ElaSQL

Getting Started

June 12th 2021

elasql.org

Outline

- Introduction to ElaSQL project
- How to test/benchmark the system?

Outline

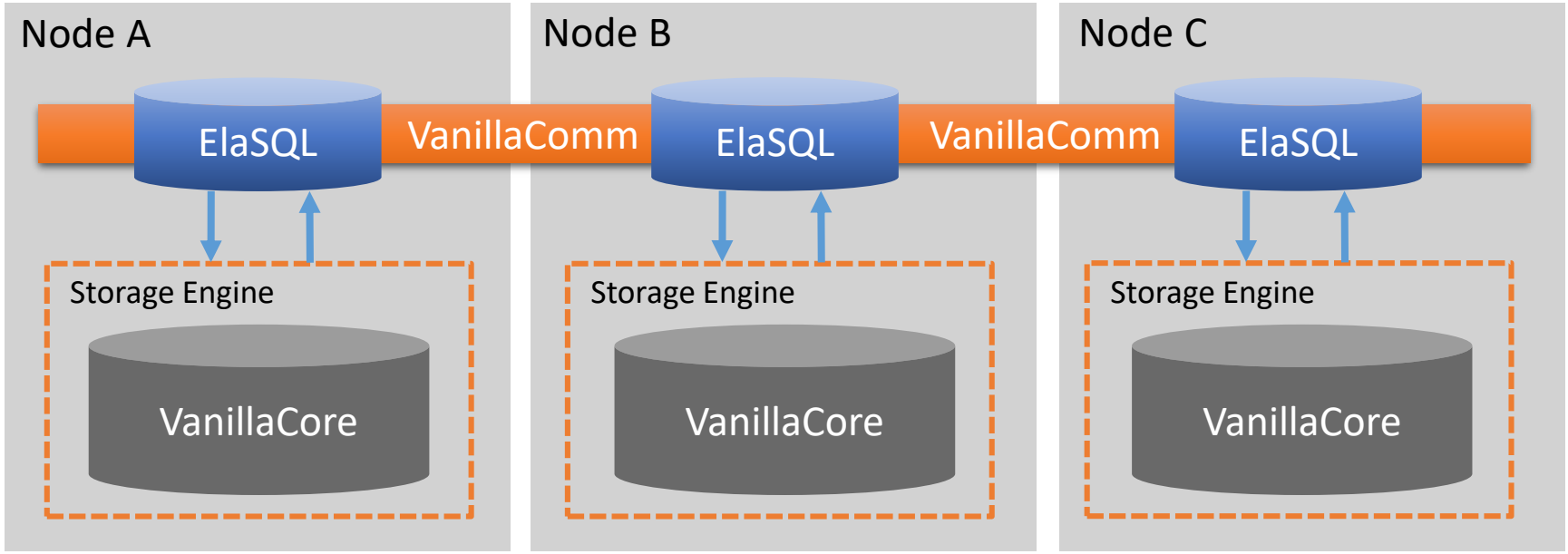
- Introduction to ElaSQL project
  - What is ElaSQL?
  - Architecture
  - Design & Key Features
  - Implemented Systems & Papers
  - The Sequencer
- How to test/benchmark the system?

ElaSQL

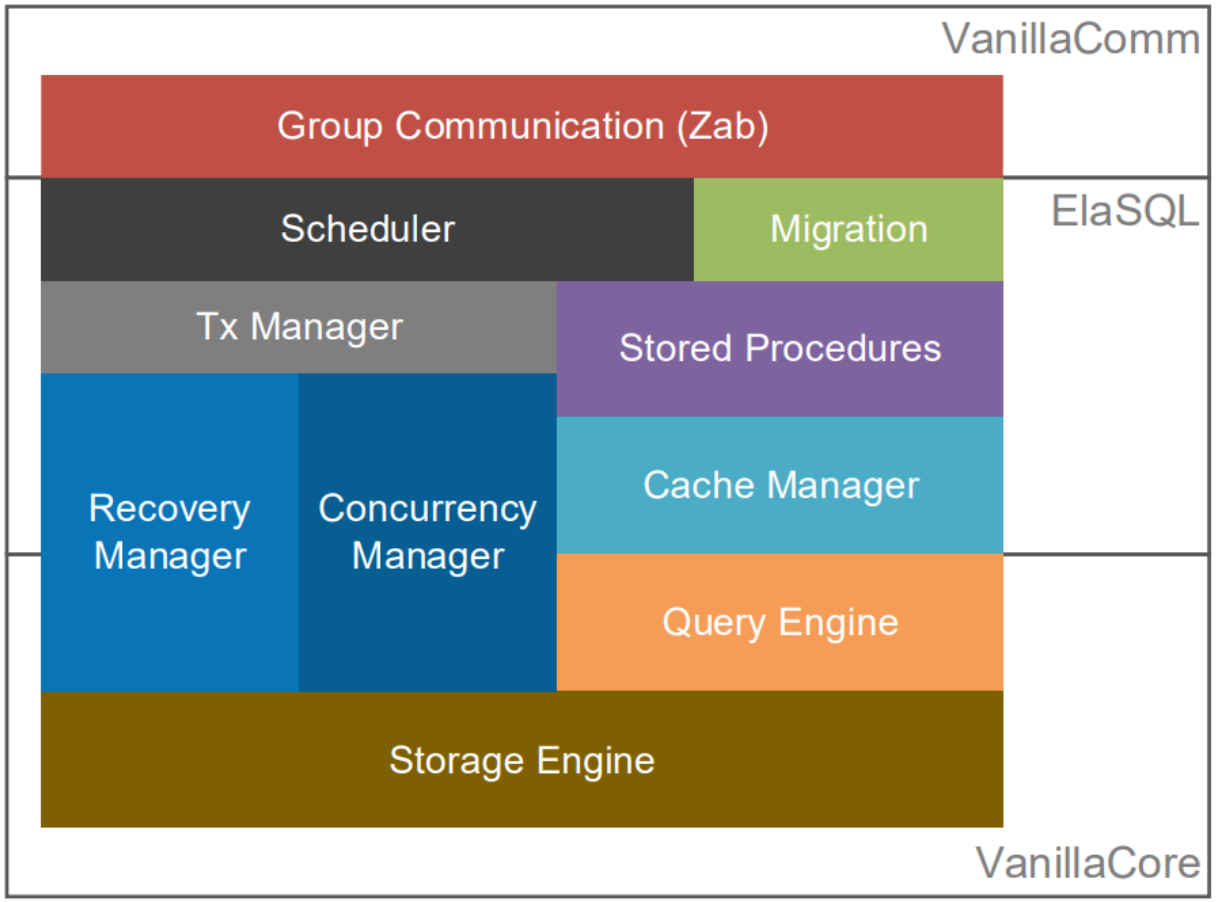
- ElaSQL is a distributed relational database system that aims to provide
  - high scalability
  - high availability
  - elasticity
- ElaSQL can be tested with ElaSQL-Bench, which is a benchmark tool that generate pressure to the system.
  - Currently, this is the only way to test ElaSQL.

The Relationship with VanillaDB

- ElaSQL is built on top of two projects of VanillaDB
  - VanillaComm: as the communication module
  - VanillaCore: as the storage engine for each machine



Architecture inside a Machine



A Deterministic Database System

- ElaSQL is a deterministic database system, which is based on the idea of the following paper:
  - Thomson, Alexander, and Daniel J. Abadi. "The case for determinism in database systems." *Proceedings of the VLDB Endowment* 3.1-2 (2010): 70-80.

- With determinism, ElaSQL can ensure a database always reach the same state from the same initial state with the same sequence of requests.

Key Features

- Strong Consistency with high availability
  - ElaSQL uses determinism to ensure consistency without relying on two phase commit.
- High Scalability
  - ElaSQL partitions a database to distribute the loads to multiple machines.
- Elasticity
  - ElaSQL implements several data migration and re-partitioning algorithm to ensure that data partitions are always up to date.

Implemented Systems

- Since ElaSQL is a research prototype, we have implemented several algorithms and systems proposed in research papers in ElaSQL.

- Please check this list for available systems and algorithms and corresponding papers.

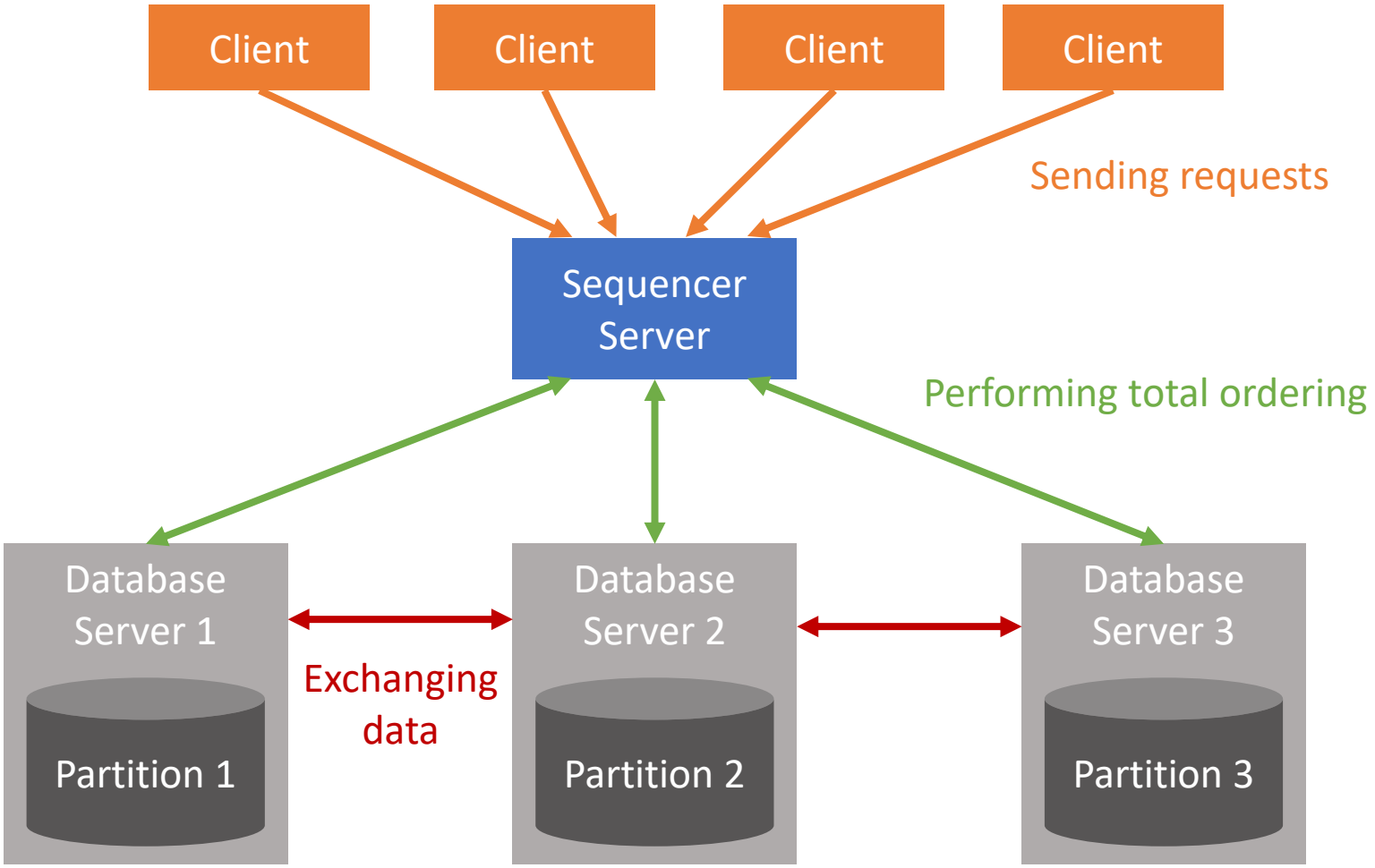
The Sequencer (ZAB Leader)

- A deterministic database system requires a total-ordering protocol to ensure the order of transactions across machines in advance.

- We implement Zookeeper Atomic Message Broadcast (ZAB) Protocol for this.
  - Which requires a machine to be the leader.
  - We call the leader as **the sequencer**.

- In our design, the sequencer is one of servers in a cluster, but it does not have database functionality.

