INSTITUTIONEN FÖR DATAVETENSKAP | LUNDS TEKNISKA HÖGSKOLA | PRESENTATIONSDAG 2018-11-01

EXAMENSARBETE Migration & Evaluation of Automatic Query Hint Generation in Persistent Systems
STUDENT Erik Jonasson
HANDLEDARE Per Andersson (LTH), Arnaud Fietzke (itestra GmbH)
EXAMINATOR Flavius Gruian (LTH)

Automating Optimization Procedure for Relational Databases

POPULÄRVETENSKAPLIG SAMMANFATTNING **Erik Jonasson**

Optimizing how and when data should be fetched from the database can be a difficult and frustrating task. The bigger the application, the more difficult it becomes. Luckily, there is a way to automate this process.

In today's software development, it is common to use so called *Object-relational-mappers*. What this software does is that it creates an extra layer between the database and the application code. The software will then take care of all database manipulation, translating between the two different worlds of object oriented program code and the relational database.

This type of software can be difficult to configure accordingly. The options are many, and the not so experienced developer can run into many misconfigurations that will cause the system performance to drop drastically.

The difficulty comes when the user has to define the *fetch strategy* for associations of entities in the database. This leaves a lot of room for customization, but also for errors. The common fetch strategies are:

- **Eager loading**: When loading this object, *eagerly* annotated associations to this object will be loaded immidiatelly.
- **Lazy loading**: When loading this object, *lazily* annotated associations will be loaded only when used in the code.

Lazy loading can seem like a better option, since the object will only get loaded when it is being accessed in the code. However, this will create a query to the database for each time when a new object or field is accessed that is not yet in memory. If the database in on a server with latency for example, this could lead to heavily decreased performance. Moreover, manually handling these settings is not optimal from a Software Engineering standpoint.

Due to these problems, we migrated and evaluated an existing tool that automates the prefetching based on statistics of the objects in the database. Since this tool was created for a legacy version of *Hibernate*, we decided to migrate the old tool to the newest version of Hibernate and then evaluate the method on a project of the software consulting company **itestra GmbH**.

The migration introduced new integration mechanisms to make it easier to use the tool. The tool decreases the amount of executed queries by 11% in our tests compared to the version loaded lazily. However, the handtuned version still performs considerably better. On the other hand, with the *Software engineering* benefits that the tool offers, the methodology has potential to become a viable option for anyone who wants to improve performance of the ORM-tool without wasting valuable development time.