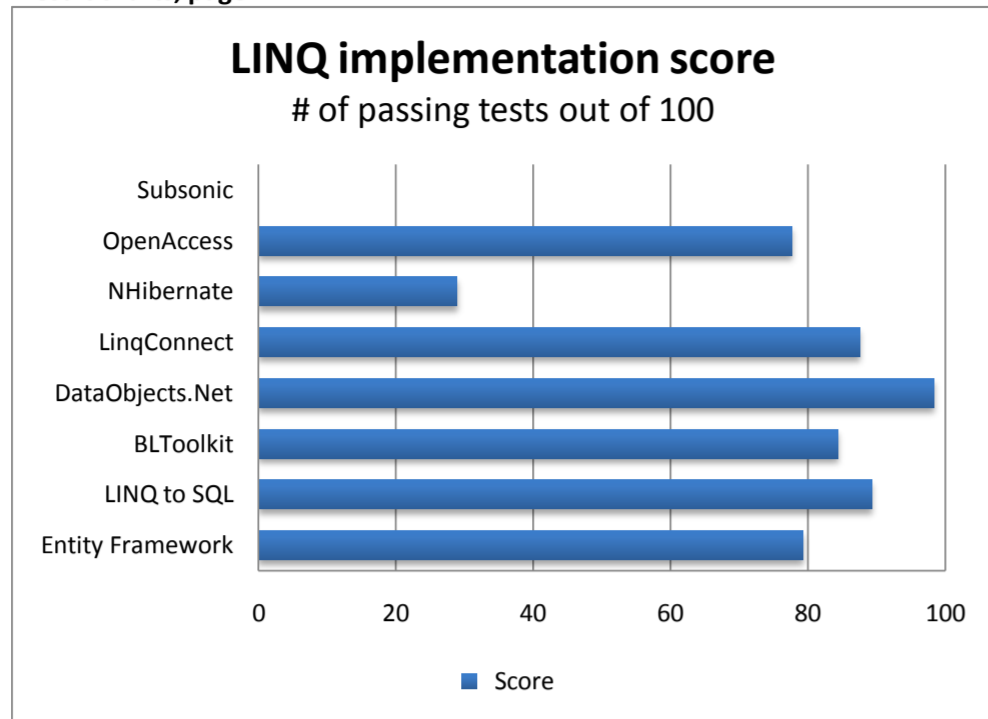


# ORMeter.NET test results

