SOL Software d.o.o.

Kosovska 28, 11000 Belgrade, Serbia
Tel: +381-11-3231310
Fax: +381-11-3343451
www.sol.rs

SOLoist OQL vs Hibernate HQL Benchmarking Report

Product: SOLoist, FastOQL	**Reviewed by:**
Document Type: Product Description	**Review Date:**
Document Version: 1.1	**Comment:**
Authors: Srđan Luković, Dragan Milićev	
Date: July 6, 2012	

Contents

Introduction

This document is a detailed report on the benchmarking of the SOLoist™ OQL technology versus Hibernate™ HQL technology conducted in SOL Software during spring 2012.

SOLoist OQL and Hibernate HQL are technologies for executing object queries against relational databases in object-relational environments (SOLoist and Hibernate, respectively). Both translate an object query (with classes and joins over class relationships) into an SQL query that is performed in a relational database. However, these two technologies are parts of slightly different environments: SOLoist is a framework based on executable UML models with an object-relational persistence layer, while Hibernate ensures persistence of Java objects. Anyhow, their query subsystems (OQL and HQL) have many similarities in purpose, syntax, and semantics.

The purpose of both technologies is to provide a means for developers to define complex object queries in an SQL-like syntax, but on a higher level of abstraction than in SQL. The queries may perform complex searches of objects over large databases. In such cases, it is particularly critical how efficiently these queries, with potentially lots of joins and 'where' conditions over class hierarchies, perform in large databases. It is well known in the database community that performance may be dramatically affected by the complexity of the query and the database size. The purpose of this experiment was to explore these effects for the two mentioned technologies.

In particular, the goals of the experiment were:

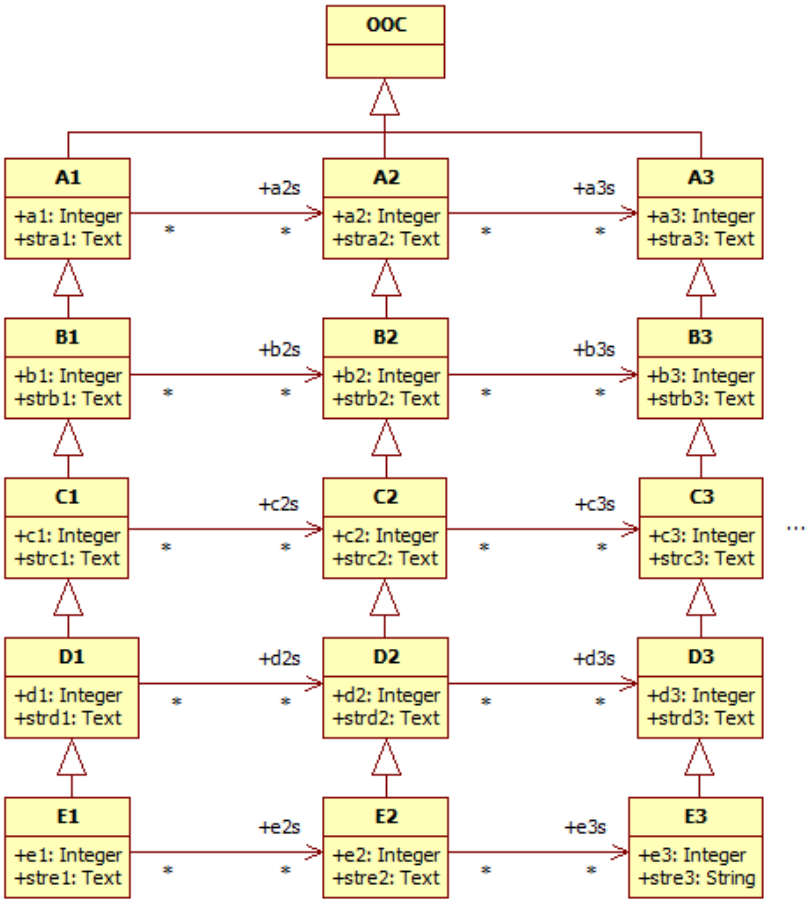
a) to benchmark and compare the performance of the two technologies, in the case of simple as well as complex class models and object queries with many joins (hops over class relationships and 'where' conditions), and
b) to evaluate the scalability of the two technologies, in terms of the size of the class model and the database size.

This document provides complete data about the conditions and assumptions of the experiment, so that it can be objectively reproduced by any motivated independent party. The document also presents the results of the experiments.

Conditions and Assumptions of the Experiment

Test Class Model

The experiment was done on the class model depicted in the UML diagram below.



There is one class hierarchy with a common base class, named *OOC*. The rationale for such a model was to emulate the situation of queries on class hierarchies with or without common base classes, but always with some base classes involved, which is very typical for object-oriented information systems. Namely, the existence of the same common base class *OOC* is equivalent (in terms of query translation to SQL and execution) with the case when each class used in a query has its own (uncommon) base class.

The class hierarchy has the "depth" of five classes (*A*-*E*) below OOC in each „vertical". Each of these classes has one integer and one textual (string) attribute (apart from its technical ID that is the primary key in the database and is not shown in the diagram because it is assumed). Each of these classes has one many-to-many, unidirectional relationship (association) with the neighboring class on the same level. The width of the model is 10 verticals (*A1-A10*, ..., *E1-E10*).

This model serves as the underlying base for queries that will be explained later and that will span over different "widths" and "depth levels" of the model. If a query does not span over some classes right to a certain vertical or below a certain horizontal, then the rest of the model does

not affect the query and is practically irrelevant (as if it does not exist). This is why the same model covers all the cases of class models with different widths (from 1 to 10) and inheritance depths (from 1 to 5, not counting *OOC*).

The idea is to construct a number of queries that will span (via class joins) over a certain number of classes and relationships on the same (but variable) horizontal, with a variable "query width," with or without including attributes (inherited or own) in the 'where' clauses. The purpose is to benchmark how the number of class joins (the query width) and the depth of the hierarchy affects the performance of the queries for the two benchmarked technologies.

The Database Schema

Both SOLoist OQL and Hibernate HQL queries were executed against the very same database. SOLoist was used to create the database schema, with small tweaks to accommodate it to Hibernate. The following are DDL statements for MySQL.

The OOC Table

The table for the base class *OOC* is created like this:

```
CREATE TABLE `ooc` (
  `id` bigint(20) NOT NULL DEFAULT '0',
  `version` bigint(20) NOT NULL,
  `DTYPE` varchar(31) DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `indexica` (`DTYPE`) USING BTREE,
  CONSTRAINT `ooc_ibfk_1` FOREIGN KEY (`id`) REFERENCES `objectofclass` (`id`)
)
```

The *id* field is the default primary key used by SOLoist for each class. The index on it is automatically created by the DBMS.

The *version* field is created by SOLoist for concurrency control. It is not used in this experiment at all, because this is a read-only benchmark. Hibernate is completely unaware of this field.

The *DTYPE* field is the type discriminator used by Hibernate's inheritance mapping. This field is indexed. It is artificially supplied to SOLoist as a class attribute just to be created in the database schema and then it is not used by SOLoist at all. SOLoist uses a type discriminator embedded in *id*.

Class Tables

The tables for the remaining classes *A1..10-E1..10* look like follows:

```
CREATE TABLE `a1` (
  `id` bigint(20) NOT NULL DEFAULT '0',
  `version` bigint(20) NOT NULL,
  `a1` int(11) DEFAULT NULL,
  `stra1` varchar(256) DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `a1_127264_btree` (`a1`) USING BTREE,
  KEY `stra1_202969_btree` (`stra1`(255)) USING BTREE
)
```

Attributes *a1* and *stra1* are mapped to the corresponding fields (`` `a1`int(11) ``, `` `stra1`varchar(256) ``). Both fields are indexed.

Association Tables

The tables for the many-to-many relationships (associations) look like this:

```
CREATE TABLE `a1#a2s_a2` (
  `a1_id` bigint(20) NOT NULL,
  `a1_pos` int(11) NOT NULL,
  `a2s_id` bigint(20) NOT NULL,
```

```
  `a2s_pos` int(11) NOT NULL,
  KEY `a1_id` (`a1_id`),
  KEY `a2s_id` (`a2s_id`),
  CONSTRAINT `a1@0023a2s_a2_ibfk_1` FOREIGN KEY (`a1_id`) REFERENCES `a1`
(`id`),
  CONSTRAINT `a1@0023a2s_a2_ibfk_2` FOREIGN KEY (`a2s_id`) REFERENCES `a2`
(`id`)
)
```

The fields *a1_id* and *a2s_id* are foreign keys referencing records in the related class tables. Both fields are indexed. The fields *a1_pos* and *a2s_pos* are fields used by SOLoist for ordered collections. In this experiment they are not used by SOLoist at all, and Hibernate is completely unaware of them.

ObjectOfClass Table

The table for the SOLoist's default class *ObjectOfClass* looks like this:

```
CREATE TABLE `objectofclass` (
  `id` bigint(20) NOT NULL DEFAULT '0',
  `version` bigint(20) NOT NULL,
  PRIMARY KEY (`id`)
)
```

The *ObjectOfClass* table is implicitly created by SOLoist. It mostly has only a conceptual significance, while its practical value is to query for all objects of all classes if necessary. In this benchmark it is not used by SOLoist at all and Hibernate is completely unaware of this table.

Indices and Constraints

Apart from the mentioned differences between the parts of the schema used by either SOLoist or Hibernate, there are two more. Hibernate (if the schema were automatically generated by it from the Java class model) would create a composite primary key on both *a1_id* and *a2s_id* fields. Second, Hibernate would create different referential constraints – the foreign key fields from association tables reference the primary key in the *OOC* table instead of the "neighboring" tables.

All these differences are irrelevant for this benchmark, as they are not used at all and/or do not affect the execution of the queries, because the data are not modified and integrity constraints do not have to be checked. The composite primary key in the association tables would only be relevant for prohibiting multiple equal links in an association table, while joins use indices on primitive foreign keys.[1]

Hibernate Mappings

We used Hibernate's „*table per subclass*" strategy with a „*type discriminator*" to map the inheritance hierarchy. It is the strategy that SOLoist uses too (apart from the type discriminator) and it is most appropriate for this test model (which should simulate a real-world scenario). In addition, it seems that this strategy is most frequently used in practice, or at least one of the favorite ones.

The „*table per subclass*" strategy does not use a type discriminator by default. Instead, Hibernate relies on outer joins and the presence of subclass fields in the resulting table (using SQL *CASE* statement). This may be, however, extremely inefficient, because Hibernate then generates SQL queries that outer join *all* classes in a certain vertical, above or below the class referenced in the query. We wanted to achieve the best from Hibernate, so we did not want to use this approach, but to obtain the most compact SQL queries with the least possible number of joins.

This is why we introduced a type discriminator. With it, Hibernate joins only the classes above the referenced class in a class vertical. Adding a type discriminator to the „*table per*

[1] Even Hibernate would not create this composite primary key if we mapped a bag instead of a set as the corresponding Java collection.

subclass“ strategy is achieved by using the „*single table per class hierarchy*“ strategy and then switching to the other strategy for a particular subclass (in this case all of the subclasses). We have used annotations to specify these mappings, as it was more convenient for generating classes for variable sized class models.

OOC class

```
@Entity
@Inheritance(strategy = InheritanceType.SINGLE_TABLE)
@DiscriminatorValue("OOC")
public class OOC {

    @Id @GeneratedValue
    private Long id;
…
```

Subclasses

```
@Entity
@DiscriminatorValue("A1")
@SecondaryTable(name = "A1")
public class A1 extends OOC {

    @Column(table = "A1")
    private int a1;
    @Column(table = "A1")
    private String stra1;

    @ManyToMany(fetch = FetchType.LAZY)
    @JoinTable(name = "`a1#a2s_a2`", joinColumns = @JoinColumn(name = "a1_ID"),
        inverseJoinColumns = @JoinColumn(name = "a2s_ID"))
    private Set<A2> a2s = new HashSet<A2>();
…
```

Associations

Associations are implemented as *HashSet* collections, with the „*lazy fetch*“ strategy. This does not affect the performance of HQL queries in our benchmark as they are always executed as single SQL queries.

The Data

The benchmarks were executed on different database sizes. The total number of objects of all classes, that is, the total number of instances of the class *OOC* range from 1 thousand to 10 million, with a step of an order of magnitude: 1K, 10K, 100K, 1M, and 10M. The total number of records in the database is 7.7 times larger. We also performed some (but not all) experiments on the database with 100M objects.

Every subclass of *OOC* has the same number of direct instances. *OOC* itself does not have direct instances. For example, in the database of 10K objects, each of the ten verticals (*A1*-*A10*) has 1K=10K/10 objects: *Ei* (*i*=1..10) has 200=1K/5 direct instances, *Di* has 200 direct instances and 200 indirect instances (the instances of *Ei*), *Ci* has 200 direct instances and 2·200 indirect instances (the instances of *Di*), *Bi* has 200 direct instances and 3·200 indirect instances (the instances of *Ci*), and finally, *Ai* has 200 direct instances and 4·200 indirect instances (the instances of *Bi*). Each class table has as many records as that class has instances (both direct and indirect). This distribution mimics typical situations in real-world systems, where base classes (and their tables) have more and more records as they are higher in the hierarchy.

Every object is linked with one random object via each association. For example, an object of *C1* is linked with one object via *c2s*, the former object can be a direct instance of *C2*, *D2*, or *E2*, randomly. Similarly, it is linked with one object via *b2s* and with one via *a2s*.

The data is loaded into the database using bulk load of CSV files prior to the benchmark. CSV files are generated programmatically. SOLoist requires prefix-coded IDs (the prefix being the type discriminator). The rest of the ID is randomly generated, and it has a uniform distribution (this may improve index performance). Randomly generated IDs are used to derive other fields in class tables. The text field is set to the string representation of the object's ID, and the integer field is the result of the conversion of ID (*long int*) to *int* – discarding the upper half (most significant) of bits.

The Benchmark Queries

The benchmark suite consists of 1040 queries. All queries have the simple form:

```
select count(*)
from X1 [join X2 join X3...]
[where <clause>]
```

We have chosen to use COUNT in the result in order not to introduce the impact of the nature of the result set into the benchmark. Instead, we are focused on the impact of the FROM and WHERE clauses. Namely, we deem that complex queries are most often used to search for and identify (very often a small) set of objects that satisfy a complex criterion in a large database, and either to identify them by their IDs and then fetch the entire identified objects one by one programmatically, by individual object fetches through the ORM persistence layer, or to fetch a small set of properties of these objects to render in the UI, such as in search results in the application's UI. In such cases, the most challenging part for the database is to select the required records, on which FROM and WHERE parts have the greatest impact. Our queries with the COUNT function in the result isolates only this part of query execution and does not pose any burden to the database to fetch concrete data. It can be expected that both Hibernate SQL and SOLoist OQL perform the best in these conditions.

Nevertheless, in order to check the soundness of this assumption, we have still conducted a separate experiment with the same form of the queries, but with data returned in the result set instead of just counting the selected records. These queries returned the values of all attributes (owned and inherited) of the object of the class in the first vertical (*X1*). We present these results in the last section of the Results chapter. These results corroborate our assumption.

The number of classes in the FROM clause varies from 1 to 10. This simulates a variable width of the model and will be referred to as the "model width" in the presentation of the results. All classes in the FROM clause are on the same horizontal (depth) in the class model. The level of the horizontal iterates down the class hierarchy from *A* to *E*, thus simulating a variable depth of the model and will be referred to as the "model depth" in the presentation of the results. (Note: This is true because neither SOLoist nor Hibernate joins class tables below the classes referenced in the object query when forming the SQL translation.)

In the simplest form, the WHERE clause is omitted. Then, the number of conditions increases from 1 to *W* (where *W* is the width of the query), adding fields from each vertical; this will be referred to as the "where count" in the presentation of the results. The conditions are merged with a conjunction expression (AND). For each number of conditions, four queries are generated. The "integer" condition is in the form `x1 > 0`, while the "string" condition is in the form `strx1 LIKE '2C0%'`. The fields *x1* and *strx1* can be from the topmost horizontal (*A*, denoted as "integer/string level=1" in the results), or from the current horizontal (denoted as "integer/string level=model depth" in the results). This is to check the impact of whether the attribute used in the WHERE clause is inherited or from the same class referenced in the query. The string literal in the "string" condition is actually the object ID prefix, i.e., the SOLoist class identifier plus two extra bits – this

filters roughly a quarter of the records in the class table (remember that the string attributes are set to the string representations of their object IDs).

As an example, the Hibernate HQL queries of width 2 for the *B* horizontal (i.e., for the model sized 2x2), are the following (the SOLoist OQL queries are equivalent, just slightly syntactically different):

```
select count(*)
from B1 x1
join x1.b2s x2

select count(*)
from B1 x1
join x1.b2s x2
where x1.a1 > 0

select count(*)
from B1 x1
join x1.b2s x2
where x1.stra1 LIKE '244%'

select count(*)
from B1 x1
join x1.b2s x2
where x1.b1 > 0

select count(*)
from B1 x1
join x1.b2s x2
where x1.strb1 LIKE '244%'
select count(*)
from B1 x1
join x1.b2s x2
where x1.a1 > 0 and x2.a2 > 0

select count(*)
from B1 x1
join x1.b2s x2
where x1.stra1 LIKE '244%' and x2.stra2 LIKE '248%'

select count(*)
from B1 x1
join x1.b2s x2
where x1.b1 > 0 and x2.b2 > 0

select count(*)
from B1 x1
join x1.b2s x2
where x1.strb1 LIKE '244%' and x2.strb2 LIKE '248%'
```

Benchmark Program

The benchmark program is written in Java. It executes queries sequentially, one at a time, for each database size and each particular ORM (SOLoist or Hibernate) in a separate batch. String representations of HQL and OQL queries were first forwarded to their respective frameworks. The frameworks did the compilation of object queries to SQL. SQL statements were then executed via JDBC drivers and their execution time was measured. This is because we wanted to take into account only the pure query execution time, without all other overhead.

Each query ran successively multiple times, until the measured execution time was within a 10% margin of the previous execution time of the same query, but the maximum of five repetitions were allowed. Then, the shortest time was recorded as the result for each query. The shortest time was taken to alleviate the (very significant) impact of DBMS caching subsystems. Namely, the execution time of one query heavily depends on how much of the data the DBMS retrieved from the disk or already had in the cache: when the same query is executed twice, very often (but not always!) its second execution may be a few orders of magnitude shorter than the first one. However, our goal was not to measure the impact of various DBMS caching mechanisms or hardware devices. Instead, we were focused on the impact of essential complexity of queries. This is why we wanted to give the DBMS the opportunity to fetch as much data to its cache as possible, so that it can perform the best. This is why we ran each query a couple of times as described (until less than a 10% difference was recorded), and took the best of the few obtained results in all cases.

The execution of each query was limited to one hour.

Both technologies (Hibernate and SOLoist) ran under the same conditions and in the same manner as described.

Software

We performed the benchmark on two different relational DBMSs:

- MySQL Database Server 5.5.24; InnoDB engine (innodb_buffer_pool_size=2G)
- Oracle Database 11g Express Edition (11.2.0.2) (limited to 1GB of memory).

No additional performance settings were changed (all defaults).

The benchmark was executed on the following software:

- Operating System: Windows 7 Professional 64 bit
- Java jdk1.7.0_03
- Hibernate 4.1.2
- JDBC:
    - MySQL 5.1.19
    - Oracle ojdbc6

Hardware

The benchmark was executed on one simple desktop PC with the following characteristics:

- CPU: AMD Athlon 64 X2 5200+ (2.6GHz, 2MB L2 Cache)
- RAM: 4GB DDR2
- HDD: Western Digital Caviar Blue 250GB (SATA-300, 7200 rpm, buffer: 16 MB).

Both the client Java benchmark and the RDBMS processes were running on the same machine.

Results

By performing the described benchmarks, we have obtained a huge number of results. We have stored all the results in a star-shaped data warehouse model that can be sliced according to a number of different dimensions:

- Database: Oracle or MySQL
- Framework: SOLoist or Hibernate
- Data size: the total number of objects (1K, 10K, 100K, 1M, 10M, 100M)
- Model Width: the width of the queries, i.e., the number of related classes (1-10)
- Model Depth: the horizontal (1-5, i.e. *A*-*E*)
- Where Count: the number of terms in the WHERE clause
- String Level: the level of the string attributes used in the WHERE clause (1-taken from *A*, or „model depth“ – taken from the current horizontal)
- Integer Level: the level of the integer attributes used in the WHERE clause (1-taken from *A*, or „model depth“ – taken from the current horizontal).

For each query, we measured two values (the facts in the data warehouse):

- the number of joins (“join count”) in the translated SQL query
- the execution time under the specified conditions; all results are expressed in milliseconds.

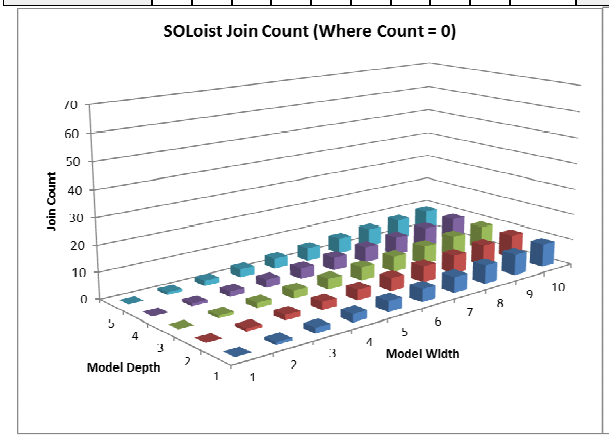
In the following sections, we present a selection of these results, sliced on some dimensions. For each result, we give the tabular representation and the graphical representation of the data from the table.

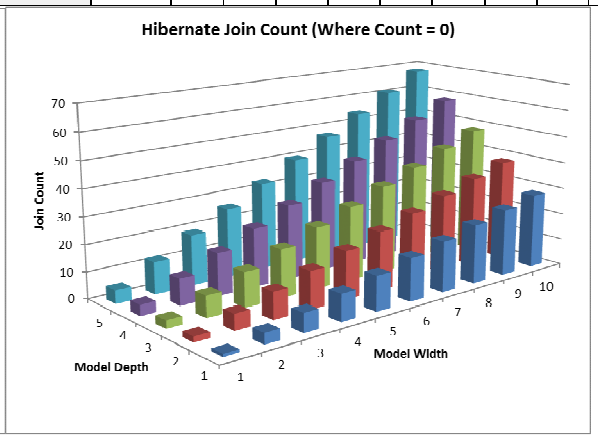
SQL vs. Object Query Complexity

In this section, we present how the number of joins in the SQL queries translated by Hibernate and SOLoist depend on the complexity of the object query, that is, for different model sizes (width and depth).

