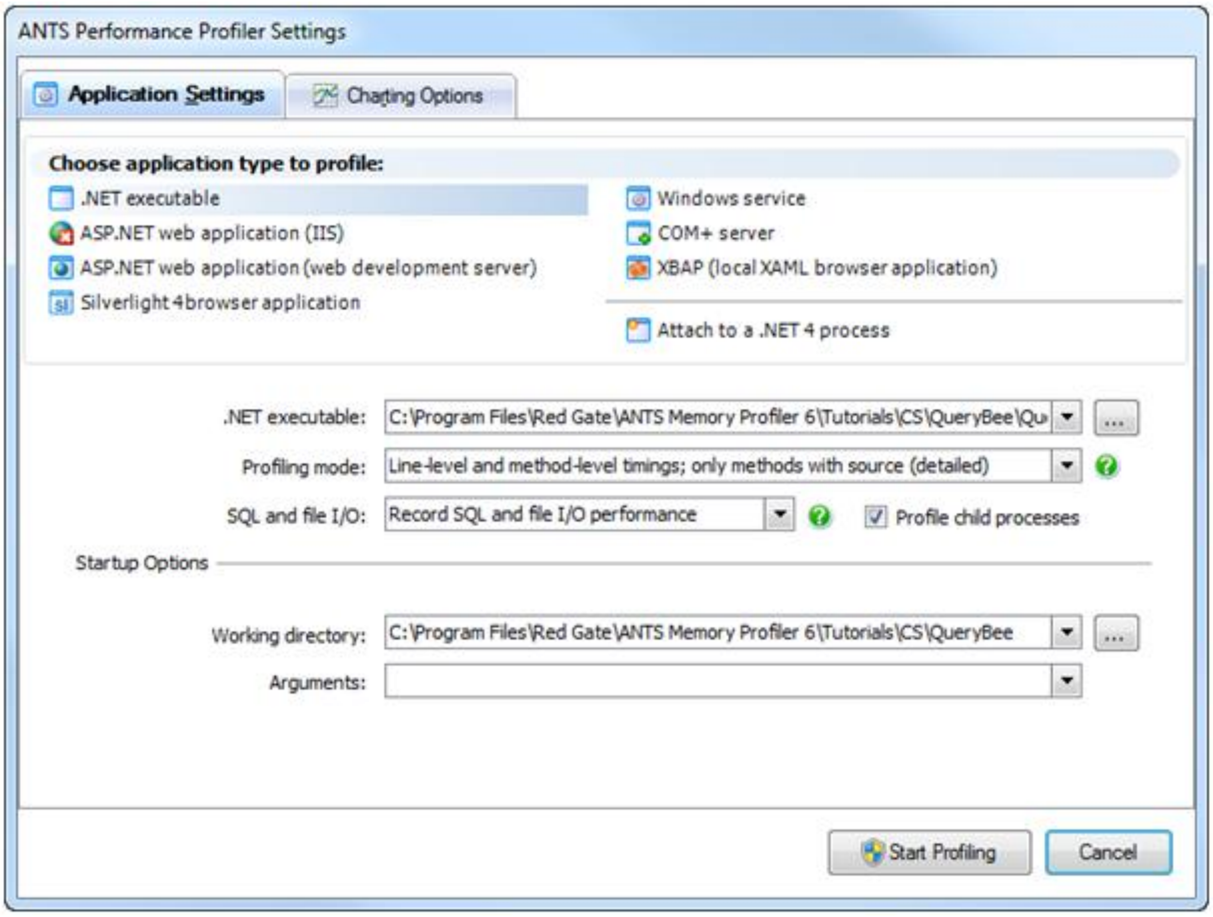
Profiling SQL with ANTS Performance Profiler 6

ANTS Performance Profiler 6 adds SQL Profiling and File I/O to the many performance counters already available in ANTS Performance Profiler 5.

