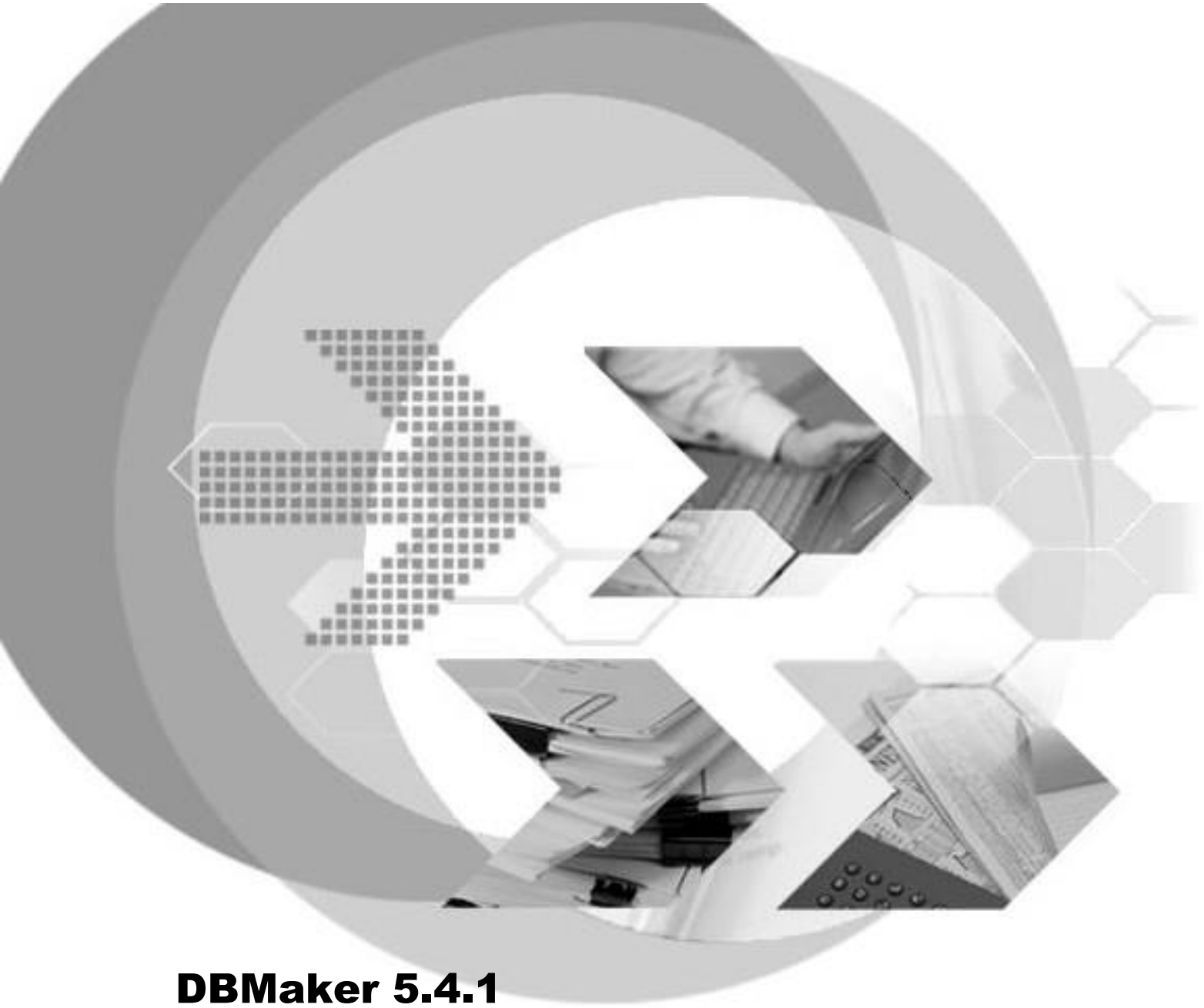


DBMaker 5.4.1

Benchmark Test Report of Ordinary Data

Version: 03.00

Author: DBMaker Test Team

SYSCOM Computer Engineering CO.

Document No: 54/DBM541-B03222013-03-BMTR(OD)

Publication Date: Jan 20, 2017

Tabela de Conteúdo:

1. Plano de Testes

1.1 Descrição dos Testes

O objetivo do teste é comparar o custo de tempo na operação dos dados ordinais no DBMaker 5.4.1, SQL Server 2012, Oracle 11g, e PostgreSQL 9.2. O test case é do tipo ODBC. Existem 18 sub-programas codificados em linguagem C que são Connect1, Insert1, Update1, Delete1, Insert3, Insert2, Update2, Delete2, Insert4, Insert5, Insert6, Select1, Select2, Select3, Select4, Select5, Select6 e Select7. Estes programas são usados para testar o desempenho através da função ODBC para operação de dados ordinais; cada um pode contar o tempo de execução, respectivamente, em si.

1.2 Ambiente de Testes

1.2.1 HARDWARE

Todos os hardwares são os mesmos como abaixo descrito na tabela.

	CPU	MEMORY	HD
Server: 1 PC	**Intel(R) Core(TM) i3-3220 CPU @ 3.30GHz**	**16G DDR3**	**Seagate 500G SATA3**
Client: 1 PC	**Intel(R) Core(TM) i3-3220 CPU @ 3.30GHz**	**16G DDR3**	**Seagate 500G SATA3**

1.2.2 SOFTWARE

Server: 1PC	**Microsoft Windows Server 2008 R2 Enterprise 64bit** **CentOs6 64bit**
Cliente: 1PC	**Microsoft Windows Server 2008 R2 Enterprise 64bit** **ODBC Driver: Microsoft Open Database Connectivity**

Database (Server side)	Configuration	ODBC driver (Client side)
DBMaker 5.4.1(#26361,20161104) (for win64 and Linux64)	Setting journal file size to avoid Journal full and default page size (8KB)	DBMaker ODBC Driver, link odbc32.lib.
SQL Server 2012 (for win64)	Default	MS SQL Server Driver
Oracle 11g (for win64 and linux64)	Default	Oracle 11g ODBC Driver
PostgreSQL 9.2 (for win64 and linux64)	Default	Postgre ANSI Driver

1.2.3 MÉTODO

- Cada Test Case será executado pelo menos 3 vezes.
- ODBC: C + ODBC API, Cliente/Servidor em máquinas diferentes

1.3 Schemas das Tabelas no Teste

Os seguintes Schemas são usados nos testes do DBMaker. Para outros RDBMS, vamos mapear os tipos de dados relacionados com o banco de dados relacionado para cada Schema.

- CREATE TABLE **hundred** (keys integer primary key, ints integer not null, signed integer default null, floats float not null, doubles double not null, decim decimal (17,2) not null, dates timestamp not null, code char (10) not null, name char (20) not null, address varchar (24) not null);
- CREATE INDEX **hundred_code** on hundred(code);
- CREATE INDEX **name_addr** on hundred(name,address);
- CREATE TABLE **updates** (keys integer primary key, ints integer not null, signed integer default null, floats float not null, doubles double not null, decim

decimal (17,2) not null, dates timestamp not null, code char (10) not null, name char (20) not null, address varchar (24) not null);

- CREATE INDEX **updates_code** on updates(code);

- CREATE TABLE **tenpct** (keys integer primary key, ints integer not null, signed integer default null, floats float not null, doubles double not null, decim decimal (17,2) not null, dates timestamp not null, code char (10) not null, name char (20) not null, address varchar (24) not null);

- CREATE INDEX **tenpct_code** on tenpct(code);

NOTA 1:

Em alguns RDBMS, o tipo de dado float são mapeados para tipo de dado real, o tipo de dado Double são mapeados para tipo de dado float, tipo de dado timestamp faz mapeamento para tipo de dado de data e hora.

NOTA 2

Alguns RDBMS suportam index clusterizados, mas só usamos NONCLUSTERD INDEX em todos os RDBMS.

1.4 Dados do Teste

O tamanho da linha é cerca de 110 bytes. Todos as linhas são geradas pelo programa.