Message Flow



Outline

- Introduction to ElaSQL project
- How to test/benchmark the system?
  - Let's meet ElaSQL-Bench
  - Setting up development environment
  - Testing inside a Java IDE
  - Testing with runnable JARs
  - Testing in a cluster

Outline

- Introduction to ElaSQL project
- How to test/benchmark the system?
  - Let's meet ElaSQL-Bench
  - Setting up development environment
  - Testing inside a Java IDE
  - Testing with runnable JARs
  - Testing in a cluster

ElaSQL-Bench

- In order to test how ElaSQL performs under an extreme circumstance, we implement a benchmark tool.
  - Which is based on another project, VanillaBench.

- This project includes two standard benchmarks:
  - The TPC-C Benchmark
  - The Yahoo! Cloud Serving Benchmark (YCSB)

Outline

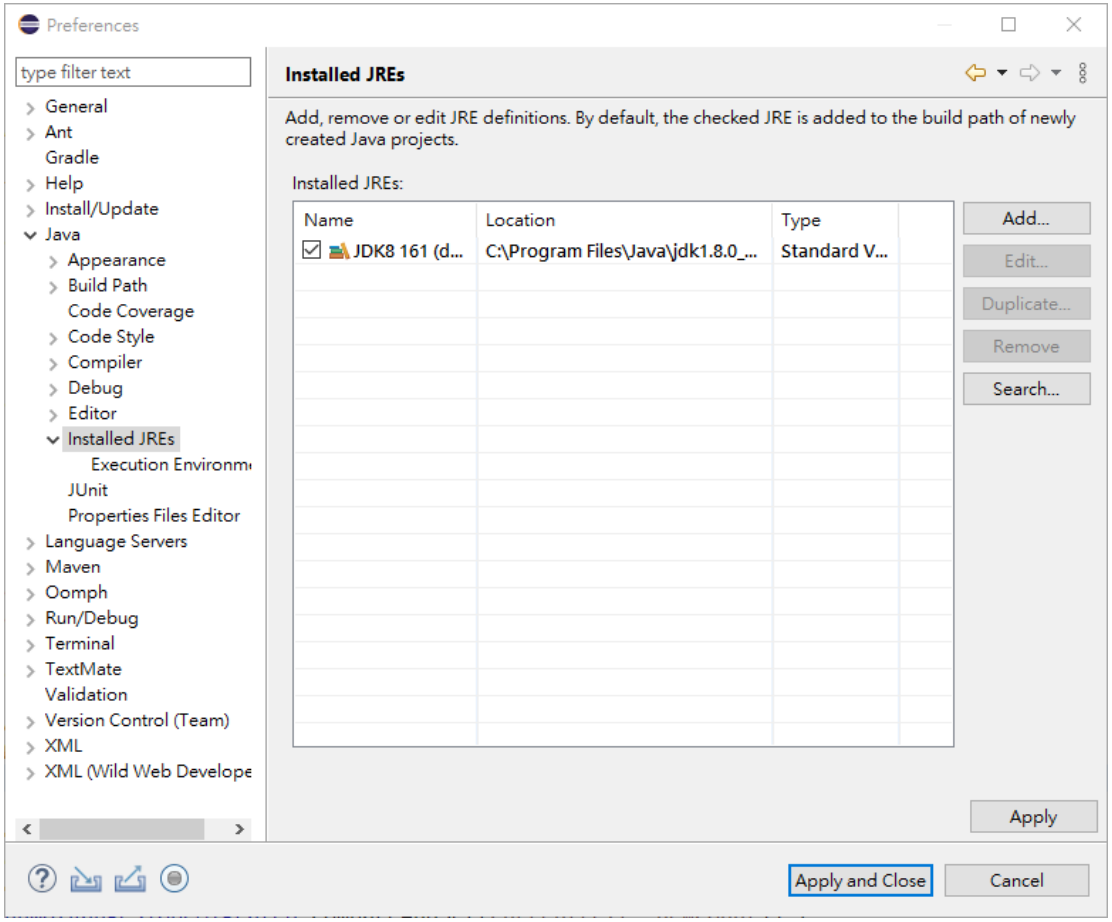
- Introduction to ElaSQL project
- How to test/benchmark the system?
  - Let's meet ElaSQL-Bench
  - Setting up development environment
  - Testing inside a Java IDE
  - Testing with runnable JARs
  - Testing in a cluster

Prerequisite

- We assume that you have the following programs in your environment.
  - Java Development Kit (JDK) **8**
    - We found some problems when running with JDK 10+. You may try, but there is no guarantee to work.
  - Eclipse
    - You may use another IDE, but we will demonstrate the following tasks in Eclipse.
  - Git
  - Bash

Changing Your JDK in Eclipse

You can change the JDK that your Eclipse uses in “Preferences”.



Steps to Setup Your Dev. Env.

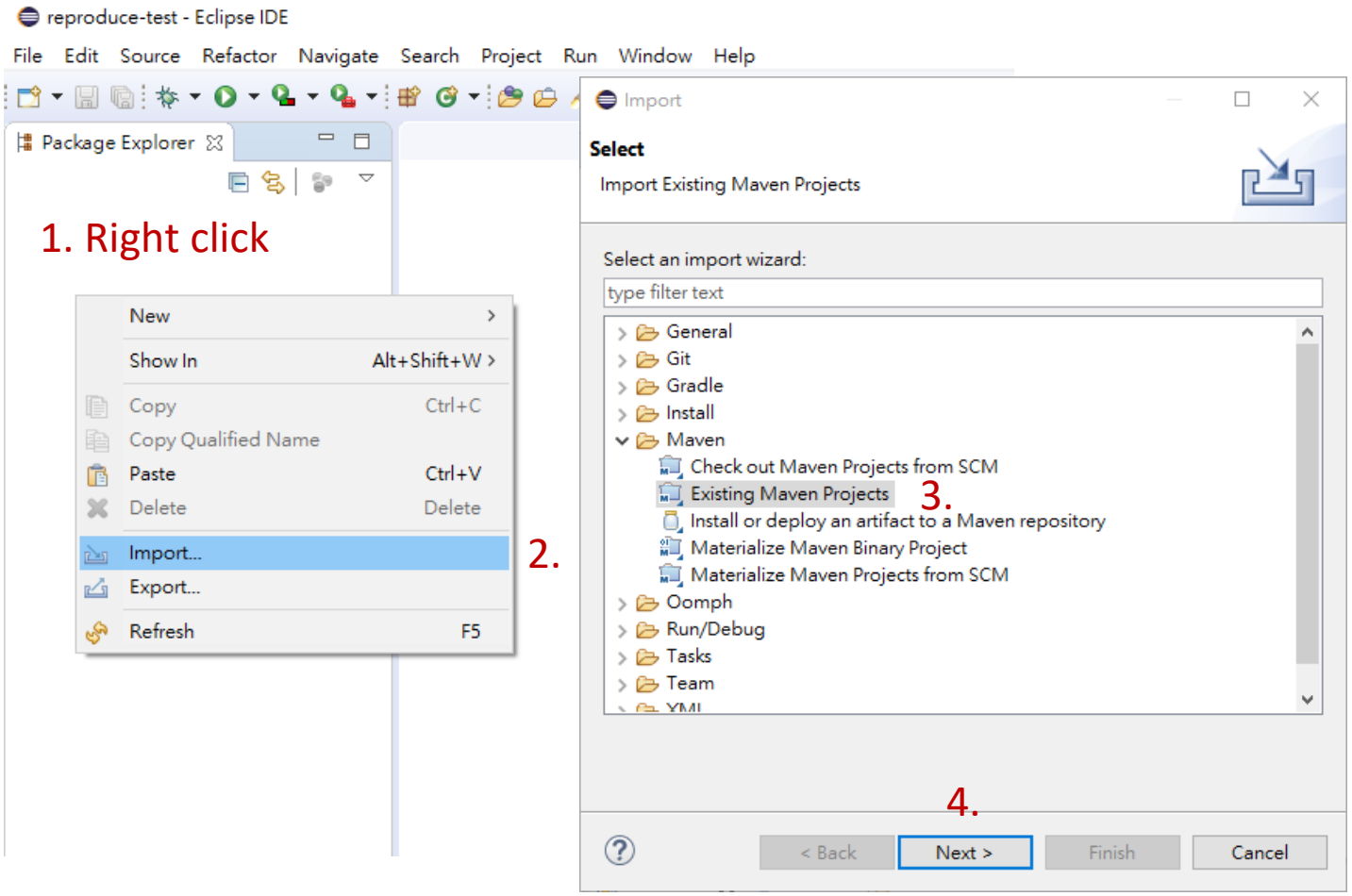
1. Clone ElaSQL and ElaSQL-Bench
2. Import the projects to Eclipse
3. Done

Cloning the Project

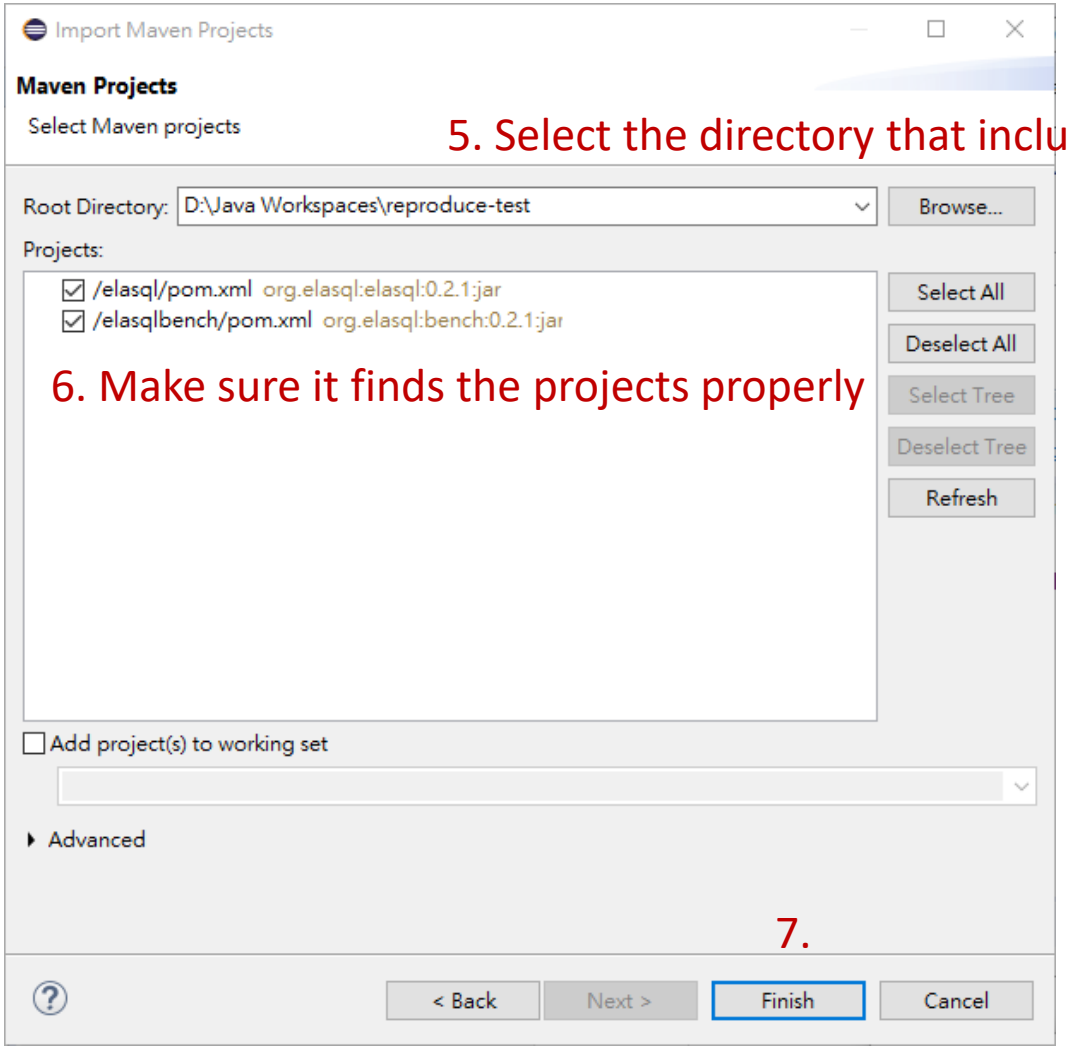
- Clone the following projects:
  - ElaSQL: https://github.com/elasql/elasql
  - ElaSQL-Bench: https://github.com/elasql/elasqlbench

- Checkout the branch you need
  - The default branch is "**master**".
  - However, if you want to reproduce certain experiments, you may want to checkout other branches.
  - For example, to reproduce MgCrab experiments, you may need to checkout "**reproduce/mgcrab**" branch.

