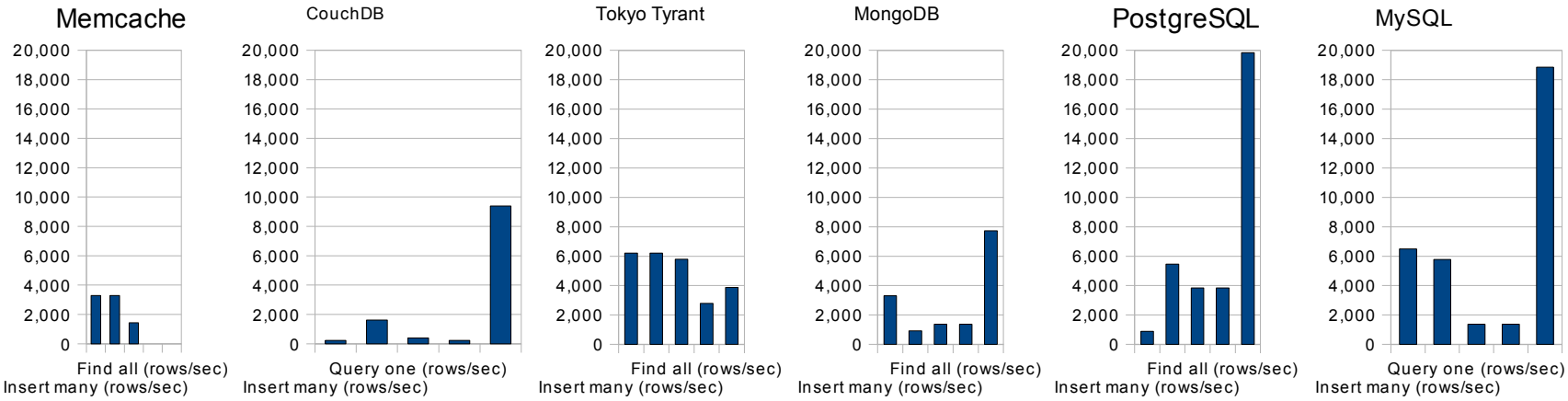
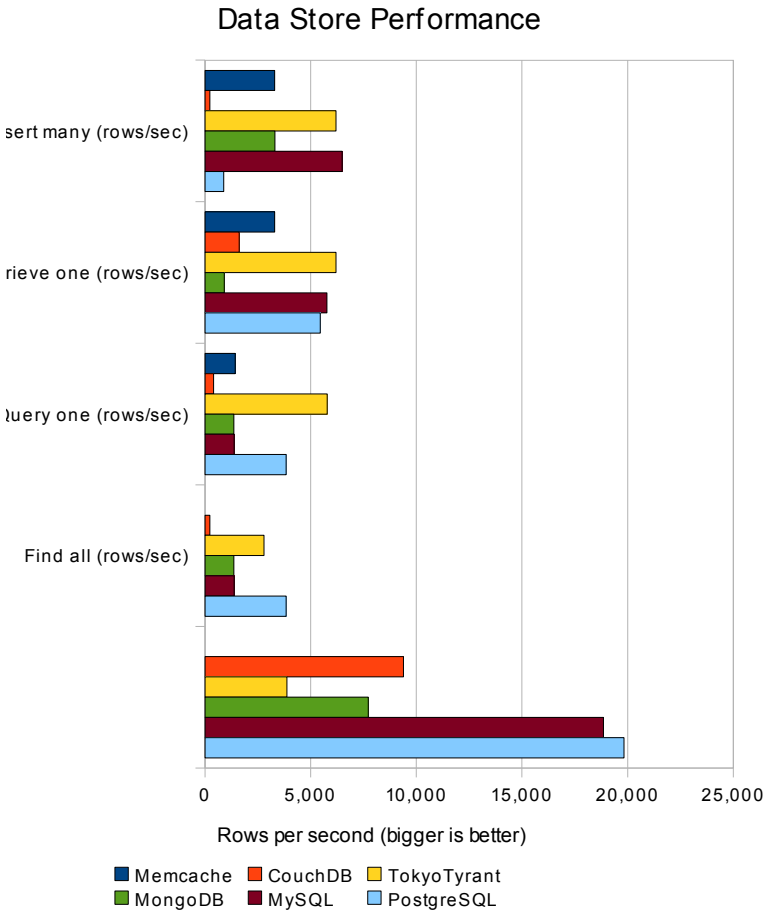
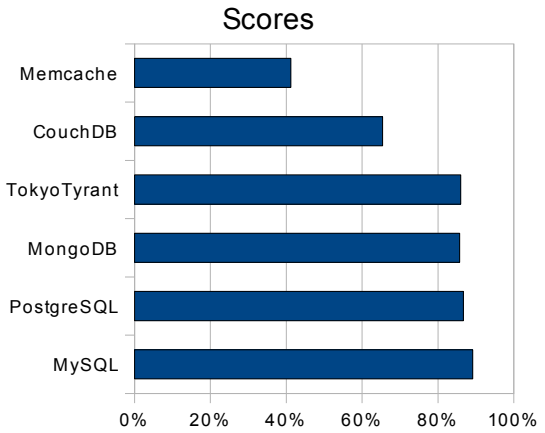
Ruby Data Stores, updated 2009-08-11