Join Count (Where Count = 0)

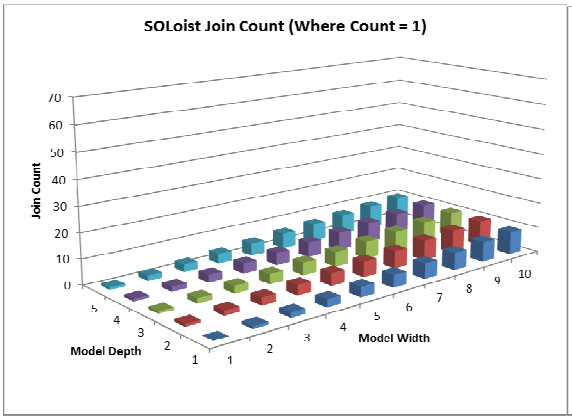
SOLoist	Join Count (Where Count = 0)										Hibernate	Join Count (Where Count = 0)									
	Model Width											Model Width									
Model Depth	1	2	3	4	5	6	7	8	9	10	Model Depth	1	2	3	4	5	6	7	8	9	10
1	0	1	2	3	4	5	6	7	8	9	1	1	4	7	10	13	16	19	22	25	28
2	0	1	2	3	4	5	6	7	8	9	2	2	6	10	14	18	22	26	30	34	38
3	0	1	2	3	4	5	6	7	8	9	3	3	8	13	18	23	28	33	38	43	48
4	0	1	2	3	4	5	6	7	8	9	4	4	10	16	22	28	34	40	46	52	58
5	0	1	2	3	4	5	6	7	8	9	5	5	12	19	26	33	40	47	54	61	68

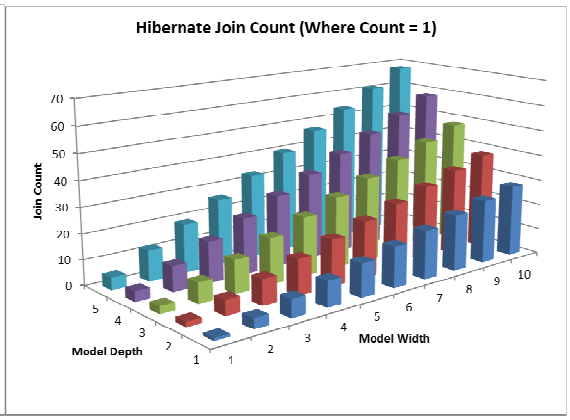




Join Count (Where Count = 1, Integer/String Level = 1)

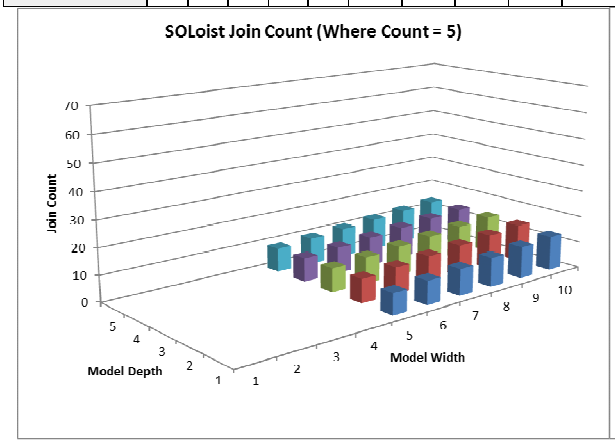
SOLoist	Join Count (Where Count = 1)										Hibernate	Join Count (Where Count = 1)									
	Model Width											Model Width									
Model Depth	1	2	3	4	5	6	7	8	9	10	Model Depth	1	2	3	4	5	6	7	8	9	10
1	0	1	2	3	4	5	6	7	8	9	1	1	4	7	10	13	16	19	22	25	28
2	1	2	3	4	5	6	7	8	9	10	2	2	6	10	14	18	22	26	30	34	38
3	1	2	3	4	5	6	7	8	9	10	3	3	8	13	18	23	28	33	38	43	48
4	1	2	3	4	5	6	7	8	9	10	4	4	10	16	22	28	34	40	46	52	58
5	1	2	3	4	5	6	7	8	9	10	5	5	12	19	26	33	40	47	54	61	68

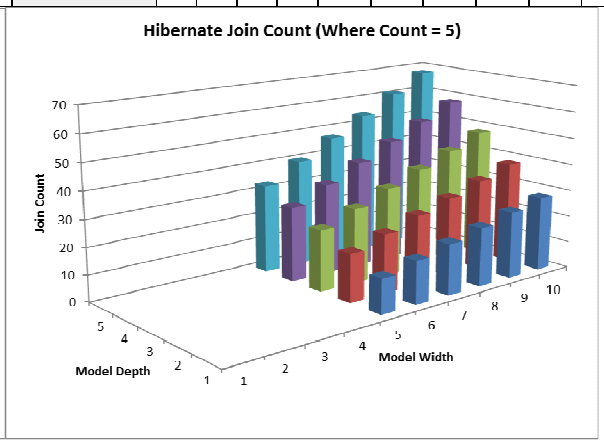




Join Count (Where Count = 5, Integer/String Level = 1)

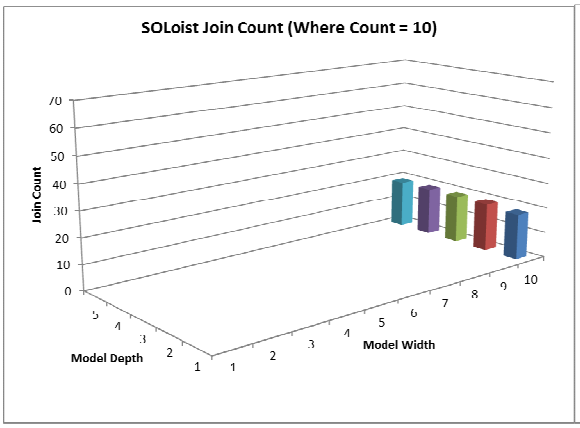
SOLoist	Join Count (Where Count = 5)										Hibernate	Join Count (Where Count = 5)									
	Model Width											Model Width									
Model Depth	1	2	3	4	5	6	7	8	9	10	Model Depth	1	2	3	4	5	6	7	8	9	10
1					8	9	10	11	12	13	1					13	16	19	22	25	28
2					9	10	11	12	13	14	2					18	22	26	30	34	38
3					9	10	11	12	13	14	3					23	28	33	38	43	48
4					9	10	11	12	13	14	4					28	34	40	46	52	58
5					9	10	11	12	13	14	5					33	40	47	54	61	68

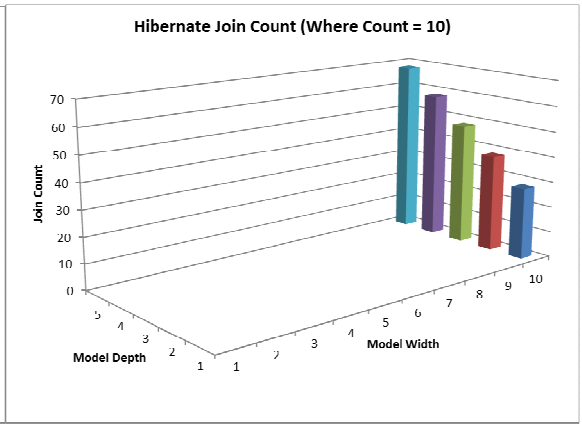




Join Count (Where Count = 10, Integer/String Level = 1)

SOLoist	Join Count (Where Count = 10)										Hibernate	Join Count (Where Count = 10)									
	Model Width											Model Width									
Model Depth	1	2	3	4	5	6	7	8	9	10	Model Depth	1	2	3	4	5	6	7	8	9	10
1										18	1										28
2										19	2										38
3										19	3										48
4										19	4										58
5										19	5										68





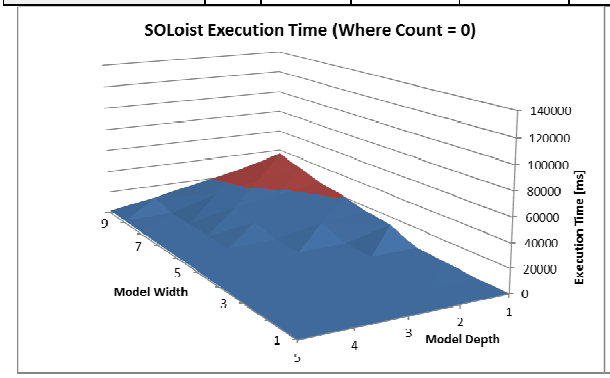
Scalability: Execution Time vs. Object Query Complexity

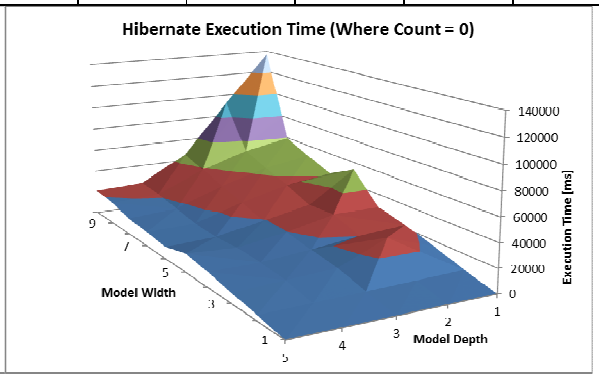
In the following set of results, we are presenting the dependency of the execution time on the complexity of the object query for SOLoist and Hibernate, that is, for different sizes of the model (width and depth). All the results are for the referential database size of 10M objects and for Oracle.

Execution Time (Where Count = 0, Data Size =10^7, DB = Oracle)

SOLoist Data Size = 10^7	Execution Time (Where Count = 0) Model Width									
Model Depth	1	2	3	4	5	6	7	8	9	10
1	47	637	1,989	3,132	12,907	15,829	20,953	24,001	30,422	36,516
2	46	500	1,429	2,313	3,235	11,001	14,657	18,157	23,392	25,704
3	31	500	1,063	1,731	2,434	3,157	7,748	11,970	15,220	17,126
4	15	296	656	1,079	1,532	1,980	2,407	2,860	3,329	8,876
5	0	132	281	828	1,048	1,256	1,469	1,689	1,922	2,137

Hibernate Data Size = 10^7	Execution Time (Where Count = 0) Model Width									
Model Depth	1	2	3	4	5	6	7	8	9	10
1	65	7,128	15,562	22,924	28,670	53,611	41,297	50,110	59,454	137,110
2	52	6,352	29,528	20,185	28,517	27,313	36,204	43,595	51,782	92,407
3	39	5,738	12,357	15,994	19,844	19,470	25,923	31,955	38,972	45,485
4	26	5,153	10,376	12,654	13,720	11,876	13,544	16,970	21,095	22,954
5	14	4,573	8,441	12,623	16,001	10,329	12,410	14,344	18,955	21,329

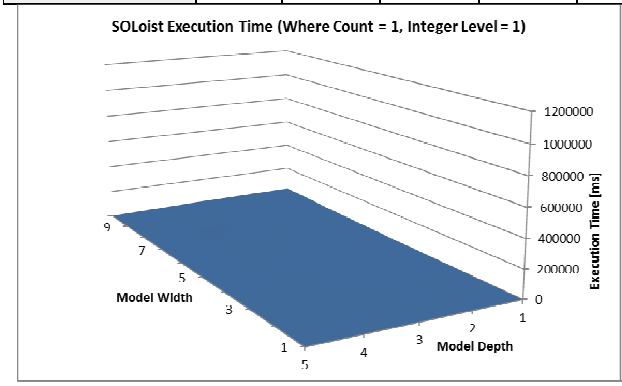


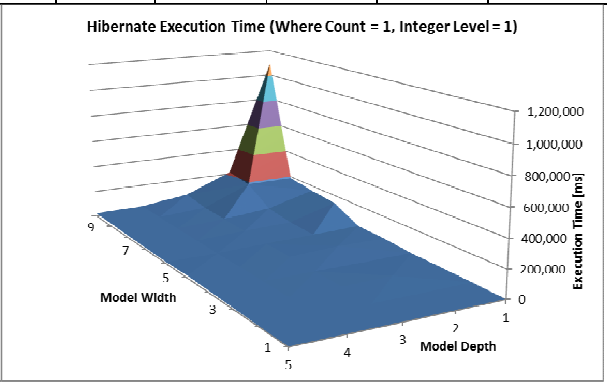


Execution Time (Where Count = 1, Integer Level = 1, Data Size =10^7, DB = Oracle)

SOLoist Data Size = 10^7	Execution Time (Where Count = 1, Integer Level = 1) Model Width									
Model Depth	1	2	3	4	5	6	7	8	9	10
1	31	1,314	2,465	2,564	3,453	6,907	10,199	12,777	19,063	23,876
2	2,580	1,946	2,152	2,840	3,454	9,407	12,141	14,829	18,766	20,298
3	2,407	1,907	1,688	2,188	2,813	6,645	7,782	11,391	14,360	16,688
4	2,235	1,485	1,672	1,627	1,948	2,220	2,541	2,860	6,340	9,485
5	1,891	1,110	1,125	1,314	2,032	2,204	2,438	2,642	2,876	3,110

Hibernate Data Size = 10^7	Execution Time (Where Count = 1, Integer Level = 1) Model Width									
Model Depth	1	2	3	4	5	6	7	8	9	10
1	1,629	10,468	15,360	19,346	35,746	41,923	139,150	160,283	188,516	1,092,189
2	1,521	6,841	20,835	15,309	38,235	31,579	40,861	67,157	187,516	213,048
3	1,313	6,172	13,490	15,584	23,001	24,017	30,032	36,001	43,251	72,563
4	1,105	5,984	11,585	12,297	14,079	13,579	14,923	19,035	22,407	27,454
5	957	5,640	9,472	13,598	17,157	11,220	13,235	15,736	20,032	22,657



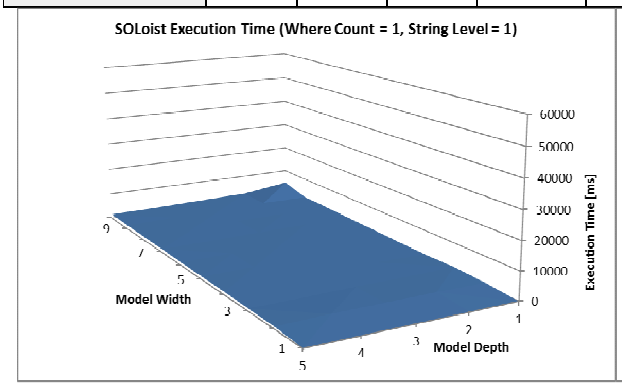


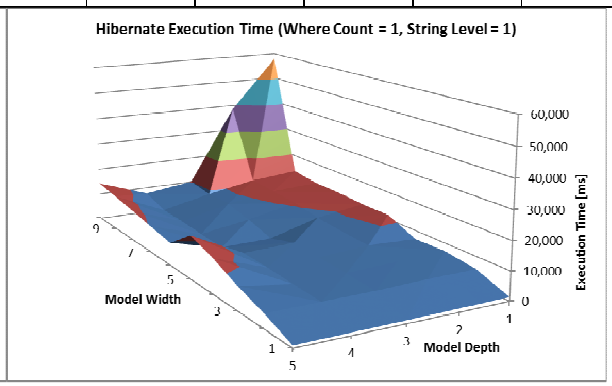
Execution Time (Where Count = 1, String Level = 1, Data Size =10^7, DB = Oracle)

SOLoist Data Size = 10^7	Execution Time (Where Count = 1, String Level = 1) Model Width									
Model Depth	1	2	3	4	5	6	7	8	9	10
1	0	1,000	1,469	1,251	1,423	1,688	1,876	2,173	2,283	4,782
2	171	1,264	1,251	1,469	1,595	1,782	1,923	2,142	2,329	2,516
3	156	1,204	1,219	1,298	1,485	1,626	1,673	1,814	2,111	2,142
4	156	1,204	985	1,189	1,282	1,376	1,438	1,605	1,715	1,799
5	156	844	923	1,088	1,079	1,115	1,268	1,309	1,423	1,595

Hibernate Data Size = 10^7	Execution Time (Where Count = 1, String Level = 1) Model Width									
Model Depth	1	2	3	4	5	6	7	8	9	10
1	1,386	5,706	7,159	6,921	11,506	11,450	12,091	12,142	13,486	57,892
2	1,250	5,572	6,515	7,571	8,157	8,939	10,532	11,236	11,954	38,564
3	1,088	5,098	5,843	6,349	6,954	3,938	6,745	8,829	8,783	10,094
4	988	5,049	5,463	5,969	6,219	3,579	3,985	4,579	4,829	7,688

5	897	4,963	8,216	12,282	15,392	8,188	9,735	11,345	12,829	14,227

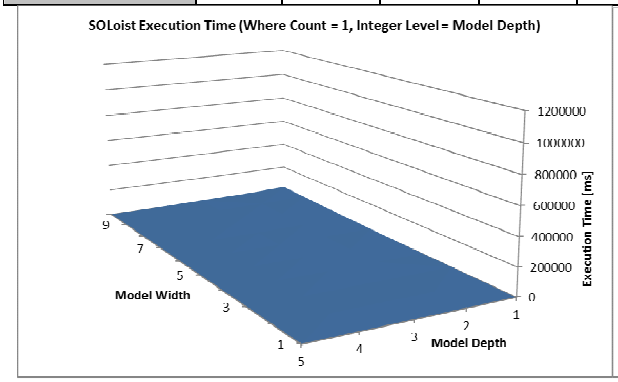


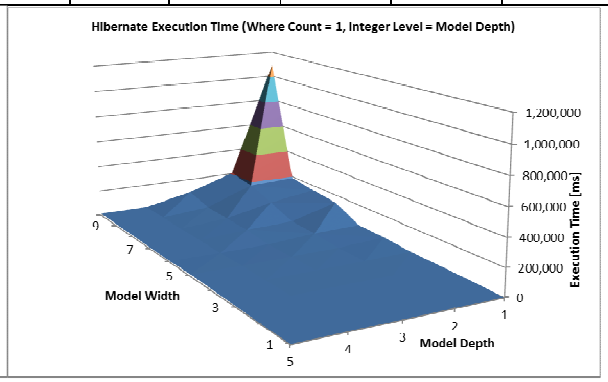


Execution Time (Where Count = 1, Integer Level = Model Depth, Data Size =10^7, DB = Oracle)

SOLoist	Execution Time (Where Count = 1, Integer Level = Model Depth)									
Data Size = 10^7	Model Width									
Model Depth	1	2	3	4	5	6	7	8	9	10
1	31	1,314	2,465	2,564	3,453	6,907	10,199	12,777	19,063	23,876
2	15	1,392	1,626	2,095	2,735	3,391	7,580	9,376	12,110	13,579
3	15	1,032	1,126	1,520	1,955	2,470	3,039	6,746	7,829	9,829
4	15	594	735	1,017	1,314	1,616	1,938	2,236	2,579	2,877
5	0	109	231	469	610	781	921	1,064	1,215	1,220

Hibernate	Execution Time (Where Count = 1, Integer Level = Model Depth)									
Data Size = 10^7	Model Width									
Model Depth	1	2	3	4	5	6	7	8	9	10
1	1,629	10,468	15,360	19,346	35,746	41,923	139,150	160,283	188,516	1,092,189
2	1,305	6,339	13,368	14,002	16,548	34,345	41,408	110,377	173,891	207,298
3	912	5,732	11,881	12,025	13,236	9,636	31,909	35,032	43,376	90,266
4	659	4,789	10,667	10,298	11,095	7,048	8,657	10,318	12,297	15,360
5	480	4,538	7,968	12,263	14,782	8,470	10,158	11,908	13,595	15,204



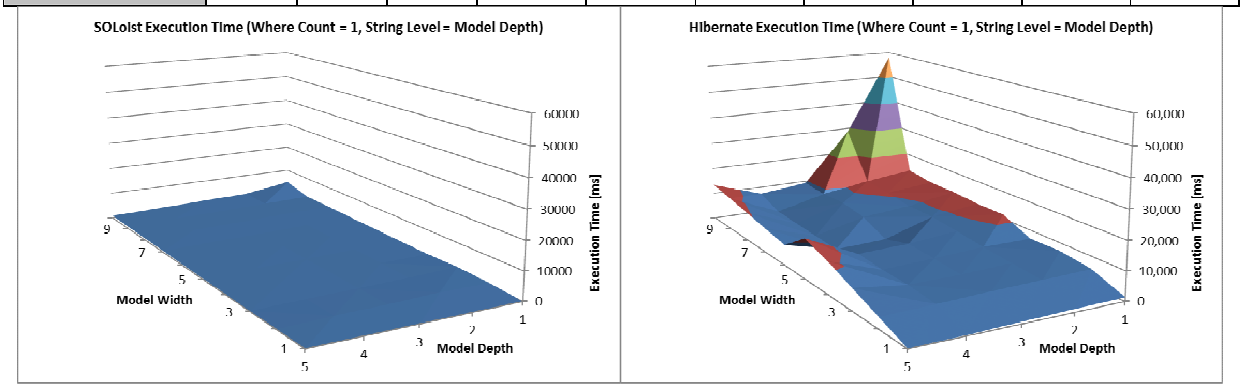


Execution Time (Where Count = 1, String Level = Model Depth, Data Size =10^7, DB = Oracle)

SOLoist	Execution Time (Where Count = 1, String Level = Model Depth)									
Data Size = 10^7	Model Width									
Model Depth	1	2	3	4	5	6	7	8	9	10
1	0	1,000	1,469	1,251	1,423	1,688	1,876	2,173	2,283	4,782
2	15	1,157	1,016	1,173	1,313	1,486	1,642	1,891	2,000	2,173

3	15	720	703	844	937	1,079	1,173	1,408	1,532	1,799
4	0	594	500	656	734	812	906	1,090	1,205	1,283
5	15	109	157	265	328	390	516	625	734	859

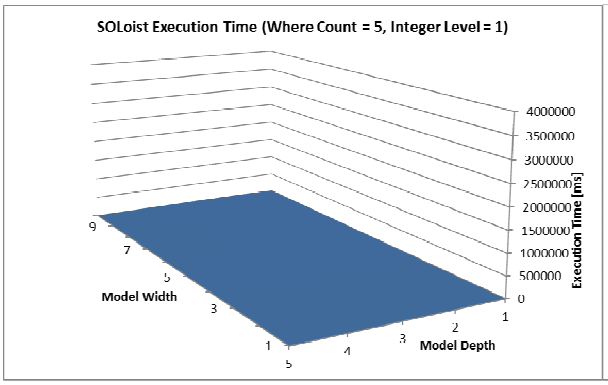
Hibernate	Execution Time (Where Count = 1, String Level = Model Depth)									
Data Size = 10^7	Model Width									
Model Depth	1	2	3	4	5	6	7	8	9	10
1	1,386	5,706	7,159	6,921	11,506	11,450	12,091	12,142	13,486	57,892
2	1,088	5,305	6,081	6,938	8,173	9,189	9,219	10,376	11,360	28,736
3	724	4,908	5,360	5,965	6,453	3,548	7,532	8,922	8,517	9,344
4	280	4,356	5,251	5,249	5,892	2,798	3,204	3,646	4,110	7,236
5	156	3,888	7,568	11,296	14,282	7,407	8,907	10,501	11,985	13,408

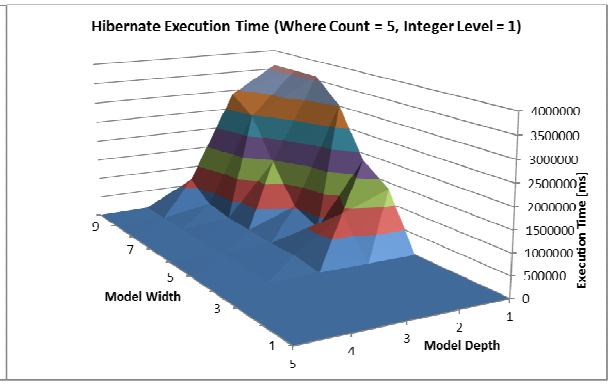


Execution Time (Where Count = 5, Integer Level = 1, Data Size =10^7, DB = Oracle)

SOLoist	Execution Time (Where Count = 5, Integer Level = 1)									
Data Size = 10^7	Model Width									
Model Depth	1	2	3	4	5	6	7	8	9	10
1					5,174	6,251	8,736	10,533	10,469	11,517
2					5,923	9,360	10,080	17,829	20,611	23,423
3					9,032	12,267	13,970	18,251	17,767	21,751
4					7,626	10,032	11,377	12,080	13,204	15,126
5					4,673	4,844	4,970	5,173	5,703	5,875

Hibernate	Execution Time (Where Count = 5, Integer Level = 1)									
Data Size = 10^7	Model Width									
Model Depth	1	2	3	4	5	6	7	8	9	10
1					1,337,751	1,824,938	3,018,251	3,600,000	3,600,000	3,600,000
2					916,532	275,689	539,813	1,460,831	2,475,876	2,897,282
3					90,595	73,205	108,204	132,657	238,392	648,423
4					13,845	14,220	13,594	14,689	33,157	35,471
5					15,063	4,987	5,047	5,241	5,361	15,032

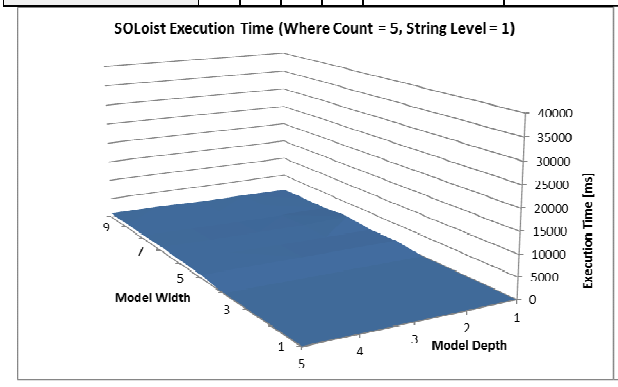


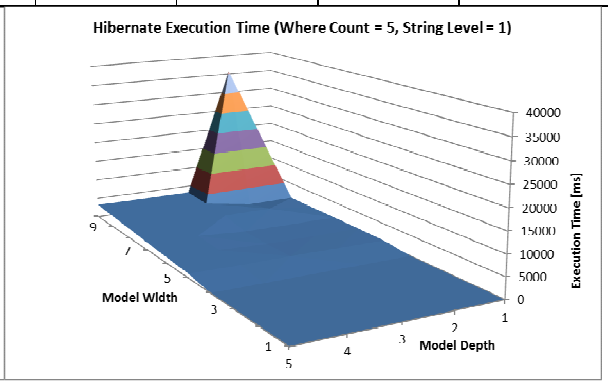


Execution Time (Where Count = 5, String Level = 1, Data Size =10^7, DB = Oracle)

SOLoist Data Size = 10^7	Execution Time (Where Count = 5, String Level = 1) Model Width									
Model Depth	1	2	3	4	5	6	7	8	9	10
1					765	610	1,371	828	844	688
2					767	625	812	687	750	756
3					750	641	843	687	782	777
4					765	772	890	766	796	1,006
5					719	859	1,017	907	938	953

Hibernate Data Size = 10^7	Execution Time (Where Count = 5, String Level = 1) Model Width									
Model Depth	1	2	3	4	5	6	7	8	9	10
1					781	861	907	875	906	906
2					845	797	927	2,344	860	35,751
3					938	823	953	2,298	2,751	3,251
4					891	1,298	1,766	2,235	2,720	3,188
5					1,028	1,423	1,845	2,345	2,782	3,267



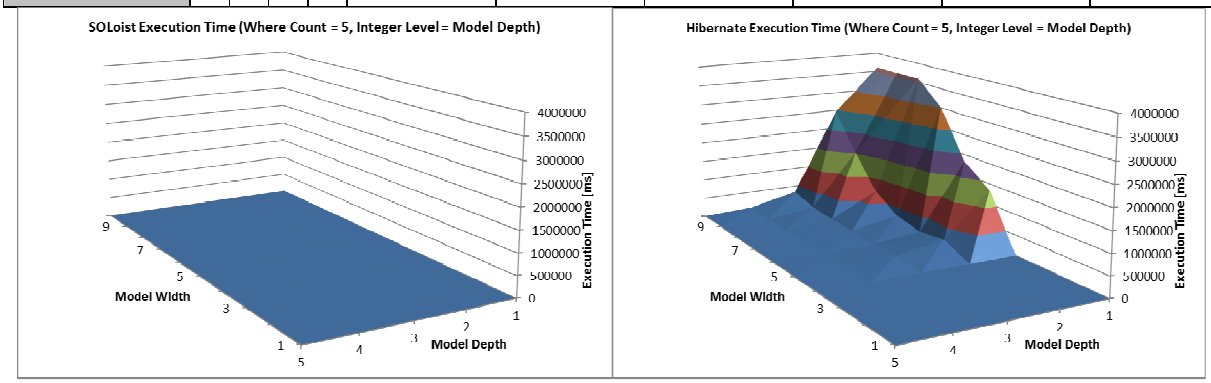


Execution Time (Where Count = 5, Integer Level = Model Depth, Data Size =10^7, DB = Oracle)

SOLoist Data Size = 10^7	Execution Time (Where Count = 5, Integer Level = Model Depth) Model Width									
Integer Level	1	2	3	4	5	6	7	8	9	10
1					5,174	6,251	8,736	10,533	10,469	11,517
2					4,907	5,173	7,626	8,251	8,392	9,219
3					3,595	4,563	3,813	4,548	4,330	4,205