Para Insert1 , Update1 , Delete1 , Insert3 , Insert2 , Update2 e Delete2 , ele só usa a tabela de hundred. E a distribuição de dados é como se segue:

```
Create table hundred
(
keys integer primary key, /* all unique; [1..N] by sequence*/
ints integer not null, /* all unique; [1..N] by random*/
signed integer default null, /* 100 unique; insert 100 different integer (100-199) circularly*/
floats float not null, /* 100 unique; insert 100 different float number circularly*/
doubles double not null, /* 100 unique; insert 100 different double number circularly*/
decim decimal (17,2) not null, /* almost unique; generate a decimal randomly */
dates timestamp not null, /* almost unique; generate a date value randomly */
code char (10) not null, /* 100 unique; insert 100 different string values circularly */
name char (20) not null, /* 100 unique; insert 100 different string values circularly */
address varchar (24) not null /* 100 unique; insert 100 different string values circularly */
)
```

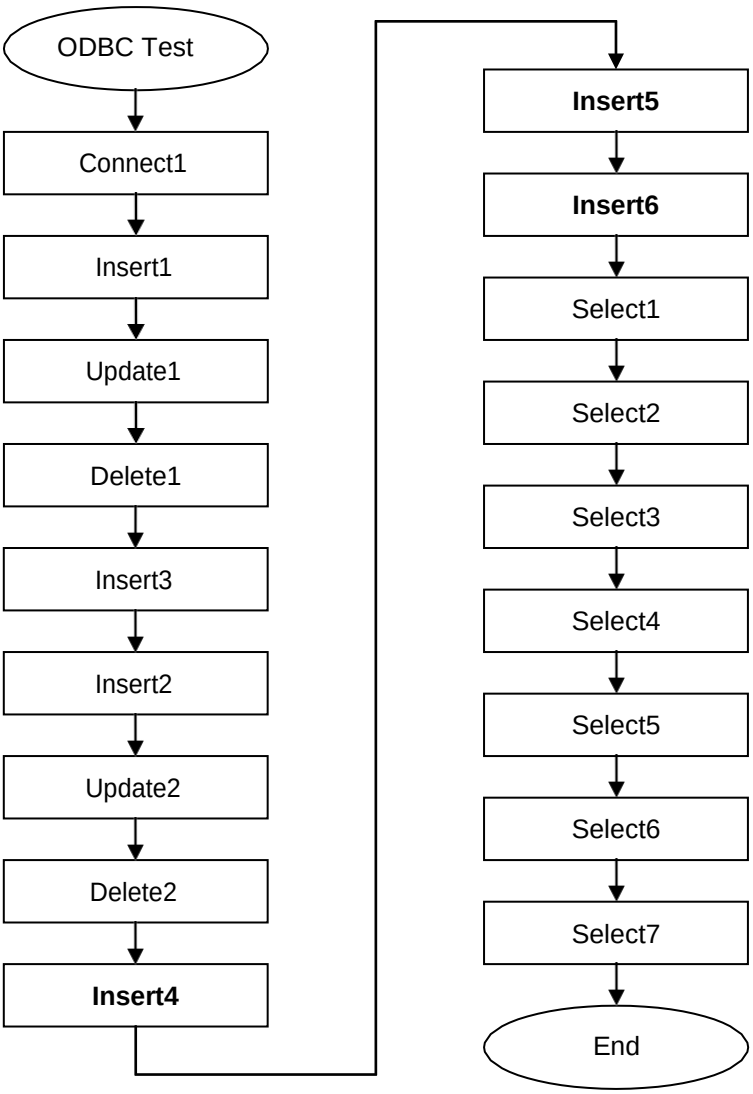
Antes de executar Select1, Select2, Select3, Select4, Select5, Select6 e Select7, os subprogramas Insert4, Insert5 e Insert6 recriará as tabelas **hundred**, **update** e

tenpct e insere 100000 linhas na tabela **hundred** (Insert4), insere 100000 linhas na tabela **update** (Insert5), insere 10000 linhas na tabela **tenpct** (Insert6). Insert4, Insert5 e Insert6 não listam no **Benchmark**, são somente para a preparação do Select1 ao Select7. A distribuição dos dados de teste é como se segue:

```
Create table hundred /* insert 100000 tuples */
(
keys integer primary key, /* all unique; [1..100000] by random*/
ints integer not null, /* all unique; [1..100000] by random*/
signed integer default null, /* 100 unique; insert 100 different integer (100-199) circularly*/
floats float not null, /* 100 unique; insert 100 different float number circularly*/
doubles double not null, /* 100 unique; insert 100 different double number circularly*/
decim decimal (17,2) not null, /* almost unique; generate a decimal randomly */
dates timestamp not null, /* almost unique; generate a date value randomly */
code char (10) not null, /* 100 unique; insert 100 different string values circularly */
name char (20) not null, /* 100 unique; insert 100 different string values circularly */
address varchar (24) not null /* 100 unique; insert 100 different string values circularly */
)
```

Os dados das tabelas **update** e **tenpct** são semelhantes aos dados da tabela **hundred**. Mas para **tenpct**, o domínio da coluna **chave** e **ints** são de 1 a 10000.

1.5 Fluxograma do Teste e Descrição de Cada Caso



Nota: *No capítulo seguinte,* ***N*** *significa* ***10000***, ***100000***.

1.5.1 CONNECT1

Conectar e desconectar do Banco de Dados N/10 vezes.

1.5.2 INSERT1

Inserir N linhas na tabela hundred em sequência de chave primária.

- Drop e recreate da tabela e índice.

- Insert usando variável do host.
- Preparar e vincular parâmetros uma vez, executar N vezes.
- Commit a cada 100 linhas.
- O Comando SQL é:

INSERT INTO HUNDRED (KEYS,INTS,SIGNED,FLOATS,DOUBLES,DECIM,DATES,CODE,NAME,ADDRESS) VALUES (?,?,?,?,?,?,?,?,?,?)

O valor do primeiro parâmetro aumentou ordenadamente a partir de 1 para N, e o valor dos outros parâmetro são aleatorios seguindo respectivamente o domínio.

1.5.3 UPDATE1

Atualize 1 linha para a tabela Hundred N vezes.

- Atualize usando a variável de Host.
- Baseado na chave primária uma linha por comando.
- Preparar e vincular parâmetros uma vez, execute N vezes.
- Commit a cada 100 linhas.
- O commando SQL é:

UPDATE HUNDRED SET INTS=?, SIGNED=?, FLOATS=?, DOUBLES=?, DECIM=?, DATES=?, CODE=?, NAME=?, ADDRESS=? WHERE KEYS=?

O valor do parâmetro é aleatorio conforme seu resectivo dominio.

1.5.4 DELETE1

Deletar uma linha para a tabela Hundred N vezes

- Deletar usando uma variável de Host.

- Baseado na chave primária, uma linha por comando.
- Preparar e vincular parâmetros uma vez, executar N vezes.
- Commit a cada 100 linhas.
- O commando SQl é:

DELETE FROM HUNDRED WHERE KEYS=?

O valor do parâmetro é aleatorio conforme seu resectivo dominio.

1.5.5 INSERT3

Inserir N linhas na tabela Hundred aleatoriamente por chave primária.

- Drop e recreate da tabela e índices.
- Insert usando uma variável de Host
- Preparar e vincular parâmetros uma vez, executar N vezes.
- Commit a cada 100 linhas.
- O comando SQL é:

INSERT INTO HUNDRED (KEYS,INTS,SIGNED,FLOATS,DOUBLES,DECIM,DATES,CODE,NAME,ADDRESS) VALUES (?,?,?,?,?,?,?,?,?,?)

O valor do parâmetro é aleatorio conforme seu resectivo dominio.

1.5.6 INSERT2

Inserir N linhas para a table hundred na sequência da chave primária.

- Drop e recreate da tabela e índices.
- Insert com comando SQL completo.