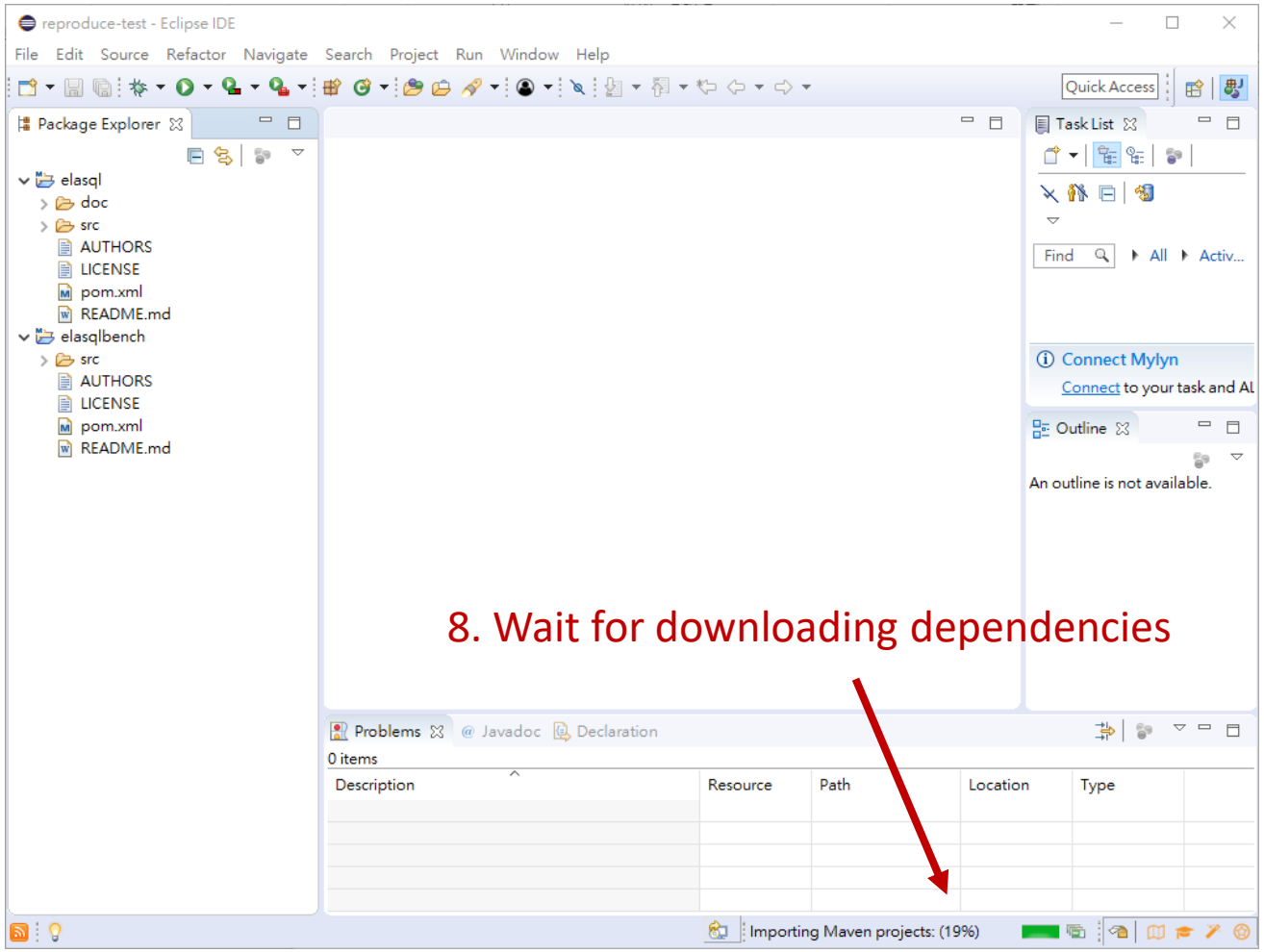
Importing into Eclipse



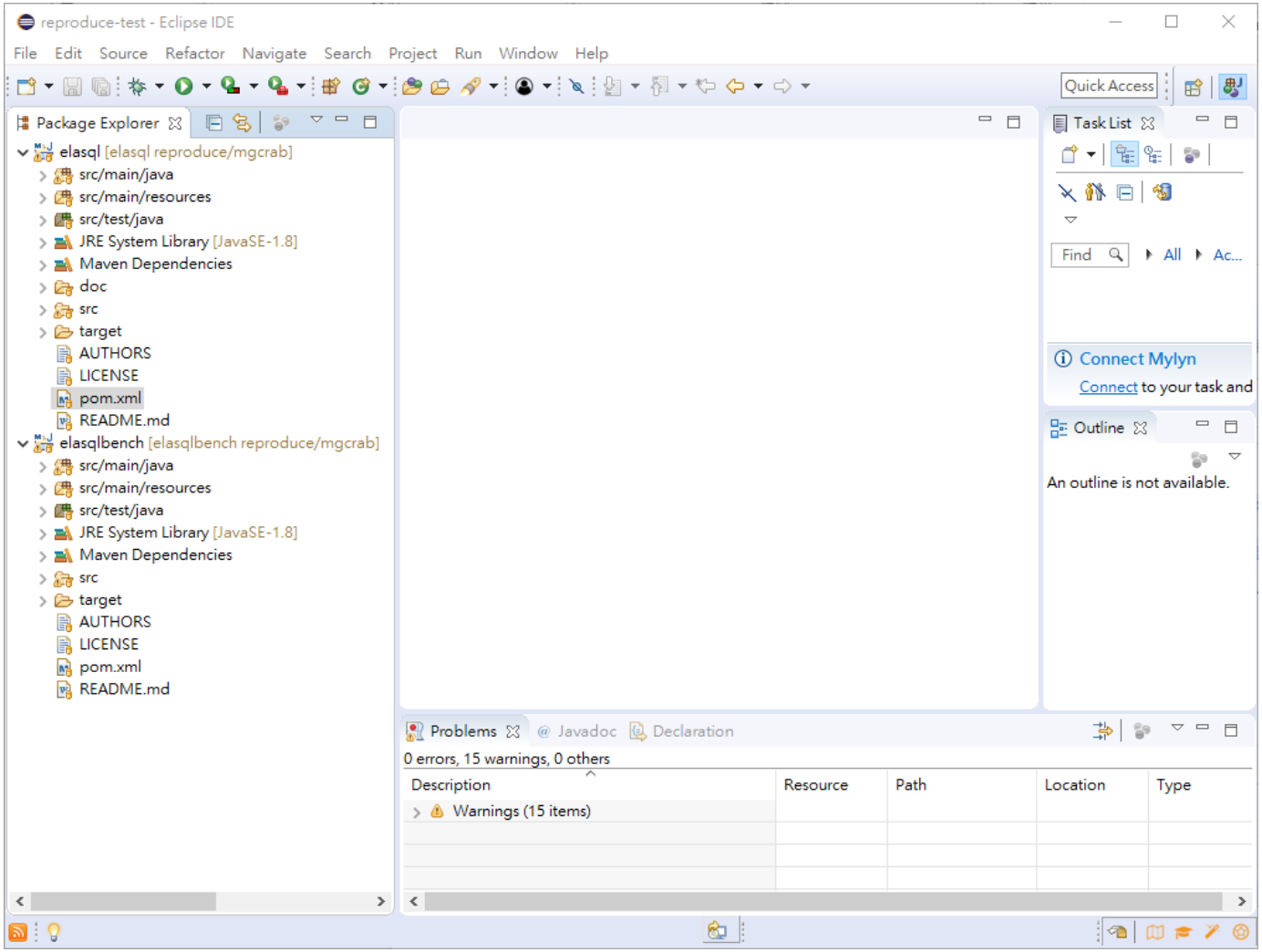
Importing into Eclipse



Importing into Eclipse



Done

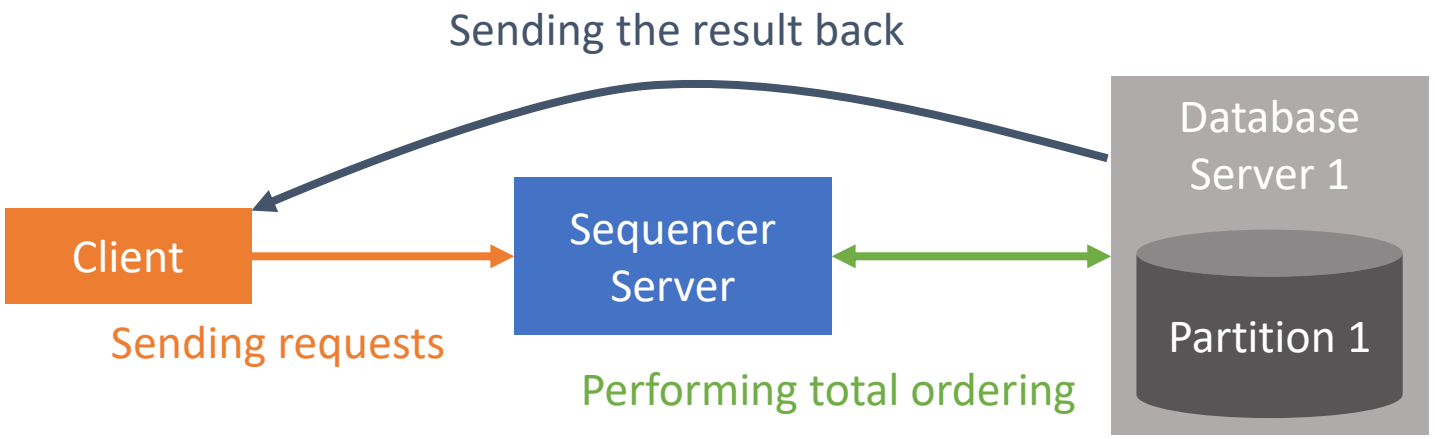

reproduce-test - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Package Explorer
elasql [elasql reproduce/mgcrab]
src/main/java
src/main/resources
src/test/java
JRE System Library [JavaSE-1.8]
Maven Dependencies
doc
src
target
AUTHORS
LICENSE
pom.xml
README.md
elasqlbench [elasqlbench reproduce/mgcrab]
src/main/java
src/main/resources
src/test/java
JRE System Library [JavaSE-1.8]
Maven Dependencies
src
target
AUTHORS
LICENSE
pom.xml
README.md
Task List
Find All Ac...
Connect Mylyn
Connect to your task and
Outline
An outline is not available.
Problems Javadoc Declaration
0 errors, 15 warnings, 0 others
Description Resource Path Location Type
Warnings (15 items)

Outline

- Introduction to ElaSQL project
- How to test/benchmark the system?
  - Let's meet ElaSQL-Bench
  - Setting up development environment
  - Testing inside a Java IDE
  - Testing with runnable JARs
  - Testing in a cluster

Testing Environments

- To launch a benchmarking test, at least three processes must be launched.
  - 1 Sequencer Server (the ZAB leader)
  - 1 Database Server
    - Adding more database servers can increase throughput
  - 1 Benchmark Client
    - Adding more clients can generate higher pressure to the system