4					2,110	2,142	2,532	2,408	2,445	2,923
5					500	532	583	641	687	735

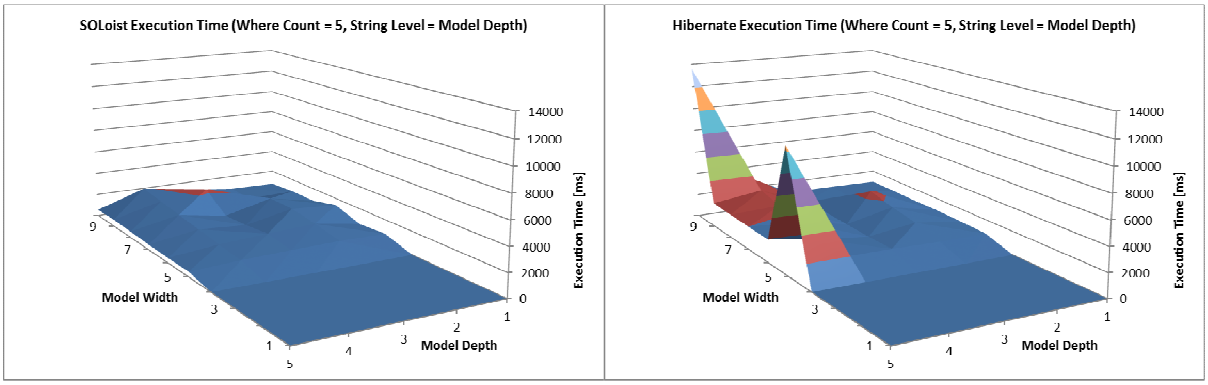
Hibernate	Execution Time (Where Count = 5, Integer Level = Model Depth)									
Data Size = 10^7	Model Width									
Integer Level	1	2	3	4	5	6	7	8	9	10
1					1,337,751	1,824,938	3,018,251	3,600,000	3,600,000	3,600,000
2					287,674	191,938	317,438	704,093	1,531,517	2,466,844
3					93,579	59,548	100,970	126,594	171,407	340,704
4					9,673	9,938	10,360	12,485	12,314	65,813
5					15,601	6,282	14,314	9,016	10,454	14,657



Execution Time (Where Count = 5, String Level = Model Depth, Data Size =10^7, DB = Oracle)

SOLoist	Execution Time (Where Count = 5, String Level = Model Depth)									
Data Size = 10^7	Model Width									
String Level	1	2	3	4	5	6	7	8	9	10
1					765	610	1,371	828	844	688
2					829	1,293	1,657	1,673	734	766
3					750	1,095	813	672	2,220	891
4					637	843	1,100	1,382	1,672	1,954
5					437	421	453	531	593	547

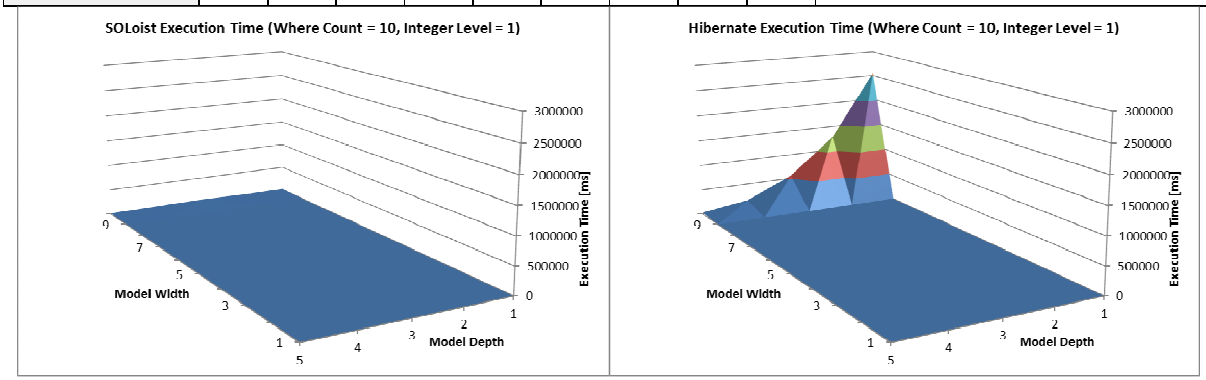
Hibernate	Execution Time (Where Count = 5, String Level = Model Depth)									
Data Size = 10^7	Model Width									
String Level	1	2	3	4	5	6	7	8	9	10
1					781	861	907	875	906	906
2					767	1,391	1,792	2,336	875	970
3					859	828	890	970	1,005	1,017
4					860	1,283	1,767	2,236	2,705	3,189
5					10,454	1,960	1,997	2,048	2,110	13,611



Execution Time (Where Count = 10, Integer Level = 1, Data Size =10^7, DB = Oracle)

SOLoist Data Size = 10^7	Execution Time (Where Count = 10, Integer Level = 1)									
	Model Width									
Model Depth	1	2	3	4	5	6	7	8	9	10
1										15,376
2										16,688
3										33,720
4										23,298
5										17,157

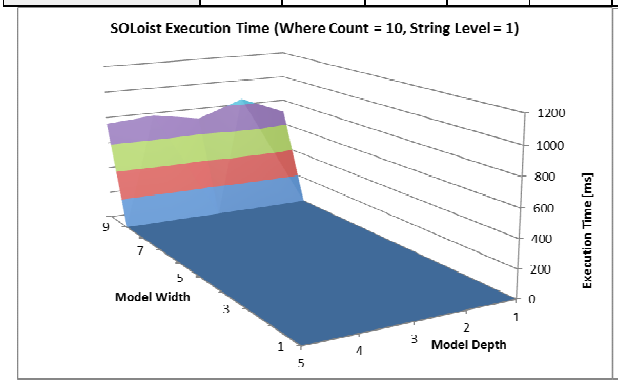
Hibernate Data Size = 10^7	Execution Time (Where Count = 10, Integer Level = 1)									
	Model Width									
Model Depth	1	2	3	4	5	6	7	8	9	10
1										2,550,281
2										1,267,595
3										556,230
4										156,893
5										31,251

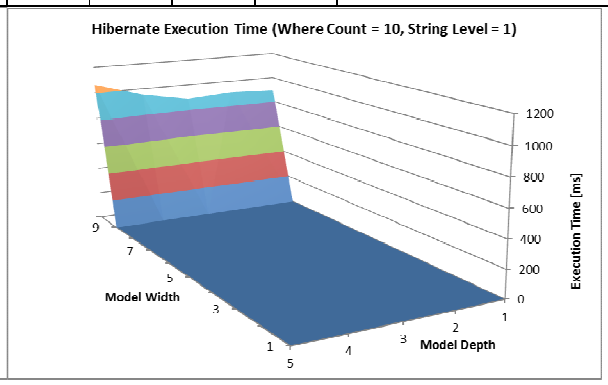


Execution Time (Where Count = 10, String Level = 1, Data Size =10^7, DB = Oracle)

SOLoist Data Size = 10^7	Execution Time (Where Count = 10, String Level = 1) Model Width									
Model Depth	1	2	3	4	5	6	7	8	9	10
1										703
2										844
3										718
4										781
5										750

Hibernate Data Size = 10^7	Execution Time (Where Count = 10, String Level = 1) Model Width									
Model Depth	1	2	3	4	5	6	7	8	9	10
1										891
2										906
3										886
4										954
5										1,063



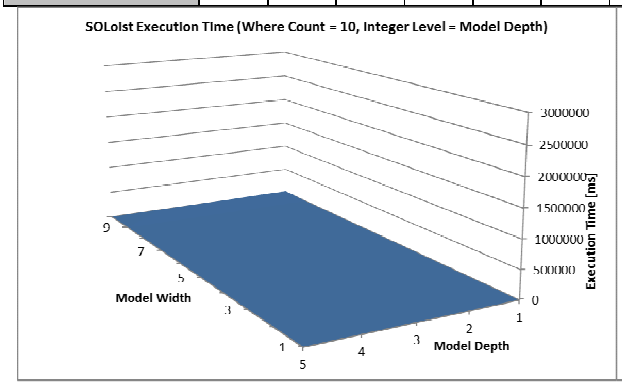


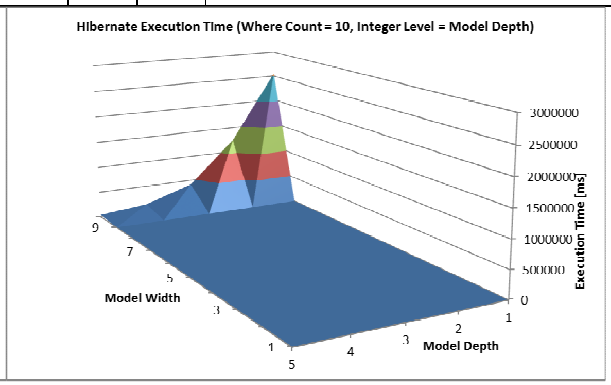
Execution Time (Where Count = 10, Integer Level =Model Depth, Data Size =10^7, DB = Oracle)

SOLoist Data Size = 10^7	Execution Time (Where Count = 10, Integer Level = Model Depth) Model Width									
Integer Level	1	2	3	4	5	6	7	8	9	10
1										15,376
2										11,970
3										6,439
4										4,110
5										750

Hibernate Data Size = 10^7	Execution Time (Where Count = 10, Integer Level = Model Depth) Model Width									
Integer Level	1	2	3	4	5	6	7	8	9	10
1										2,550,281
2										1,235,704
3										412,160

4										124,704
5										40,110

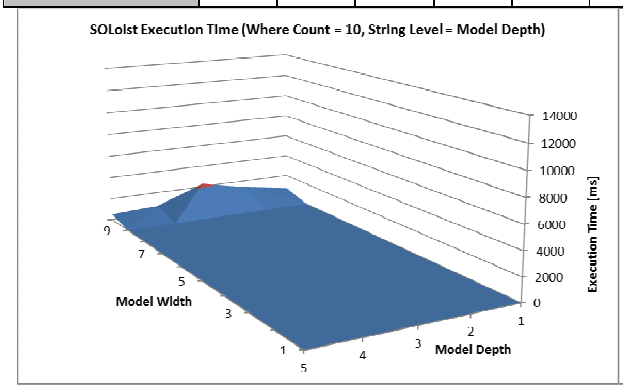


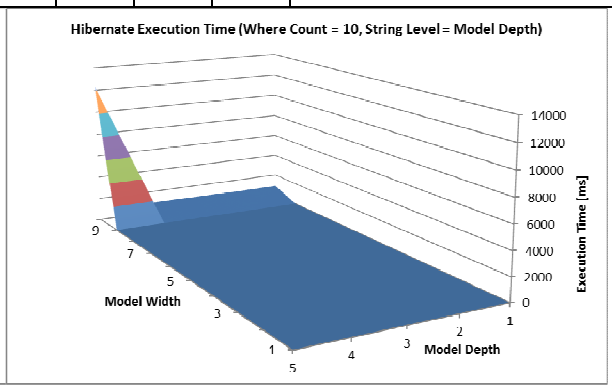


Execution Time (Where Count = 10, String Level = Model Depth, Data Size =10^7, DB = Oracle)

SOLoist Data Size = 10^7	Execution Time (Where Count = 10, String Level = Model Depth) Model Width									
String Level	1	2	3	4	5	6	7	8	9	10
1										703
2										1,371
3										2,266
4										718
5										516

Hibernate Data Size = 10^7	Execution Time (Where Count = 10, String Level = Model Depth) Model Width									
String Level	1	2	3	4	5	6	7	8	9	10
1										891
2										876
3										890
4										914
5										12,376



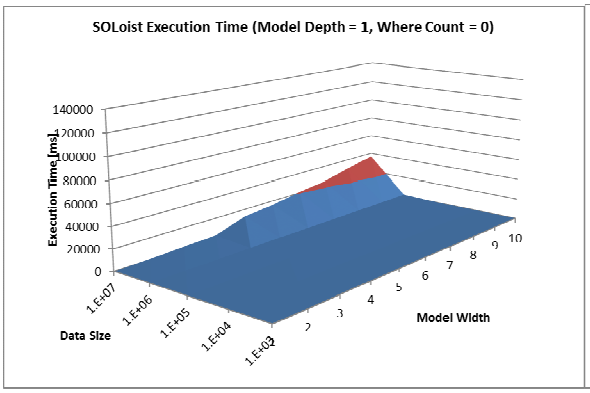


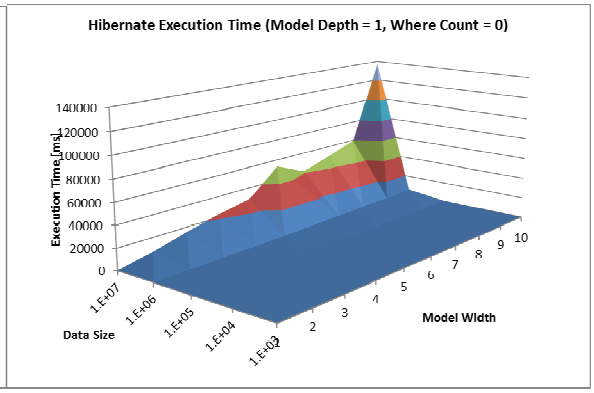
Scalability: Execution Time vs. Database Size

In the following set of results, we are presenting the dependency of the execution time on the volume of the data in the database for SOLoist and Hibernate, for different query complexity. All the results are for Oracle.

Execution Time (Model Depth = 1, Where Count = 0, DB = Oracle)

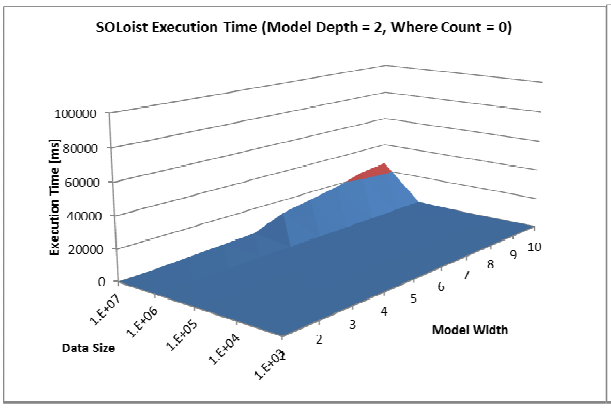
SOLoist	Execution Time (Model Depth = 1, Where Count = 0)					Hibernate	Execution Time (Model Depth = 1, Where Count = 0)				
	Data Size						Data Size				
Model Width	1.E+03	1.E+04	1.E+05	1.E+06	1.E+07	Model Width	1.E+03	1.E+04	1.E+05	1.E+06	1.E+07
1	2	1	2	6	47	1	1	1	2	8	65
2	0	2	7	62	637	2	4	5	26	347	7,128
3	2	3	13	137	1,989	3	5	7	51	698	15,562
4	4	5	24	396	3,132	4	7	11	77	987	22,924
5	6	8	30	522	12,907	5	7	16	131	1,766	28,670
6	7	9	42	606	15,829	6	6	21	152	2,083	53,611
7	8	11	52	702	20,953	7	7	25	210	2,811	41,297
8	10	14	61	802	24,001	8	7	29	275	3,546	50,110
9	12	16	72	1,046	30,422	9	7	32	339	4,321	59,454
10	12	20	81	1,055	36,516	10	7	39	404	5,483	137,110

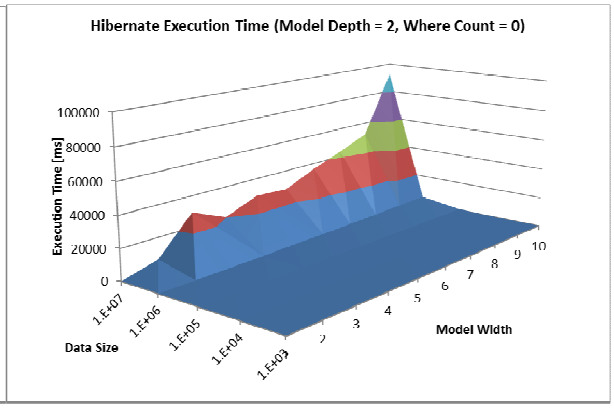




Execution Time (Model Depth = 2, Where Count = 0, DB = Oracle)

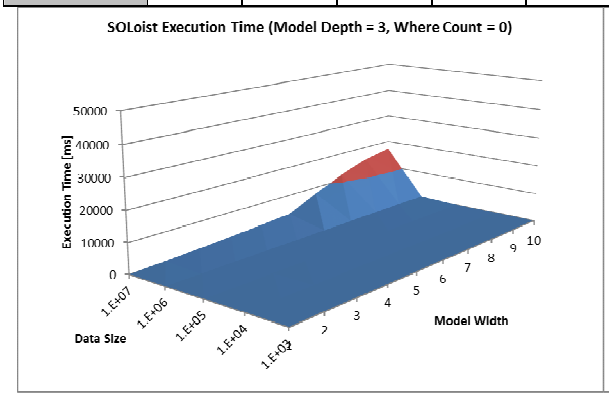
SOLoist	Execution Time (Model Depth = 2, Where Count = 0)					Hibernate	Execution Time (Model Depth = 2, Where Count = 0)				
	Data Size						Data Size				
Model Width	1.E+03	1.E+04	1.E+05	1.E+06	1.E+07	Model Width	1.E+03	1.E+04	1.E+05	1.E+06	1.E+07
1	1	1	2	6	46	1	1	1	2	7	52
2	1	1	5	52	500	2	3	4	21	309	6,352
3	3	2	11	113	1,429	3	4	8	41	593	29,528
4	6	5	18	313	2,313	4	7	11	60	905	20,185
5	5	7	25	393	3,235	5	7	14	102	1,497	28,517
6	7	9	34	482	11,001	6	7	17	119	1,773	27,313
7	7	10	41	554	14,657	7	7	21	166	2,318	36,204
8	9	12	49	638	18,157	8	8	23	217	2,899	43,595
9	12	14	57	733	23,392	9	7	27	269	3,556	51,782
10	13	16	65	856	25,704	10	7	31	320	4,187	92,407

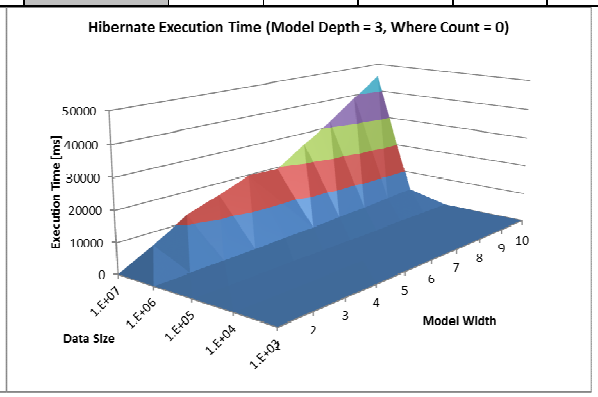




Execution Time (Model Depth = 3, Where Count = 0, DB = Oracle)

SOLoist	Execution Time (Model Depth = 3, Where Count = 0)					Hibernate	Execution Time (Model Depth = 3, Where Count = 0)				
	Data Size						Data Size				
Model Width	1.E+03	1.E+04	1.E+05	1.E+06	1.E+07	Model Width	1.E+03	1.E+04	1.E+05	1.E+06	1.E+07
1	1	1	1	4	31	1	1	2	1	4	39
2	2	1	4	37	500	2	3	4	16	255	5,738
3	3	3	8	84	1,063	3	4	7	32	494	12,357
4	4	4	15	222	1,731	4	5	9	46	727	15,994
5	6	6	20	297	2,434	5	6	12	76	1,165	19,844
6	6	7	31	375	3,157	6	6	14	89	1,364	19,470
7	8	9	33	408	7,748	7	7	16	122	1,778	25,923
8	10	11	68	464	11,970	8	6	20	162	2,218	31,955
9	11	13	46	561	15,220	9	7	22	201	2,806	38,972
10	12	16	56	591	17,126	10	7	24	238	3,445	45,485

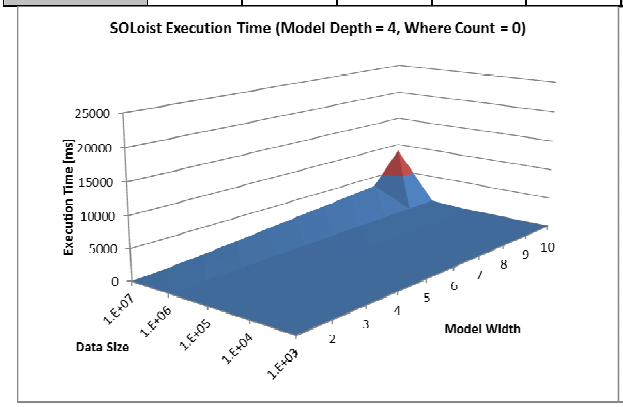


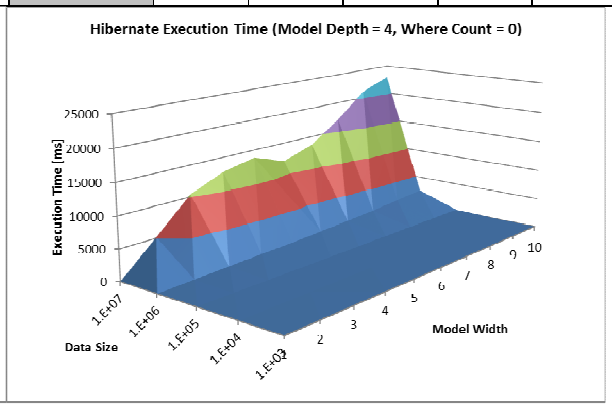


Execution Time (Model Depth = 4, Where Count = 0, DB = Oracle)

SOLoist	Execution Time (Model Depth = 4, Where Count = 0)					Hibernate	Execution Time (Model Depth = 4, Where Count = 0)				
	Data Size						Data Size				
Model Width	1.E+03	1.E+04	1.E+05	1.E+06	1.E+07	Model Width	1.E+03	1.E+04	1.E+05	1.E+06	1.E+07
1	1	1	1	3	15	1	1	1	2	4	26
2	1	1	3	29	296	2	2	4	11	216	5,153
3	3	3	6	53	656	3	3	6	22	412	10,376
4	4	5	10	145	1,079	4	5	7	31	618	12,654
5	5	6	15	180	1,532	5	7	10	50	735	13,720
6	6	7	19	222	1,980	6	6	12	61	1,019	11,876

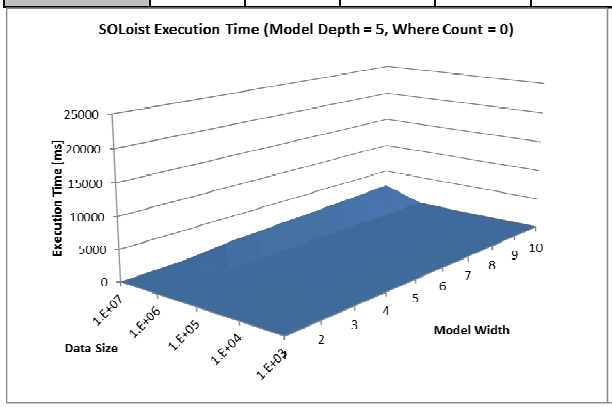
7	7	8	23	297	2,407	7	6	13	80	1,298	13,544
8	9	11	28	298	2,860	8	6	15	104	1,587	16,970
9	10	12	32	348	3,329	9	7	17	128	2,034	21,095
10	13	15	39	390	8,876	10	6	19	152	2,438	22,954

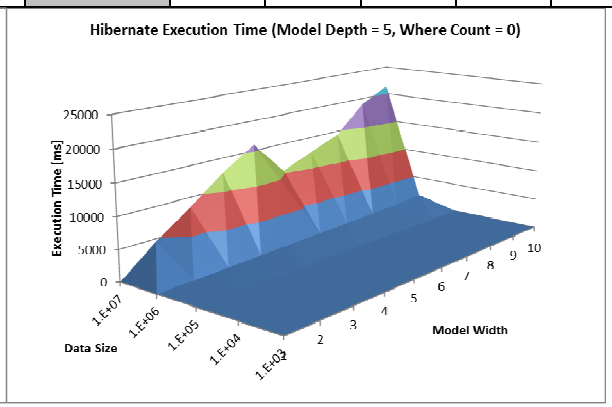




Execution Time (Model Depth = 5, Where Count = 0, DB = Oracle)

SOLoist	Execution Time (Model Depth = 5, Where Count = 0)					Hibernate	Execution Time (Model Depth = 5, Where Count = 0)				
	Data Size						Data Size				
Model Width	1.E+03	1.E+04	1.E+05	1.E+06	1.E+07	Model Width	1.E+03	1.E+04	1.E+05	1.E+06	1.E+07
1	1	1	1	2	0	1	1	1	1	2	14
2	1	2	3	14	132	2	3	3	7	170	4,573
3	3	2	4	28	281	3	4	4	13	326	8,441
4	5	3	6	43	828	4	6	7	18	492	12,623
5	6	5	9	61	1,048	5	7	8	24	645	16,001
6	6	7	13	83	1,256	6	9	10	36	801	10,329
7	8	7	15	97	1,469	7	10	12	36	954	12,410
8	9	10	19	135	1,689	8	11	13	43	1,135	14,344
9	10	11	21	134	1,922	9	13	14	51	1,313	18,955
10	12	12	24	156	2,137	10	13	16	58	1,639	21,329

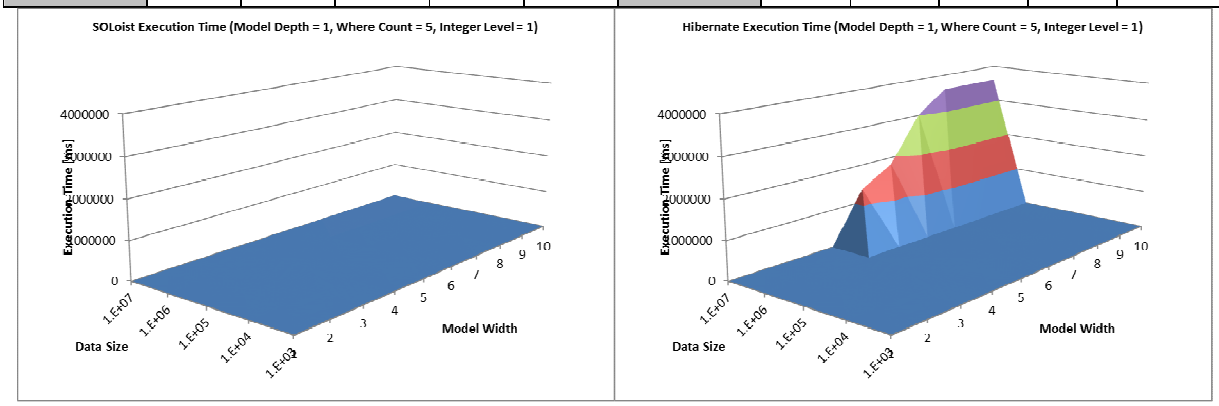




Execution Time (Model Depth = 1, Where Count = 5, Integer Level = 1)