- Prepare N vezes, execute N vezes.
- Commit a cada 100 linhas.
- O commando SQL é:

INSERT INTO HUNDRED (KEYS,INTS,SIGNED,FLOATS,DOUBLES,DECIM,DATES,CODE,NAME,ADDRESS) VALUES

(c1,c2,c3,c4,c5,c6,c7,c8,c9,c10)

A variável c1 recebe o valor ordenado a partir do seu domínio. Outra variável pode obter o valor aleatoriamente a partir do seu domínio.

1.5.7 UPDATE2

Atualização em MASSA (N/100) linhas por colunas não indexadas N vezes.

- Baseado em colunas não indexadas, N/100 linhas por commando.
- Preparar e vincular parâmetros uma vez, executa 100 vezes.
- Commit a cada 1 update.
- O comando SQL é:

UPDATE HUNDRED SET INTS=?,SIGNED=?,FLOATS=?, DOUBLES=?, DECIM=?,DATES=?,CODE=?,ADDRESS=? WHERE NAME=?

O valor do parâmetro é aleatorio conforme seu resectivo dominio..

1.5.8 DELETE2

Delete em MASSA (N/100) linhas por colunas não indexadas N vezes.

- Baseado em colunas não indexadas, N/100 linha por comando.
- Preparar e vincular um parâmetro, executar 100 vezes.

- Commit a cada um delete.

- O comando SQL é:

DELETE FROM HUNDRED WHERE SIGNED=?

O valor do parâmetro é aleatorio conforme seu respectivo domínio.

1.5.9 INSERT4

Insert 100000 linhas na tabela hundred aleatoriamente.

1.5.10 INSERT5

Insert 100000 linha na tabela updates aleatoriamente.

1.5.11 INSERT6

Insert 10000 linhas na tabela tenpct aleatoriamente.

1.5.12 SELECT1

Select em MASSA de linhas na tabela hundred sem chave com a espressão EQUAL por N/10 vezes.

- Select com clausula sem chave usando o comando SQL complete.

- Espressão EQUAL sem usar chave.

- Select usando table scan (Varredura de tabela).

- Preparar N vezes, executar N vezes.

- Use SQLGetData() para obter dados

- Procure num_linha/100 linhas por vez.
- Commit a cada100 select´s.
- O comando SQL é:

SELECT KEYS, INTS, SIGNED, FLOATS, DOUBLES, DECIM, DATES, CODE, NAME, ADDRESS FROM HUNDRED WHERE SIGNED=$$.

A variável $$ obtém o valor aleatoriamente do domínio na coluna assinada.

1.5.13 SELECT2

Select de linhas em MASSA para a tabela hundred na chave primária pela expressão RANGE por N/10 vezes.

- Select com clausula na chave primária usando o comando SQL completo.
- Expressão RANGE na chave primária.
- Usando a chave primária e o índice para varredura.
- Preparar N vezes, executar N vezes.
- Use SQLGetData() para obter dados.
- Busca de linha em MASSA a todo tempo.
- Commit a cada 100 select.
- O comando SQL é:

SELECT KEYS, INTS, SIGNED, FLOATS, DOUBLES, DECIM, DATES, CODE, NAME, ADDRESS FROM HUNDRED WHERE KEYS<=$$.

A variável $$ recebe o valor aleatoriamente conforme o domínio para a coluna chave.

1.5.14 SELECT3

Select de linhas em MASSA da tabela hundred sem usar a chave primária na expressão RANGE por N/10 vezes.

- Select sem usar a clausula na chave usando o comando SQL complete.
- Não usar chave na expressão RANGE
- Usando o table scan (Varredura de tabela)
- Preparar N vezes, executar N vezes.
- Use SQLGetData() para obter dados.
- Busca de linhas em MASSA a todo tempo.
- Commit a cada 100 select.
- O comando SQL é:

SELECT KEYS, INTS, SIGNED, FLOATS, DOUBLES, DECIM, DATES, CODE, NAME, ADDRESS FROM HUNDRED WHERE INTS<=$$.

A variável $$ recebe o valor aleatoriamente conforme o dominio para a coluna INTS.

1.5.15 SELECT4

Select de linhas em MASSA para as três tabelas juntas sem usar chave por N/100 vezes.

- Join entre três tabelas sem usar chave, expressão EQUAL sem usar chave.
- Select sem chave na clausula usandoo comando SQL complete.
- Preparar N vezes, executar N vezes.
- Use SQLGetData() para obter dados.

- Buscar de linhas em MASSA a todo tempo.
- Commit a cada 100 selects.
- O comando SQL é:

SELECT UPDATES.SIGNED, UPDATES.FLOATS, HUNDRED.SIGNED, HUNDRED.FLOATS,TENPCT.SIGNED, TENPCT.FLOATS FROM UPDATES, HUNDRED, TENPCT WHERE UPDATES.NAME = HUNDRED.NAME AND UPDATES.NAME = TENPCT.NAME AND UPDATES.INTS = $$.

O valor da variável $$ é aleatorio conforme o domínio da coluna INTS.

1.5.16 SELECT5

Select de linhas em MASSA para a tabela hundred com chave primária fazendo sub consulta por N/10 vezes.

- Chave primária na expressão de sub-consulta.
- Use comando SQL completo.
- Prepare N vezes, execute N vezes.
- Use SQLGetData() para obter dados.
- Buscar linhas em MASSA a todo tempo.
- Commit para cada 100 selects.
- O Comando SQL é:

SELECT KEYS, INTS, SIGNED FROM HUNDRED WHERE KEYS IN (SELECT INTS FROM TENPCT)

1.5.17 SELECT6

Select de linhas em MASSA na tabela hundred com índice para a expressão LIKE por N vezes

- Expressão LIKE no índice.
- Select com chave de índice na clausula utilizando o comando SQL completo.
- Prepare N vezes, execute N vezes.
- Use SQLGetData() para obter dados.
- Commit para cada 100 selects.
- O comando SQL é:

SELECT CKEYS, INTS, SIGNED, FLOATS, DOUBLES, DECIM, DATES, CODE, NAME, ADDRESS FROM HUNDRED WHERE CODE LIKE '$$%'.

A variável $$ vem dos 4 primeiros caracteres da coluna CODE conforme domnio.

1.5.18 SELECT7

Select de linhas em MASSA para a tabela hundred com índices compostos para expressão AND e OR por N/10 vezes.

- Select com índice composto na clausula usando o comando SQL complete.
- Expressão AND e OR no índice composto.
- Prepare N vezes, execute N vezes.
- Use SQLGetData() para obter dados.
- Commit a cada 100 selects.
- O comando SQL é:

SELECT KEYS, INTS, SIGNED, FLOATS, DOUBLES, DECIM, DATES, CODE, NAME, ADDRESS FROM HUNDRED WHERE (NAME=$$1 AND ADDRESS=$$2) OR (NAME=$$3 AND ADDRESS=$$4) OR (NAME=$$5 AND ADDRESS=$$6).

Cada uma das variáveis $$1, $$3 e $$5 vem com valores distintos conforme

domínio da coluna NOME.

E as variáveis $$2, $$4 e $$6 vem com valores distintos conforme domínio da coluna ADDRESS.

2. Conclusão dos Testes

A seguir temos o resultado da comparação baseado em N = 100000.

2.1 Windows 2008 R2 64bit + DB

2.1.1 PARA CONNECT1

DBMaker 5.4.1 é mais rápido que os outros RDBMS. O Rank de desempenho é:

DBMaker 5.4.1 > SQL Server 2012 >> PostgreSQL 9.2 >> Oracle 11g.

O gráfico 1 mostra o custo de tempo.

2.1.2 PARA INSERT1, UPDATE1, DELETE1, INSERT3, INSERT2, UPDATE2 E DELETE2

Para Insert1, Update1, Delete1, Insert3, Insert2 and Update2, DBMaker 5.4.1 é muito mais rápido do que outros RDBMS.

Para Delete2, DBMaker 5.4.1 é mais lento que PostgreSQL 9.2, mas é mais rápido do que os outros RDBMS.