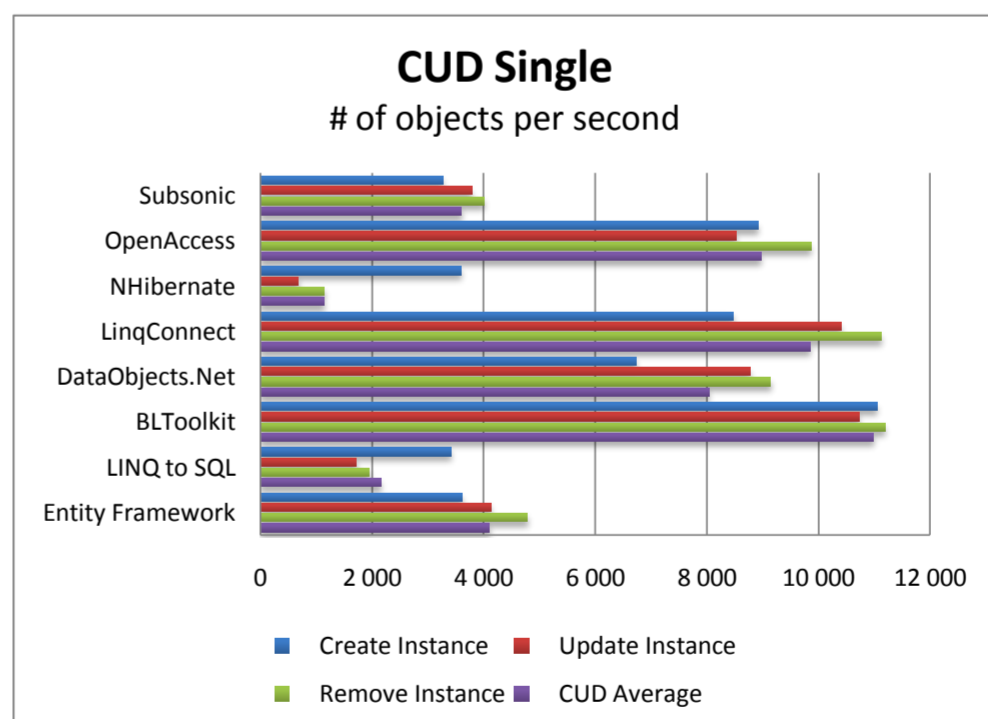
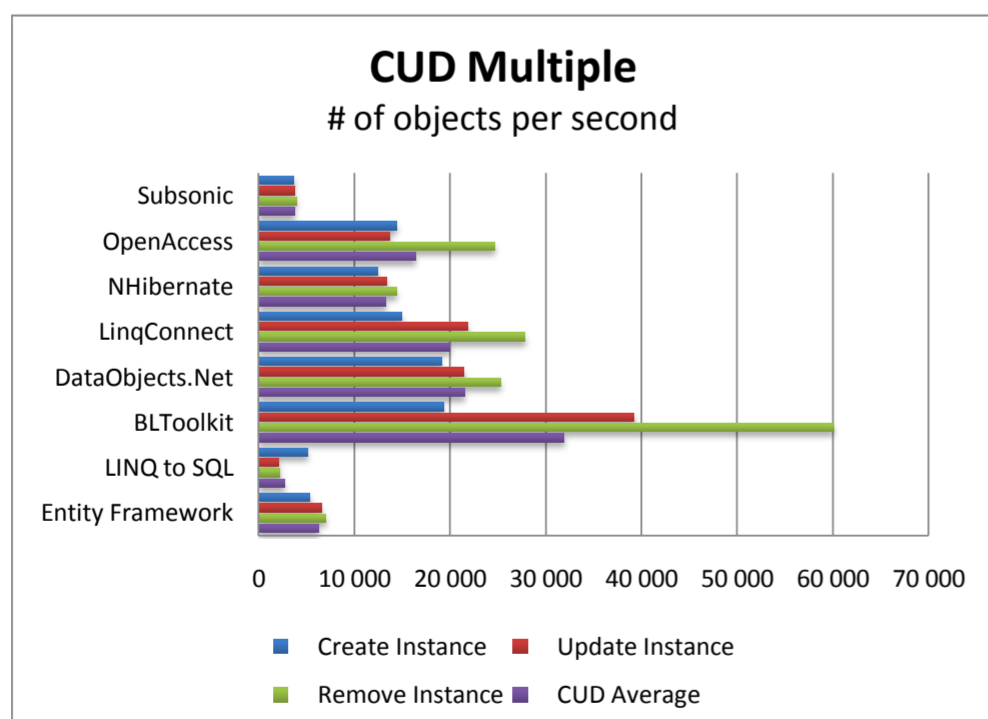
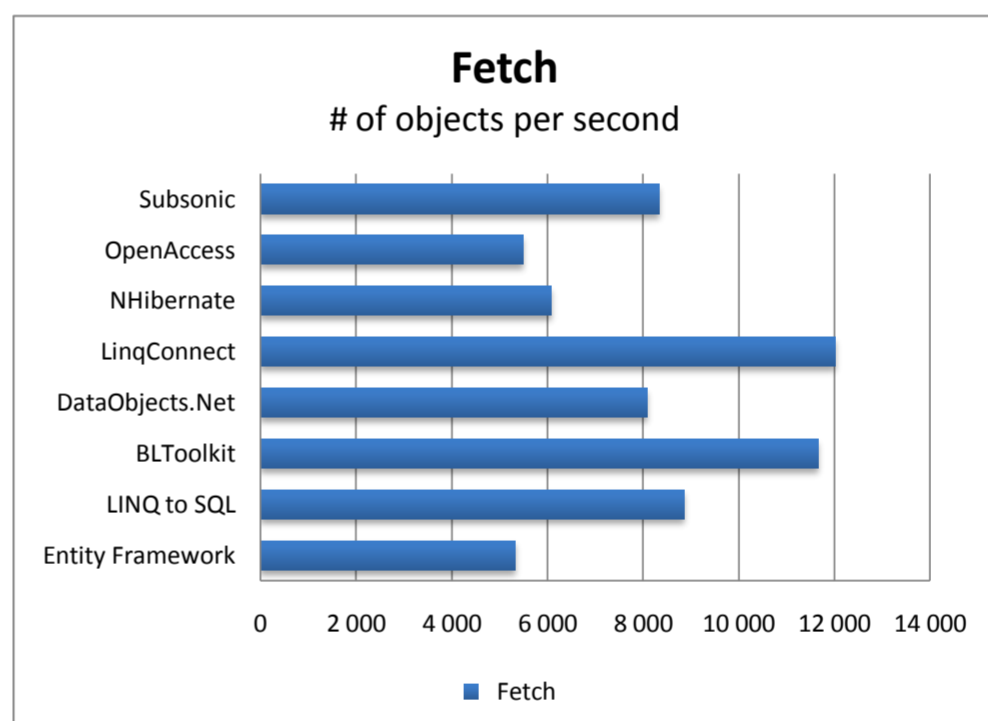