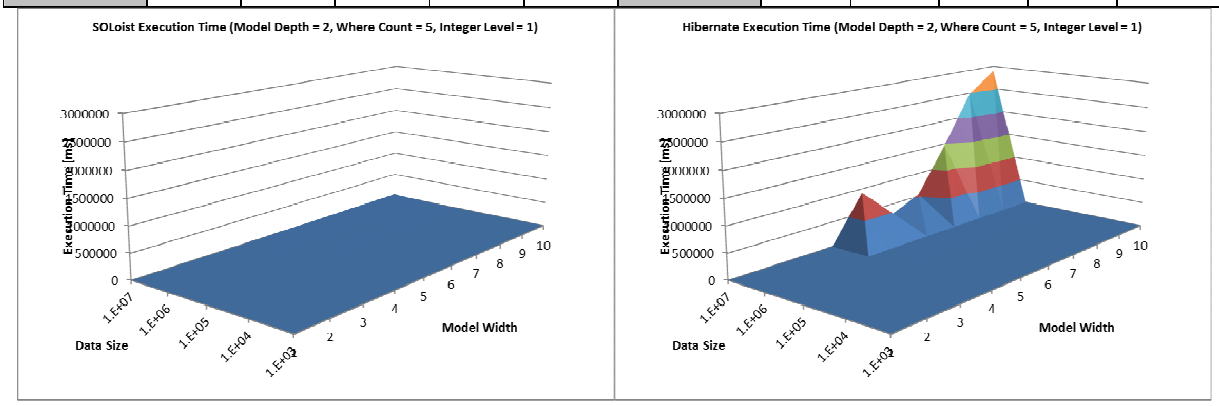
SOLoist	Execution Time (Model Depth = 1, Where Count = 5, Integer Level = 1)					Hibernate	Execution Time (Model Depth = 1, Where Count = 5, Integer Level = 1)				
	Data Size						Data Size				
Model Width	1.E+03	1.E+04	1.E+05	1.E+06	1.E+07	Model Width	1.E+03	1.E+04	1.E+05	1.E+06	1.E+07
1						1					
2						2					

3						3					
4						4					
5	12	14	32	233	5,174	5	3	11	93	1,148	1,337,751
6	12	14	34	258	6,251	6	5	15	101	1,397	1,824,938
7	14	16	38	279	8,736	7	3	17	122	1,506	3,018,251
8	15	18	42	311	10,533	8	4	25	123	1,617	3,600,000
9	16	22	47	334	10,469	9	5	34	131	1,734	3,600,000
10	18	24	51	362	11,517	10	5	26	272	1,839	3,600,000



Execution Time (Model Depth = 2, Where Count = 5, Integer Level = 1)

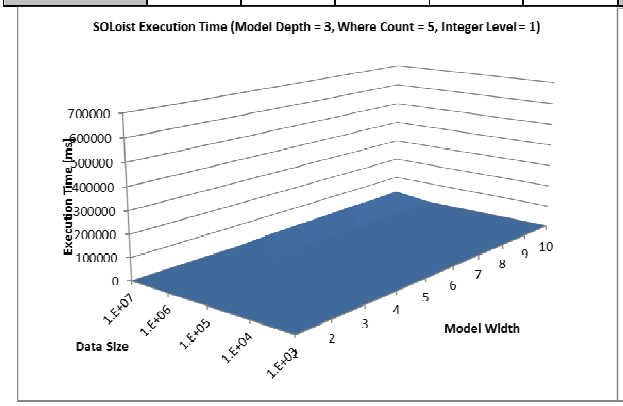
SOLoist	Execution Time (Model Depth = 2, Where Count = 5, Integer Level = 1)					Hibernate	Execution Time (Model Depth = 2, Where Count = 5, Integer Level = 1)				
	Data Size						Data Size				
Model Width	1.E+03	1.E+04	1.E+05	1.E+06	1.E+07	Model Width	1.E+03	1.E+04	1.E+05	1.E+06	1.E+07
1						1					
2						2					
3						3					
4						4					
5	13	14	34	246	5,923	5	4	9	74	984	916,532
6	13	16	43	334	9,360	6	4	14	81	1,172	275,689
7	14	18	48	326	10,080	7	3	14	97	1,273	539,813
8	12	25	60	394	17,829	8	5	20	98	1,655	1,460,831
9	15	22	60	420	20,611	9	5	24	102	1,499	2,475,876
10	18	26	63	464	23,423	10	5	12	182	1,698	2,897,282

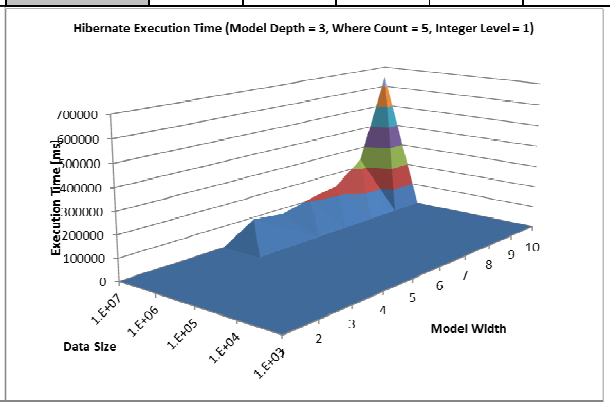


Execution Time (Model Depth = 3, Where Count = 5, Integer Level = 1)

SOLoist	Execution Time (Model Depth = 3,	Hibernate	Execution Time (Model Depth = 3,

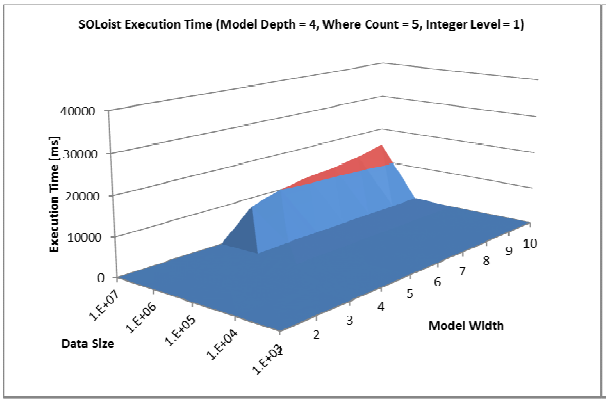
	Where Count = 5, Integer Level = 1)						Where Count = 5, Integer Level = 1)				
	Data Size						Data Size				
Model Width	1.E+03	1.E+04	1.E+05	1.E+06	1.E+07	Model Width	1.E+03	1.E+04	1.E+05	1.E+06	1.E+07
1						1					
2						2					
3						3					
4						4					
5	11	13	34	263	9,032	5	3	7	57	750	90,595
6	14	17	39	359	12,267	6	5	12	62	898	73,205
7	12	17	42	313	13,970	7	3	13	73	971	108,204
8	13	19	57	336	18,251	8	4	17	73	1,031	132,657
9	16	21	51	370	17,767	9	5	19	79	1,182	238,392
10	18	25	56	441	21,751	10	5	11	138	1,289	648,423

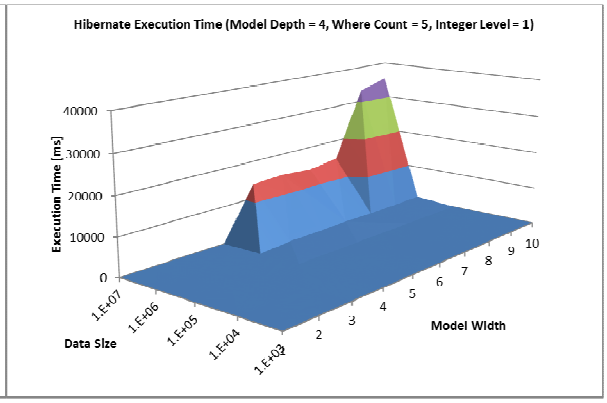




Execution Time (Model Depth = 4, Where Count = 5, Integer Level = 1)

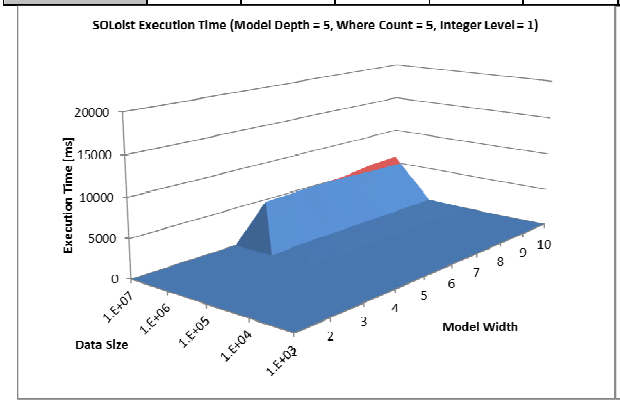
SOLoist	Execution Time (Model Depth = 4, Where Count = 5, Integer Level = 1)					Hibernate	Execution Time (Model Depth = 4, Where Count = 5, Integer Level = 1)				
	Data Size						Data Size				
Model Width	1.E+03	1.E+04	1.E+05	1.E+06	1.E+07	Model Width	1.E+03	1.E+04	1.E+05	1.E+06	1.E+07
1						1					
2						2					
3						3					
4						4					
5	14	14	32	249	7,626	5	4	7	39	556	13,845
6	12	16	37	282	10,032	6	3	9	38	606	14,220
7	12	17	42	305	11,377	7	4	10	46	788	13,594
8	14	20	44	327	12,080	8	4	10	76	617	14,689
9	17	20	45	327	13,204	9	4	11	57	715	33,157
10	17	22	52	335	15,126	10	4	9	65	701	35,471

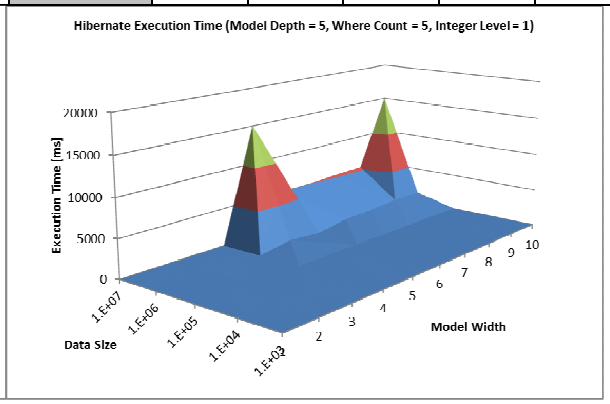




Execution Time (Model Depth = 5, Where Count = 5, Integer Level = 1)

SOLoist	Execution Time (Model Depth = 5, Where Count = 5, Integer Level = 1)					Hibernate	Execution Time (Model Depth = 5, Where Count = 5, Integer Level = 1)				
	Data Size						Data Size				
Model Width	1.E+03	1.E+04	1.E+05	1.E+06	1.E+07	Model Width	1.E+03	1.E+04	1.E+05	1.E+06	1.E+07
1						1					
2						2					
3						3					
4						4					
5	9	14	26	165	4,673	5	3	6	29	674	15,063
6	12	13	28	174	4,844	6	4	15	30	339	4,987
7	12	17	31	169	4,970	7	3	18	32	977	5,047
8	14	18	33	187	5,173	8	3	19	47	1,102	5,241
9	10	20	35	191	5,703	9	3	20	37	1,367	5,361
10	11	21	40	208	5,875	10	3	17	38	1,361	15,032





Scalability: SOLoist vs. Hibernate Performance on MySQL

On MySQL, Hibernate HQL performed extremely poorly. It was able to execute (in reasonable time of the 15 minute limit) only a very small number of the narrower and shallower queries even for the smallest database size of 1K! The larger data sets (of 1M and more) were even not worth trying. This is why we performed just a very small set of benchmarks for Hibernate on MySQL. SOLoist, on the other hand, performed quite well on MySQL and completed the full suite of queries for all database sizes!

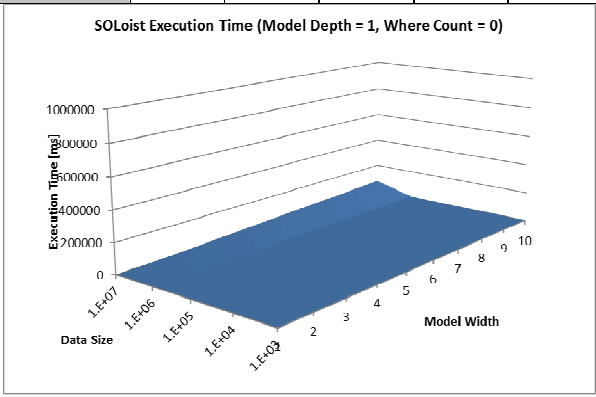
For both technologies, the execution time was limited to 15 minutes. SOLoist has completed all the queries well before that time has expired, while Hibernate missed the deadline many times. The results of 90,000 ms in the tables indicate an expiration of the 15 min time limit.

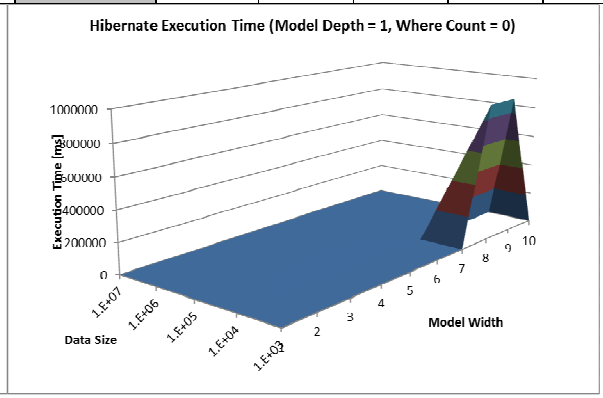
In addition, MySQL has a limit of about 60 joins in a query, so it was not able even to compile all the queries in the benchmark suite.

Here are the comparative results of SOLoist vs. Hibernate (where available) for different database sizes on MySQL and some selected queries.

Execution Time (Model Depth = 1, Where Count = 0)

SOLoist	Execution Time (Model Depth = 1, Where Count = 0)					Hibernate	Execution Time (Model Depth = 1, Where Count = 0)				
	Data Size						Data Size				
Model Width	1.E+03	1.E+04	1.E+05	1.E+06	1.E+07	Model Width	1.E+03	1.E+04	1.E+05	1.E+06	1.E+07
1	0	0	0	32	438	1	0	0	31		
2	0	0	31	455	4,654	2	0	0	78		
3	0	15	94	1,105	12,190	3	0	15	172		
4	0	16	172	1,739	21,099	4	0	31	266		
5	0	15	218	2,423	31,316	5	0	31	359		
6	0	31	281	3,126	40,411	6	0	62	468		
7	0	46	328	3,833	48,552	7	141				
8	0	47	422	4,557	57,302	8	900,000				
9	0	47	484	5,288	65,689	9	900,000				
10	0	47	562	6,074	74,719	10	15				

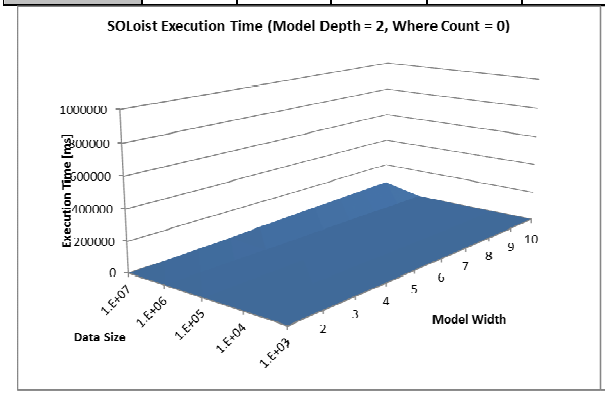


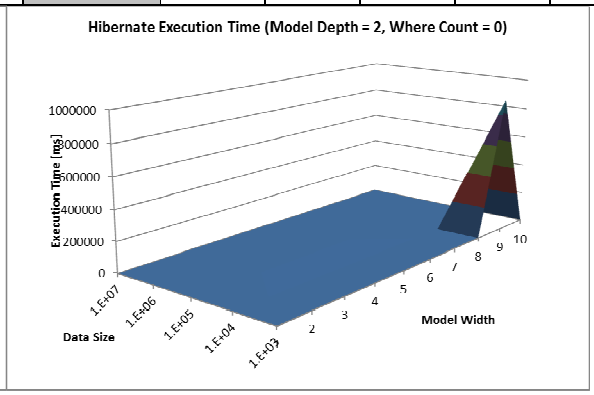


Execution Time (Model Depth = 2, Where Count = 0)

SOLoist	Execution Time (Model Depth = 2, Where Count = 0)					Hibernate	Execution Time (Model Depth = 2, Where Count = 0)				
	Data Size						Data Size				
Model Width	1.E+03	1.E+04	1.E+05	1.E+06	1.E+07	Model Width	1.E+03	1.E+04	1.E+05	1.E+06	1.E+07
1	0	0	0	31	344	1	0	0	32		
2	0	0	31	235	3,908	2	0	15	94		
3	0	16	62	734	10,392	3	0	15	203		
4	0	15	125	1,067	17,032	4	0	31	297		
5	0	15	172	1,567	25,657	5	16	47	391		
6	0	31	203	2,312	32,970	6	0	47	484		
7	0	31	266	2,583	40,391	7	750				
8	0	31	359	3,131	47,865	8	692				
9	0	31	422	3,677	54,866	9	900,000				

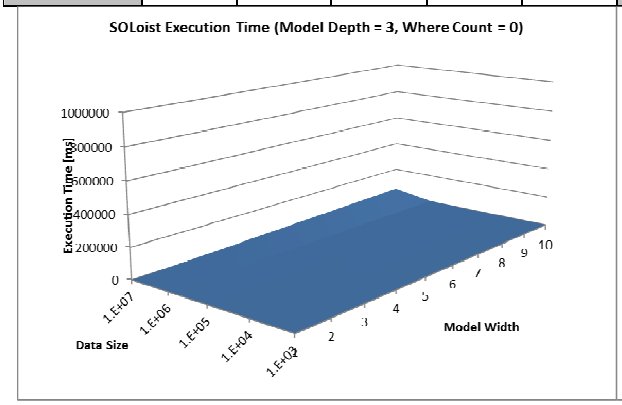
10	0	47	439	4,266	61,376	10	25				

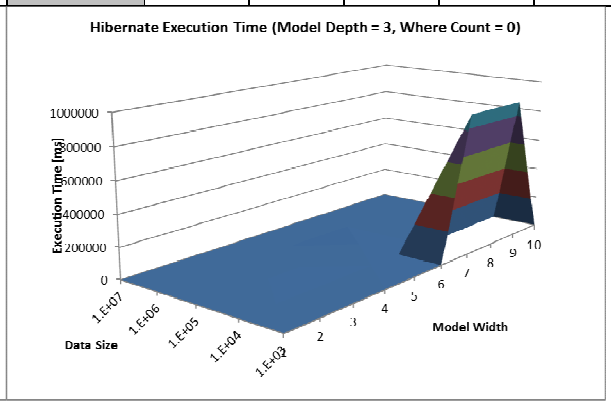




Execution Time (Model Depth = 3, Where Count = 0)

SOLoist	Execution Time (Model Depth = 3, Where Count = 0)					Hibernate	Execution Time (Model Depth = 3, Where Count = 0)				
	Data Size						Data Size				
Model Width	1.E+03	1.E+04	1.E+05	1.E+06	1.E+07	Model Width	1.E+03	1.E+04	1.E+05	1.E+06	1.E+07
1	0	0	0	15	250	1	0	0	31		
2	0	0	16	250	2,974	2	0	15	109		
3	0	0	62	626	8,023	3	0	15	172		
4	0	15	78	1,023	12,974	4	0	31	266		
5	0	16	125	1,439	19,268	5	0	1,298	375		
6	0	16	187	1,860	24,392	6	0	1,505	5,173		
7	0	31	203	2,286	29,992	7	900,000				
8	0	31	266	2,724	35,302	8	900,000				
9	0	31	312	3,157	40,438	9	900,000				
10	0	31	344	3,626	46,021	10	31				

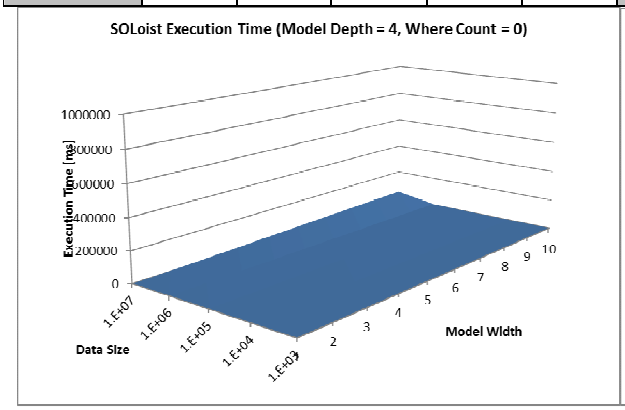


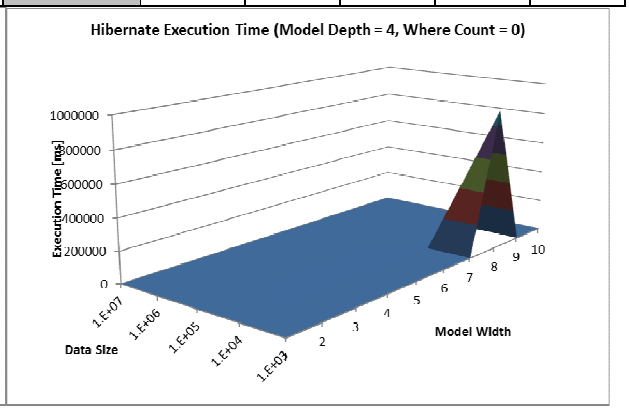


Execution Time (Model Depth = 4, Where Count = 0)

SOLoist	Execution Time (Model Depth = 4, Where Count = 0)					Hibernate	Execution Time (Model Depth = 4, Where Count = 0)				
	Data Size						Data Size				
Model Width	1.E+03	1.E+04	1.E+05	1.E+06	1.E+07	Model Width	1.E+03	1.E+04	1.E+05	1.E+06	1.E+07
1	0	0	0	15	250	1	0	0	31		
2	0	0	16	250	2,974	2	0	15	78		
3	0	0	62	626	8,023	3	0	15	158		
4	0	15	78	1,023	12,974	4	0	15	203		
5	0	16	125	1,439	19,268	5	0	31	266		

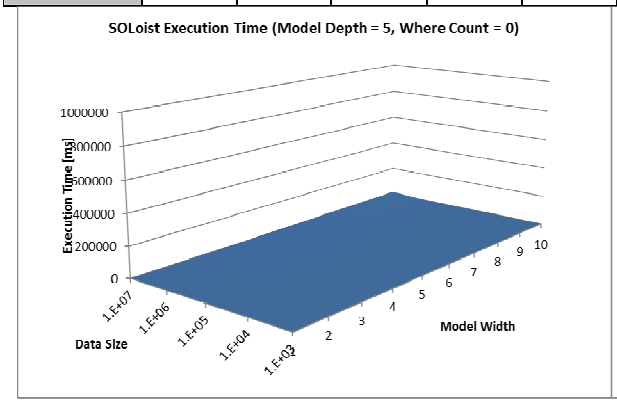
6	0	16	187	1,860	24,392	6	0	31	359		
7	0	31	203	2,286	29,992	7	14				
8	0	31	266	2,724	35,302	8	900,000				
9	0	31	312	3,157	40,438	9	15				
10	0	31	344	3,626	46,021	10	63				

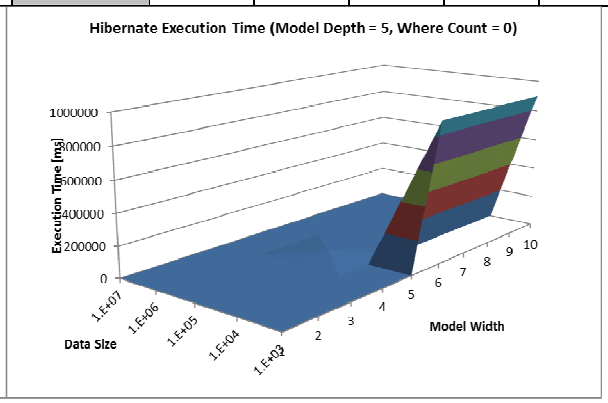




Execution Time (Model Depth = 5, Where Count = 0)

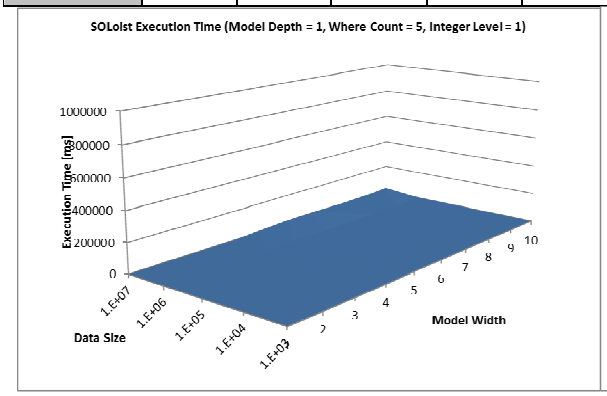
SOLoist	Execution Time (Model Depth = 5, Where Count = 0)					Hibernate	Execution Time (Model Depth = 5, Where Count = 0)				
	Data Size						Data Size				
Model Width	1.E+03	1.E+04	1.E+05	1.E+06	1.E+07	Model Width	1.E+03	1.E+04	1.E+05	1.E+06	1.E+07
1	0	0	0	0	78	1	0	0	16		
2	0	0	0	62	638	2	0	0	46		
3	0	0	15	172	1,786	3	0	0	78		
4	0	0	16	265	3,442	4	0	15	484		
5	0	0	47	375	5,115	5	0	15	29,360		
6	0	0	47	500	6,786	6	900,000	15			
7	0	0	62	641	8,505	7	900,000				
8	0	0	78	766	10,177	8	900,000				
9	0	15	93	909	11,864	9	900,000				
10	0	15	109	1,036	13,552	10	900,000				