O Rank de desempenho é o seguinte:

Para Insert1: **DBMaker 5.4.1** > SQL Server 2012 > Oracle 11g > PostgreSQL 9.2.

Para Update1: **DBMaker 5.4.1** > SQL Server 2012 > Oracle 11g > PostgreSQL 9.2.

Para Delete1: **DBMaker 5.4.1** > SQL Server 2012 > Oracle 11g > PostgreSQL 9.2.

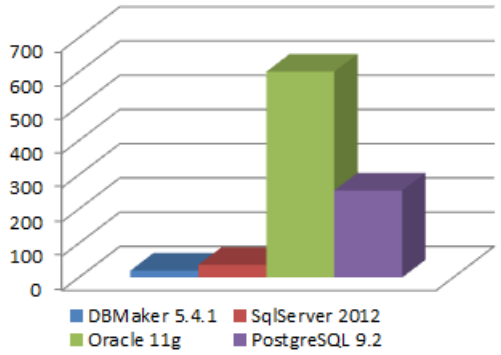
Para Insert3: **DBMaker 5.4.1** > SQL Server 2012 > Oracle 11g > PostgreSQL 9.2.

Para Insert2: **DBMaker 5.4.1** > SQL Server 2012 > PostgreSQL 9.2 >> Oracle 11g.

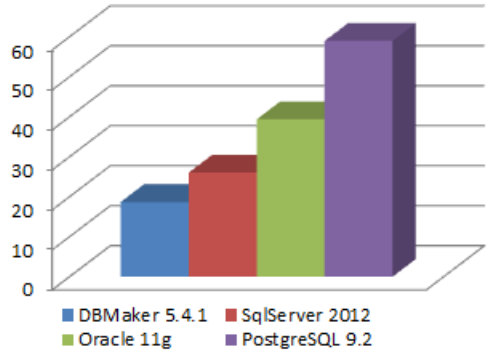
Para Update2: **DBMaker 5.4.1** > PostgreSQL 9.2 > SQL Server 2012 > Oracle 11g.

Para Delete2: PostgreSQL 9.2 > **DBMaker 5.4.1** > SQL Server 2012 > Oracle 11g.

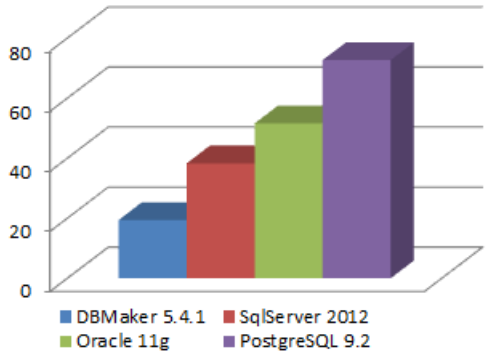
Do gráfico 1 ~ gráfico 8 mostram o custo de tempo do Connect1,Insert1, Update1, Delete1, Insert3, Insert2, Update2 e Delete2 respectivamente. A unidade da barra vertical é "segundos", quanto menor a barra significa melhor desempenho do RDBMS.



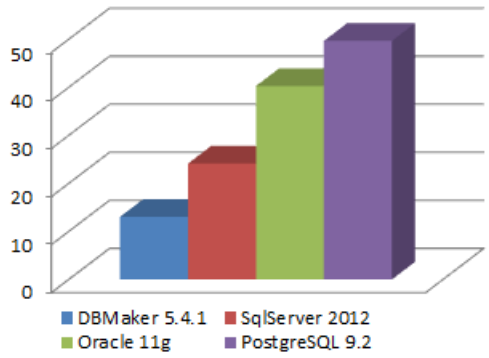
Connect1



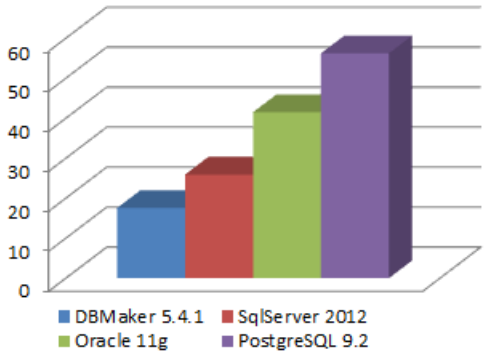
Insert1



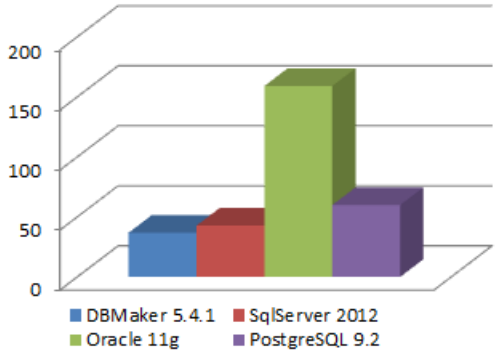
Update1



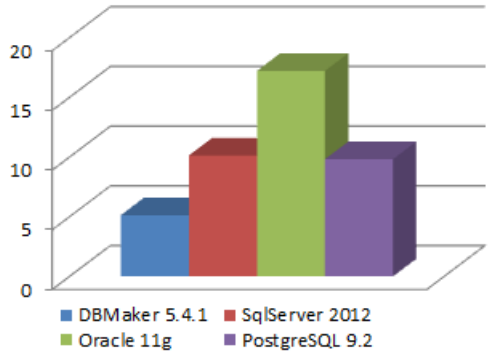
Delete1

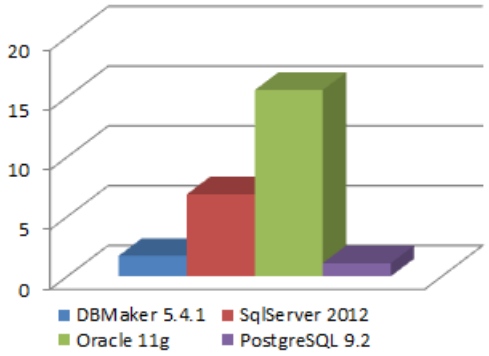


Insert3



Insert2





Update2 **Delete2**

2.1.3 PARA SELECT1, SELECT2, SELECT3, SELECT4, SELECT5, SELECT6 E SELECT7

Para Select2, Select6 e Select7, DBMaker 5.4.1 é mais rápido que outros RDBMS.

Para Select1, Select3, Select4, e Select5, DBMaker 5.4.1 é mais lento do que o SQL Server 2012, mas é mais rápido do que o Oracle 11g e PostgreSQL 9.2.

O Rank de desempenho é:

Para Select1: SQL Server 2012 > **DBMaker 5.4.1** > Oracle 11g > PostgreSQL 9.2.

Para Select2: **DBMaker 5.4.1** > SQL Server 2012 > PostgreSQL 9.2 > Oracle 11g.

Para Select3: SQL Server 2012 > **DBMaker 5.4.1** > Oracle 11g > PostgreSQL 9.2.

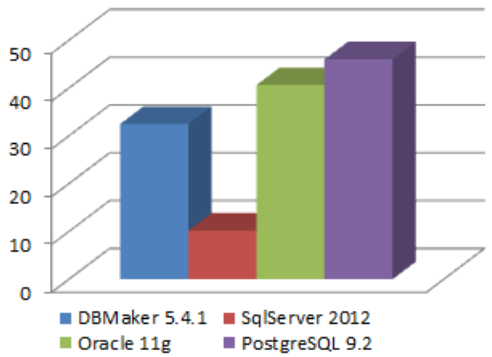
Para Select4: SQL Server 2012 > **DBMaker 5.4.1** > Oracle 11g > PostgreSQL 9.2.

Para Select5: SQL Server 2012 > **DBMaker 5.4.1** > PostgreSQL 9.2 > Oracle 11g.