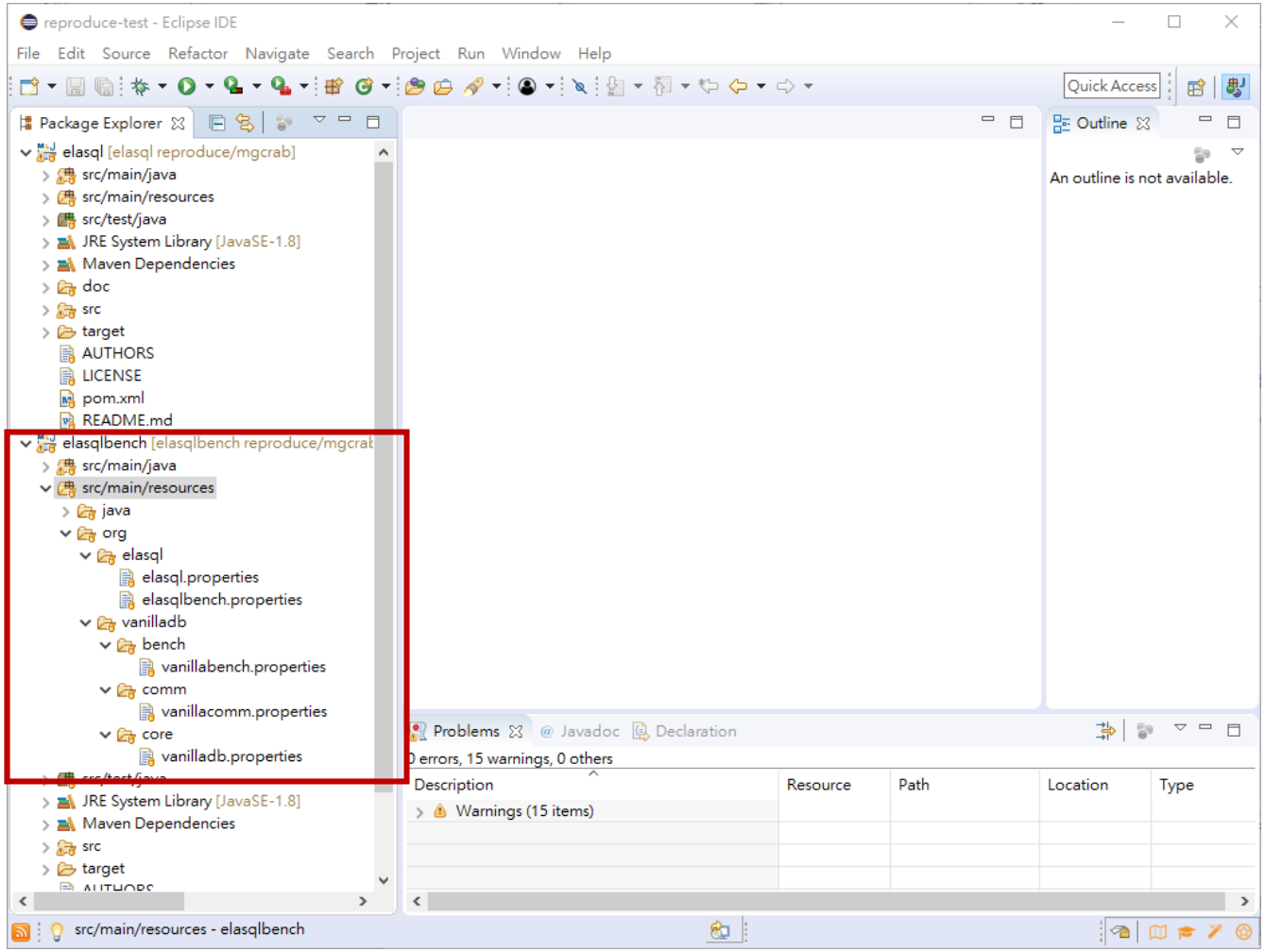
Testing inside Eclipse

1. Setup the properties files
   - Which includes the configurations for ElaSQL and ElaSQL-Bench
2. Setup run configurations
3. Loading a testbed
   1. Launch servers
   2. Launch clients
4. Benchmarking
   1. Launch servers
   2. Launch clients

Testing inside Eclipse

1. Setup the properties files
   - Which includes the configurations for ElaSQL and ElaSQL-Bench
2. Setup run configurations
3. Loading a testbed
   1. Launch servers
   2. Launch clients
4. Benchmarking
   1. Launch servers
   2. Launch clients

The Properties Files



Setting Up Network Addresses

- ElaSQL uses VanillaComm to communicate through networks.

- We need to tell VanillaComm where to find all the machines (including servers and clients).
  - The addresses should be put in vanillacomm.properties

Setting Up Network Addresses

- Here is an example to setup the addresses for 2 servers and 1 client.
  - The last server will become the sequencer.

vanillacomm.properties

```
1 #
2 # VanillaDB Comm configuration file
3 #
4 # This file is a single place for controlling all constant fields defined in
5 # VanillaDB Communication Module classes. The path of this file should be set as a system property
6 # keyed "org.vanilladb.comm.config.file" so the content will to be
7 # processed during VanillaDB Comm initiation.
8 #
9
10 #
11 # Module general settings
12 #
13
14 # The views of the machine
15 # A machine is represented by "ID IP PORT"
16 # Each machine is split by a comma (,)
17 org.vanilladb.comm.view.ProcessView.SERVER_VIEW=0 127.0.0.1 42961, 1 127.0.0.1 42962
18 org.vanilladb.comm.view.ProcessView.CLIENT_VIEW=0 127.0.0.1 30000
19
```

The Sequencer = Server No.1

Setting Up The Storage Engine

- ElaSQL uses VanillaCore as a storage engine to store data on each machine.
  - vanillacore.properties includes the configurations for the storage engine.
- Most configurations have been tuned for benchmarking.
  - Only some of them should be checked carefully.

Setting Up The Storage Engine

vanilladb.properties

```
25
26 #
27 # File package settings
28 #
29
30 # The number of bytes in a block.  A common value is 4K.
31 org.vanilladb.core.storage.file.Page.BLOCK_SIZE=4096
32 # The parent directory of database files.
33 org.vanilladb.core.storage.file.FileMgr.DB_FILES_DIR=
34 # The directory of log files.
35 org.vanilladb.core.storage.file.FileMgr.LOG_FILES_DIR=
36 org.vanilladb.core.storage.file.io.IoAllocator.USE_O_DIRECT=false
37
38
39 #
40 # Buffer package settings
41 #
42
43 # The maximum waiting time for pinning a buffer. Original value is 10 seconds.
44 org.vanilladb.core.storage.buffer.BufferMgr.MAX_TIME=10000
45 # The epsilon value for tuning waiting time.
46 org.vanilladb.core.storage.buffer.BufferMgr.EPSILON=50
47 # The size of buffer pool (default 1GB).
48 org.vanilladb.core.storage.buffer.BufferMgr.BUFFER_POOL_SIZE=1048576
49
50
51 #
52 # Log package settings
53 #
54
55 # The name of vanilladb's log file.
```

where to put the database files
(default: home directory)

O_DIRECT should be used if the system runs on Linux environments

controls how much data are cached in memory
(note that this number means “the number of blocks”)

Setting Up ElaSQL (The Distributed Modules)

- ElaSQL also has many configurations:
  - How many data partitions are there?
  - Which system to run? Calvin? Hermes?
  - Which data migration algorithm to use?

- All these are put in elasql.properties
  - The file contains a comprehensive explanations for each parameter, so we will not go through all the parameters here.

Setting Up ElaSQL (The Distributed Modules)

elasql.properties

```
52
53
54 #
55 # Schedule package settings
56 #
57
58 # The stored procedure factory class of different types of scheduler
59 # Note that this is only used when no factory class is assigned.
60 org.elasql.schedule.naive.NaiveScheduler.FACTORY_CLASS=
61 org.elasql.schedule.calvin.CalvinScheduler.FACTORY_CLASS=
62
63
64 #
65 # Metadata package settings
66 #
67
68 # The number of data partitions.
69 # Usually, this should be the number of database servers.
70 org.elasql.storage.metadata.PartitionMetaMgr.NUM_PARTITIONS=1
71
72
73 #
74 # T-Part package settings
75 #
76
77 # How many requests are queued for processing at once.
78 org.elasql.schedule.tpart.TPartPartitioner.NUM_TASK_PER_SINK=10
79 # To control if T-Part should weight more on minimizing distributed transactions.
80 org.elasql.schedule.tpart.CostAwareNodeInserter.BETA=1.0
81 # The maximum size of the fusion table
82 # Note that the actual size may exceed this number at little bit.
83 org.elasql.schedule.tpart.hermes.FusionTable.EXPECTED_MAX_SIZE=100000
```

We set it to 1 because we only have 1 database server

Setting Up a Benchmarking Test

- ElaSQL-Bench reuses the codebase of VanillaBench, which is a benchmarking tool for single-node DBMS.

- So, the configurations are separated in two files:
  - vanillabench.properties (only the basic configrations)
  - elasqlbench.properties

vanillabench.properties

vanillabench.properties

```
16
17 #
18 # Basic Parameters
19 #
20
21 # The running time for warming up before benchmarking
22 org.vanilladb.bench.BenchmarkerParameters.WARM_UP_INTERVAL=60000
23 # The running time for benchmarking
24 org.vanilladb.bench.BenchmarkerParameters.BENCHMARK_INTERVAL=60000
25 # The number of remote terminal executors for benchmarking
26 org.vanilladb.bench.BenchmarkerParameters.NUM_RTES=2
27 # The sleeping time (in milliseconds) between transactions for each RTE
28 # 0 = no sleeping, 100 is a generally good number for under-loaded workloads
29 org.vanilladb.bench.BenchmarkerParameters.RTE_SLEEP_TIME=0
30 # The IP of the target database server
31 org.vanilladb.bench.BenchmarkerParameters.SERVER_IP=127.0.0.1
32 # 1 = JDBC, 2 = Stored Procedures
33 org.vanilladb.bench.BenchmarkerParameters.CONNECTION_MODE=2
34 # 1 = Micro, 2 = TPC-C, 3 = TPC-E, 4 = YCSB
35 # TPC-E dose not work for now
36 org.vanilladb.bench.BenchmarkerParameters.BENCH_TYPE=2
37 # Whether it enables the built-in profiler on the server
38 org.vanilladb.bench.BenchmarkerParameters.PROFILING_ON_SERVER=false
39 # The path to the generated reports
40 org.vanilladb.bench.StatisticMgr.OUTPUT_DIR=
41 # The granularity for summarizing the performance of benchmarking
42 org.vanilladb.bench.StatisticMgr.GRANULARITY=1000
43 # Whether the RTEs display the results of each transaction
44 org.vanilladb.bench.rte.TransactionExecutor.DISPLAY_RESULT=false
```

Only these parameters will take effect on ElaSQL-Bench

vanillabench.properties

vanillabench.properties

```
16
17 #
18 # Basic Parameters
19 #
20
21 # The running time for warming up before benchmarking
22 org.vanilladb.bench.BenchmarkerParameters.WARM_UP_INTERVAL=60000
23 # The running time for benchmarking
24 org.vanilladb.bench.BenchmarkerParameters.BENCHMARK_INTERVAL=60000
25 # The number of remote terminal executors for benchmarking
26 org.vanilladb.bench.BenchmarkerParameters.NUM_RTES=2
27 # The sleeping time (in milliseconds) between transactions for each RTE
28 # 0 = no sleeping, 100 is a generally good number for under-loaded workloads
29 org.vanilladb.bench.BenchmarkerParameters.RTE_SLEEP_TIME=0
30 # The IP of the target database server
31 org.vanilladb.bench.BenchmarkerParameters.SERVER_IP=127.0.0.1
32 # 1 = JDBC, 2 = Stored Procedures
33 org.vanilladb.bench.BenchmarkerParameters.CONNECTION_MODE=2
34 # 1 = Micro, 2 = TPC-C, 3 = TPC-E, 4 = YCSB
35 # TPC-E dose not work for now
36 org.vanilladb.bench.BenchmarkerParameters.BENCH_TYPE=2
37 # Whether it enables the built-in profiler on the server
38 org.vanilladb.bench.BenchmarkerParameters.PROFILING_ON_SERVER=false
39 # The path to the generated reports
40 org.vanilladb.bench.StatisticMgr.OUTPUT_DIR=
41 # The granularity for summarizing the performance of benchmarking
42 org.vanilladb.bench.StatisticMgr.GRANULARITY=1000
43 # Whether the RTEs display the results of each transaction
44 org.vanilladb.bench.rte.TransactionExecutor.DISPLAY_RESULT=false
```