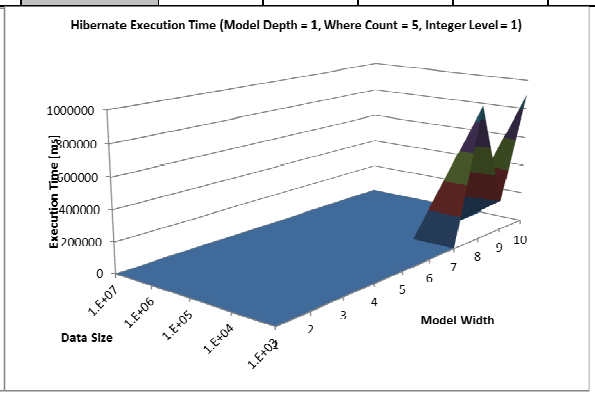




Execution Time (Model Depth = 1, Where Count = 5, Integer Level = 1)

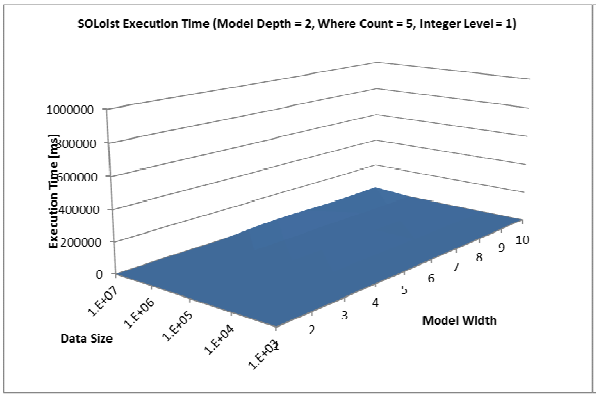
SOLoist	Execution Time (Model Depth = 1, Where Count = 5, Integer Level = 1)					Hibernate	Execution Time (Model Depth = 1, Where Count = 5, Integer Level = 1)				
	Data Size						Data Size				
Model Width	1.E+03	1.E+04	1.E+05	1.E+06	1.E+07	Model Width	1.E+03	1.E+04	1.E+05	1.E+06	1.E+07
1						1					
2						2					
3						3					
4						4					
5	0	0	93	891	11,188	5	16	15	94		
6	0	0	141	1,118	20,095	6	78	31	125		
7	0	15	187	1,203	20,801	7	218				
8	0	15	156	1,297	14,235	8	900,000				
9	0	15	203	1,360	15,348	9	200,219				
10	0	15	187	1,453	24,272	10	900,000				

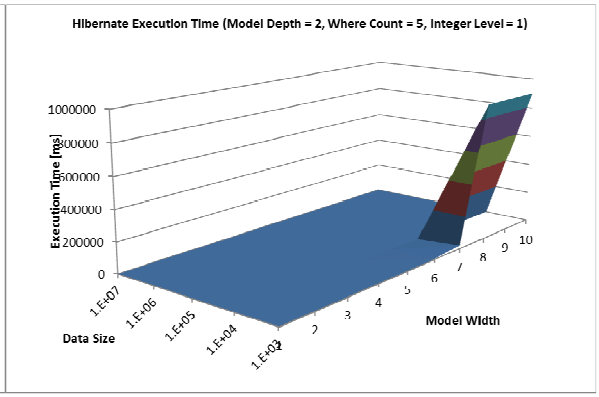




Execution Time (Model Depth = 2, Where Count = 5, Integer Level = 1)

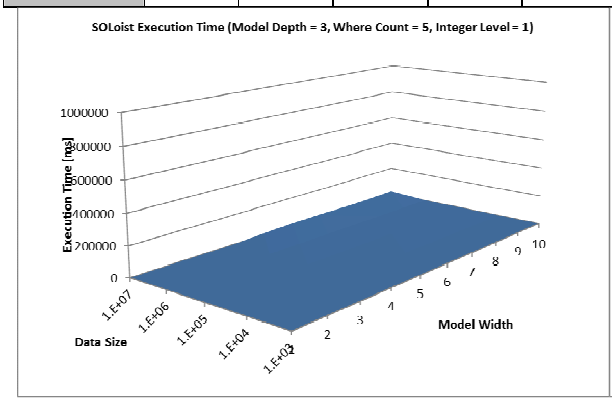
SOLoist	Execution Time (Model Depth = 2, Where Count = 5, Integer Level = 1)					Hibernate	Execution Time (Model Depth = 2, Where Count = 5, Integer Level = 1)				
	Data Size						Data Size				
Model Width	1.E+03	1.E+04	1.E+05	1.E+06	1.E+07	Model Width	1.E+03	1.E+04	1.E+05	1.E+06	1.E+07
1						1					
2						2					
3						3					
4						4					
5	0	0	109	890	10,818	5	1,271	1,282	157		
6	0	0	109	922	19,048	6	4,336	891	296		
7	0	15	110	1,046	19,020	7	22,110				
8	0	15	156	1,126	14,115	8	900,000				
9	0	15	156	1,203	14,911	9	900,000				
10	0	16	141	1,298	22,177	10	900,000				

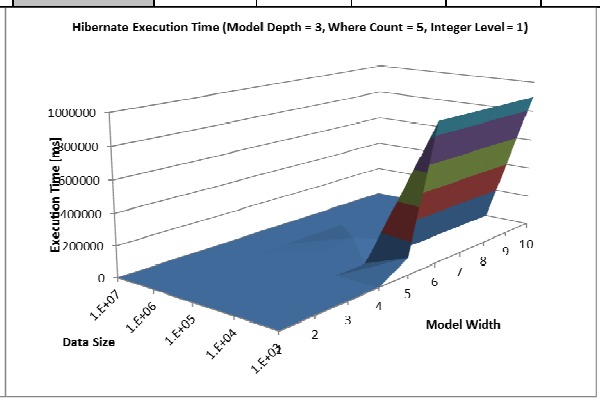




Execution Time (Model Depth = 3, Where Count = 5, Integer Level = 1)

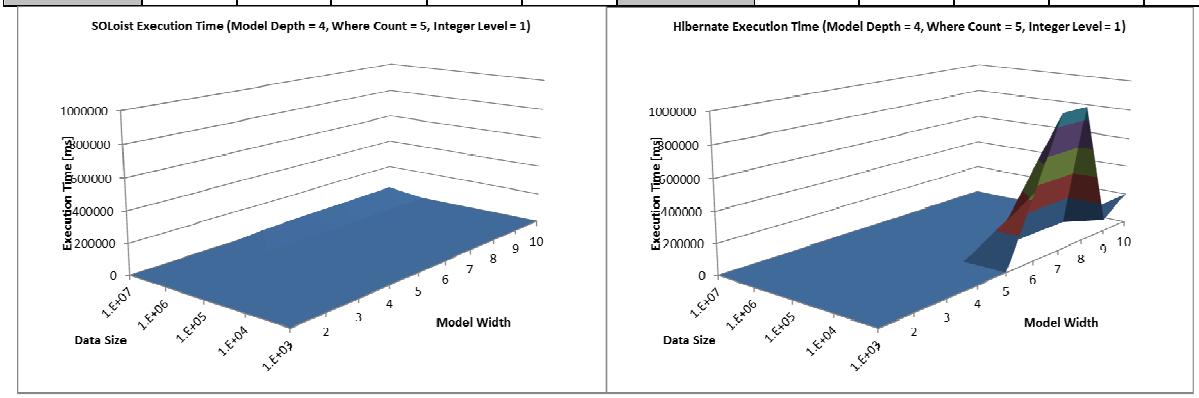
SOLoist	Execution Time (Model Depth = 3, Where Count = 5, Integer Level = 1)					Hibernate	Execution Time (Model Depth = 3, Where Count = 5, Integer Level = 1)				
	Data Size						Data Size				
Model Width	1.E+03	1.E+04	1.E+05	1.E+06	1.E+07	Model Width	1.E+03	1.E+04	1.E+05	1.E+06	1.E+07
1						1					
2						2					
3						3					
4						4					
5	0	0	63	875	9,037	5	108,613	2,146	7,704		
6	0	0	93	984	14,848	6	900,000	6,407	41,911		
7	0	0	109	1,048	15,235	7	900,000				
8	0	0	109	1,141	11,642	8	900,000				
9	0	15	109	1,235	12,161	9	900,000				
10	0	16	110	1,313	17,411	10	900,000				





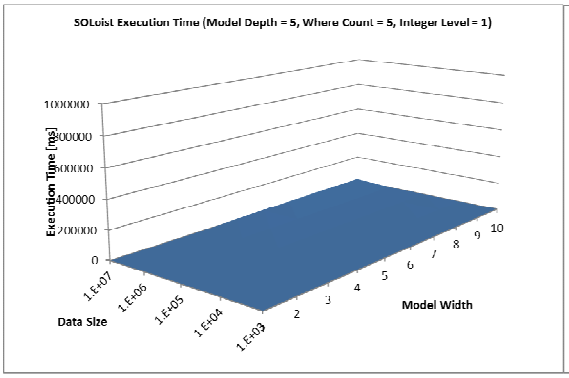
Execution Time (Model Depth = 4, Where Count = 5, Integer Level = 1)

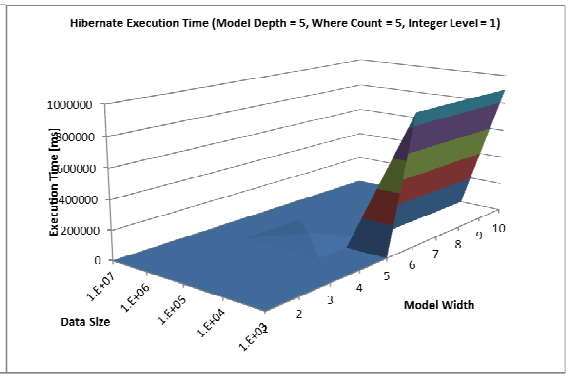
SOLoist	Execution Time (Model Depth = 4, Where Count = 5, Integer Level = 1)					Hibernate	Execution Time (Model Depth = 4, Where Count = 5, Integer Level = 1)				
	Data Size						Data Size				
Model Width	1.E+03	1.E+04	1.E+05	1.E+06	1.E+07	Model Width	1.E+03	1.E+04	1.E+05	1.E+06	1.E+07
1						1					
2						2					
3						3					
4						4					
5	0	16	109	1,000	16,270	5	172	234	125		
6	0	15	141	1,156	19,517	6	440,532	15			
7	0	15	172	1,297	23,188	7	900,000				
8	0	15	187	1,453	26,548	8	900,000				
9	0	31	219	1,625	30,032	9	77,424				
10	0	31	250	1,782	33,844	10	199,506				



Execution Time (Model Depth = 5, Where Count = 5, Integer Level = 1)

SOLoist	Execution Time (Model Depth = 5, Where Count = 5, Integer Level = 1)					Hibernate	Execution Time (Model Depth = 5, Where Count = 5, Integer Level = 1)				
	Data Size						Data Size				
Model Width	1.E+03	1.E+04	1.E+05	1.E+06	1.E+07	Model Width	1.E+03	1.E+04	1.E+05	1.E+06	1.E+07
1						1					
2						2					
3						3					
4						4					
5	0	0	46	503	6,677	5	166	15	37,988		
6	0	16	63	625	8,271	6	900,000	0			
7	0	15	78	781	10,126	7	900,000				
8	0	16	94	907	11,974	8	900,000				
9	0	15	109	1,051	13,520	9	900,000				
10	0	16	125	1,189	15,068	10	900,000				





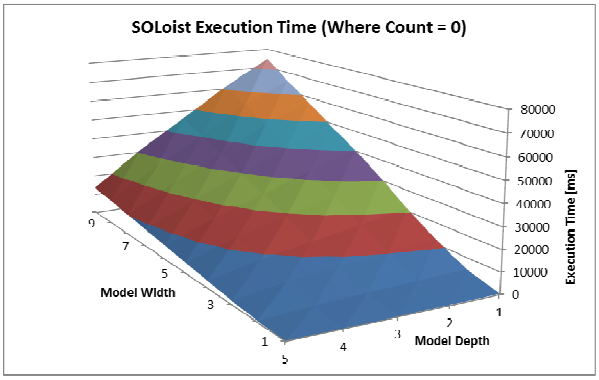
Scalability: SOLoist Performance on MySQL

On MySQL, Hibernate HQL performed extremely poorly. It was able to execute (in reasonable time of the 15 minute limit) only a very small number of the narrower and shallower queries even for the smallest database size of 1K! The larger data sets (of 1M and more) were even not worth trying. This is why we performed just a very small set of benchmarks for Hibernate on MySQL. SOLoist, on the other hand, performed quite well on MySQL and completed the full suite of queries for all database sizes!

Here we are presenting the results for MySQL for SOLoist only, for the referential data size of 10M and for different query complexity.

Execution Time (Where Count = 0, Data Size =10^7, DB = MySQL)

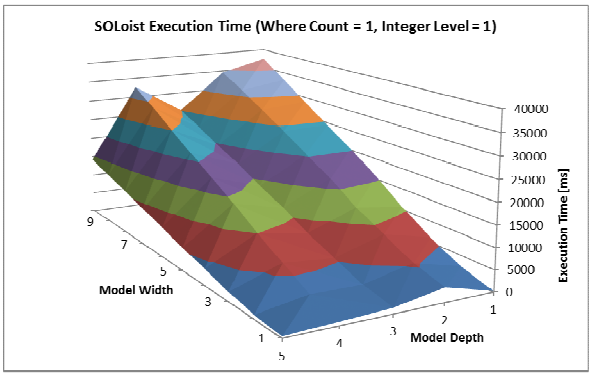
SOLoist Data Size = 10^7	Execution Time (Where Count = 0) Model Width									
Model Depth	1	2	3	4	5	6	7	8	9	10
1	438	4,654	12,190	21,099	31,316	40,411	48,552	57,302	65,689	74,719
2	344	3,908	10,392	17,032	25,657	32,970	40,391	47,865	54,866	61,376
3	250	2,974	8,023	12,974	19,268	24,392	29,992	35,302	40,438	46,021
4	177	2,051	5,383	8,844	12,844	16,392	19,797	23,392	27,236	31,032
5	78	638	1,786	3,442	5,115	6,786	8,505	10,177	11,864	13,552



Execution Time (Where Count = 1, Integer Level = 1, Data Size =10^7, DB = MySQL)

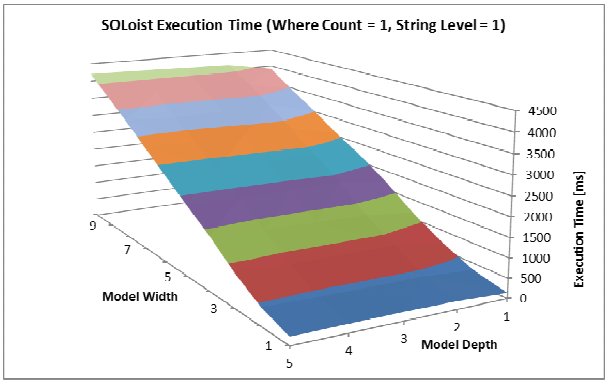
SOLoist Data Size = 10^7	Execution Time (Where Count = 1, Integer Level = 1) Model Width									
Model Depth	1	2	3	4	5	6	7	8	9	10
1	317	2,456	6,099	10,661	15,782	20,344	24,410	28,833	33,017	37,517

2	3,455	5,387	8,876	12,318	16,718	20,427	24,130	27,802	31,255	34,579
3	1,361	2,830	5,386	8,083	11,303	14,005	17,079	19,661	22,412	25,161
4	875	2,975	6,442	10,080	14,112	17,772	21,364	25,142	28,954	32,631
5	437	939	2,254	3,911	5,536	7,257	9,063	10,755	12,438	14,193



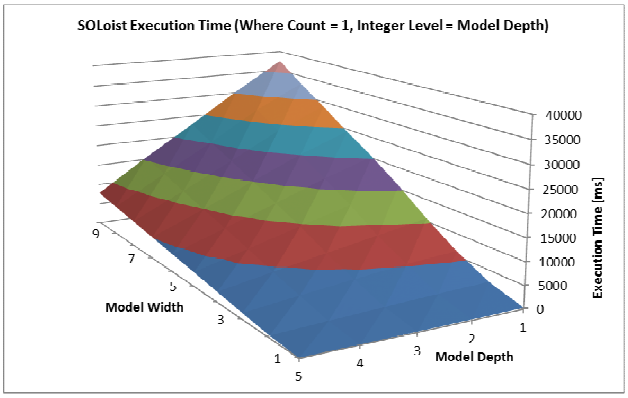
Execution Time (Where Count = 1, String Level = 1, Data Size =10^7, DB = MySQL)

SOLoist	Execution Time (Where Count = 1, String Level = 1)									
Data Size = 10^7	Model Width									
Model Depth	1	2	3	4	5	6	7	8	9	10
1	140	328	687	1,145	1,707	2,177	2,579	3,036	3,458	3,911
2	203	422	843	1,298	1,892	2,360	2,845	3,313	3,745	4,142
3	219	422	844	1,318	1,859	2,333	2,833	3,271	3,724	4,163
4	218	422	843	1,333	1,849	2,333	2,786	3,239	3,739	4,188
5	208	454	878	1,407	1,864	2,338	2,817	3,298	3,755	4,208



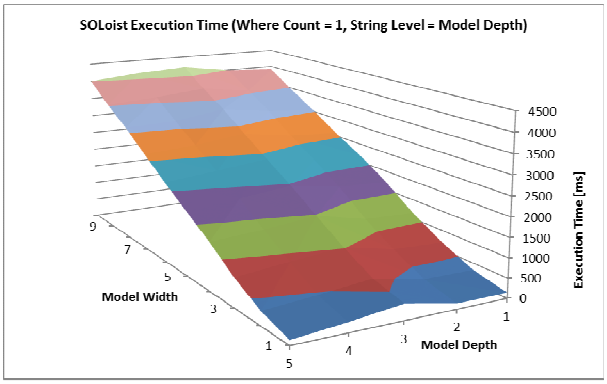
Execution Time (Where Count = 1, Integer Level = Model Depth, Data Size =10^7, DB = MySQL)

SOLoist	Execution Time (Where Count = 1, Integer Level = Model Depth)									
Data Size = 10^7	Model Width									
Model Depth	1	2	3	4	5	6	7	8	9	10
1	317	2,456	6,099	10,661	15,782	20,344	24,410	28,833	33,017	37,517
2	219	2,062	5,255	8,583	12,882	16,580	20,114	23,860	27,284	30,474
3	172	1,552	4,051	6,567	9,708	12,329	15,052	17,736	20,250	22,924
4	110	1,047	2,748	4,490	6,468	8,255	9,942	11,766	13,631	15,580
5	62	517	1,363	2,317	3,239	4,145	5,079	6,001	6,896	7,814



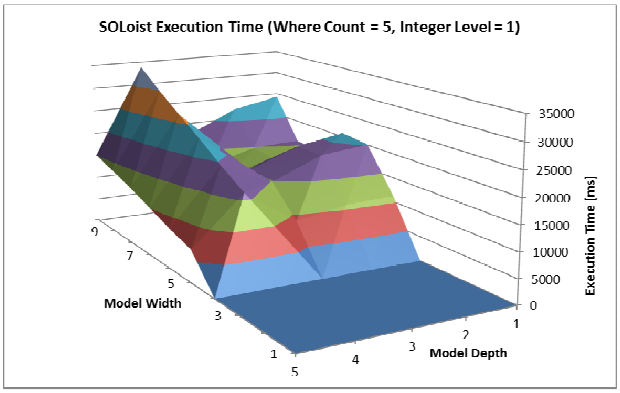
Execution Time (Where Count = 1, String Level = Model Depth, Data Size =10^7, DB = MySQL)

SOLoist Data Size = 10^7	Execution Time (Where Count = 1, String Level = Model Depth) Model Width									
Model Depth	1	2	3	4	5	6	7	8	9	10
1	140	328	687	1,145	1,707	2,177	2,579	3,036	3,458	3,911
2	141	343	734	1,192	1,755	2,223	2,661	3,146	3,564	3,991
3	406	609	1,017	1,485	1,974	2,411	2,864	3,338	3,733	4,194
4	250	469	892	1,345	1,835	2,270	2,707	3,142	3,610	4,126
5	125	359	750	1,255	1,693	2,141	2,614	3,099	3,521	4,005



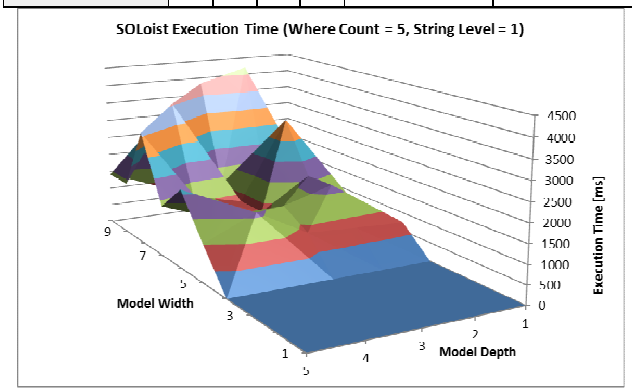
Execution Time (Where Count = 5, Integer Level = 1, Data Size =10^7, DB = MySQL)

SOLoist Data Size = 10^7	Execution Time (Where Count = 5, Integer Level = 1) Model Width									
Model Depth	1	2	3	4	5	6	7	8	9	10
1					11,188	20,095	20,801	14,235	15,348	24,272
2					10,818	19,048	19,020	14,115	14,911	22,177
3					9,037	14,848	15,235	11,642	12,161	17,411
4					16,270	19,517	23,188	26,548	30,032	33,844
5					6,677	8,271	10,126	11,974	13,520	15,068



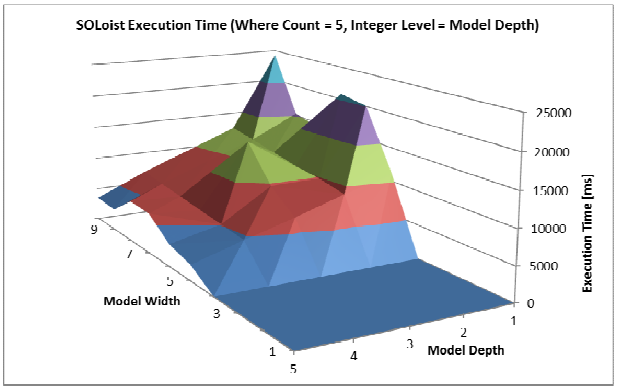
Execution Time (Where Count = 5, String Level = 1, Data Size =10^7, DB = MySQL)

SOLoist	Execution Time (Where Count = 5, String Level = 1)									
Data Size = 10^7	Model Width									
Model Depth	1	2	3	4	5	6	7	8	9	10
1					734	750	746	766	750	750
2					797	1,673	1,657	3,032	953	4,204
3					875	1,286	813	813	1,036	3,833
4					1,614	812	843	1,145	3,552	875
5					1,016	2,103	1,365	3,099	1,411	1,412



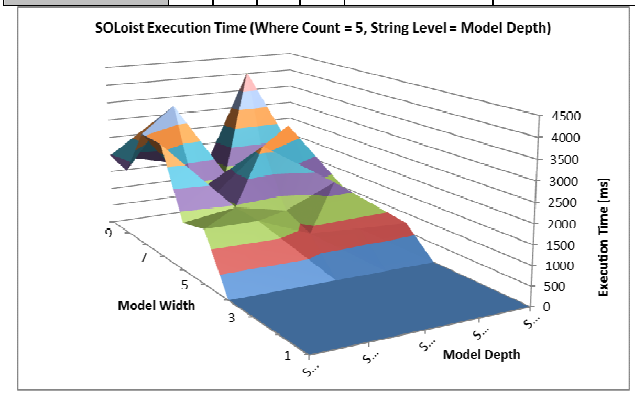
Execution Time (Where Count =5, Integer Level = Model Depth, Data Size =10^7, DB = MySQL)

SOLoist	Execution Time (Where Count = 5, Integer Level = Model Depth)									
Data Size = 10^7	Model Width									
Integer Level	1	2	3	4	5	6	7	8	9	10
1					11,188	20,095	20,801	14,235	15,348	24,272
2					9,052	9,864	10,298	11,053	11,802	12,474
3					6,719	9,735	14,954	8,704	9,130	9,615
4					4,616	5,083	5,482	5,955	6,346	6,782
5					2,344	3,330	6,069	6,223	3,257	3,474



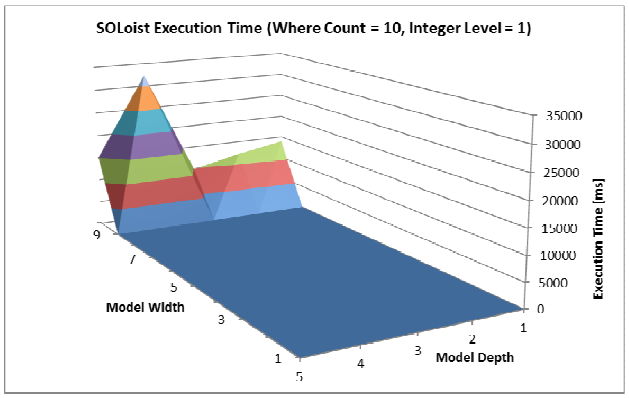
Execution Time (Where Count =5, String Level = Model Depth, Data Size =10^7, DB = MySQL)

SOLoist Data Size = 10^7	Execution Time (Where Count = 5, String Level = Model Depth) Model Width									
String Level	1	2	3	4	5	6	7	8	9	10
1					734	750	746	766	750	750
2					797	1,758	1,646	2,860	991	4,032
3					875	1,177	2,657	958	1,021	1,266
4					1,414	1,455	1,489	1,255	3,481	1,643
5					1,413	1,251	3,021	3,064	1,788	1,974