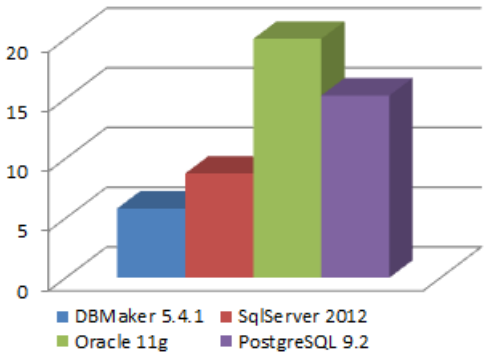
Para Select6: **DBMaker 5.4.1** > SQL Server 2012 > Oracle 11g > PostgreSQL 9.2.

Para Select7: **DBMaker 5.4.1** > SQL Server 2012 > Oracle 11g > PostgreSQL 9.2.

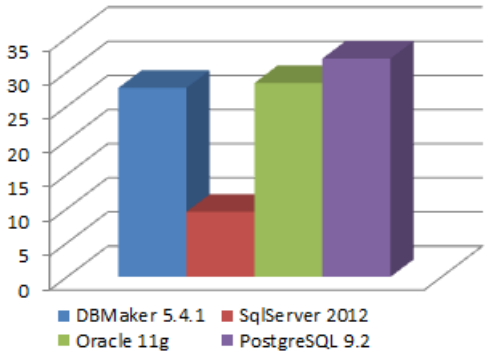
Do gráfico 9 ao gráfico 15 mostra o custo de tempo do Select1 ao Select7 respectivamente. A unidade da barra vertical é "segundos", as menores barras significam melhor desempenho de RDBMS.



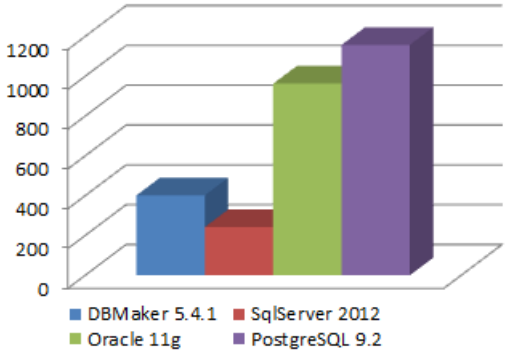
Select1



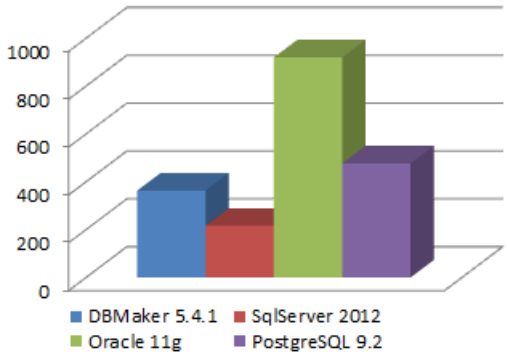
Select2



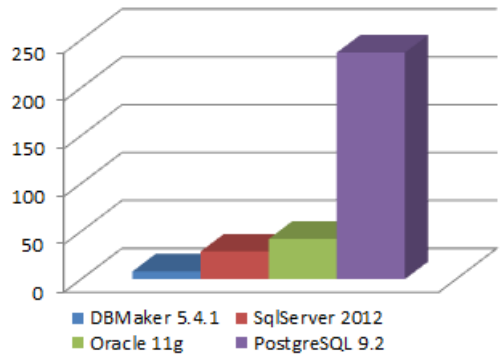
Select3



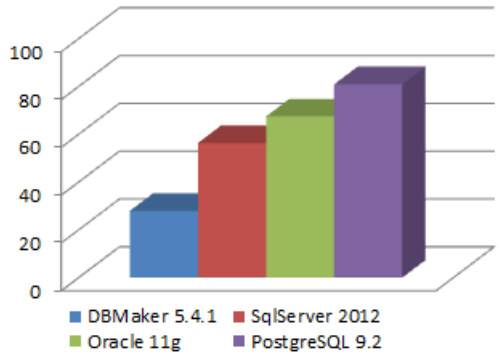
Select4



Select5



Select6



Select7

2.2 CentOS Release 6.3 64bit+ DB

2.2.1 PARA CONNECT1

DBMaker 5.4.1 é muito mais rápido que os outros RDBMS.

O Rank de desempenho é:

DBMaker 5.4.1 > PostgreSQL 9.2 >> Oracle 11g.

O Gráfico 16 mostra o custo do tempo de Connect1.

2.2.2 PARA INSERT1, UPDATE1, DELETE1, INSERT3, INSERT2, UPDATE2 E DELETE2

DBMaker 5.4.1 é muito mais rápido do que outros RDBMS, mas para os testes Insert2,Update2 e **Delete2**, PostgreSQL 9.2 é melhor do que DBMaker 5.4.1.

O Rank de desempenho é:

Para Insert1: **DBMaker 5.4.1** > PostgreSQL 9.2 > Oracle 11g.

Para Update1: **DBMaker 5.4.1** > Oracle 11g > PostgreSQL 9.2.

Para Delete1: **DBMaker 5.4.1** > PostgreSQL 9.2 > Oracle 11g.

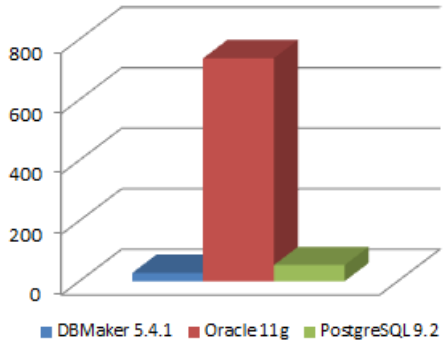
Para Insert3: **DBMaker 5.4.1** > Oracle 11g > PostgreSQL 9.2.

Para Insert2: **DBMaker 5.4.1** > PostgreSQL 9.2> Oracle 11g.

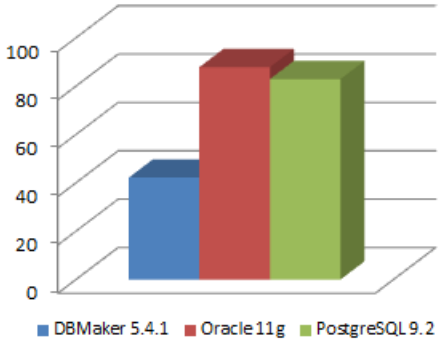
Para Update2: PostgreSQL 9.2 > **DBMaker 5.4.1** > Oracle 11g.

Para Delete2: PostgreSQL 9.2 > **DBMaker 5.4.1** > Oracle 11g.

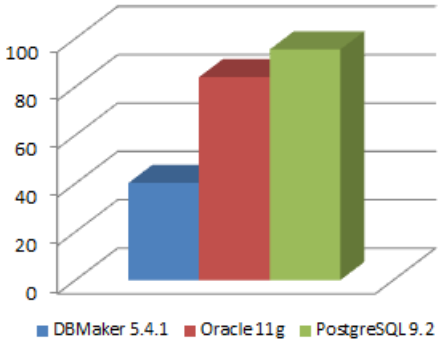
Do gráfico 16 ao gráfico 23 mostra o custo do tempo para o Insert1, Update1, Delete1, Insert3, Insert2, Update2 e Delete2 respectivamente. A unidade da barra vertical é "segundos", a barra menor significa melhor desempenho do RDBMS.