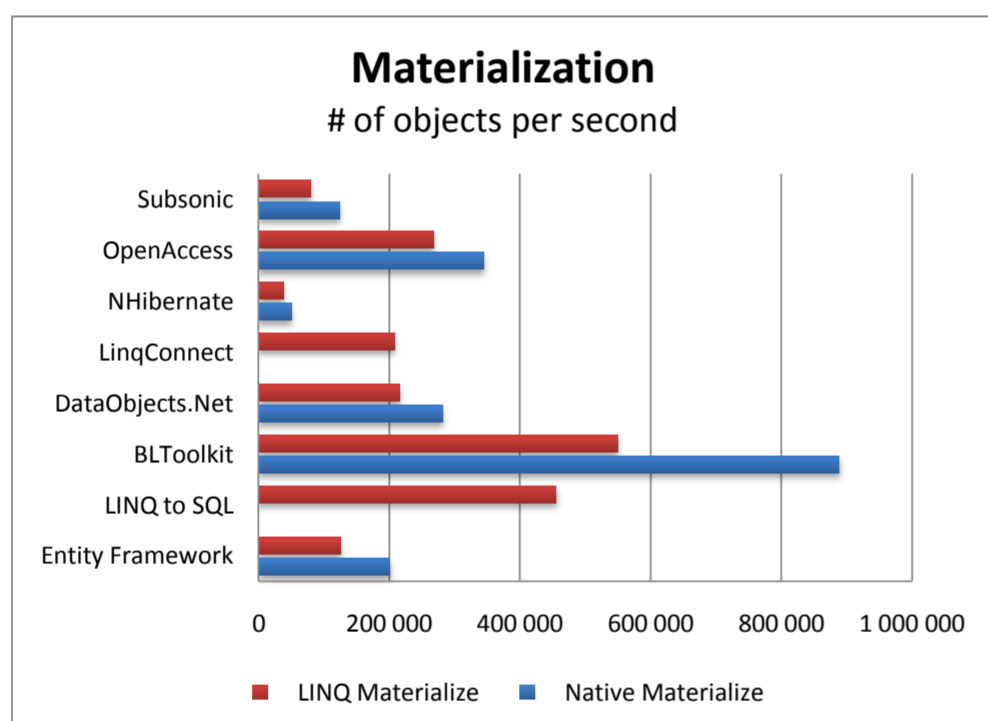
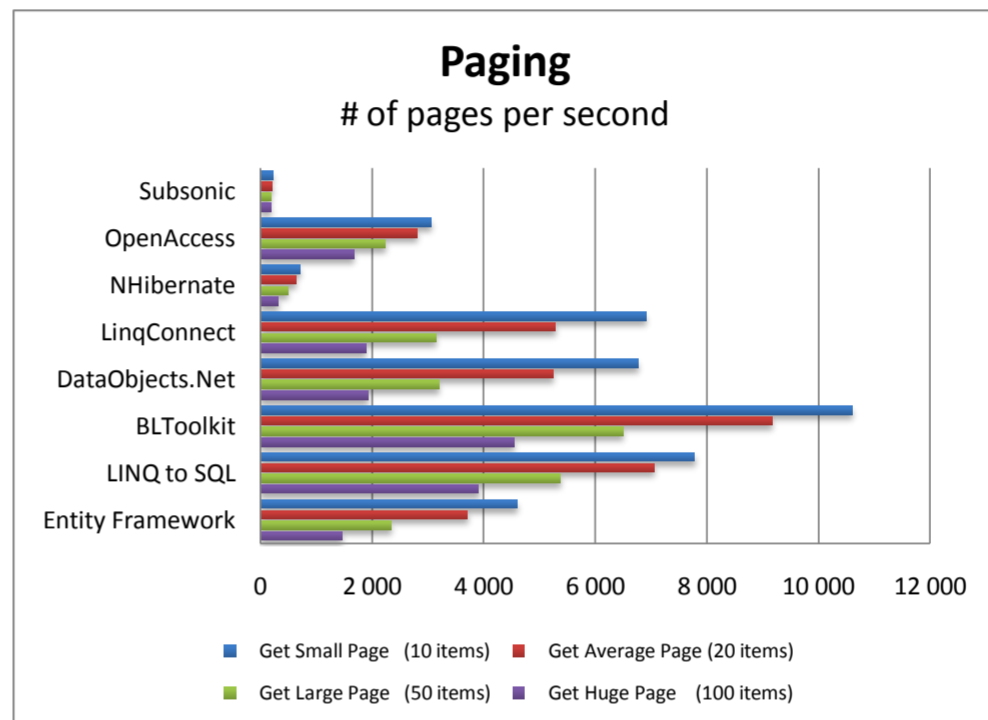