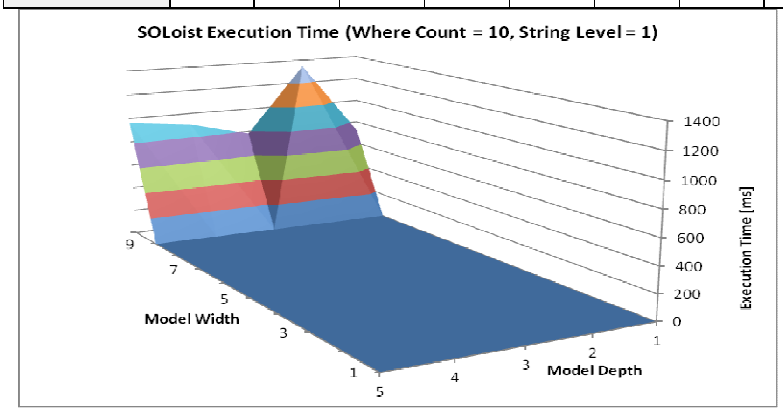
Execution Time (Where Count = 10, Integer Level = 1, Data Size =10^7, DB = MySQL)

SOLoist Data Size = 10^7	Execution Time (Where Count = 10, Integer Level = 1) Model Width									
Model Depth	1	2	3	4	5	6	7	8	9	10
1	0									13,907
2										12,099
3										9,751
4										32,376
5										15,047



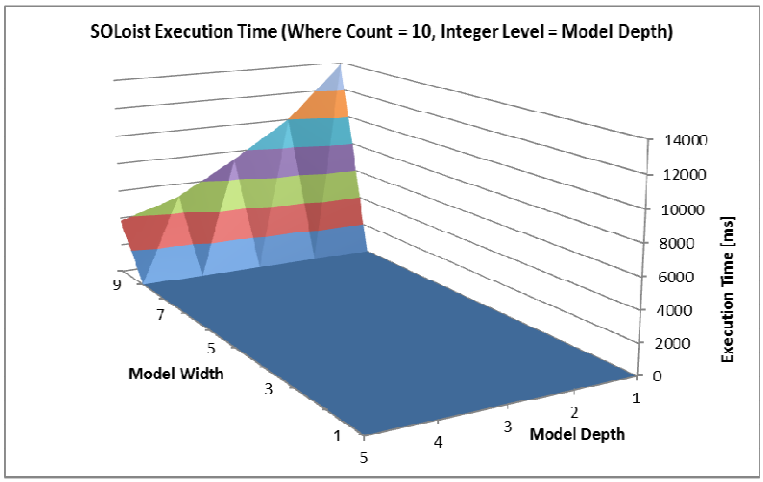
Execution Time (Where Count = 10, String Level = 1, Data Size =10^7, DB = MySQL)

SOLoist	Execution Time (Where Count = 10, String Level = 1)									
Data Size = 10^7	Model Width									
Model Depth	1	2	3	4	5	6	7	8	9	10
1	0									718
2										1,344
3										781
4										906
5										961



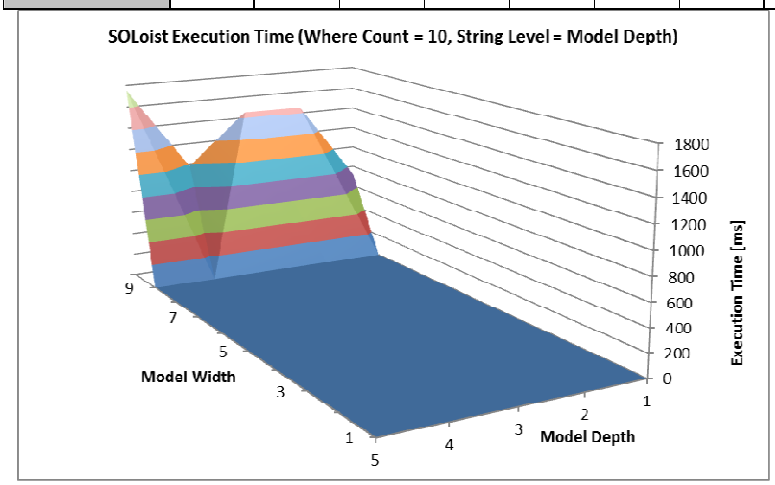
Execution Time (Where Count = 10, Integer Level=Model Depth, Data Size =10^7, DB = MySQL)

SOLoist	Execution Time (Where Count = 10, Integer Level = Model Depth)									
Data Size = 10^7	Model Width									
Integer Level	1	2	3	4	5	6	7	8	9	10
1	0									13,907
2										10,040
3										7,409
4										5,068
5										3,771



Execution Time (Where Count = 10, String Level = Model Depth, Data Size =10^7, DB = MySQL)

SOLoist Data Size = 10^7	Execution Time (Where Count = 10, String Level = Model Depth) Model Width									
String Level	1	2	3	4	5	6	7	8	9	10
1										718
2										1,452
3										1,454
4										986
5										1,746



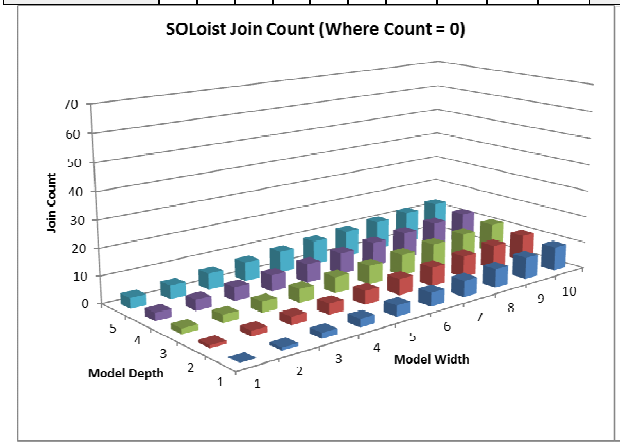
Queries Returning Data in Results

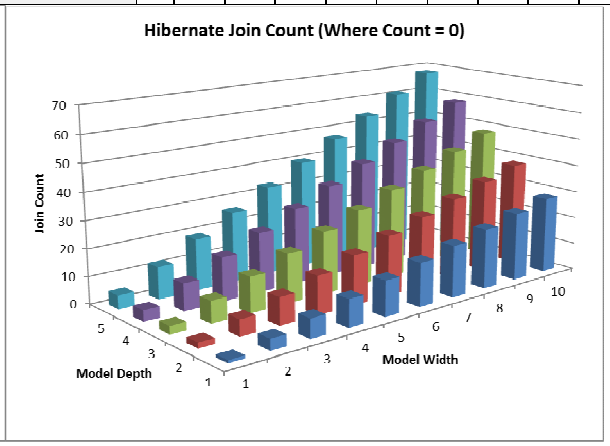
In order to verify that the queries that return only the COUNT in their result set represent a correct general benchmarking, we have repeated the same experiments (with Oracle only) with the same queries, modified so that they return the values of all attributes (owned and inherited) of the objects of the class in the first vertical (*x1* of class *A1..E1*). The results presented in this section show that these queries, quite expectedly, do perform a bit more slowly than with COUNT results (because they have to fetch and return some data instead just to count the selected records), and do that for both technologies, SOLoist and Hibernate. However, the relative performance of SOLoist vs. Hibernate is unchanged: SOLoist outperforms Hibernate in all cases, and does that significantly in most cases.

SQL vs. Object Query Complexity

Join Count (Where Count = 0)

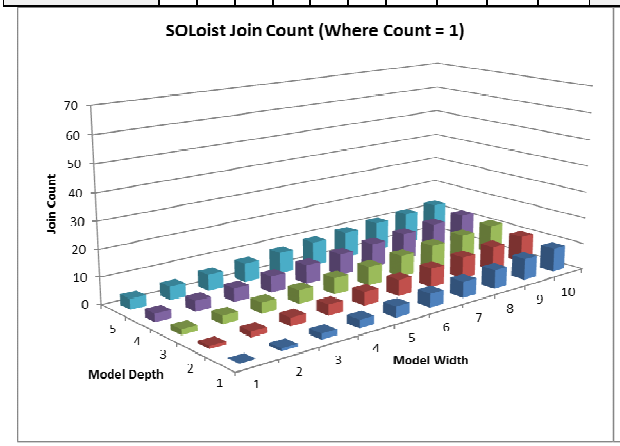
SOLoist	Join Count (Where Count = 0)										Hibernate	Join Count (Where Count = 0)									
	Model Width											Model Width									
Model Depth	1	2	3	4	5	6	7	8	9	10	Model Depth	1	2	3	4	5	6	7	8	9	10
1	0	1	2	3	4	5	6	7	8	9	1	1	4	7	10	13	16	19	22	25	28
2	1	2	3	4	5	6	7	8	9	10	2	2	6	10	14	18	22	26	30	34	38
3	2	3	4	5	6	7	8	9	10	11	3	3	8	13	18	23	28	33	38	43	48
4	3	4	5	6	7	8	9	10	11	12	4	4	10	16	22	28	34	40	46	52	58
5	4	5	6	7	8	9	10	11	12	13	5	5	12	19	26	33	40	47	54	61	68

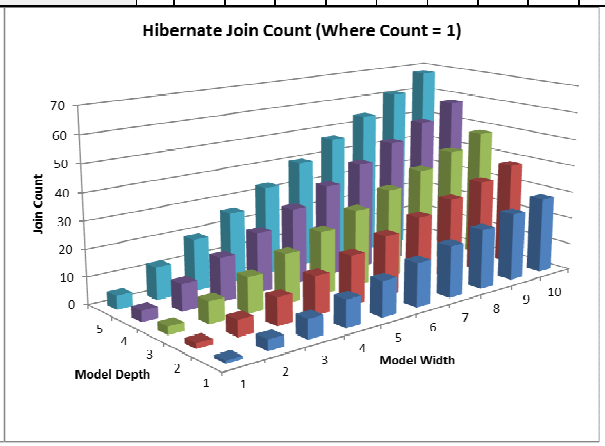




Join Count (Where Count = 1, Integer/String Level = 1)

SOLoist	Join Count (Where Count = 1)										Hibernate	Join Count (Where Count = 1)									
	Model Width											Model Width									
Model Depth	1	2	3	4	5	6	7	8	9	10	Model Depth	1	2	3	4	5	6	7	8	9	10
1	0	1	2	3	4	5	6	7	8	9	1	1	4	7	10	13	16	19	22	25	28
2	1	2	3	4	5	6	7	8	9	10	2	2	6	10	14	18	22	26	30	34	38
3	2	3	4	5	6	7	8	9	10	11	3	3	8	13	18	23	28	33	38	43	48
4	3	4	5	6	7	8	9	10	11	12	4	4	10	16	22	28	34	40	46	52	58
5	4	5	6	7	8	9	10	11	12	13	5	5	12	19	26	33	40	47	54	61	68



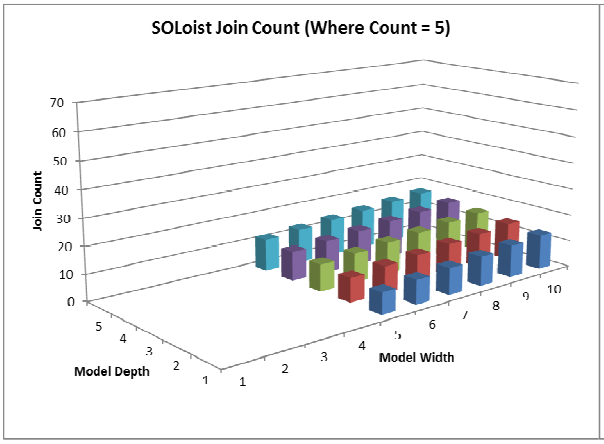


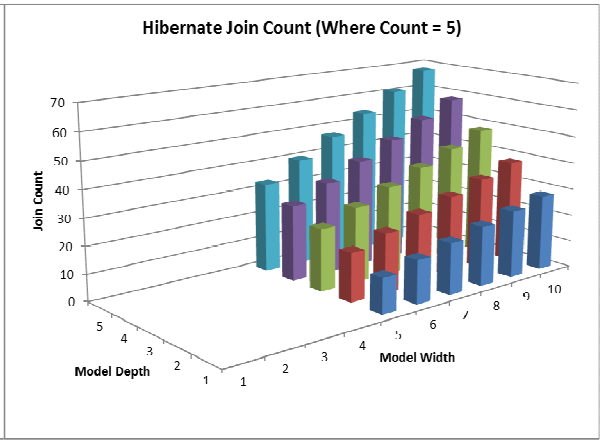
Join Count (Where Count = 5, Integer/String Level = 1)

SOLoist	Join Count (Where Count = 5)	Hibernate	Join Count (Where Count = 5)

	Model Width									
Model Depth	1	2	3	4	5	6	7	8	9	10
1					8	9	10	11	12	13
2					9	10	11	12	13	14
3					10	11	12	13	14	15
4					11	12	13	14	15	16
5					12	13	14	15	16	17

	Model Width									
Model Depth	1	2	3	4	5	6	7	8	9	10
1					13	16	19	22	25	28
2					18	22	26	30	34	38
3					23	28	33	38	43	48
4					28	34	40	46	52	58
5					33	40	47	54	61	68

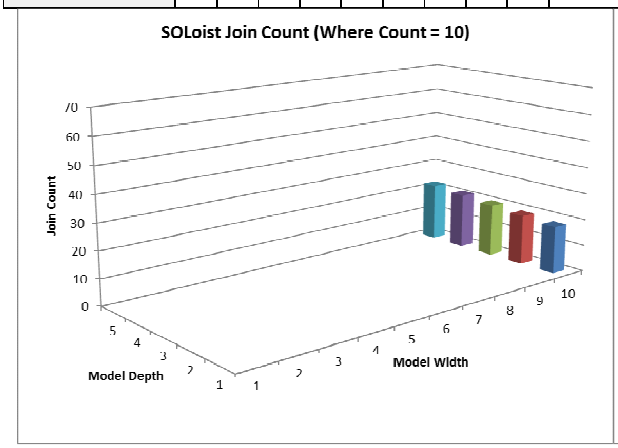


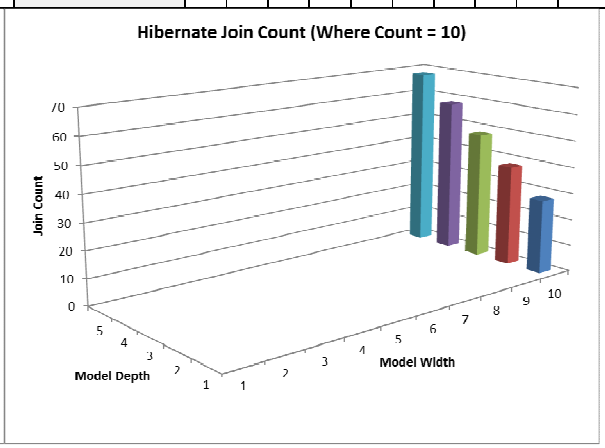


Join Count (Where Count = 10, Integer/String Level = 1)

SOLoist	Join Count (Where Count = 10)									
	Model Width									
Model Depth	1	2	3	4	5	6	7	8	9	10
1										18
2										19
3										20
4										21
5										22

Hibernate	Join Count (Where Count = 10)									
	Model Width									
Model Depth	1	2	3	4	5	6	7	8	9	10
1										28
2										38
3										48
4										58
5										68

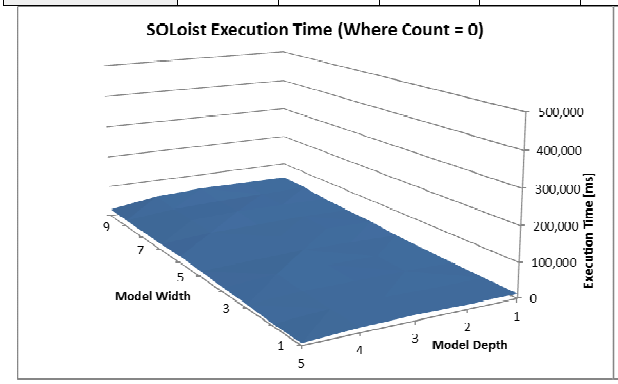


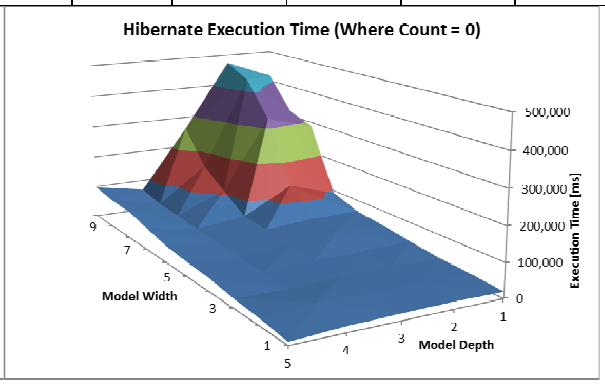


Scalability: Execution Time vs. Object Query Complexity

Execution Time (Where Count = 0, Data Size =10^7, DB = Oracle)

SOLoist	Execution Time (Where Count = 0)									
Data Size = 10^7	Model Width									
Model Depth	1	2	3	4	5	6	7	8	9	10
1	13,956	15,048	17,313	20,313	23,594	30,970	33,781	38,595	46,689	50,628
2	13,141	15,126	19,844	23,657	25,969	32,141	35,548	39,532	43,251	51,031
3	15,611	17,158	18,391	23,079	27,719	30,876	31,579	40,095	39,673	51,722
4	13,376	15,485	18,538	23,845	22,315	26,486	29,673	33,907	34,485	44,487
5	6,251	5,704	10,407	12,705	16,266	20,047	18,095	19,376	19,657	21,896
Hibernate	Execution Time (Where Count = 0)									
Data Size = 10^7	Model Width									
Model Depth	1	2	3	4	5	6	7	8	9	10
1	19,087	27,079	31,953	42,766	55,160	64,219	92,391	286,835	315,364	417,314
2	21,267	30,798	31,641	37,562	59,551	63,441	135,516	327,970	435,017	473,782
3	19,579	29,111	31,032	34,985	43,189	51,751	58,626	108,454	159,480	262,079
4	17,204	32,407	24,251	27,673	30,078	36,469	42,204	48,394	53,141	68,656
5	9,141	20,252	27,220	38,719	43,345	50,720	66,656	83,807	86,328	97,908

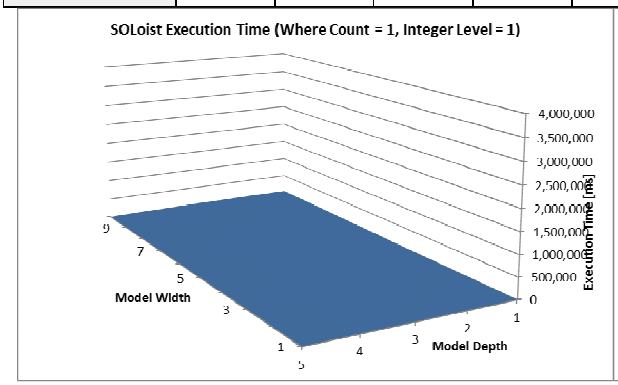


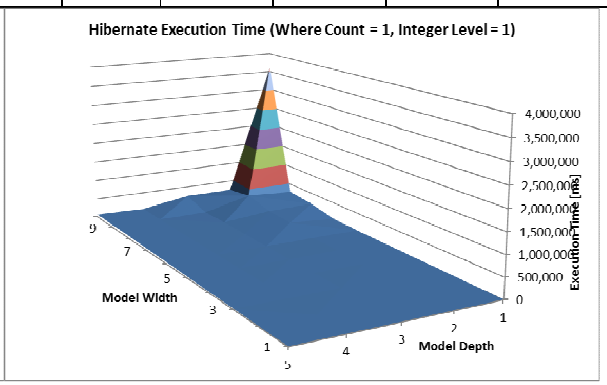


Execution Time (Where Count = 1, Integer Level = 1, Data Size =10^7, DB = Oracle)

SOLoist	Execution Time (Where Count = 1, Integer Level = 1)									
Data Size = 10^7	Model Width									
Model Depth	1	2	3	4	5	6	7	8	9	10
1	6,626	7,418	8,298	9,880	16,063	25,079	26,308	36,079	40,157	46,231
2	7,173	8,038	7,938	13,266	17,758	22,001	24,126	32,099	38,267	48,270
3	7,300	7,407	9,206	13,736	16,313	19,001	20,902	30,407	29,782	41,968
4	8,157	9,197	10,408	14,189	15,642	14,458	20,845	22,626	25,033	30,463
5	4,168	3,224	8,282	9,360	11,345	11,532	12,438	11,954	12,361	13,611
Hibernate	Execution Time (Where Count = 1, Integer Level = 1)									
Data Size = 10^7	Model Width									
Model Depth	1	2	3	4	5	6	7	8	9	10
1	9,032	15,829	23,824	30,064	40,256	45,844	70,343	206,485	277,985	3,600,000
2	9,469	19,391	23,642	27,673	40,750	45,193	151,220	177,980	345,354	245,657
3	10,548	19,908	22,418	24,079	30,517	34,142	51,022	60,392	51,348	240,812
4	11,159	22,392	18,782	21,079	23,156	25,642	27,454	30,385	38,578	49,782

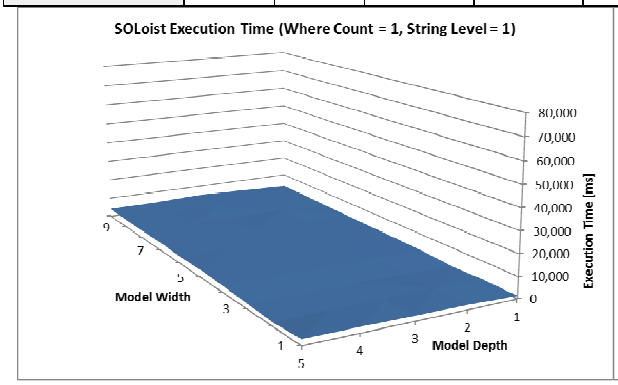
5	5,563	16,001	24,033	25,813	35,909	46,345	49,329	63,455	62,407	72,156

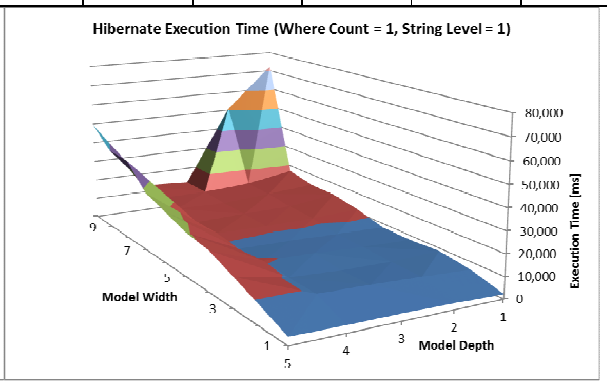




Execution Time (Where Count = 1, String Level = 1, Data Size =10^7, DB = Oracle)

SOLoist	Execution Time (Where Count = 1, String Level = 1)									
Data Size = 10^7	Model Width									
Model Depth	1	2	3	4	5	6	7	8	9	10
1	1,376	1,735	1,267	2,141	2,438	2,563	2,783	2,986	3,298	3,501
2	2,188	2,563	2,001	2,985	3,251	3,454	3,610	3,970	4,189	4,236
3	2,392	2,111	2,136	2,891	3,282	3,470	3,814	3,594	4,517	4,997
4	2,689	2,298	2,141	3,016	3,157	3,314	3,715	3,611	3,689	4,242
5	2,759	1,739	2,423	3,064	3,548	3,297	3,823	3,621	3,720	4,162
Hibernate	Execution Time (Where Count = 1, String Level = 1)									
Data Size = 10^7	Model Width									
Model Depth	1	2	3	4	5	6	7	8	9	10
1	2,157	6,485	8,658	7,829	8,065	11,657	14,095	14,345	17,113	72,298
2	2,939	7,579	7,798	8,423	8,564	10,719	11,360	12,612	13,299	50,938
3	3,361	8,595	8,110	9,594	9,157	10,532	11,017	11,173	11,972	12,157
4	3,407	9,126	9,314	10,158	9,344	10,782	10,454	11,767	11,876	13,596
5	3,720	9,079	14,126	19,033	20,844	24,767	28,564	39,611	40,142	49,221

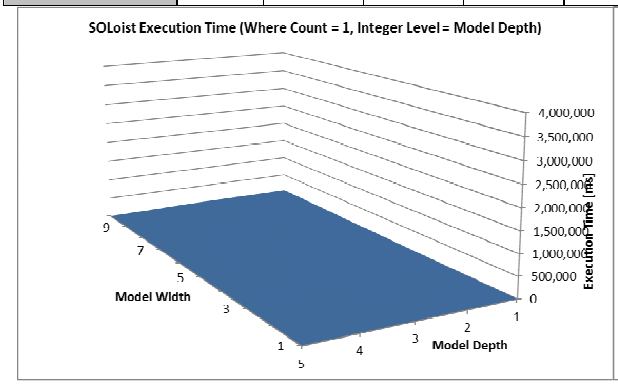


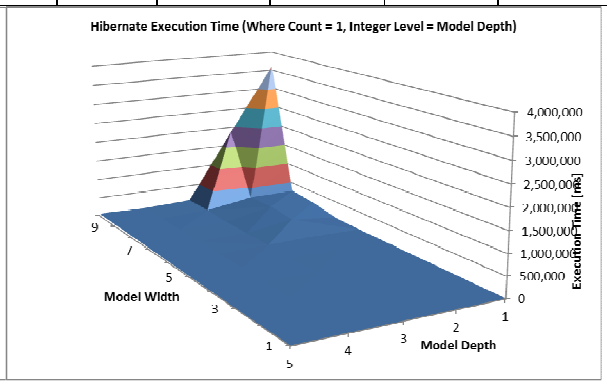


Execution Time (Where Count = 1, Integer Level = Model Depth, Data Size =10^7, DB = Oracle)

SOLoist	Execution Time (Where Count = 1, Integer Level = Model Depth)									
Data Size = 10^7	Model Width									
Model Depth	1	2	3	4	5	6	7	8	9	10
1	6,626	7,418	8,298	9,880	16,063	25,079	26,308	36,079	40,157	46,231
2	7,094	7,986	8,110	9,048	14,012	21,971	25,611	29,033	37,143	41,786
3	6,298	6,313	7,551	12,407	15,064	22,353	23,423	29,501	31,157	44,006