	Memcache	CouchDB	TokyoTyrant	MongoDB	PostgreSQL	MySQL
Persistent	N	Y	Y	Y	Y	Y
Schema replication	Y	Y	Y	Y	N	Y [4]
Easy to install	Y	Y	Y	Y	Y	Y
Easy to use	Y	N	Y	Y	Y	Y
Well-documented	Y	N	Y	Y	Y	Y
Console	N	Y	Y	Y	Y	Y
Fetch by id	Y	Y	Y	Y	Y	Y
Fetch by query	N	Y	Y	Y	Y	Y
Fetch by substring	N	Y	Y	Y	Y	Y
Fetch by subset	N	Y [1]	Y [2]	Y	Y	Y
Fetch count	N	Y	Y	Y	Y	Y
Fetch min/max	N	Y [1]	Y [2]	Y	Y	Y
Data types	N	N	N	Y	Y	Y
Increment/decrement	Y	Y [1]	Y [2]	Y	Y	Y
Push/pop value	N	Y [1]	Y [2]	Y	Y	N
Index a column	N	Y	Y	Y	Y	Y
Virtual filesystem	N	N	N	Y	N	N
Sensible import/export	N	Y	Y	Y	Y	Y
Multi-master replication	N	Y	Y	Y	Y [3]	Y [3]
Master-slave replication	N	Y	Y	Y	Y [3]	Y [3]
Transactions	N	Y	Y	N	Y	Y
Extensible	N	Y	Y	Y	Y	Y
Proven	Y	N	N	N	Y	Y
Well-understood & common	Y	N	N	N	Y	Y
Insert one (rows/sec)	3,293	235	**6,204**	3,316	891	**6,488**
Insert many (rows/sec)	*3,293*	*1,620*	***6,204***	917	5,457	5,774
Retrieve one (rows/sec)	1,438	404	**5,787**	1,375	3,848	1,378
Query one (rows/sec)		*237*	**2,793**	1,375	**3,848**	1,378
Find all (rows/sec)		*9,394*	3,882	7,725	**19,830**	18,854
Score (bigger is better)	41%	65%	86%	86%	87%	**89%**
Pros:	N/A	Flexible	Quick, multi stores	Easy, complete	Safe, simple	Safe, simple
Cons:	Not persistent	Very slow, tricky	Fewer features, tricky	Slower than DB	Schema replication	Schema replication
Conclusion:	Not an option	Probably not	For speed	For general purpose	Grampa is still spry	Quirky kid grew up

Notes:
WARNING: These are naive benchmarks that do serial operations in tight loops with small datasets from single host to localhost.
[1] CouchDB uses JavaScript programs to provide this functionality.
[2] Tokyo Tyrant uses Lua programs to provide this functionality.
[3] Replication is tricky, requiring third-party packages and custom infrastructure
[4] MySQL DDL replication is slow, risky, and may require restarting cluster


Memcache
CouchDB
Tokyo Tyrant
MongoDB
PostgreSQL
MySQL
20,000
18,000
16,000
14,000
12,000
10,000
8,000
6,000
4,000
2,000
0
Find all (rows/sec)
Query one (rows/sec)
Insert many (rows/sec)


Data Store Performance
sert many (rows/sec)
rieve one (rows/sec)
uery one (rows/sec)
Find all (rows/sec)
0
5,000
10,000
15,000
20,000
25,000
Rows per second (bigger is better)
Memcache
CouchDB
TokyoTyrant
MongoDB
MySQL
PostgreSQL


Scores
Memcache
CouchDB
TokyoTyrant
MongoDB
PostgreSQL
MySQL
0%
20%
40%
60%
80%
100%

Scoring system:

Value for X:		Weights	Memcache	CouchDB	Tokyo Tyrant	MongoDB	PostgreSQL	MySQL
Y	Persistent	100	0	100	100	100	100	100
1	Schema replic	30	30	30	30	30	0	15
N	Easy to install	10	10	10	10	10	10	10
0	Easy to use	15	15	0	15	15	15	15
M	Well-documen	10	10	0	10	10	10	10
0.5	Console	2	0	2	2	2	2	2
	Fetch by id	10	10	10	10	10	10	10
	Fetch by quer	10	0	10	10	10	10	10
	Fetch by subs	2	0	2	2	2	2	2
	Fetch by subs	2	0	1	1	2	2	2
	Fetch count	2	0	2	2	2	2	2
	Fetch min/max	2	0	1	1	2	2	2
	Data types	5	0	0	0	5	5	5
	Increment/dec	5	5	2.5	2.5	5	5	5
	Push/pop valu	1	0	0.5	0.5	1	1	0
	Index a colum	10	0	10	10	10	10	10
	Virtual filesyst	1	0	0	0	1	0	0
	Sensible impo	10	0	10	10	10	10	10
	Multi-master re	5	0	5	5	5	2.5	2.5
	Master-slave r	30	0	30	30	30	15	15
	Transactions	2	0	2	2	0	2	2
	Extensible	10	0	10	10	10	10	10
	Proven	30	30	0	0	0	30	30
	Well-understo	10	10	0	0	0	10	10
	Insert one (rov	50	30	0	50	40	50	45
	Insert many (rows/sec)							
	Retrieve one (rows/sec)							
	Query one (rows/sec)							
	Find all (rows/sec)							
	Score (bigger	364	41%	65%	86%	86%	87%	89%