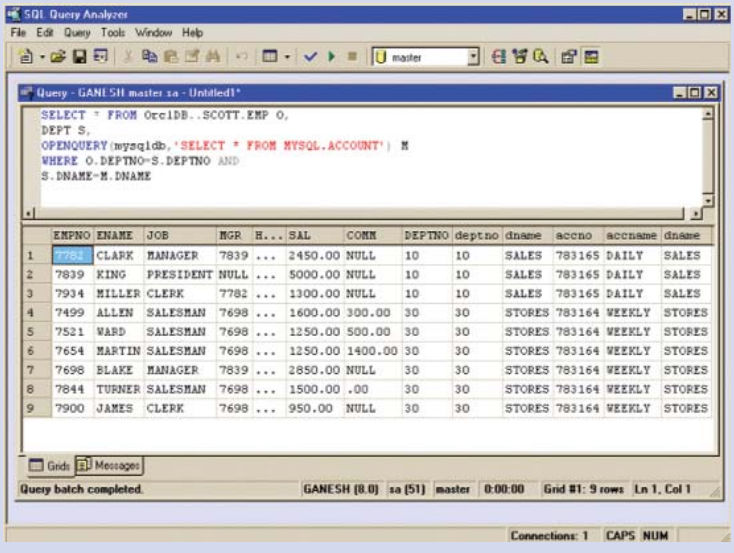
Distributed Queries in MS SQL

With a single query from My SQL Server 2000, you can query data from multiple databases. These are called distributed queries. In this article, we see how to create one

What makes MS SQL Server 2000 differ from other databases? One of the answers is distributed queries. With distributed queries you can query data in multiple databases with a single query from MS SQL Server. For instance, imagine a situation where you want data from a table named EMP in one of the databases (say Oracle), a table named DEPT in another database (say SQL Server) and a table named ACCOUNT in yet another database (say MySQL). In such situations we use distributed queries.

What is the need to query different databases? Think you are in a client-server environment and each computer maintains its own database, in fact a different database. Now, if the client needs to be given data from all the sources but need not know what the underlying databases are, then the system administrator can create a view by querying different databases and give the client permission to access the view. Now, the client can query all the sources as if it's a single source. Further, they can be used in migration from one or more legacy databases *(see screenshot below)*.

Snapshot	
Applies to	Database developers
USP	Query different databases at once



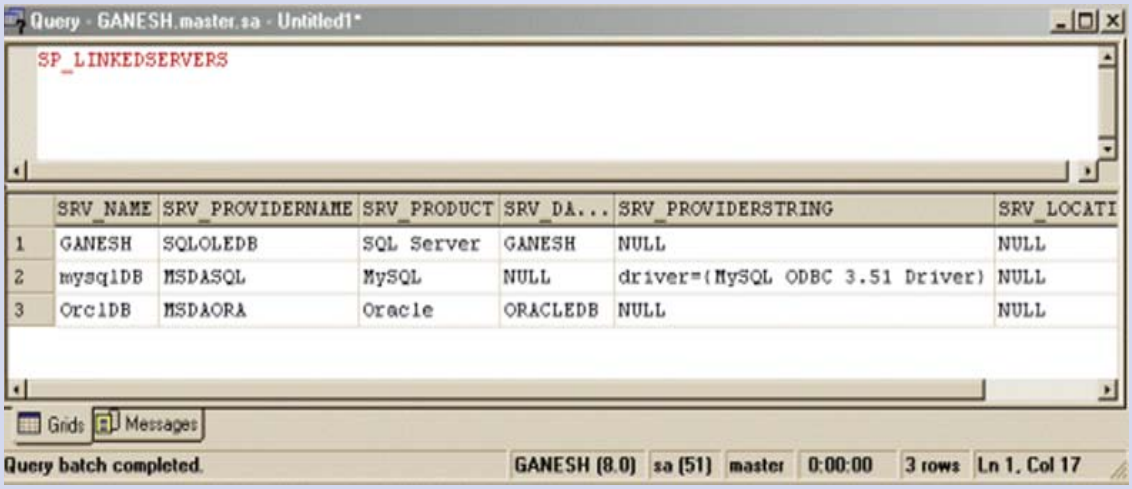
The screenshot shows a distributed query and its result involving three databases (Oracle, MS SQL Server and MySQL). The result shows all employee details (EMP) of a certain department (DEPT) having an account (ACCOUNT)