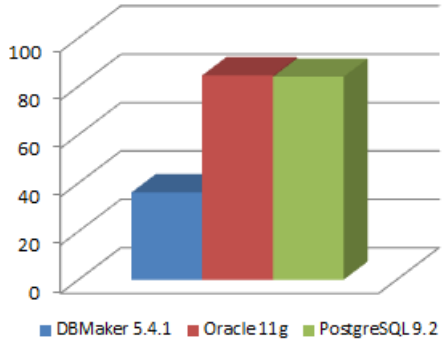
Connect1



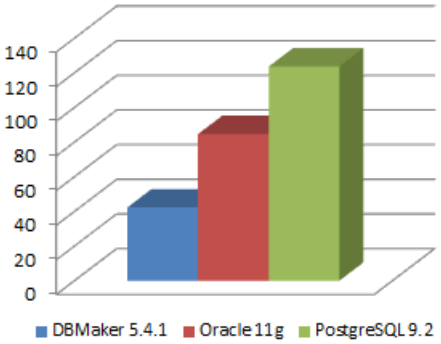
Insert1



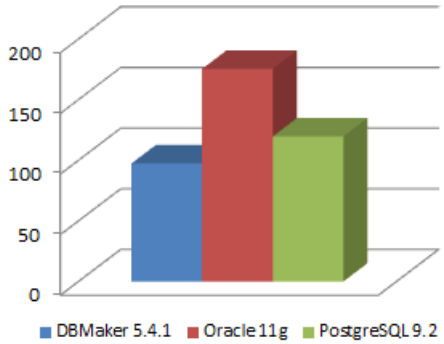
Update1



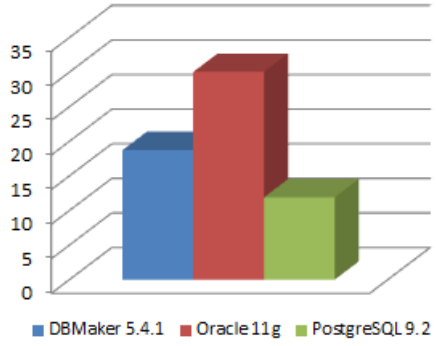
Delete1



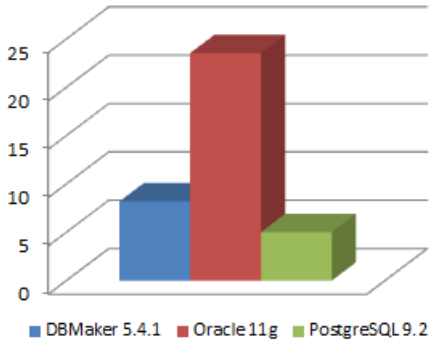
Insert3



Insert2



Update2



Delete2

2.2.3 PARA SELECT1, SELECT2, SELECT3, SELECT4, SELECT5, SELECT6 E SELECT7

DBMaker 5.4.1 é mais rápido do que outros RDBMS, mas para os testes **Select1** e **Select3**, PostgreSQL 9.2 e Oracle 11g são um pouco melhores do que DBMaker 5.4.1.

O Rank de performance é:

Para Select1: **DBMaker 5.4.1**> PostgreSQL 9.2 > Oracle 11g.

ParaSelect2: **DBMaker 5.4.1** > PostgreSQL 9.2 > Oracle 11g.

Para Select3: **DBMaker 5.4.1**> PostgreSQL 9.2 > Oracle 11g.

Para Select4: **DBMaker 5.4.1** > PostgreSQL 9.2 > Oracle 11g.

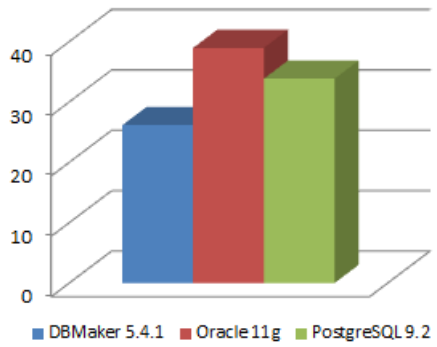
Para Select5: **DBMaker 5.4.1** > PostgreSQL 9.2 > Oracle 11g.

Para Select6: **DBMaker 5.4.1** > Oracle 11g > PostgreSQL 9.2.

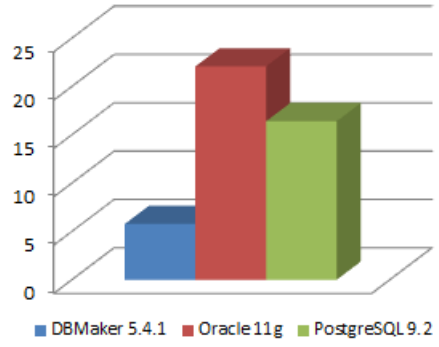
Para Select7: **DBMaker 5.4.1** > PostgreSQL 9.2 > Oracle 11g.

Do gráfico 24 ao gráfico 30 mostra o custo do tempo de Select1 ao Select7 respectivamente. A unidade de barra vertical é " segunda ", a barra menor significa melhor desempenho de RDBMS.

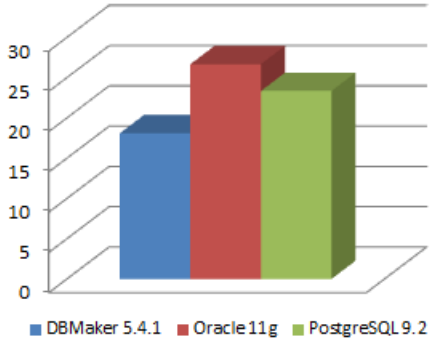
Do gráfico 24 ao gráfico 30 mostra o custo do tempo do Select1 ao Select7 respectivamente. A unidade de barra vertical é "segundos", a barra menor significa melhor desempenho de RDBMS.



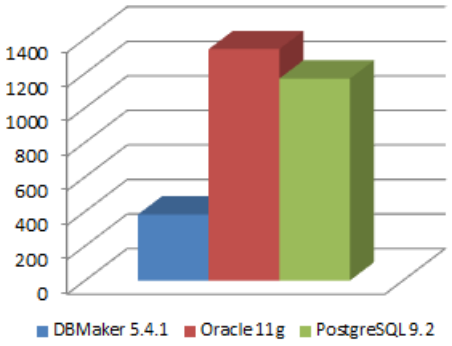
Select1



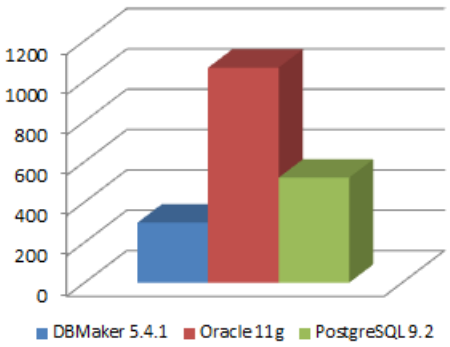
Select2



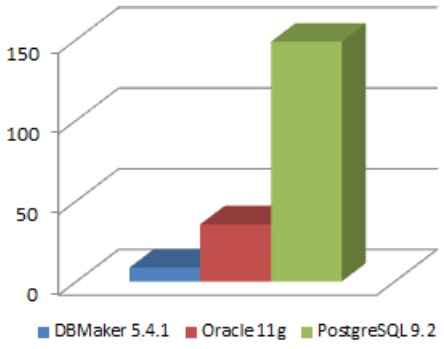
Select3



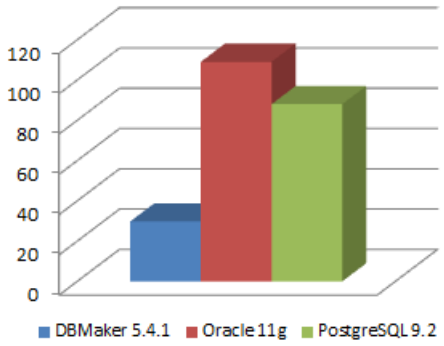
Select4



Select5



Select6



Select7

3. Apêndice

Existe um ponto necessário a ser explicado:

Cada caso de teste será executado apenas uma vez enquanto o teste for N=500 mil, porque isso pode custar muito tempo. Pegue o exemplo do Oracle 11g, o teste Select4 foi necessário executar somente uma vez, pois custou cerca de 14 horas. Se for 18 casos de teste e testarmos três vezes, isso vai custar muito mais tempo. Assim, a tabela de resultados para testes sobre N=500 mil são listados em uma tabela separadamente fazendo referencia aos bancos de dados.