This technical paper demonstrates how you can profile the SQL queries generated by your application. A demonstration database table called 'Customers' has been populated with 100,000 records using Red Gate’s SQL Data Generator (a tool designed to populate database tables with realistic data suitable for use in testing). The table has an ID column called 'ID1' but no primary key or other indices.

SQL queries are entered manually in QueryBee, a simple program for querying SQL databases, and profiled.   Profiling an application which generates the queries itself is possible using exactly the same procedure.

1. Set-up ANTS Performance Profiler to profile QueryBee. This is the application that will send the SQL queries to the SQL server.
2. Set the path to QueryBee in the **ANTS Performance Profiler Settings** dialog, and ensure that **Record SQL and file I/O performance** is enabled.


ANTS Performance Profiler Settings
Application Settings
Charting Options
Choose application type to profile:
.NET executable
ASP.NET web application (IIS)
ASP.NET web application (web development server)
Silverlight 4 browser application
Windows service
COM+ server
XBAP (local XAML browser application)
Attach to a .NET 4 process
Startup Options
Start Profiling
Cancel

.NET executable: C:\Program Files\Red Gate\ANTS Memory Profiler 6\Tutorials\CS\QueryBee\Qu

Profiling mode: Line-level and method-level timings; only methods with source (detailed)

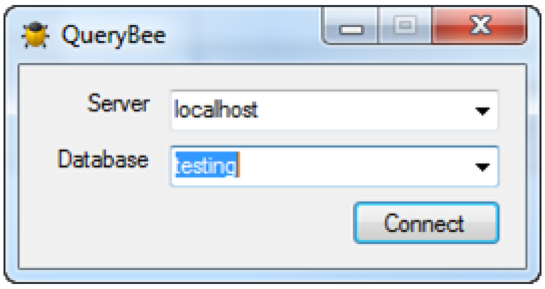
SQL and file I/O: Record SQL and file I/O performance

☑ Profile child processes

Working directory: C:\Program Files\Red Gate\ANTS Memory Profiler 6\Tutorials\CS\QueryBee