How long the benchmarking test

Number of RTE threads, each of which simulates a user

Which benchmark to use

elasqlbench.properties

elasqlbench.properties

Nothing need to be changed

```
33
34 #
35 # TPC-C Parameters
36 #
37
38 # Partition strategies
39 # 1: Normal, 2: MgCrab scaling-out, 3: MgCrab consolidation
40 org.elasql.bench.benchmarks.tpcc.ElasqlTpccConstants.PARTITION_STRATEGY=1
41
42 # These parameters only work with the normal partitioning strategy
43 # Controls the skewness (hotness) of a partition
44 org.elasql.bench.benchmarks.tpcc.ElasqlTpccConstants.WAREHOUSE_PER_PART=1
45 org.elasql.bench.benchmarks.tpcc.TpccStandardRteGenerator.SKEW_RATIO=0.0
46
47 # Parameters for MgCrab scale-out experiments
48 # Note that when NUM_HOT_PARTS = 2 and HOT_WAREHOUSE_PER_HOT_PART = 2,
49 # it will create 2 source partitions and 4 destination partitions,
50 # because each hot partition must migrate a hot warehouse to a destination partition.
51 # How many partitions are hot
52 org.elasql.bench.server.metadata.migration.scaleout.TpccScaleoutBeforePartPlan.NUM_HOT_PARTS=1
53 # How many warehouses each hot partition has
54 org.elasql.bench.server.metadata.migration.scaleout.TpccScaleoutBeforePartPlan.HOT_WAREHOUSE_PER_HOT_PART=1
55
56
57
58 #
59 # YCSB Parameters
60 #
61 # Database mode
62 # 1: Single Table, 2: Multi-Table (works better for multi-tenant settings)
63 org.elasql.bench.benchmarks.ycsb.ElasqlYcsbConstants.DATABASE_MODE=1
```

Testing inside Eclipse

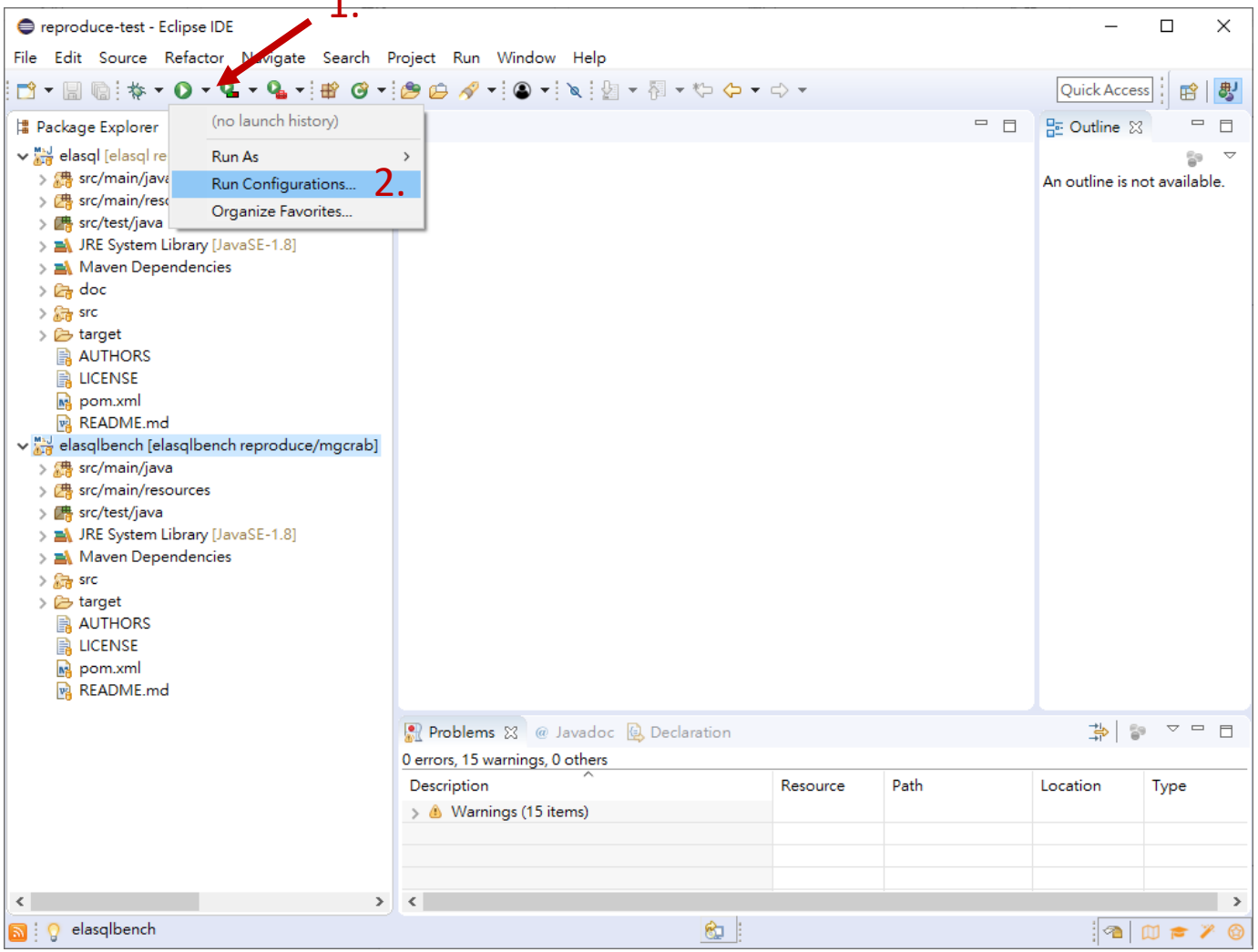
1. Setup the properties files
   - Which includes the configurations for ElaSQL and ElaSQL-Bench
2. Setup run configurations
3. Loading a testbed
   1. Launch servers
   2. Launch clients
4. Benchmarking
   1. Launch servers
   2. Launch clients

Run Configurations

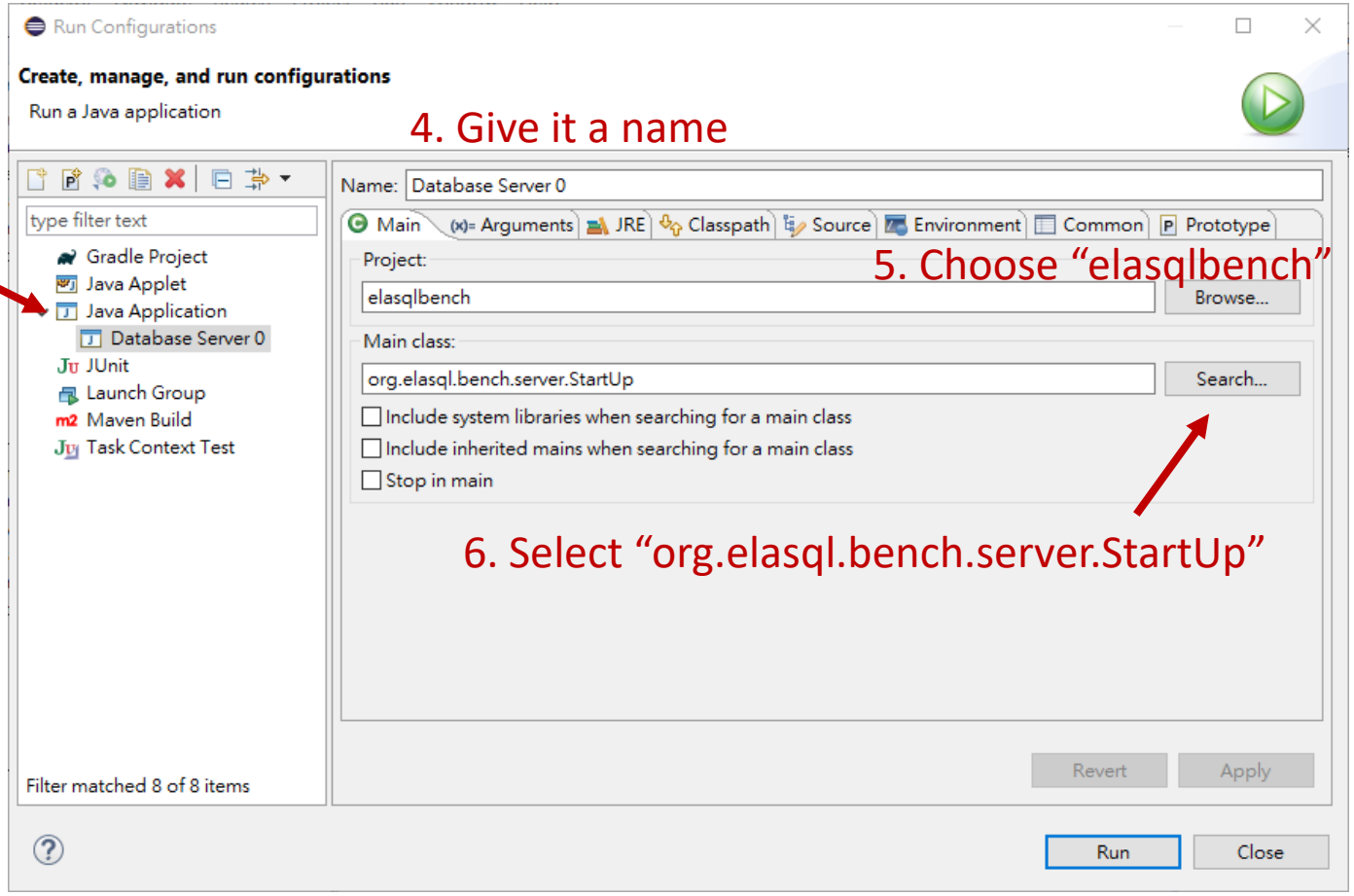
- A run configuration configures how eclipse launch a Java process.

- Each process must have its own run configuration.
  - 3 configurations for a sequencer server, a database server, and a client.

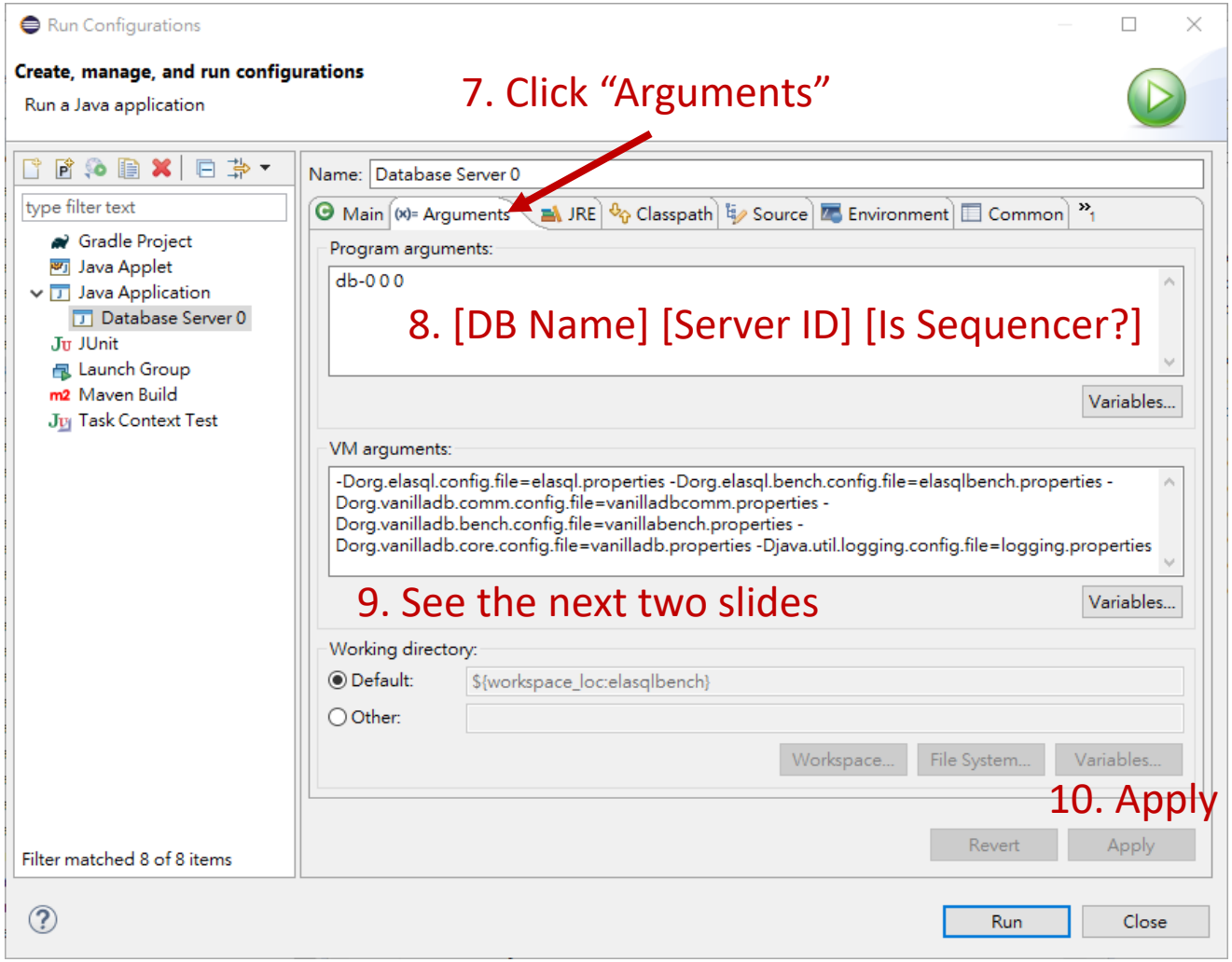
Setting Up Run Configurations



Setting Up Run Configurations (A Database Server)



Setting Up Run Configurations (A Database Server)



Program Arguments (For Servers)

- Program Arguments
  - DB Name: the database name
    - Note that if you run servers on the same machine, each server should have an unique name for its database.
  - Server ID: the ID of the server process
  - Is Sequencer: to set if it is running in sequencer mode.
    - The server with the greatest ID should turn this ON.