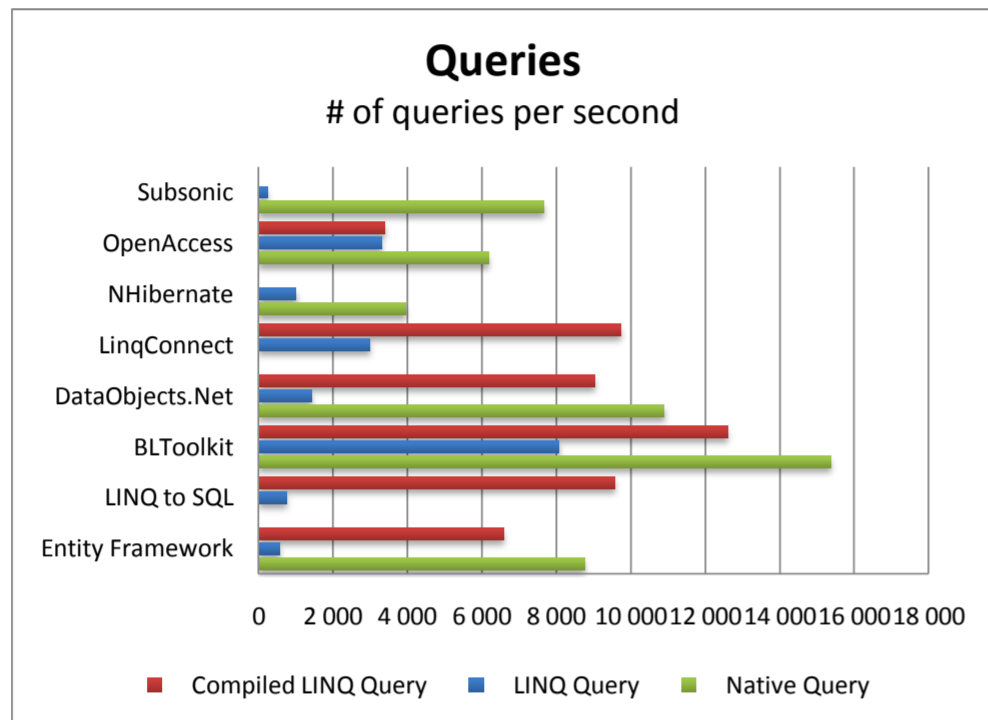
Visit <http://ormeter.net> for details