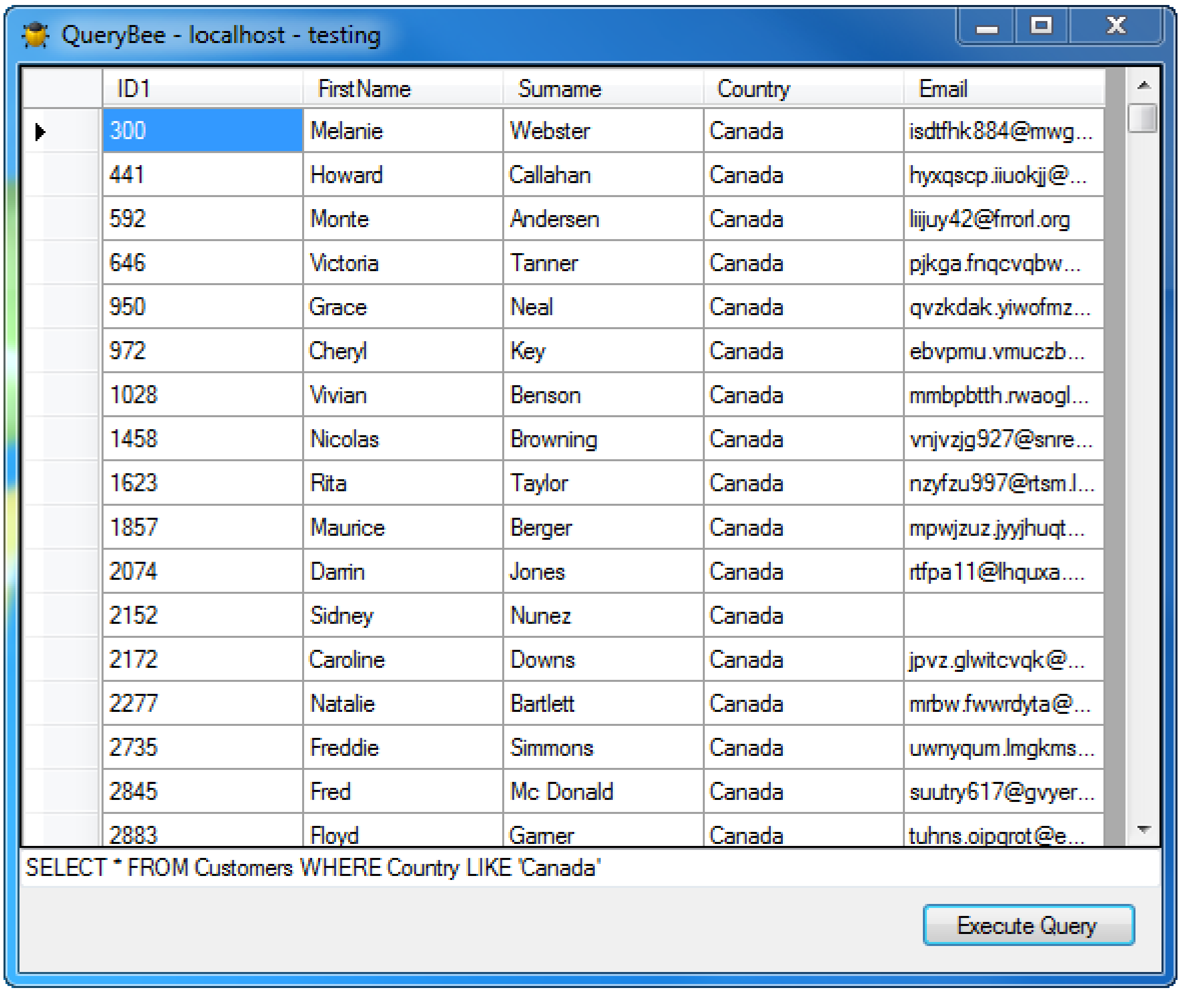
Arguments:

3. Click **Start Profiling** to start QueryBee. Select the test database and click **Connect**.



Server: localhost
Database: testing

4. Run the query `SELECT * FROM Customers` to ensure that all records are read into memory, and that further queries will not include file I/O overheads.
5. Run the query `SELECT * FROM Customers WHERE Country LIKE 'Canada'`

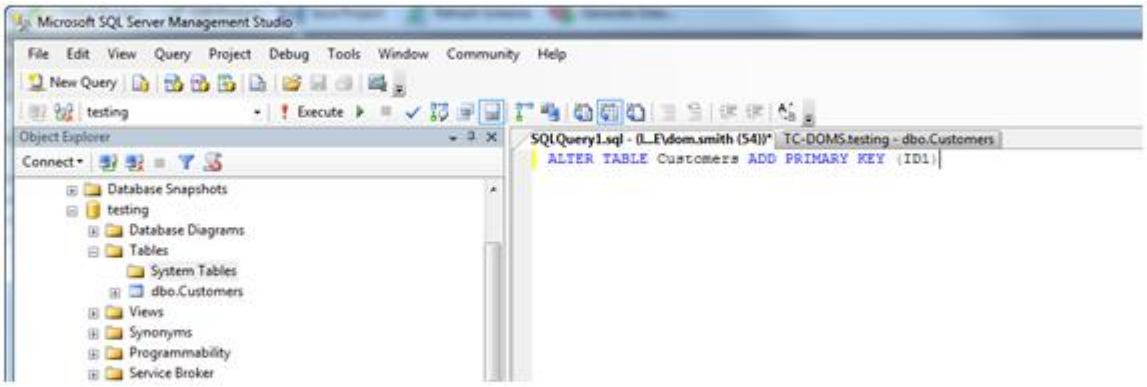


ID1	FirstName	Surname	Country	Email
300	Melanie	Webster	Canada	isdtfhk884@mwg...
441	Howard	Callahan	Canada	hyxqscp.iiuokjj@...
592	Monte	Andersen	Canada	liijuy42@frrorl.org
646	Victoria	Tanner	Canada	pjkga.fnqcvqbw...
950	Grace	Neal	Canada	qvzkdak.yiwofmz...
972	Cheryl	Key	Canada	ebvpmu.vmuczb...
1028	Vivian	Benson	Canada	mmbpbtth.rwaogl...
1458	Nicolas	Browning	Canada	vnjvzjg927@snre...
1623	Rita	Taylor	Canada	nzyfzu997@rtsm.l...
1857	Maurice	Berger	Canada	mpwjzuz.jyyjhuqt...
2074	Darrin	Jones	Canada	rtfpa11@lhquxa....
2152	Sidney	Nunez	Canada	
2172	Caroline	Downs	Canada	jpvz.glwitcvqk@...
2277	Natalie	Bartlett	Canada	mrbw.fwwrdyta@...
2735	Freddie	Simmons	Canada	uwnyqum.lmgkms...
2845	Fred	Mc Donald	Canada	suutry617@gvyer...
2883	Floyd	Garner	Canada	tuhns.oipqrot@e...

```
SELECT * FROM Customers WHERE Country LIKE 'Canada'
```

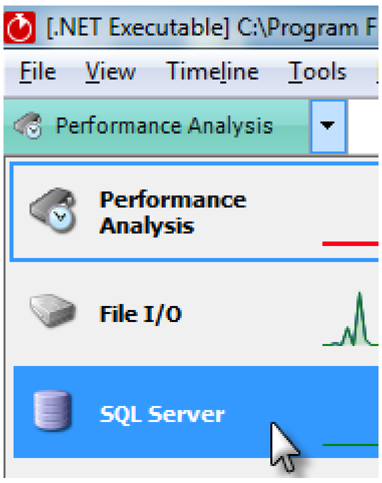
6. The following query is executed, using the equals operator instead:
   `SELECT * FROM Customers WHERE Country='Canada'`
7. To test updated information, run the following query:
   `UPDATE Customers SET Email='a@example.com' WHERE ID1=100`

8. To show that a primary key will make this query quicker, a primary key is applied to the ID1 field in SQL Server Management Studio:

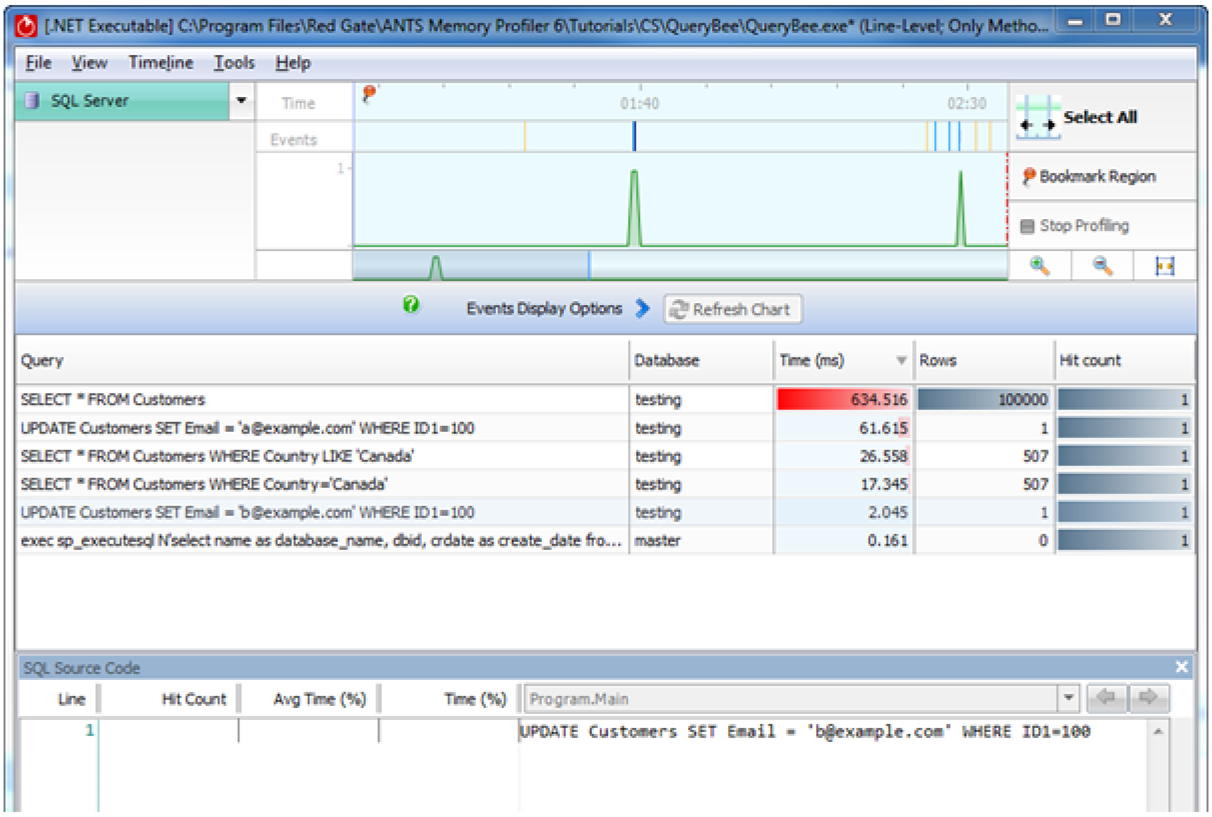


```
ALTER TABLE Customers ADD PRIMARY KEY (ID1)
```

9. Run the following query in QueryBee:
   `UPDATE Customers SET Email='b@example.com' WHERE ID1=100`
10. In the ANTS Performance Profiler window, click **Stop Profiling**
11. On the **Performance Analysis** menu, set ANTS Performance Profiler to **SQL Server** profiling mode:



ANTS Performance Profiler displays the time taken for each query and the number of rows affected or returned. The LIKE operator is slower than the = operator for text comparison, and that the update query was much quicker with a primary key than without a key.


[.NET Executable] C:\Program Files\Red Gate\ANTS Memory Profiler 6\Tutorials\CS\QueryBee\QueryBee.exe* (Line-Level; Only Metho...
File View Timeline Tools Help
SQL Server
Time
Events
01:40
02:30
Select All
Bookmark Region
Stop Profiling
Events Display Options
Refresh Chart

Query	Database	Time (ms)	Rows	Hit count
SELECT * FROM Customers	testing	634.516	100000	1
UPDATE Customers SET Email = 'a@example.com' WHERE ID1=100	testing	61.615	1	1
SELECT * FROM Customers WHERE Country LIKE 'Canada'	testing	26.558	507	1
SELECT * FROM Customers WHERE Country='Canada'	testing	17.345	507	1
UPDATE Customers SET Email = 'b@example.com' WHERE ID1=100	testing	2.045	1	1
exec sp_executesql N'select name as database_name, dbid, crdate as create_date fro...	master	0.161	0	1

```
1    UPDATE Customers SET Email = 'b@example.com' WHERE ID1=100
```

The examples in this technical paper have been kept deliberately simple. A more complex use case for SQL profiling that you may consider is when complex code generates an SQL query at runtime. In such cases, it might be difficult to check exactly what query has been run, but profiling an application with ANTS Performance Profiler allows you to see immediately how your application has interacted with the SQL server.

Note: SQL Profiling is only available to users of Windows Vista or later, who are profiling a locally-hosted SQL server. It is not possible to use SQL profiling with Microsoft SQL Server Express editions.