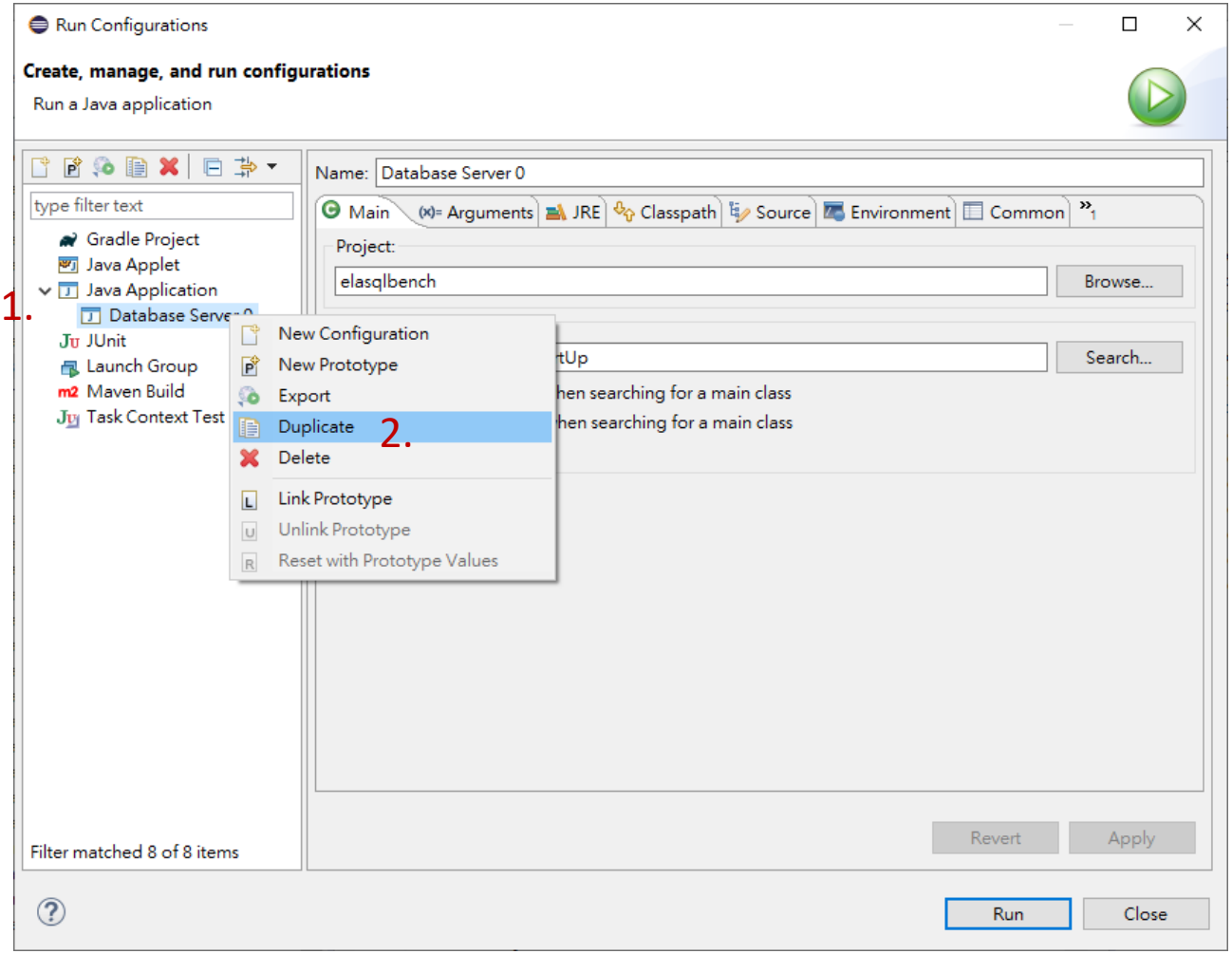
VM Arguments

- VM Arguments:
  - To tell ElaSQL where to find those properties files
  - Copy and paste this:

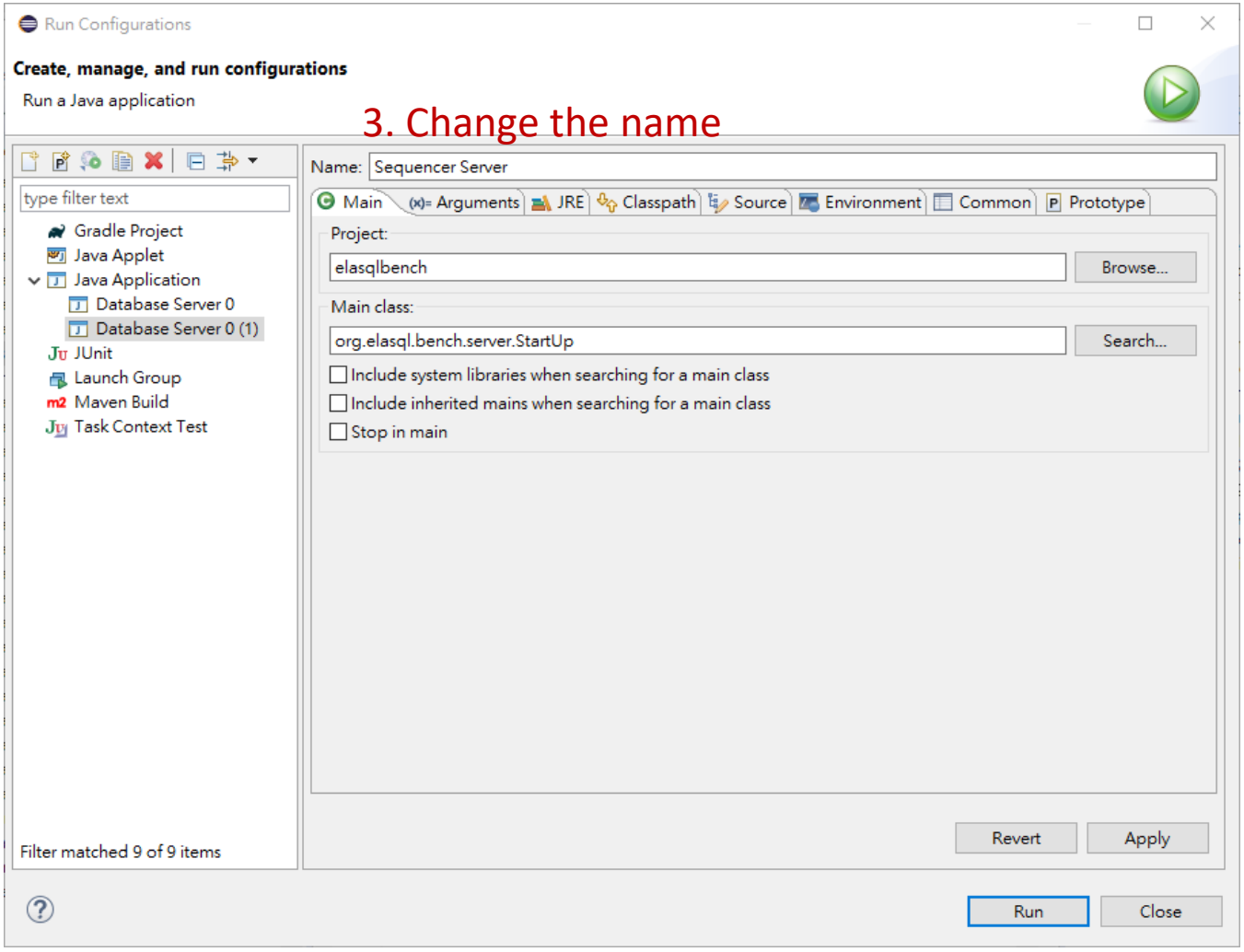
```
-Dorg.elasql.config.file=target/classes/org/elasql/elasql.properties
-Dorg.elasql.bench.config.file=target/classes/org/elasql/elasqlbench.properties
-Dorg.vanilladb.comm.config.file=target/classes/org/vanilladb/comm/vanillacomm.properties
-Dorg.vanilladb.bench.config.file=target/classes/org/vanilladb/bench/vanillabench.properties
-Dorg.vanilladb.core.config.file=target/classes/org/vanilladb/core/vanilladb.properties
-Djava.util.logging.config.file=target/classes/java/util/logging/logging.properties
```

- If you encounter any problem when copying the arguments from this slide, you can copy from here.

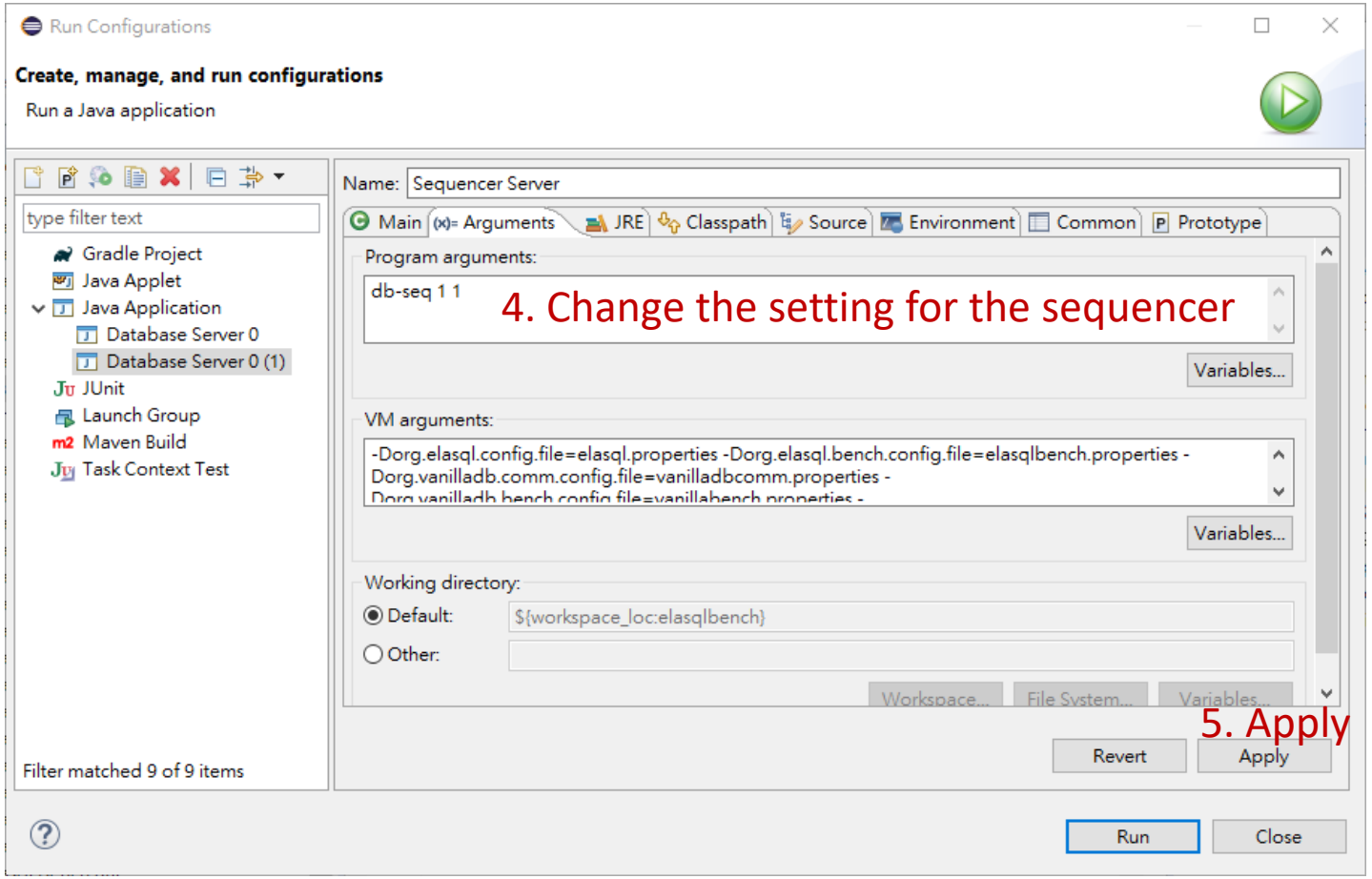
Setting Up Run Configurations (The Sequencer Server)