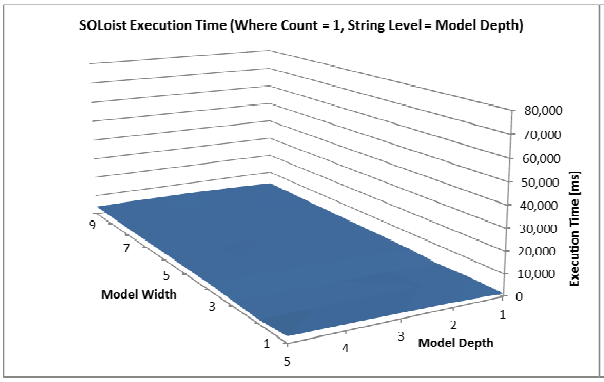
4	5,333	5,360	5,552	8,298	8,033	11,892	17,314	22,392	22,751	26,563
5	3,704	3,017	3,723	4,361	4,430	4,622	4,594	7,673	8,298	10,139
Hibernate	Execution Time (Where Count = 1, Integer Level = Model Depth)									
Data Size = 10^7	Model Width									
Model Depth	1	2	3	4	5	6	7	8	9	10
1	9,032	15,829	23,824	30,064	40,256	45,844	70,343	206,485	277,985	3,600,000
2	9,407	16,173	22,704	26,011	49,909	65,517	211,486	278,196	259,902	1,871,798
3	8,658	14,704	21,142	23,407	29,844	46,813	39,001	44,438	79,002	91,563
4	7,610	14,079	17,610	18,500	22,266	21,813	26,532	29,181	31,704	45,996
5	5,001	9,813	14,503	19,845	22,813	31,329	41,751	51,581	56,024	64,219

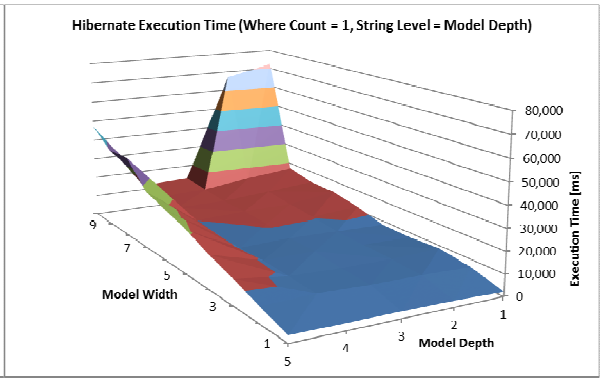




Execution Time (Where Count = 1, String Level = Model Depth, Data Size =10^7, DB = Oracle)

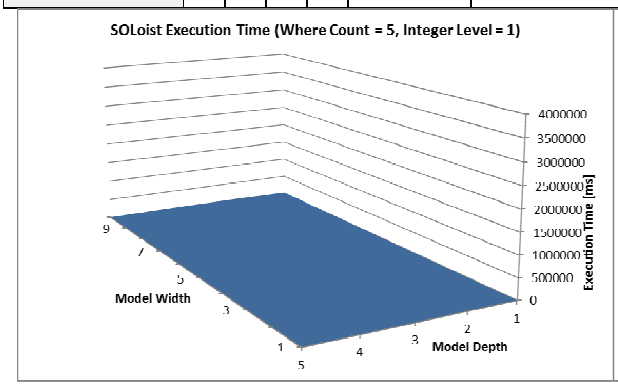
SOLoist	Execution Time (Where Count = 1, String Level = Model Depth)									
Data Size = 10^7	Model Width									
Model Depth	1	2	3	4	5	6	7	8	9	10
1	1,376	1,735	1,267	2,141	2,438	2,563	2,783	2,986	3,298	3,501
2	2,158	2,407	2,016	2,787	2,939	3,204	3,501	3,501	3,892	4,087
3	2,486	2,048	2,097	2,845	3,130	3,619	3,547	3,684	3,704	4,346
4	2,783	2,212	2,246	3,027	3,298	3,199	3,376	3,730	3,804	4,308
5	2,892	1,782	2,349	3,033	3,048	3,282	3,408	3,411	3,751	3,989
Hibernate	Execution Time (Where Count = 1, String Level = Model Depth)									
Data Size = 10^7	Model Width									
Model Depth	1	2	3	4	5	6	7	8	9	10
1	2,157	6,485	8,658	7,829	8,065	11,657	14,095	14,345	17,113	72,298
2	2,908	7,157	7,922	8,220	8,681	10,189	11,235	12,533	14,779	66,986
3	3,173	7,391	7,980	9,235	9,237	10,283	11,001	10,986	11,865	12,985
4	3,220	8,205	8,788	9,234	8,595	9,121	10,017	11,049	11,266	12,206
5	3,407	7,532	12,046	15,938	21,032	24,517	28,142	39,855	38,954	47,085

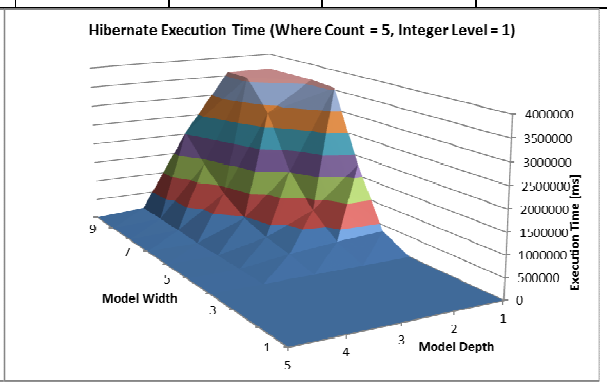




Execution Time (Where Count = 5, Integer Level = 1, Data Size =10^7, DB = Oracle)

SOLoist	Execution Time (Where Count = 5, Integer Level = 1)									
Data Size = 10^7	Model Width									
Model Depth	1	2	3	4	5	6	7	8	9	10
1					4,977	6,501	9,235	11,610	14,657	18,315
2					8,283	10,439	17,532	23,641	20,016	28,957
3					16,860	17,407	24,626	24,391	21,923	33,301
4					16,236	20,454	19,782	22,345	24,860	26,694
5					13,094	15,298	14,533	13,907	14,376	15,367
Hibernate	Execution Time (Where Count = 5, Integer Level = 1)									
Data Size = 10^7	Model Width									
Model Depth	1	2	3	4	5	6	7	8	9	10
1					344,580	1,778,298	3,553,173	3,600,000	3,600,000	3,600,000
2					269,486	773,766	1,528,847	2,833,844	3,600,000	3,600,000
3					165,735	278,063	479,985	851,486	1,390,329	2,047,126
4					18,516	18,297	21,329	58,688	71,407	94,513
5					13,954	12,391	14,767	13,538	14,736	14,023



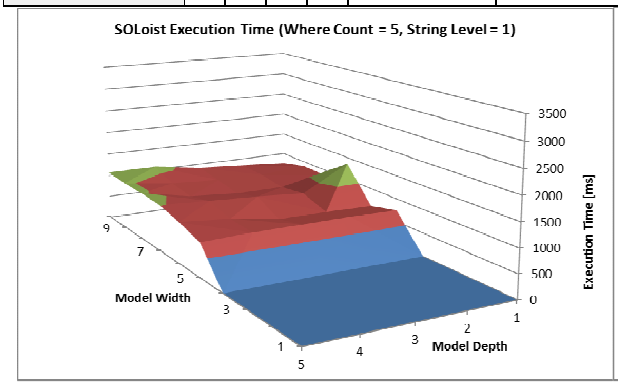


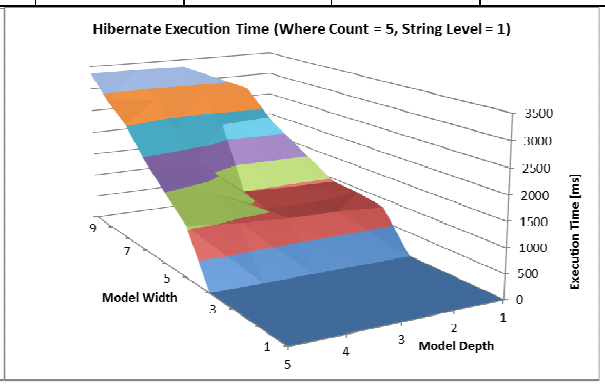
Execution Time (Where Count = 1, String Level = 1, Data Size =10^7, DB = Oracle)

SOLoist	Execution Time (Where Count = 5, String Level = 1)									
Data Size = 10^7	Model Width									
Model Depth	1	2	3	4	5	6	7	8	9	10
1					750	609	1,360	875	891	759

2					766	625	813	735	813	838
3					766	625	890	734	828	912
4					765	797	945	813	891	1,072
5					735	890	1,008	1,032	1,033	1,067

Hibernate	Execution Time (Where Count = 5, String Level = 1)									
Data Size = 10^7	Model Width									
Model Depth	1	2	3	4	5	6	7	8	9	10
1					813	907	937	906	922	937
2					875	828	953	2,360	2,876	1,845
3					891	859	996	2,298	2,833	3,329
4					953	1,329	1,808	2,344	2,813	3,346
5					1,064	1,501	1,954	2,528	2,876	3,344

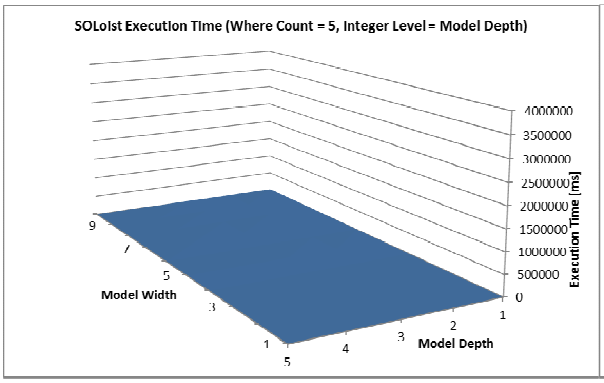


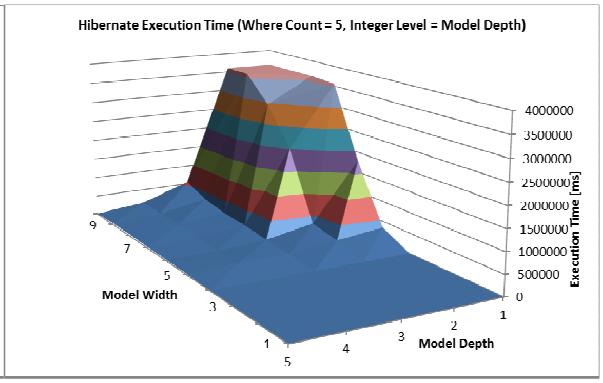


Execution Time (Where Count = 1, Integer Level = Model Depth, Data Size =10^7, DB = Oracle)

SOLoist	Execution Time (Where Count = 5, Integer Level = Model Depth)									
Data Size = 10^7	Model Width									
Integer Level	1	2	3	4	5	6	7	8	9	10
1					4,977	6,501	9,235	11,610	14,657	18,315
2					5,938	6,188	7,392	9,673	12,345	13,939
3					6,501	6,251	7,126	7,564	9,860	13,255
4					2,938	3,017	6,985	6,751	5,767	6,722
5					2,282	2,611	2,533	2,611	2,829	3,247

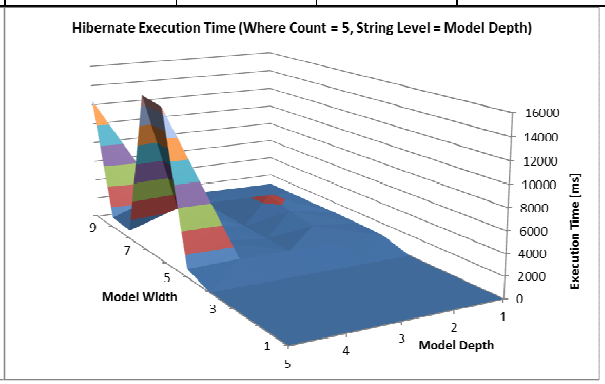
Hibernate	Execution Time (Where Count = 5, Integer Level = Model Depth)									
Data Size = 10^7	Model Width									
Integer Level	1	2	3	4	5	6	7	8	9	10
1					344,580	1,778,298	3,553,173	3,600,000	3,600,000	3,600,000
2					213,657	342,736	2,011,082	2,958,720	3,600,000	3,600,000
3					87,063	124,719	197,782	230,751	336,813	549,735
4					99,407	47,626	56,516	111,415	118,345	143,532
5					25,282	27,565	32,954	41,712	39,469	17,298





Execution Time (Where Count = 1, String Level = Model Depth, Data Size =10^7, DB = Oracle)

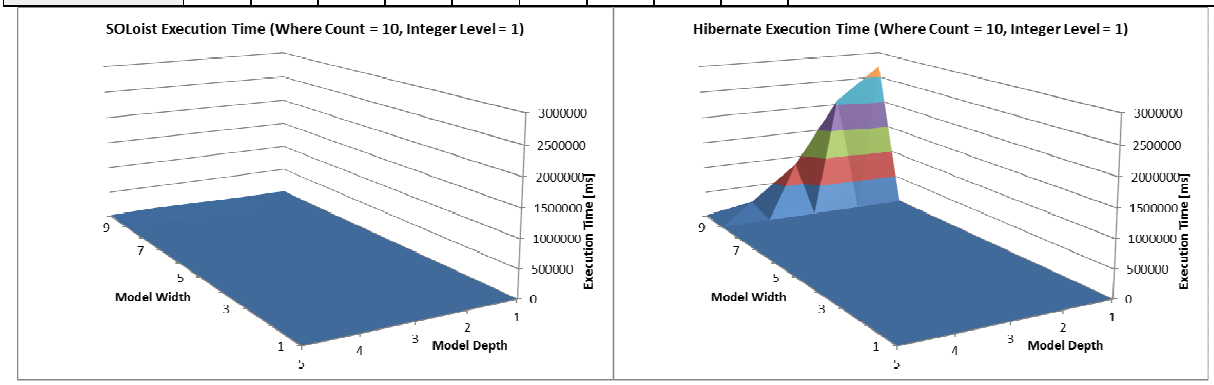
SOLoist	Execution Time (Where Count = 5, String Level = Model Depth)									
Data Size = 10^7	Model Width									
String Level	1	2	3	4	5	6	7	8	9	10
1					750	609	1,360	875	891	759
2					829	1,283	1,652	1,747	796	832
3					766	1,110	875	750	2,267	973
4					610	860	1,142	1,449	1,704	2,011
5					469	531	563	547	594	662
Hibernate	Execution Time (Where Count = 5, String Level = Model Depth)									
Data Size = 10^7	Model Width									
String Level	1	2	3	4	5	6	7	8	9	10
1					813	907	937	906	922	937
2					781	1,345	1,778	2,360	921	984
3					859	843	922	1,011	1,110	1,045
4					906	1,345	1,798	2,377	2,797	3,236
5					796	14,705	15,110	812	1,001	12,470



Execution Time (Where Count = 10, Integer Level = 1, Data Size =10^7, DB = Oracle)

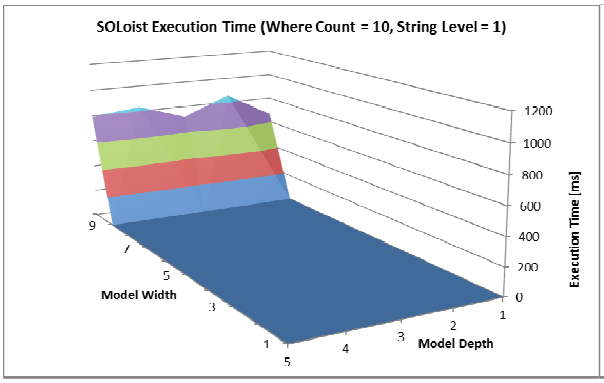
SOLoist	Execution Time (Where Count = 10, Integer Level = 1)									
Data Size = 10^7	Model Width									
Model Depth	1	2	3	4	5	6	7	8	9	10
1										17,004

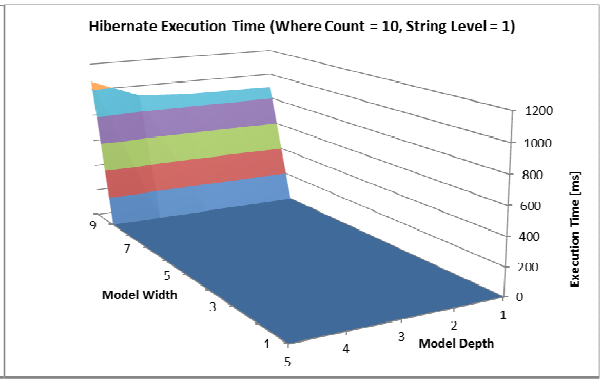
2										19,488
3										32,503
4										48,392
5										31,637
Hibernate	Execution Time (Where Count = 10, Integer Level = 1)									
Data Size = 10^7	Model Width									
Model Depth	1	2	3	4	5	6	7	8	9	10
1										2,719,019
2										2,086,736
3										842,337
4										179,096
5										30,486



Execution Time (Where Count = 10, String Level = 1, Data Size =10^7, DB = Oracle)

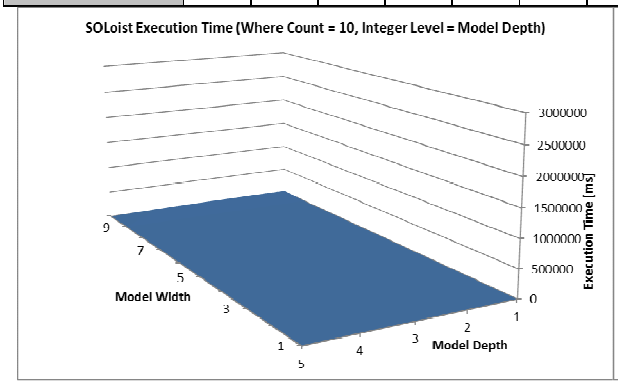
SOLoist	Execution Time (Where Count = 10, String Level = 1)									
Data Size = 10^7	Model Width									
Model Depth	1	2	3	4	5	6	7	8	9	10
1										665
2										856
3										717
4										833
5										791
Hibernate	Execution Time (Where Count = 10, String Level = 1)									
Data Size = 10^7	Model Width									
Model Depth	1	2	3	4	5	6	7	8	9	10
1										891
2										895
3										898
4										922
5										1,063

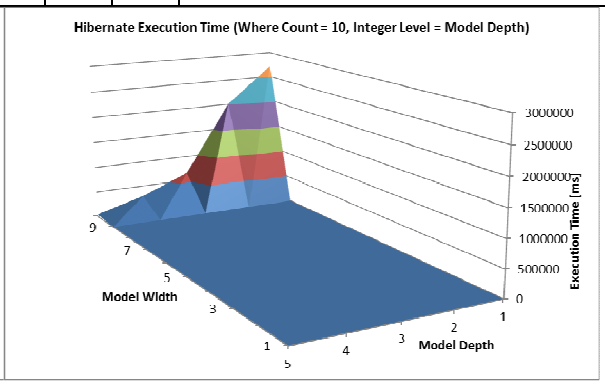




Execution Time (Where Count = 10, Integer Level = Model Depth, Data Size =10^7, DB = Oracle)

SOLoist	Execution Time (Where Count = 10, Integer Level = Model Depth)									
Data Size = 10^7	Model Width									
Integer Level	1	2	3	4	5	6	7	8	9	10
1										17,004
2										18,122
3										21,672
4										16,088
5										837
Hibernate	Execution Time (Where Count = 10, Integer Level = Model Depth)									
Data Size = 10^7	Model Width									
Integer Level	1	2	3	4	5	6	7	8	9	10
1										2,719,019
2										2,009,641
3										655,486
4										291,782
5										40,985

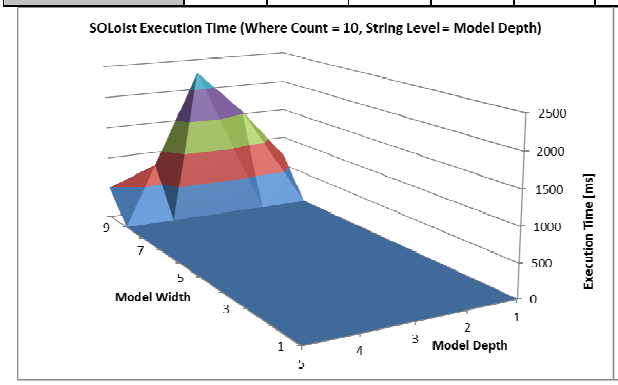


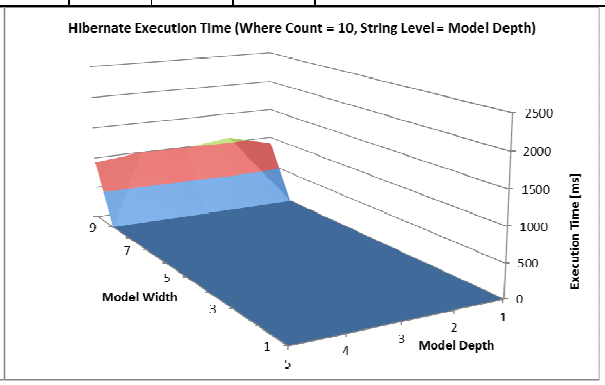


Execution Time (Where Count = 10, String Level = Model Depth, Data Size =10^7, DB = Oracle)

SOLoist	Execution Time (Where Count = 10, String Level = Model Depth)									
Data Size = 10^7	Model Width									
String Level	1	2	3	4	5	6	7	8	9	10
1										665

2										1,555
3										2,276
4										786
5										494
Hibernate	Execution Time (Where Count = 10, String Level = Model Depth)									
Data Size = 10^7	Model Width									
String Level	1	2	3	4	5	6	7	8	9	10
1										891
2										1,079
3										992
4										1,002
5										922

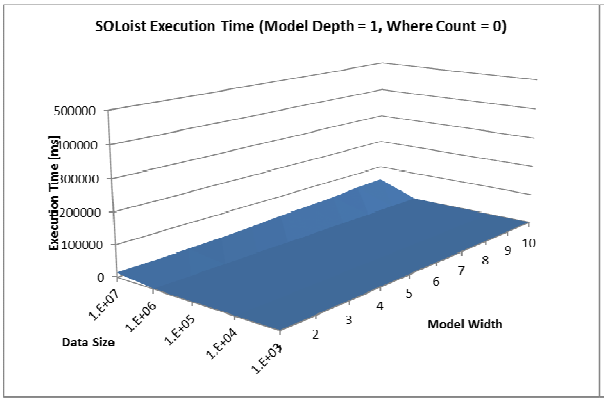


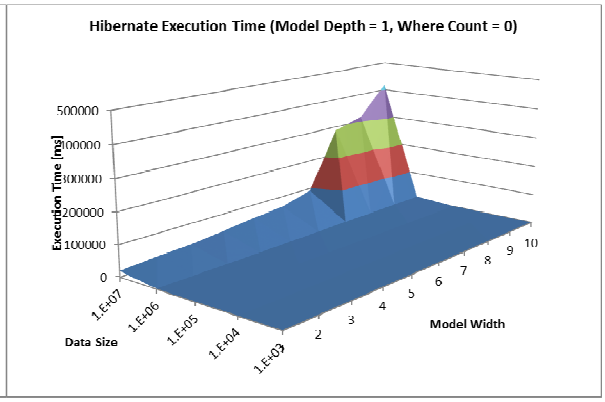


Scalability: Execution Time vs. Database Size

Execution Time (Model Depth = 1, Where Count = 0, DB = Oracle)

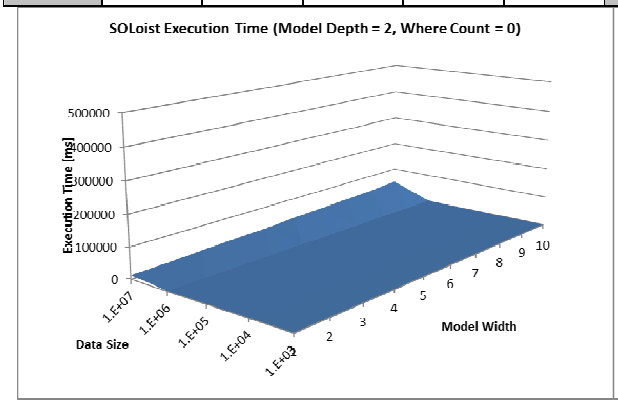
SOLoist	Execution Time (Model Depth = 1, Where Count = 0)					Hibernate	Execution Time (Model Depth = 1, Where Count = 0)				
	Data Size						Data Size				
Model Width	1.E+03	1.E+04	1.E+05	1.E+06	1.E+07	Model Width	1.E+03	1.E+04	1.E+05	1.E+06	1.E+07
1	5	40	164	1,412	13,956	1	10	21	167	1,689	19,087
2	4	19	156	1,456	15,048	2	6	29	208	2,076	27,079
3	6	22	170	1,498	17,313	3	8	28	232	2,316	31,953
4	4	26	174	1,985	20,313	4	10	32	292	2,861	42,766
5	7	29	191	1,885	23,594	5	9	34	382	4,499	55,160
6	8	34	204	1,881	30,970	6	9	41	342	5,398	64,219
7	9	28	216	1,989	33,781	7	13	49	437	7,239	92,391
8	11	30	225	2,115	38,595	8	11	52	528	7,420	286,835
9	13	33	239	2,536	46,689	9	12	62	622	8,401	315,364
10	18	43	293	2,428	50,628	10	12	70	713	9,755	417,314

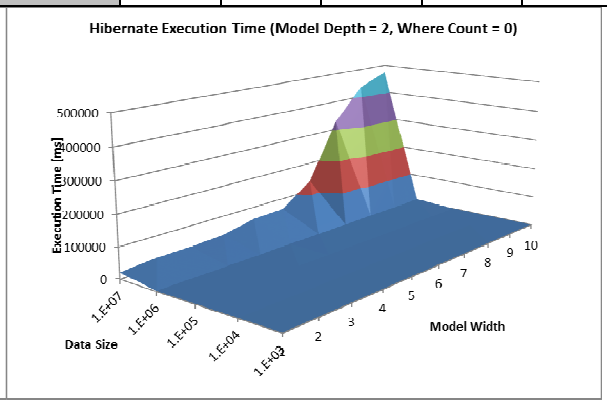




Execution Time (Model Depth = 2, Where Count = 0, DB = Oracle)