3.1 Resultados de Testes

3.1.1 TABELA 1: WINDOWS 2008 64BIT SERVER + DB PARA INSERT, UPDATE, DELETE

3.1.1.1 Para resultados de 50,000 / 100,000 registros e testados 3 vezes:

		DBMaker 5.4.1		SQL Server 2012		Oracle 11g		PostgreSQL 9.2	
N=		50,000	100,000	50,000	100,000	50,000	100,000	50,000	100,000
Connect1 [N/10 times]	1st	10.000	20.748	18.000	36.000	499.000	602.843	124.000	255.871
	2nd	9.921	16.707	18.454	36.000	197.745	600.000	124.410	253.000
	3rd	10.000	20.389	18.314	36.301	188.807	599.000	123.957	252.533
	AVG	9.974	19.281	18.256	36.100	295.184	600.614	124.122	253.801
Insert1	1st	12.558	24.000	14.913	29.000	25.000	39.000	27.534	60.824
	2nd	8.175	15.000	15.000	24.000	20.967	40.000	27.956	58.188
	3rd	8.596	17.000	14.000	25.000	18.767	39.000	27.000	57.845
	AVG	9.776	18.667	14.638	26.000	21.578	39.333	27.497	58.952
Update1	1st	10.000	21.000	18.000	51.963	30.000	51.730	33.000	69.982
	2nd	9.000	18.549	15.694	33.000	25.474	52.000	33.000	72.000
	3rd	9.937	18.455	15.320	29.781	26.895	50.965	34.000	76.000
	AVG	9.646	19.335	16.338	38.248	27.456	51.565	33.333	72.661
Delete1	1st	6.411	13.000	15.226	25.000	23.000	40.887	24.914	49.624
	2nd	6.567	13.000	11.606	23.728	18.000	41.870	25.000	49.000
	3rd	6.000	13.000	11.000	23.525	18.627	38.000	23.978	50.341
	AVG	6.326	13.000	12.611	24.084	19.876	40.252	24.631	49.655
Insert3	1st	8.000	18.000	15.000	25.000	27.000	42.000	27.659	56.000
	2nd	7.987	17.000	14.000	26.000	20.764	41.000	28.345	56.643
	3rd	8.000	17.566	13.478	26.645	21.715	41.418	28.000	56.000
	AVG	7.996	17.522	14.159	25.882	23.160	41.473	28.001	56.214
Insert2	1st	17.909	37.000	20.920	39.656	73.000	147.000	29.297	59.000
	2nd	17.799	36.722	22.760	44.000	82.633	165.000	29.000	61.355
	3rd	16.521	35.958	21.000	44.000	80.933	164.986	29.000	59.124
	AVG	17.410	36.560	21.56	42.552	78.855	158.995	29.099	59.826
Update2 [100 times]	1st	2.512	5.367	6.755	10.420	11.434	19.624	4.000	9.438
	2nd	2.000	5.000	7.000	10.109	10.935	16.000	3.000	10.000
	3rd	2.000	5.000	5.772	9.828	9.766	16.000	3.494	10.000
	AVG	2.171	5.122	6.509	10.119	10.712	17.208	3.498	9.813
Delete2 [100 times]	1st	0.921	1.000	3.000	8.000	9.796	17.534	0.764	1.000
	2nd	0.889	1.591	5.819	6.000	8.939	15.865	0.764	1.279
	3rd	0.967	1.701	3.557	6.521	6.302	13.322	0.733	1.000
	AVG	0.926	1.701	4.125	6.840	8.346	15.574	0.754	1.093

3.1.1.2 Para resultados de 500,000 registros e teste único:

	DBMaker 5.4.1	SQL Server 2012	Oracle 11g	PostgreSQL 9.2
N=	500, 000	500, 000	500, 000	500, 000
Connect1 [N/10 times]	102. 000	197. 808	5110. 000	1249. 000
Insert1	109. 653	203. 000	304. 746	384. 759
Update1	98. 000	290. 612	518. 000	1153. 000
Delete1	62. 931	142. 000	545. 689	696. 354
Insert3	91. 837	170. 742	293. 000	343. 000
Insert2	199. 633	254. 624	878. 000	435. 000
Update2 [100 times]	28. 000	34. 882	280. 000	596. 966
Delete2 [100 times]	6. 411	28. 000	156. 000	296. 000

3.1.2 TABELA 2: WINDOWS 2008 64BIT SERVER + DB PARASELECT

3.1.2.1 Para resutados de 50,000 / 100,000 registros e testados 3 vezes:

		DBMaker 5.4.1		SQL Server 2012		Oracle 11g		PostgreSQL9.2	
N= / tuples=		5000 / 50000	10000 / 100000	5000 / 50000	10000 / 100000	5000 / 50000	10000 / 100000	5000 / 50000	10000 / 100000
Select1 [N/10 times]	1st	8.658	32.791	6.973	9.469	19.282	40.000	23.556	45.942
	2nd	8.533	32.000	6.927	10.000	19.780	40.794	24.000	46.000
	3rd	7.940	32.589	7.000	10.967	19.843	41.000	24.000	45.989
	AVG	8.377	32.460	6.967	10.145	19.635	40.598	23.852	45.977
Select2 [N/10 times]	1st	1.000	5.881	8.000	8.000	19.937	20.000	16.000	15.000
	2nd	1.000	5.788	8.000	9.157	19.656	20.000	15.803	15.381
	3rd	1.731	5.694	8.000	9.000	20.000	20.000	15.959	15.397
	AVG	1.244	5.788	8.000	8.719	19.864	20.000	15.921	15.259
Select3 [N/10 times]	1st	7.473	27.000	8.000	8.829	21.684	28.189	23.883	32.000
	2nd	7.425	28.000	8.658	9.844	22.000	28.000	23.930	32.000
	3rd	7.000	28.000	8.627	9.859	22.000	29.000	23.000	32.000
	AVG	7.299	27.667	8.428	9.511	21.895	28.396	23.604	32.000
Select4 [N/100 times]	1st	52.650	402.000	66.846	224.000	260.738	964.000	294.840	1176.304
	2nd	52.000	401.000	67.922	251.000	261.722	932.366	291.000	1140.393
	3rd	52.821	401.000	67.000	251.000	253.000	991.000	291.000	1155.977
	AVG	52.490	401.333	67.256	242	258.487	962.455	292.28	1157.558
Select5 [N/10 times]	1st	91.000	370.828	115.000	203.908	448.000	922.000	264.000	479.000
	2nd	91.432	359.347	114.000	219.000	446.473	921.000	263.500	475.723
	3rd	91.000	359.799	116.657	225.000	452.000	917.000	263.453	475.000
	AVG	91.144	363.325	115.219	215.969	448.824	920. 000	263.651	476.574
Select6 [N times]	1st	4.000	7.426	26.224	27.393	37.861	41.824	89.856	214.000
	2nd	3.978	8.000	26.000	29.796	38.000	40.935	93.000	266.000
	3rd	3.994	8.564	27.000	29.281	38.000	43.000	89.000	230.000
	AVG	3.991	7.997	26.408	28.823	37.954	41.920	90.619	236.667
Select7 [N/10 times]	1st	8.000	20.000	30.000	53.000	28.000	60.372	42.000	80.886
	2nd	8.000	31.855	30.000	57.000	29.312	74.000	42.416	81.000
	3rd	8.548	31.949	30.420	59.000	29.530	67.938	42.000	80.792
	AVG	8.183	27.935	30.14	56.333	28.947	67.437	42.139	80.893

3.1.2.2 Para resultados de 500,000 registros e teste único:

	DBMaker 5.4.1	SQL Server 2012	Oracle 11g	PostgreSQL 9.2
N= / tuples=	10, 000 / 500, 000	10, 000 / 500, 000	10, 000 / 500, 000	10, 000 / 500, 000
Select1 [N/10 times]	149. 000	49. 000	188. 000	250. 000
Select2 [N/10 times]	3. 000	7. 660	23. 447	17. 706
Select3 [N/10 times]	69. 357	24. 000	54. 662	122. 881
Select4 [N/100 times]	9720. 000	5957. 853	34262. 932	30273. 000
Select5 [N/10 times]	1843. 877	779. 000	5294. 540	3174. 000
Select6 [N times]	11. 000	35. 459	34. 000	1005. 905
Select7 [N/10 times]	117. 702	158. 000	544. 862	446. 613