Setting Up Run Configurations (The Sequencer Server)

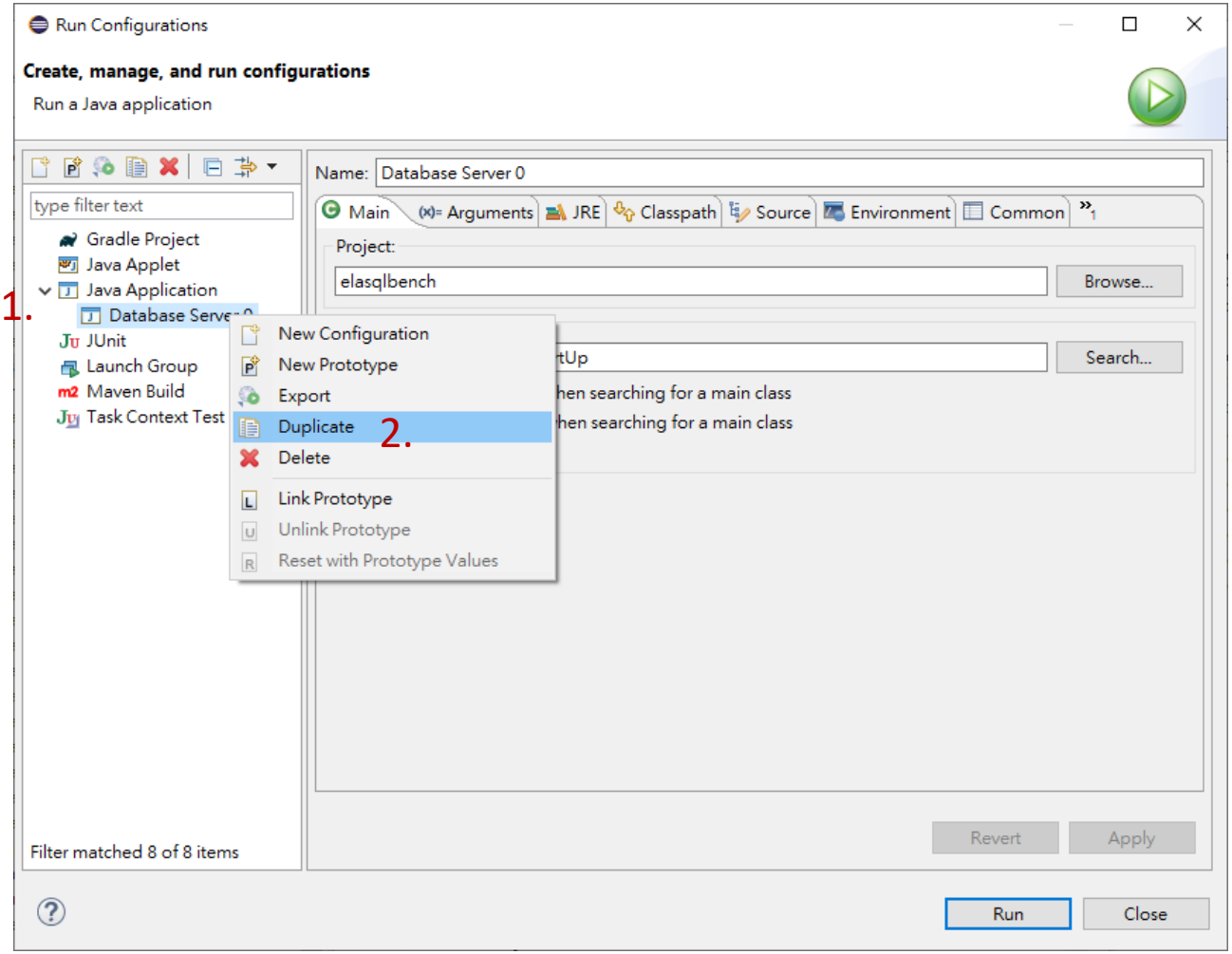


Setting Up Run Configurations (The Sequencer Server)

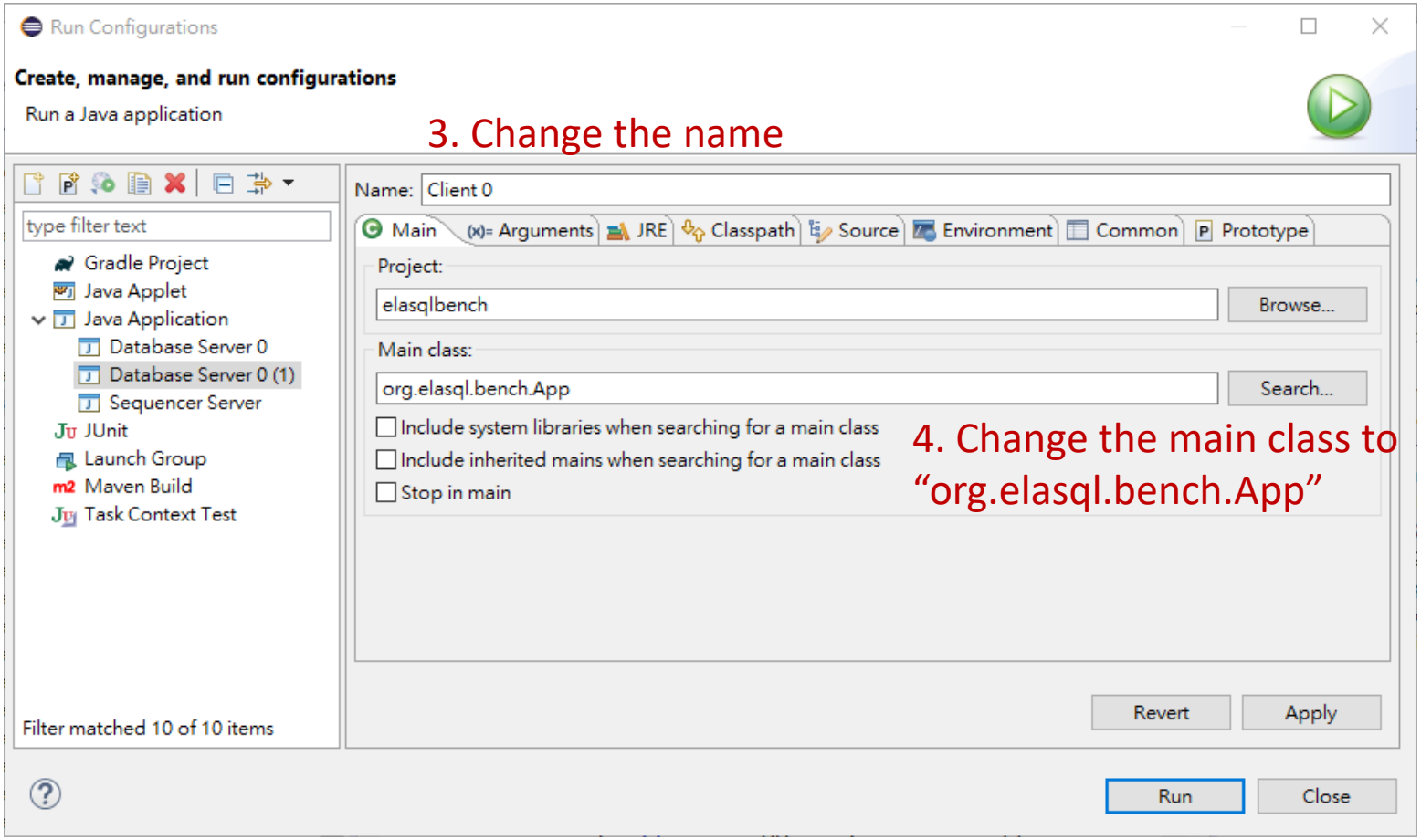


Note: the sequencer will not create any database file, so 'db-seq' is just a placeholder.

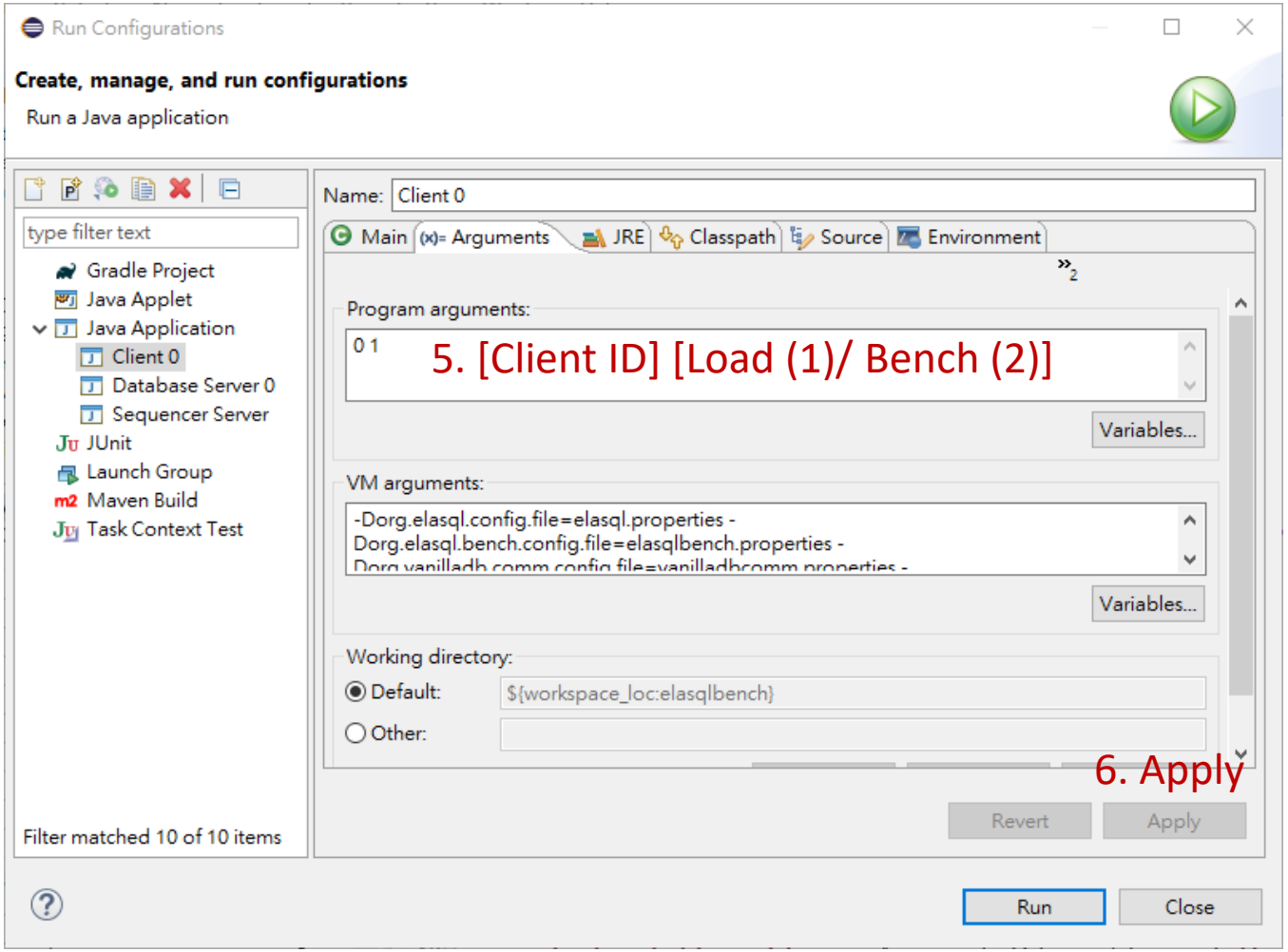
Setting Up Run Configurations (A Benchmarking Client)



Setting Up Run Configurations (A Benchmarking Client)



Setting Up Run Configurations (A Benchmarking Client)



Program & VM Arguments (For Clients)

- Program Arguments
  - Client ID: the ID of the client process
  - Load/Bench: to controls the action of this client
    - 1: Loading a new testbed on a clean database.
    - 2: Benchmarking on an existing testbed.

