



# Developer Test Task

Nortal Summer University 2017

## Jack's Garage

### 1. The Story

Jack has always had an entrepreneur mindset but he also is persistent in doing everything by himself and not asking for help. It took him years before he hired a mechanic so he could work on managing his shop instead of working on the cars. Once he had some time on his hands he developed a booking system for his garage. He called the application "Jack's Garage booking". It was a desktop application allowing him to get an overview of the available times. Years passed and now he has two mechanics working on two repair stands in his garage. While developing the application he didn't see this coming and the data model does not support that well. Additionally, he has encountered several issues with the current model he is using.

### 2. Assignment

Help Jack to refactor current data model, making sure that the existing data is retained.

You have 3 major tasks:


- Change the existing database structure
- Migrate existing data to the new structure
- Implement queries for reports to prove Jack that he can achieve what he needs with this new structure

All these tasks need to be solved within corresponding files under: `sql-jacks-garage/src/main/resources/db/assignment`



## 2.1. Current structure

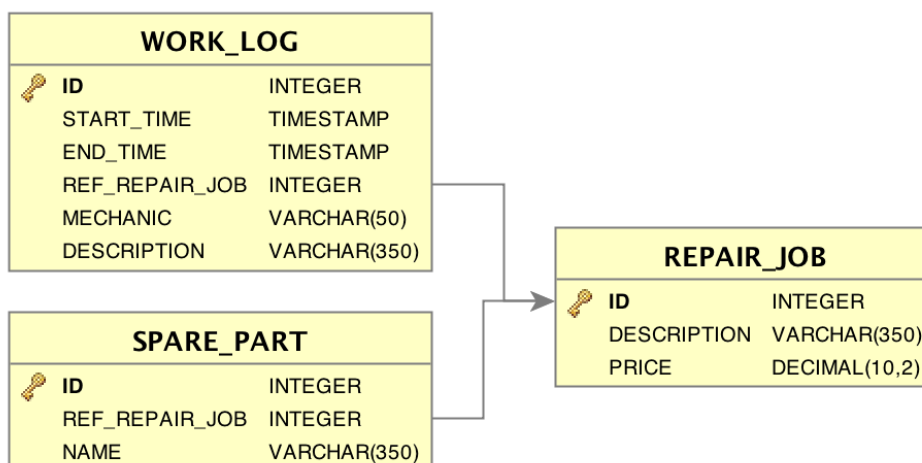
The current data model of his application can be seen on the right. Each job has a start time, end time, semicolon separated list of parts required, description and agreed price.

REPAIR_JOB	
 ID	INTEGER
START_TIME	TIMESTAMP
END_TIME	TIMESTAMP
PARTS	VARCHAR(35)
DESCRIPTION	VARCHAR(350)
PRICE	DECIMAL(10,2)

## 2.2. Expected structure

Repair job can be done in multiple sessions by different mechanics. To allow this, the WORK\_LOG table will contain data about working sessions for a job.

At the moment Jack has also difficulties in warehouse management since the required spare parts have only one field in the database. On some occasions these have been separated by comma or semicolon.





## 2.3. Data migration

Existing data migration from REPAIR\_JOB -> WORK\_LOG should be done like this

WORK_LOG column	column value	Comment
START_TIME	REPAIR_JOB.START_TIME	
END_TIME	REPAIR_JOB.END_TIME	
DESCRIPTION	REPAIR_JOB.DESCRPTION	For existing data this value will be duplicated, since each REPAIR_JOB will have exactly one WORK_LOG created. Not to be removed from REPAIR_JOB
REF_REPAIR_JOB	REPAIR_JOB.ID	
MECHANIC	"Mike Mechan"	Constant value (until now just one mechanic hired)

Existing data migration from REPAIR\_JOB -> SPARE\_PART should be done like this.

SPARE_PART column	column value	Comment
REF_REPAIR_JOB	REPAIR_JOB.ID	
NAME	REPAIR_JOB.PARTS	The existing values do not need to be split in any way. One REPAIR_JOB record creates one SPARE_PART row

## 3. Set up

### 3.1. Gradle

Remember that gradle commands can be executed in three different ways:

1. When you have Gradle installed: *gradle <command>*
2. When working on Windows command line or PowerShell: *gradlew.bat <command>*



3. When working on MacOS, Linux or Windows 10 Shell: `./gradlew <command>`  
In current guide all examples will use the first format that you can adapt if needed.

## 3.2. Database

This task requires you to open up two terminal or command line windows. To start the database use `"gradle startDb"`. This command will start HSQLDB server and keeps it running - the command does not exit automatically.

To start a fresh copy of the database, you can use `"gradle cleanDb startDb"`. The cleanDb command will remove your existing database and requires the HSQLDB server to be stopped. Note that this will also remove all modifications you have manually applied directly to the database

To view the database structure you can use `"gradle startDbManager"` but you can also use other Database Management Software such as Cubrid, Firebird or DbVisualizer.

## 3.3. Liquibase

In another terminal window you can use `"gradle updateDb"` that instructs Liquibase library to connect to your local database and apply (in alphabetical order) all the changes it finds under: `"sql-jacks-garage/src/main/resources/db"` This is also where your assignment resides. If successfully executed, Liquibase does not apply same file twice. It also verifies the hash of the file, so you cannot edit it once it is executed against the database. You can edit it and re-execute if some error occurred during the last run.