LINQ implementation												
Test name	Minimum	Maximum	SqlClient	Entity Framework	LINQ to SQL	BLToolkit	DataObjects.Net	LinqConnect	NHibernate	OpenAccess	Subsonic	Unit
<b>LINQ implementation</b>												
Aggregates	0	5	n/a	0	0	1	0	1	3	1	n/a	f,a
All/Any/Contains	0	6	n/a	0	1	1	0	1	4,2	3	n/a	f,a
Complex	0	6	n/a	0	0	6	0	2	6	4	n/a	f,a
Element operations	0	9	n/a	4	2	5	0	0	6	5	n/a	f,a
Filtering	0	12	n/a	4,2	2,2	0	0	0	6,1	0	n/a	f,a
Grouping	0	10	n/a	1	1	3	0	2	10,2	2	n/a	f,a
Join	0	4	n/a	0	0	0	0	1	4	1	n/a	f,a
Ordering	0	8	n/a	2,1	2	0	0	2	6	1,1	n/a	f,a
Projections	0	13	n/a	1	1	2,1	0	1	6,1	1,1	n/a	f,a
References	0	4	n/a	0	0	0	0	0	3	0	n/a	f,a
Set operations	0	9	n/a	0	0	0	0	1	6,2	0	n/a	f,a
Standard functions	0	25	n/a	11,1	3,1	0	2	3,1	20	8,1	n/a	f,a
Take/Skip	0	5	n/a	1	0	1	0	0	2,1	1	n/a	f,a
Type casts	0	5	n/a	1	1	0	0	1	4	0	n/a	f,a
<b>LINQ Implementation total:</b>												
Performed	0	121	n/a	121	121	121	121	121	121	121	n/a	#
Passed	0	121	n/a	96	108	102	119	106	35	94	n/a	#
Failed	0	121	n/a	25	13	19	2	15	86	27	n/a	#
Properly	0	121	n/a	21	10	18	2	14	77	24	n/a	#
Asserted	0	121	n/a	4	3	1	0	1	9	3	n/a	#
Score	0	100	n/a	79,3	89,3	84,3	98,3	87,6	28,9	77,7	n/a	%
<b>Color bar</b> <span style="float:right">Worst result <span style="display:inline-block; width:100px; height:10px; background: linear-gradient(to right, red, orange, yellow, green);"></span> Best result</span>												
<b>Units:</b>												
f/a	total count of failed tests [, count of tests failed with assertion ], less is better (0 is ideal)											
#	count											
%	percentage (% of passed tests), more is better											
												1000 item sequence
Test name	Minimum	Maximum	SqlClient	Entity Framework	LINQ to SQL	BLToolkit	DataObjects.Net	LinqConnect	NHibernate	OpenAccess	Subsonic	Unit
<b>CRUD Performance:</b>												
Fetch	0		20 808	5 316	8 855	11 663	8 086	12 007	6 070	5 493	8 332	op/s
<b>Single Operation:</b>												
Create Instance	0		17 048	3 619	3 409	11 047	6 737	8 478	3 599	8 914	3 269	op/s
Update Instance	0		16 738	4 128	1 723	10 730	8 770	10 405	683	8 531	3 788	op/s
Remove Instance	0		17 666	4 781	1 950	11 193	9 144	11 130	1 149	9 870	4 004	op/s
CUD Average	0		17 142	4 105	2 163	10 986	8 036	9 853	1 148	8 972	3 602	op/s
<b>Multiple Operations:</b>												
Create Instance	0		39 342	5 383	5 124	19 329	19 086	14 929	12 417	14 443	3 711	op/s
Update Instance	0		41 532	6 571	2 101	39 244	21 465	21 876	13 413	13 729	3 788	op/s
Remove Instance	0		25 463	7 026	2 244	60 092	25 306	27 844	14 416	24 660	3 982	op/s
CUD Average	0		33 750	6 246	2 683	31 926	21 519	19 968	13 277	16 421	3 738	op/s
<b>Data Access Performance:</b>												
<b>Query:</b>												
LINQ Query	0		n/a	563	743	8 064	1 414	2 980	1 008	3 298	231	queries/s
Compiled LINQ Query	0		n/a	6 586	9 575	12 601	9 029	9 732	n/a	3 383	n/a	queries/s
Native Query	0		18 017	8 758	n/a	15 374	10 898	n/a	3 945	6 174	7 656	queries/s
<b>Paging (LINQ only):</b>												
Get Small Page (10 items)	0		n/a	4 600	7 779	10 608	6 769	6 912	707	3 057	221	pages/s
Get Average Page (20 items)	0		n/a	3 708	7 050	9 174	5 246	5 290	631	2 805	212	pages/s
Get Large Page (50 items)	0		n/a	2 348	5 373	6 494	3 198	3 151	496	2 235	191	pages/s
Get Huge Page (100 items)	0		n/a	1 467	3 910	4 538	1 929	1 890	325	1 671	189	pages/s
<b>Materialization:</b>												
LINQ Materialize	0		n/a	125 922	455 497	549 480	215 973	208 281	37 802	268 046	79 624	objects/s
Native Materialize	0		1 186 661	201 328	n/a	888 651	281 056	n/a	49 879	344 317	124 055	objects/s
<b>File sizes (in bytes):</b>												
Runtime libraries (.dlls)			n/a	n/a	n/a	1 584 128	4 345 856	911 872	2 481 844	4 755 968	285 184	bytes
<b>Color bar</b> <span style="float:right">Worst result <span style="display:inline-block; width:100px; height:10px; background: linear-gradient(to right, red, orange, yellow, green);"></span> Best result</span>												
<b>Units:</b>												
op/s	operations per second, more is better											
queries/s	queries per second, more is better											
pages/s	pages per second, more is better											
objects/s	# of materialized objects per second, more is better											