3.1.3 TABELA 3: CENTOS RELEASE 3 64BIT + DB PARA INSERT, UPDATE E DELETE

3.1.3.1 Para resultados de 50,000 / 100,000 registros e testados 3 vezes:

		DBMaker 5.4.1		Oracle 11g		PostgreSQL 9.2	
N=		50, 000	100, 000	50, 000	100, 000	50, 000	100, 000
Connect1 [N/10 times]	1st	13. 000	26. 145	367. 000	735. 000	26. 000	52. 603
	2nd	13. 000	31. 824	382. 590	742. 732	28. 000	54. 000
	3rd	13. 000	26. 000	384. 259	733. 934	32. 884	54. 000
	AVG	13. 000	27. 990	377. 950	737. 222	28. 961	53. 534
Insert1	1st	22. 605	46. 000	44. 000	89. 000	41. 464	82. 773
	2nd	19. 780	40. 341	43. 290	87. 750	61. 000	83. 000
	3rd	19. 875	40. 233	43. 399	86. 845	41. 059	83. 000
	AVG	20. 753	42. 191	43. 563	87. 865	47. 841	82. 924
Update1	1st	20. 857	41. 000	47. 315	87. 000	45. 000	96. 393
	2nd	20. 000	39. 734	44. 000	81. 915	71. 000	94. 832
	3rd	19. 921	39. 874	44. 616	83. 000	47. 000	94. 942
	AVG	20. 259	40. 202	45. 438	83. 972	54. 333	95. 389
Delete1	1st	17. 737	37. 409	44. 000	87. 579	40. 342	79. 000
	2nd	15. 272	35. 000	41. 000	83. 000	63. 000	92. 820
	3rd	15. 662	36. 000	40. 000	83. 000	59. 935	80. 000
	AVG	16. 224	36. 136	41. 667	84. 526	54. 426	83. 94
Insert3	1st	20. 873	41. 000	42. 635	87. 000	41. 000	119. 000
	2nd	21. 000	43. 000	41. 823	82. 992	68. 000	122. 850
	3rd	19. 000	43. 000	41. 621	84. 000	58. 734	129. 000
	AVG	20. 291	42. 333	42. 026	84. 664	55. 911	123. 617
Insert2	1st	28. 719	106. 000	88. 000	178. 000	47. 517	117. 000
	2nd	28. 579	93. 000	86. 000	175. 000	58. 000	126. 454
	3rd	32. 760	93. 000	86. 799	172. 911	59. 872	115. 533
	AVG	30. 019	97. 333	86. 933	175. 30	55. 130	119. 662
Update2 [100 times]	1st	7. 000	18. 767	16. 000	34. 000	8. 565	12. 000
	2nd	8. 000	18. 642	12. 000	29. 079	6. 895	12. 760
	3rd	8. 000	18. 751	13. 000	27. 000	8. 000	11. 000
	AVG	7. 667	18. 720	13. 667	30. 026	7. 632	11. 92
Delete2 [100 times]	1st	6. 271	8. 000	14. 492	28. 626	2. 075	4. 000
	2nd	6. 000	8. 580	11. 000	19. 000	2. 527	4. 992
	3rd	6. 614	8. 000	12. 714	23. 000	3. 931	6. 069
	AVG	6. 295	8. 193	12. 735	23. 542	2. 844	5. 020

3.1.3.2 Para resultados de 500,000 registros e teste único:

	DBMaker 5.4.1	Oracle 11g	PostgreSQL 9.2
N=	500, 000	500, 000	500, 000
Connect1 [N/10 times]	208. 000	138. 000	262. 813
Insert1	260. 817	430. 904	502. 000
Update1	306. 993	548. 871	4758. 000
Delete1	339. 000	480. 000	2148. 951
Insert3	527. 999	454. 662	536. 000
Insert2	516. 563	886. 409	549. 729
Update2 [100 times]	63. 000	524. 000	779. 000
Delete2 [100 times]	28. 423	288. 000	874. 000

3.1.4 TABELA 4: CENTOS RELEASE 3 64BIT+ DB PARA SELECT

3.1.4.1 Para resultados de 50,000 / 100,000 registros e testados 3 vezes:

		DBMaker 5.4.1		Oracle 11g		PostgreSQL 9.2	
N= tuples		5000 / 50, 000	10, 000 / 100, 000	5000 / 50, 000	10, 000 / 100, 000	5000 / 50, 000	10, 000 / 100, 000
Select 1 [N/10 times]	1st	7. 000	26. 000	20. 000	38. 000	19. 983	33. 727
	2nd	7. 000	26. 000	20. 000	39. 000	20. 000	33. 790
	3rd	7. 000	26. 442	21. 169	39. 811	20. 000	34. 000
	AVG	7. 000	26. 147	20. 390	38. 937	19. 994	33. 839
Select 2 [N/10 times]	1st	1. 684	5. 757	22. 000	22. 000	16. 676	16. 614
	2nd	1. 701	5. 787	22. 000	22. 215	16. 458	16. 000
	3rd	1. 716	5. 772	22. 000	22. 000	15. 990	16. 583
	AVG	1. 700	5. 772	22. 000	22. 072	16. 375	16. 399
Select 3 [N/10 times]	1st	5. 000	18. 000	23. 571	26. 000	20. 000	23. 000
	2nd	5. 000	18. 252	23. 000	27. 000	20. 000	23. 416
	3rd	5. 000	18. 000	24. 149	27. 000	20. 000	23. 634
	AVG	5. 000	18. 084	23. 573	26. 667	20. 000	23. 35
Select 4 [N/100 times]	1st	50. 000	393. 000	336. 000	1363. 000	309. 000	1169. 000
	2nd	50. 169	373. 000	348. 000	1332. 570	307. 742	1168. 000
	3rd	50. 200	393. 000	338. 786	1324. 000	299. 474	1171. 000
	AVG	50. 123	383. 333	340. 929	1339. 857	305. 405	1169. 333
Select 5 [N/10 times]	1st	75. 000	294. 310	536. 000	1052. 814	272. 000	521. 072
	2nd	75. 000	306. 494	547. 389	1055. 000	271. 000	521. 852
	3rd	75. 000	292. 703	539. 000	1090. 551	269. 000	519. 824
	AVG	75. 000	297. 836	540. 796	1066. 122	270. 667	520. 916
Select 6 [N times]	1st	4. 000	8. 000	32. 000	36. 317	93. 865	150. 000
	2nd	4. 524	8. 985	33. 000	35. 100	94. 895	147. 000
	3rd	4. 000	9. 000	35. 366	35. 000	95. 504	149. 000
	AVG	4. 175	8. 662	33. 455	35. 472	94. 755	148. 667
Select 7 [N/10 times]	1st	8. 000	29. 843	54. 818	108. 888	45. 000	88. 000
	2nd	7. 863	29. 874	55. 000	108. 873	45. 000	88. 000
	3rd	7. 909	29. 890	55. 000	109. 387	44. 944	88. 000
	AVG	7. 924	29. 749	54. 939	109. 049	44. 981	88. 000

3.1.4.2 Para resultados de 500,000 registros e teste único:

	DBMaker 5.4.1	Oracle 11g	PostgreSQL 9.2
N= / tuples=	10, 000 / 500, 000	10, 000 / 500, 000	10, 000 / 500, 000
Select1 [N/10 times]	111. 000	183. 000	155. 000
Select2 [N/10 times]	5. 756	22. 000	16. 879
Select3 [N/10 times]	52. 618	43. 602	61. 635
Select4 [N/100 times]	9596. 918	27022. 976	29334. 000
Select5 [N/10 times]	1582. 000	5132. 000	2541. 510
Select6 [N times]	10. 733	50. 000	706. 000
Select7 [N/10 times]	145. 000	540. 322	415. 429