SOLoist	Execution Time (Model Depth = 2, Where Count = 0)					Hibernate	Execution Time (Model Depth = 2, Where Count = 0)				
	Data Size						Data Size				
Model Width	1.E+03	1.E+04	1.E+05	1.E+06	1.E+07	Model Width	1.E+03	1.E+04	1.E+05	1.E+06	1.E+07
1	8	26	147	1,319	13,141	1	14	22	164	1,810	21,267
2	5	19	134	1,578	15,126	2	10	25	198	2,060	30,798
3	6	21	158	1,416	19,844	3	9	28	215	2,364	31,641
4	6	28	156	1,828	23,657	4	13	33	294	3,096	37,562
5	8	22	211	1,768	25,969	5	9	32	286	4,442	59,551
6	10	26	181	1,716	32,141	6	9	39	310	4,599	63,441
7	13	29	195	1,858	35,548	7	10	46	396	5,637	135,516
8	14	29	204	2,107	39,532	8	10	53	460	6,591	327,970
9	15	33	217	2,093	43,251	9	11	60	532	6,668	435,017
10	18	36	301	2,417	51,031	10	12	63	599	7,995	473,782

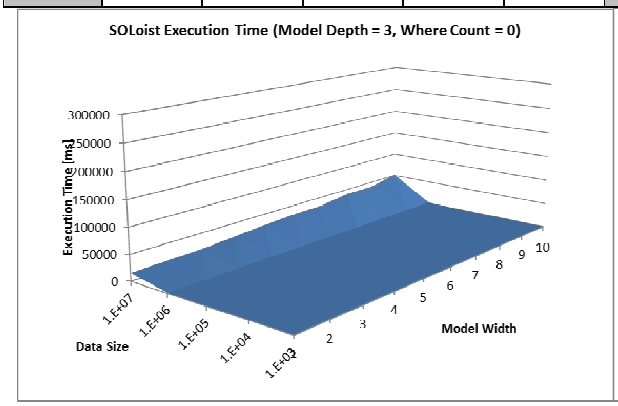


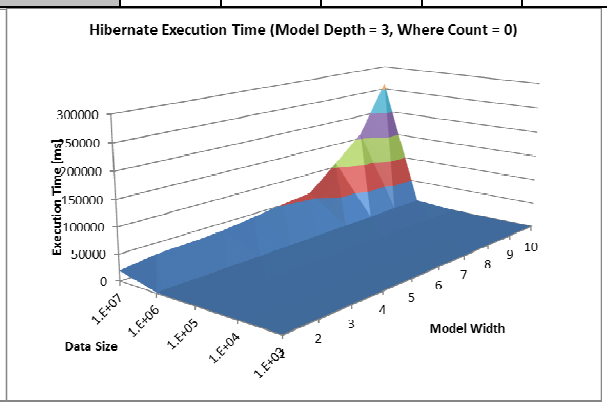


Execution Time (Model Depth = 3, Where Count = 0, DB = Oracle)

SOLoist	Execution Time (Model Depth = 3, Where Count = 0)					Hibernate	Execution Time (Model Depth = 3, Where Count = 0)				
	Data Size						Data Size				
Model Width	1.E+03	1.E+04	1.E+05	1.E+06	1.E+07	Model Width	1.E+03	1.E+04	1.E+05	1.E+06	1.E+07
1	9	18	115	1,156	15,611	1	12	21	149	1,470	19,579
2	5	20	116	1,113	17,158	2	14	27	178	1,841	29,111
3	7	21	149	1,187	18,391	3	11	27	190	2,104	31,032
4	7	21	133	1,399	23,079	4	11	28	247	2,832	34,985

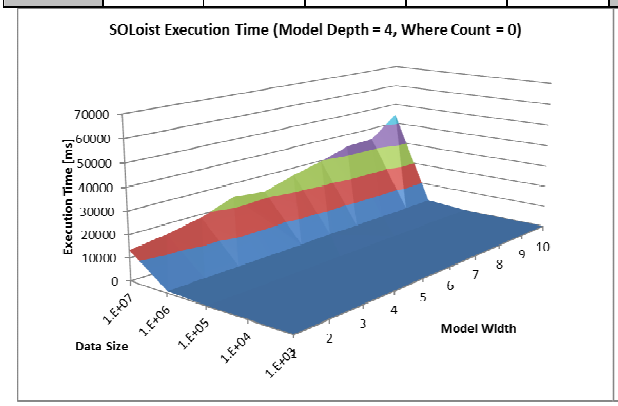
5	10	22	158	1,364	27,719	5	10	32	257	3,305	43,189
6	11	23	184	1,424	30,876	6	12	36	261	3,940	51,751
7	12	26	162	1,498	31,579	7	12	39	316	4,284	58,626
8	14	30	177	1,627	40,095	8	12	44	369	5,092	108,454
9	16	41	195	2,049	39,673	9	12	49	427	6,156	159,480
10	17	37	197	1,831	51,722	10	12	55	483	6,393	262,079

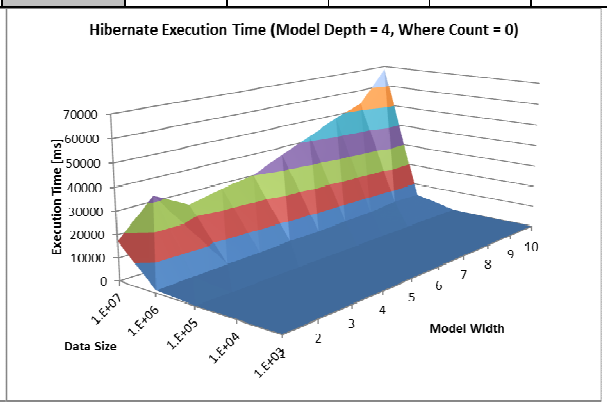




Execution Time (Model Depth = 4, Where Count = 0, DB = Oracle)

SOLoist	Execution Time (Model Depth = 4, Where Count = 0)					Hibernate	Execution Time (Model Depth = 4, Where Count = 0)				
	Data Size						Data Size				
Model Width	1.E+03	1.E+04	1.E+05	1.E+06	1.E+07	Model Width	1.E+03	1.E+04	1.E+05	1.E+06	1.E+07
1	6	15	91	897	13,376	1	9	20	123	1,245	17,204
2	7	19	126	908	15,485	2	6	22	141	1,514	32,407
3	7	19	94	974	18,538	3	6	24	154	1,960	24,251
4	8	18	109	1,014	23,845	4	9	31	202	2,213	27,673
5	9	19	117	1,087	22,315	5	7	29	202	2,792	30,078
6	12	23	135	1,417	26,486	6	8	33	203	2,770	36,469
7	13	24	129	1,153	29,673	7	8	32	235	3,197	42,204
8	15	27	132	1,297	33,907	8	8	40	275	3,631	48,394
9	16	53	144	1,310	34,485	9	8	38	313	4,658	53,141
10	18	31	164	1,369	44,487	10	8	42	346	4,414	68,656

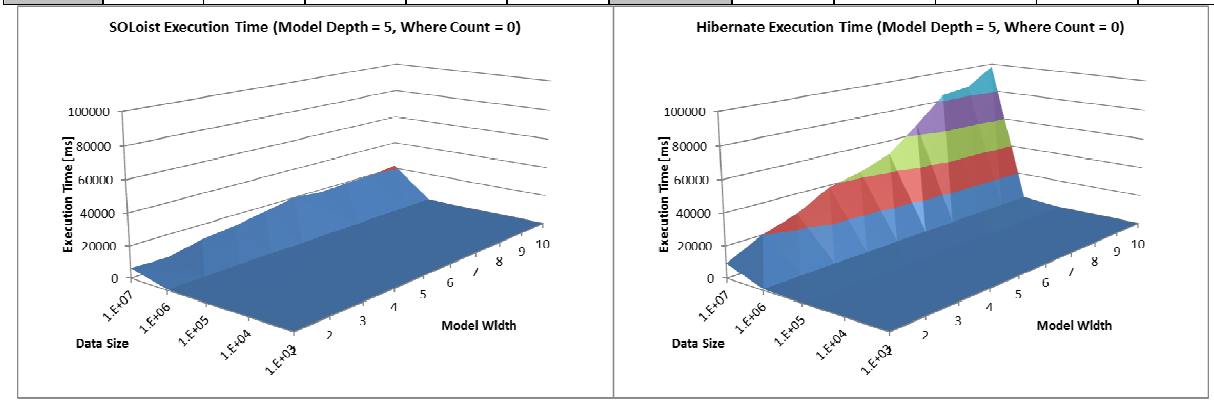




Execution Time (Model Depth = 5, Where Count = 0, DB = Oracle)

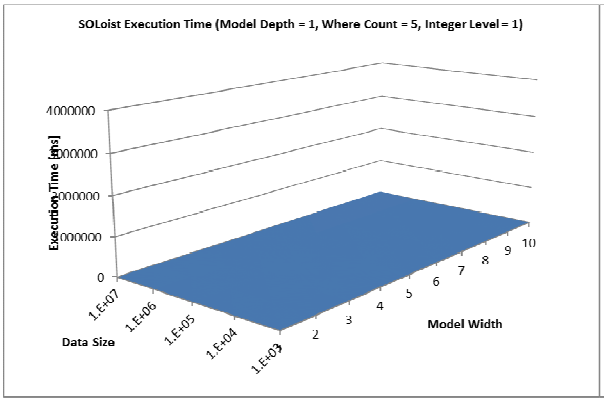
SOLoist	Execution Time (Model Depth = 5, Where Count = 0)	Hibernate	Execution Time (Model Depth = 5, Where Count = 0)
	Data Size		Data Size

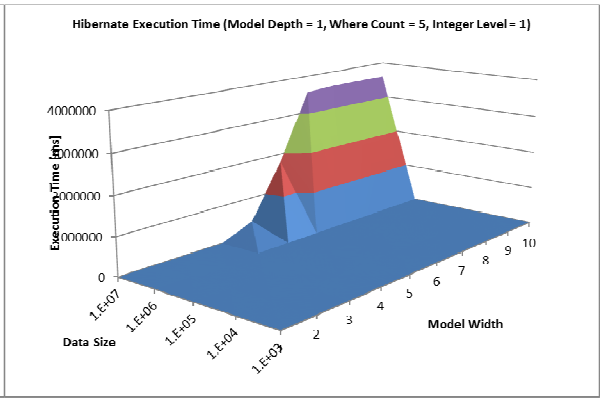
Model Width	1.E+03	1.E+04	1.E+05	1.E+06	1.E+07	Model Width	1.E+03	1.E+04	1.E+05	1.E+06	1.E+07
1	7	12	57	530	6,251	1	3	16	83	781	9,141
2	5	14	68	530	5,704	2	4	19	86	1,038	20,252
3	7	14	59	830	10,407	3	5	23	95	1,347	27,220
4	8	15	65	679	12,705	4	6	20	103	1,396	38,719
5	9	16	70	606	16,266	5	8	23	118	1,802	43,345
6	13	19	80	623	20,047	6	10	24	127	1,821	50,720
7	14	19	82	684	18,095	7	12	27	134	2,006	66,656
8	15	23	85	710	19,376	8	11	29	147	2,202	83,807
9	15	27	98	733	19,657	9	14	31	160	2,339	86,328
10	17	30	86	757	21,896	10	14	32	169	2,547	97,908



Execution Time (Model Depth = 1, Where Count = 5, Integer Level = 1, DB = Oracle)

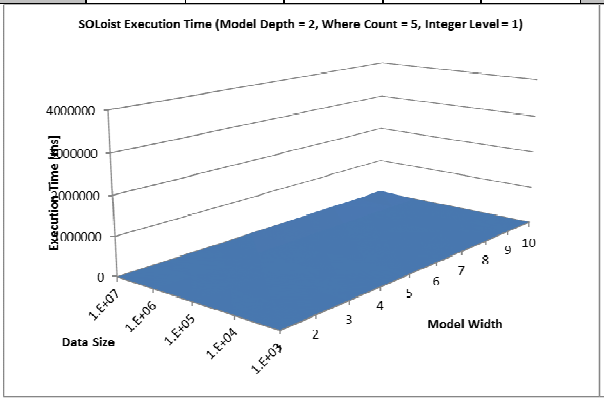
SOLoist	Execution Time (Model Depth = 1, Where Count = 5, Integer Level = 1)					Hibernate	Execution Time (Model Depth = 1, Where Count = 5, Integer Level = 1)				
	Data Size						Data Size				
Model Width	1.E+03	1.E+04	1.E+05	1.E+06	1.E+07	Model Width	1.E+03	1.E+04	1.E+05	1.E+06	1.E+07
1						1					
2						2					
3						3					
4						4					
5	9	13	41	317	4,977	5	4	12	106	1,398	344,580
6	11	15	43	326	6,501	6	5	15	118	1,587	1,778,298
7	12	19	47	357	9,235	7	3	17	135	1,783	3,553,173
8	13	21	52	393	11,610	8	5	26	137	1,770	3,600,000
9	15	23	60	422	14,657	9	5	29	148	1,929	3,600,000
10	18	24	65	459	18,315	10	4	28	286	2,090	3,600,000

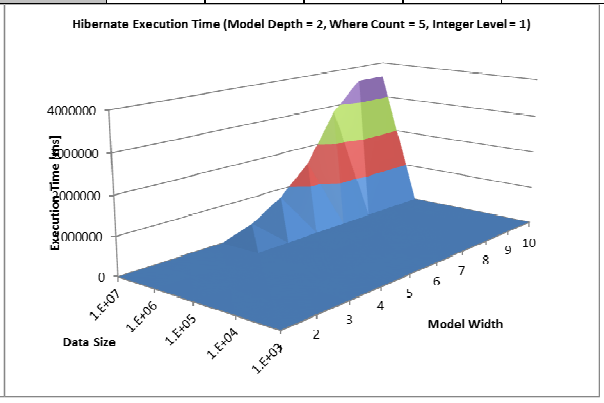




Execution Time (Model Depth = 2, Where Count = 5, Integer Level = 1, DB = Oracle)

SOLoist	Execution Time (Model Depth = 2, Where Count = 5, Integer Level = 1)					Hibernate	Execution Time (Model Depth = 2, Where Count = 5, Integer Level = 1)				
	Data Size						Data Size				
Model Width	1.E+03	1.E+04	1.E+05	1.E+06	1.E+07	Model Width	1.E+03	1.E+04	1.E+05	1.E+06	1.E+07
1						1					
2						2					
3						3					
4						4					
5	9	14	40	314	8,283	5	3	11	93	1,165	269,486
6	11	16	46	401	10,439	6	5	12	102	1,430	773,766
7	12	19	56	455	17,532	7	3	17	114	1,396	1,528,847
8	13	22	59	497	23,641	8	5	22	119	1,391	2,833,844
9	15	23	64	548	20,016	9	5	24	123	1,508	3,600,000
10	16	25	73	590	28,957	10	5	13	149	1,514	3,600,000

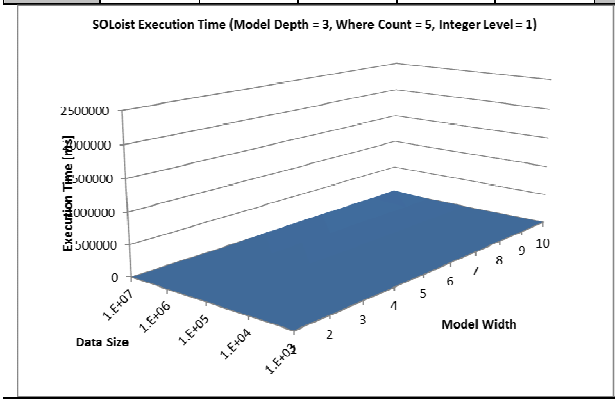


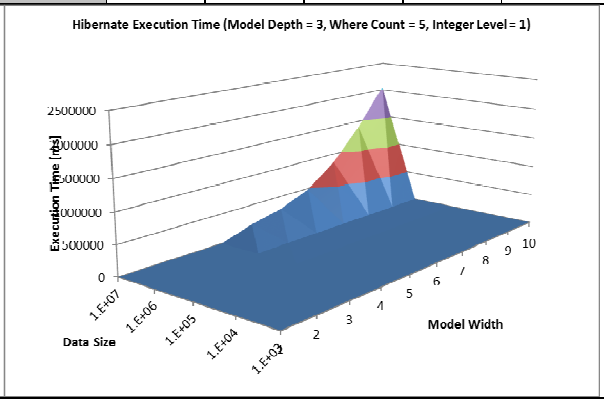


Execution Time (Model Depth = 3, Where Count = 5, Integer Level = 1, DB = Oracle)

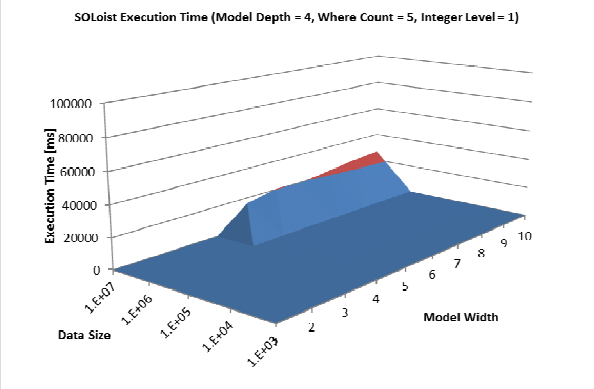
SOLoist	Execution Time (Model Depth = 3, Where Count = 5, Integer Level = 1)					Hibernate	Execution Time (Model Depth = 3, Where Count = 5, Integer Level = 1)				
	Data Size						Data Size				
Model Width	1.E+03	1.E+04	1.E+05	1.E+06	1.E+07	Model Width	1.E+03	1.E+04	1.E+05	1.E+06	1.E+07
1						1					
2						2					

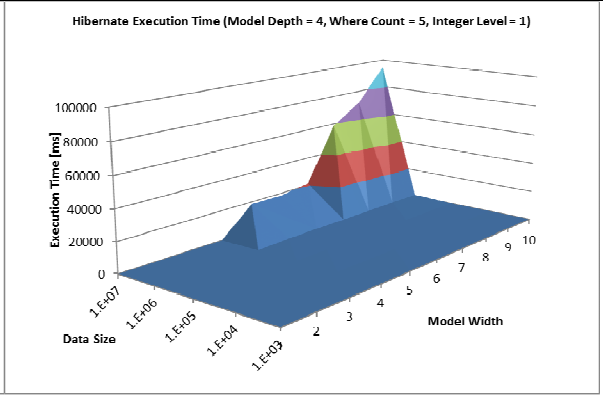
3						3					
4						4					
5	13	16	51	376	16,860	5	3	10	73	942	165,735
6	15	23	54	410	17,407	6	5	9	81	905	278,063
7	17	22	54	458	24,626	7	3	10	87	1,129	479,985
8	18	24	59	479	24,391	8	4	14	89	1,031	851,486
9	20	29	62	511	21,923	9	6	17	96	1,122	1,390,329
10	23	29	70	548	33,301	10	5	13	155	1,158	2,047,126





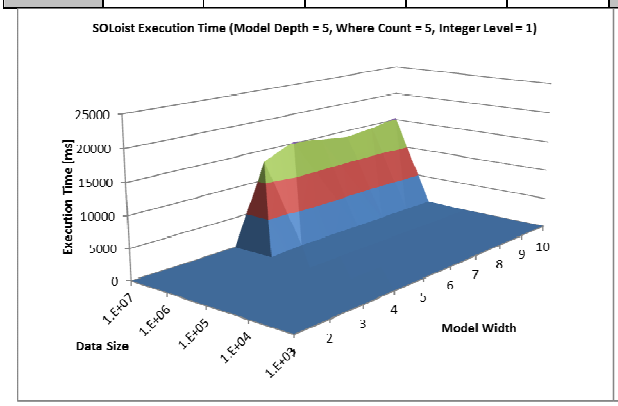
SOLoist	Execution Time (Model Depth = 4, Where Count = 5, Integer Level = 1)					Hibernate	Execution Time (Model Depth = 4, Where Count = 5, Integer Level = 1)				
	Data Size						Data Size				
Model Width	1.E+03	1.E+04	1.E+05	1.E+06	1.E+07	Model Width	1.E+03	1.E+04	1.E+05	1.E+06	1.E+07
1						1					
2						2					
3						3					
4						4					
5	14	17	50	371	16,236	5	4	7	47	723	18,516
6	15	21	49	401	20,454	6	4	16	62	811	18,297
7	18	22	52	416	19,782	7	4	17	57	759	21,329
8	19	23	57	479	22,345	8	4	17	89	818	58,688
9	22	30	59	485	24,860	9	5	17	70	1,013	71,407
10	22	31	69	512	26,694	10	5	9	75	910	94,513

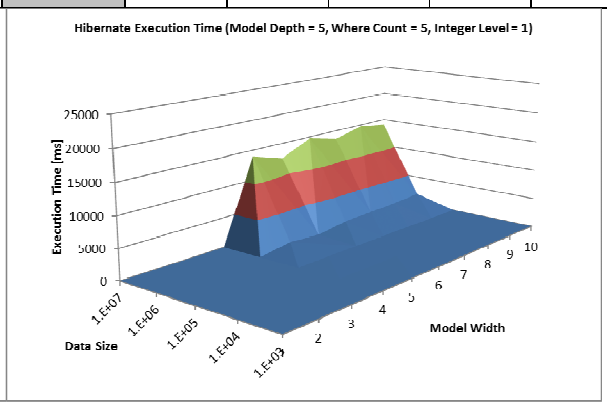




Execution Time (Model Depth = 5, Where Count = 5, Integer Level = 1, DB = Oracle)

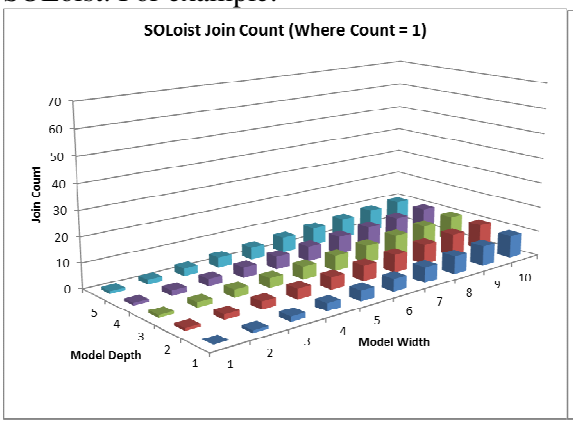
SOLoist	Execution Time (Model Depth = 5, Where Count = 5, Integer Level = 1)					Hibernate	Execution Time (Model Depth = 5, Where Count = 5, Integer Level = 1)				
	Data Size						Data Size				
Model Width	1.E+03	1.E+04	1.E+05	1.E+06	1.E+07	Model Width	1.E+03	1.E+04	1.E+05	1.E+06	1.E+07
1						1					
2						2					
3						3					
4						4					
5	10	20	39	254	13,094	5	3	5	44	783	13,954
6	11	20	43	264	15,298	6	3	23	33	414	12,391
7	16	26	45	276	14,533	7	2	24	36	1,141	14,767
8	16	24	47	282	13,907	8	3	25	57	1,278	13,538
9	20	26	49	290	14,376	9	3	27	41	1,422	14,736
10	23	27	54	299	15,367	10	3	17	43	1,746	14,023

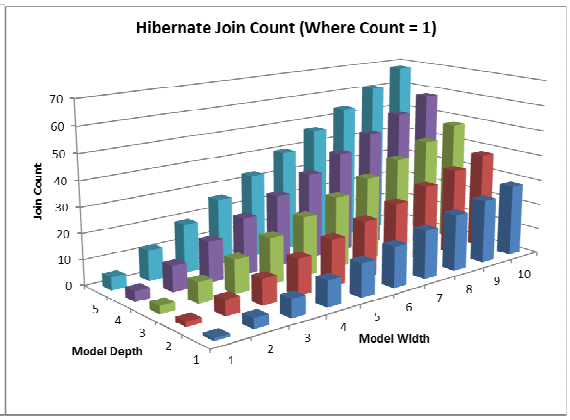




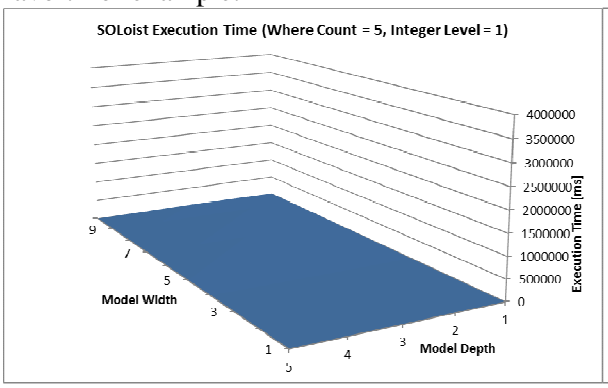
Conclusions

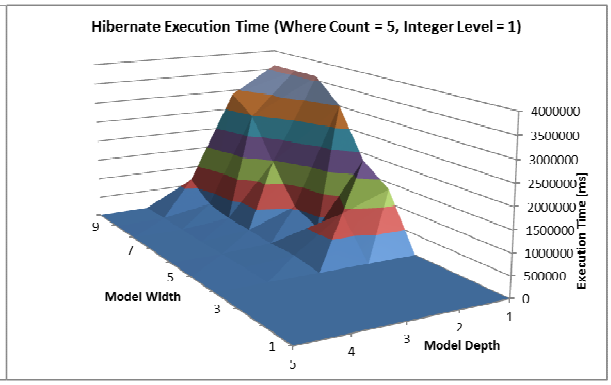
In most cases, SOLoist OQL produces SQL queries with several times fewer joins than Hibernate HQL. In very rare cases of simple queries, the numbers of joins are comparable, while SOLoist is never worse than Hibernate. The more complex the queries, the bigger the difference in favor of SOLoist. For example:





In almost all cases, SOLoist OQL outperforms Hibernate HQL from a few times to a few orders of magnitude. It scales better with the complexity of the object query, especially for large databases. In very rare cases of simple queries, the performance is comparable, while SOLoist is never worse than Hibernate. The more complex the queries, the bigger the difference in SOLoist's favor. For example:





SOLoist OQL scales better than Hibernate HQL with the size of the database, for all kinds of queries. The more complex the queries, the bigger the difference in SOLoist's favor. For example: