Jeremiah Flowers

Results:

I was only able to get two database loaded with data and two server scripted connected to the database.

The queries I used were:

1. SELECT FieldNum, Random FROM Field1
2. SELECT FieldChar, Random FROM Field2
3. SELECT FieldNum, FieldChar FROM Fields
4. SELECT FieldNum, FieldChar, Address, Phone, Number FROM Lots

   //WHERE clause
5. SELECT FieldNum, Random FROM Field1 WHERE (FieldNum%10) = 1
6. SELECT FieldChar, Random FROM Field2 WHERE FieldChar LIKE '_______'

   //JOIN clause
7. SELECT Fields.FieldNum, Field1.Random, Fields.FieldChar, Field2.Random FROM Fields
   JOIN Field1 ON Fields.FieldNum = Field1.FieldNum
   JOIN Field2 ON Fields.FieldChar = Field2.FieldChar
8. SELECT Lots.ID, Lots.FieldNum, Field1.Random, Lots.FieldChar, Field2.Random, Lots.Address, Lots.Phone, Lots.Number FROM Lots
   JOIN Field1 ON Lots.FieldNum = Field1.FieldNum
   JOIN Field2 ON Lots.FieldChar = Field2.FieldChar

These will be the times for a server-side script to execute a query to a database and return to the database. The scripts does not do anything with the retrieved data, only releases it upon closing the connection. All tables have 5 million rows that are being retrieved (except Fields table with 100 thousand rows)

PHP:

(in seconds)	1	2	3	4	5	6	7	8
MySQL	6.279	5.973	5.847	5.886	5.860	5.801	5.882	6.445
SQL Server	.384	.061	.066	.003	.003	.002	1.282	30.222
Oracle	---	---	---	---	---	---	---	---

ASP.NET

(in seconds)	1	2	3	4	5	6	7	8
MySQL	16.281	16.031	0.141	3.688	6.094	10.484	(Timed Out)	(Timed Out)
SQL Server	.422	.031	.062	.078	.125	.078	2.016	(Timed Out)
Oracle	---	---	---	---	---	---	---	---

Other – the tools I used to manage the databases. Because these tools are displaying the query, the times will be longer (or in this case to much to handle). This is just extra data to display.

(in seconds)	1	2	3	4	5	6	7	8
MySQL Workbench	-Crashes	-Crashes	4.461	-Crashes	-Crashes	-Crashes	Timed Out	Timed Out
SQL Server -- Management Studio	58	82	1	100 – Completes /w errors	6	67	6	150

Conclusion:

The data set I was able to get were PHP with MySQL, PHP with SQL Server, ASP.NET with MySQL, and ASP.NET with SQL Server. In both cases PHP beat out ASP.NET. SQL Server inside of PHP won out on all queries except for the last, where the query time sky rocketed up. MySQL remains the same throughout all execution, which is very odd. MySQL on ASP.NET did the worst performance out of all other times. It looks like ASP.NET starts to fail when using the JOIN clause queries. Using ASP.NET I could debug the program as it ran and I noticed that closing the connection (or freeing the data) took the longest when executing a query.

Queries one through six were setup to be more of a sprint, since they were straight forward simple queries. The main reason why the queries had to retrieve so many rows is because those queries would complete too quickly to be able to judge them. Queries seven and eight are a lot more stressful because they require joining of tables and because the tables have more columns that need to be retrieved.

For since straight forward simple queries, PHP with SQL server would be the best choice, but to handle tables with multiple columns PHP with MySQL seems to be the better choice.

Oracle was unable create the data set I need before running out of time. Migrating the data from one database to Oracle was not working and running a script to insert all the data was surprisingly very slow.

Website: 52.1.182.175