Let us see how we can create one. First, you need to create a linked server. A linked server is one, which once created, is used by SQL Server 2000 to implicitly connect to and retrieve data from it. To create a linked server for Oracle, follow the steps.

```
EXEC sp_addlinkedserver
'OrclDB','Oracle','MSDAORA','ORACLEDB'

EXEC sp_addlinkedsrvlogin 'Or
clDB',false,'sa','SCOTT','TIGER'
```

In the first step, we created a linked server where first argument refers to the server name, second to server product, third to the server provider name and the fourth to the server data source. The server data source name is the name of the oracle instance or an SQL * Net alias name that points to an Oracle data in-



The screenshot shows all the registered linked servers and their details

stance.

In the second step, we created the login credentials that are used by SQL Server to log to a particular database. The second argument specifies whether SQL Server login credentials be used or not. The default value is TRUE, which specifies that SQL Server uses its own username and password to connect with the remote server with remote_userid and remote_pwd arguments ignored .The third argument refers to the SQL Server user account name. The last two arguments refers to the remote_userid and remote_pwd. Now you can query Oracle from MS SQL as

```
SELECT * FROM OrclDB..SCOTT.EMP
```

SQL Server uses a four-part name to access a linked server. The syntax for this is

```
LinkedServerName.Catalog.Schema.Object-
Name
```

For Oracle the syntax slightly differs as:

```
OracleLinkedServerName..OwnerUserName.O
bjectName
```

For MySQL, the code slightly differs. Since MySQL provider does not have a friendly name (such as MSDAORA, MSDASQL etc) and instance name, we give MSDASQL as the provider name and Null as the data source.

```
sp_addlinkedserver 'mysqlDB', 'MySQL', 'MS-
DASQL', Null, Null, 'Driver= {MySQL ODBC
3.51 Driver}'
```

That's all for MySQL and no need to create a login, as they are not needed for MySQL. But observe that we have used two more arguments. The fifth argument here refers to the location, which is again a Null and sixth refers to the provider string. Now to query it we use a rowset function named 'OpenQuery ()'. This is because the MySQL provider does not support a four-part name as yet. We query it as:

```
SELECT * FROM OPENQUERY (mysqlDB,'SE-
LECT * FROM MYSQL.ACCOUNT')
```

Now to list all the linked servers we have just created along with their details, use the following stored procedure.

```
EXEC sp_linkedservers
```

To drop a linked remote server we need to drop all the existing logins to it and then drop the server. To drop all logins to a linked server we use the keyword 'droplogins' along with sp_dropserver as:

```
Exec sp_dropserver Linked_ServerName,
droplogins
```

Now that we have created the linked servers, we can query the 3 tables in 3 databases in Query Analyzer as:

```
SELECT * FROM OrclDB..SCOTT.EMP O,
DEPT S,
OPENQUERY (mysqlDB,'SELECT * FROM
MYSQL.ACCOUNT') M
WHERE O.DEPTNO=S.DEPTNO AND
        S.DNAME=M.DNAME
```

For those who use MSDE (MS SQL Server Desktop Engine) there is a small (192 kb) but powerful free GUI utility called DBTray that is very simple and easy to use and can be downloaded from www.sulaco.co.za or http://home.global.co.za/~jhorn.

MySQL users can use any GUI utility given on the February 2004 PCQEssential CD, of which SQLyog is a free utility and is more convenient.

K Praveen Babu