………………………………………………………………………………………………………………………………………………3

Audience………………………………………………………………………………………………………………………………………………………..3

Technologies…………………………………………………………………………………………………………………………………………………..3

Script Code and How to use……………………………………………………………………………………………………………………………4

**Overview**

This query optimization technique can be used where a query is running very long (bad performing query) due to presence of NOT EXISTS / EXISTS sql function. These two functions do full table scan. Hence if a table has high volume of data (in Millions) then query consumes high I/O and CPU. In a result query runs for longer time(more than an hour) and output does not come. Hence to overcome such issue, NOT EXISTS / EXISTS can be converted to INNER JOIN so that query optimizer will not go for full table scan and will do join on required rows.

**Benefits**

* Consumption of I/O and CPU time will be less
* Query takes less time and gives output
* Performance of a query improves significantly

**Features**

* This technique can be used in all SQL poor performing queries

**Audience**

ETL Developer and SQL/PLSQL developer

**Technologies**

* Database (Teradata, Oracle etc)

**Script code and how to use**

Here comparison happens with the same table. Table1 has high volume of data that is in Millions

**Original Query:**

Select A.\* from **Table1 A**

NOT exists/exists (sel A.col1, A.col2,B.col3,B.col4,B.col5

from **Table1 B**

Where A.col1=B.col1

and A.col2=B.col2

and A.col3=B.col3

and A.col4=B.col4

and A.col5=B.col5 and **some other condition with Table1** B)

**Query Conversion to INNER JOIN:**

🡪Create a volatile table with all conditions available in the query

Create volatile table Table\_name

As (select \* from Table1 with data—with conditions

Primary index ( same as Table1)

On commit preserve rows;

🡪Collect statistics on volatile table as present in original table

Sel A.\*

From Table1 A INNER JOIN Table\_name K

A.col1=K.col1

and A.col2=K.col2

and A.col3=K.col3

and A.col4=K.col4

and A.col5=K.col5