Result charts, page 1:



Larger = better!



**Cell:** R46C7

**Comment:** Alex Yakunin:  
Specialized API:  
SqlQuery<T>.Insert(..., int batchSize, IEnumerable<T> sequence)

**Cell:** R46C10

**Comment:** Alex Yakunin:  
Specialized API: ISessionFactory.OpenStatelessSession is used in this test.

**Cell:** R47C7

**Comment:** Alex Yakunin:  
Specialized API:  
SqlQuery<T>.Update(..., int batchSize, IEnumerable<T> sequence)

**Cell:** R47C10

**Comment:** Alex Yakunin:  
Specialized API: ISessionFactory.OpenStatelessSession is used in this test.

**Cell:** R48C7

**Comment:** Alex Yakunin:  
Specialized API:  
SqlQuery<T>.Delete(..., int batchSize, IEnumerable<T> sequence)

**Cell:** R48C10

**Comment:** Alex Yakunin:  
Specialized API: ISessionFactory.OpenStatelessSession is used in this test.

**Cell:** R54C10

**Comment:** Alex Yakunin:  
Complied queries are not supported, result is copied from above cell.

**Cell:** R54C12

**Comment:** Alex Yakunin:  
Complied queries are not supported, result is copied from above cell.

**Cell:** R55C6

**Comment:** Alex Yakunin:  
No native queries, result is copied from above cell.