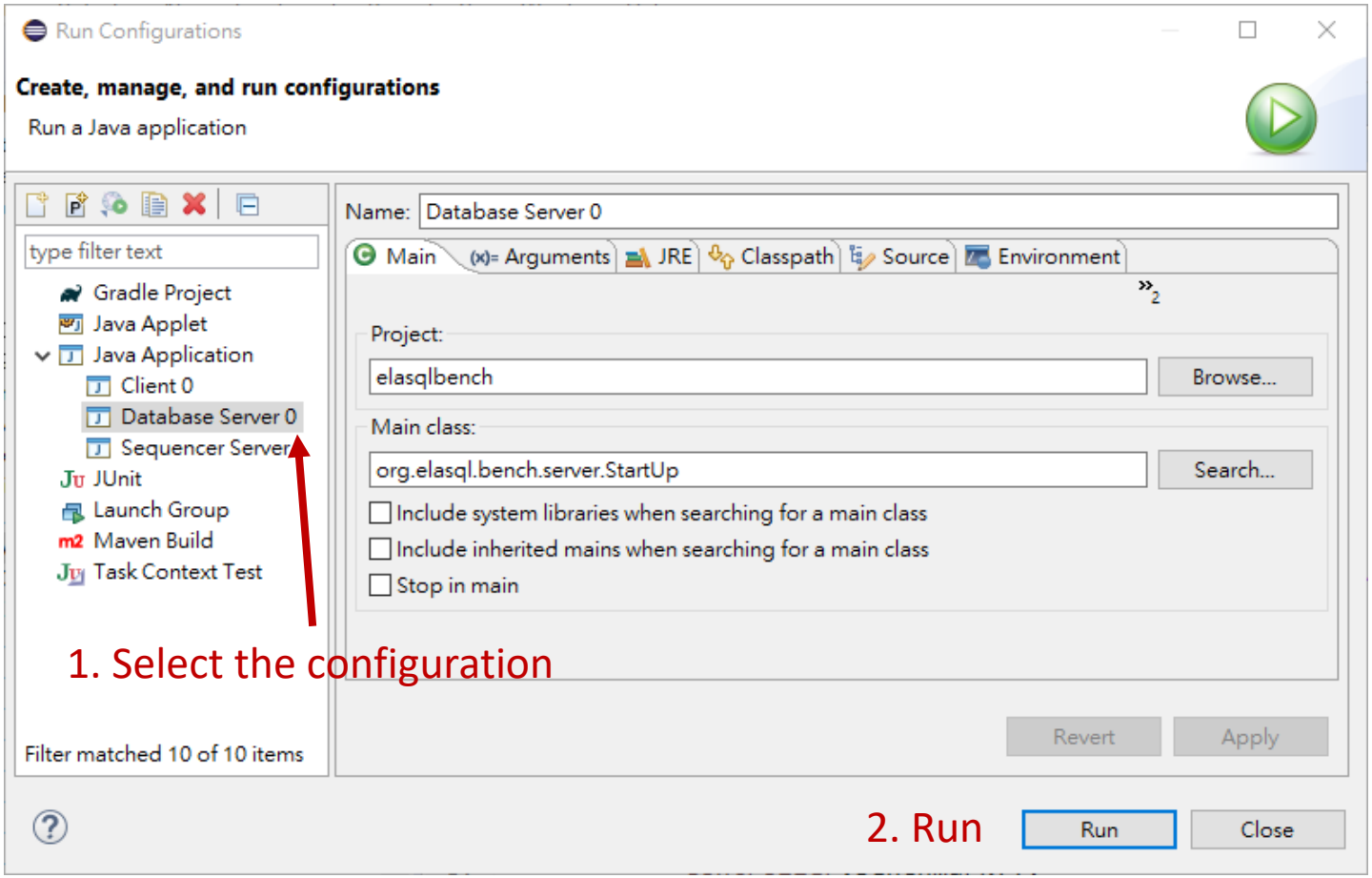
- Note that a client must first load a new testbed on a system before benchmarking it.

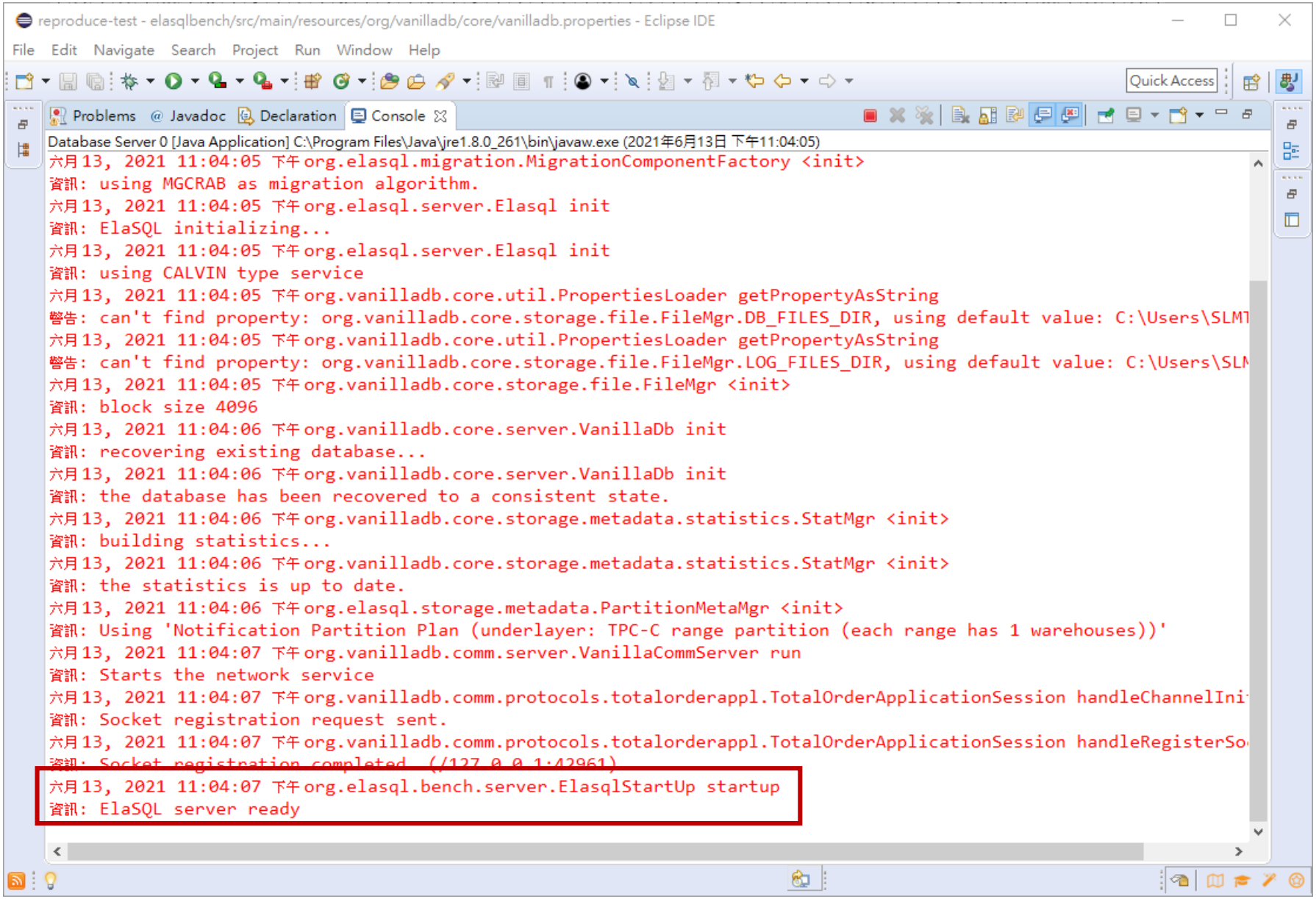
- VM Arguments: same as the servers

Testing inside Eclipse

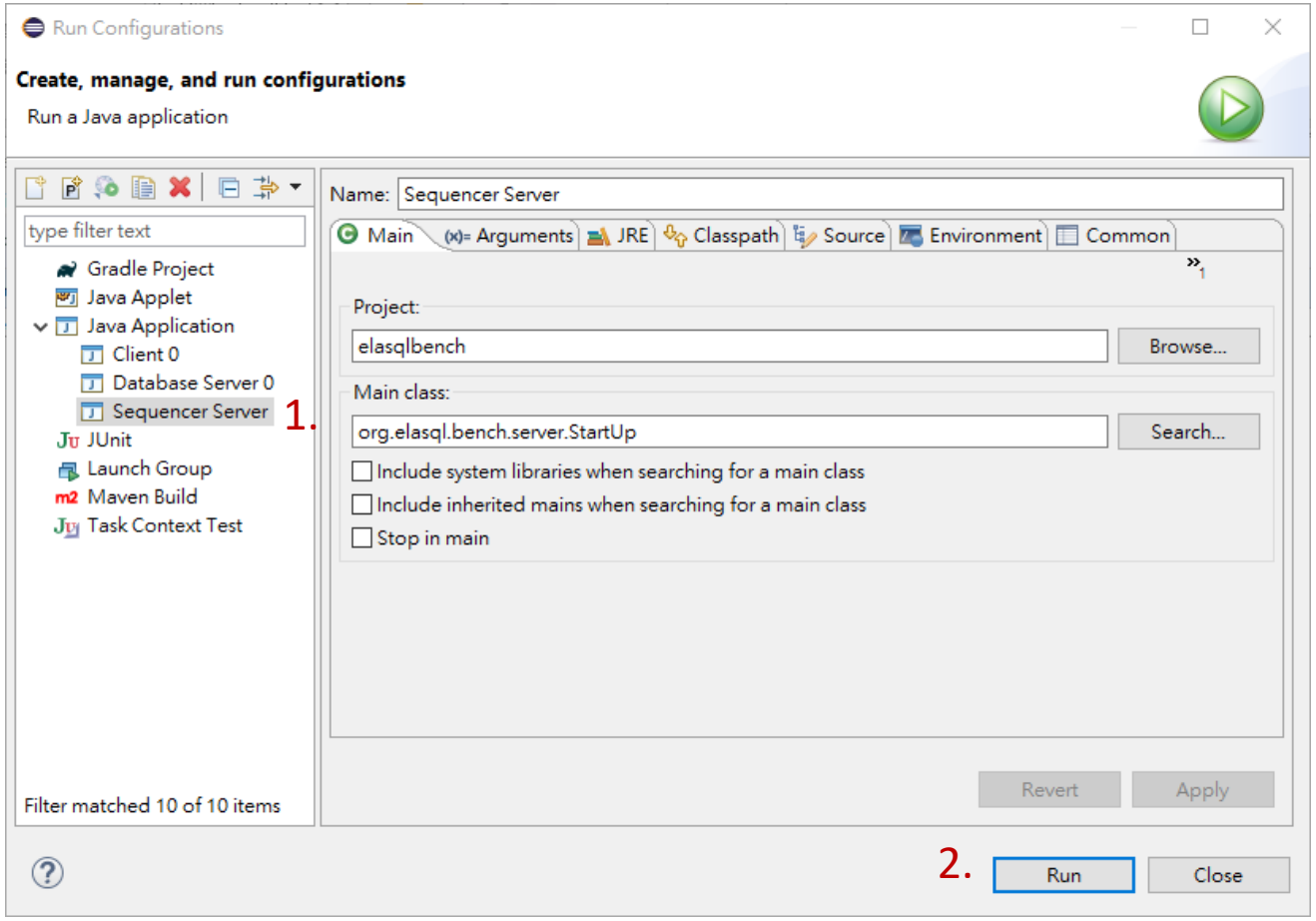
1. Setup the properties files
   - Which includes the configurations for ElaSQL and ElaSQL-Bench
2. Setup run configurations
3. Loading a testbed
   1. Launch servers
   2. Launch clients
4. Benchmarking
   1. Launch servers
   2. Launch clients

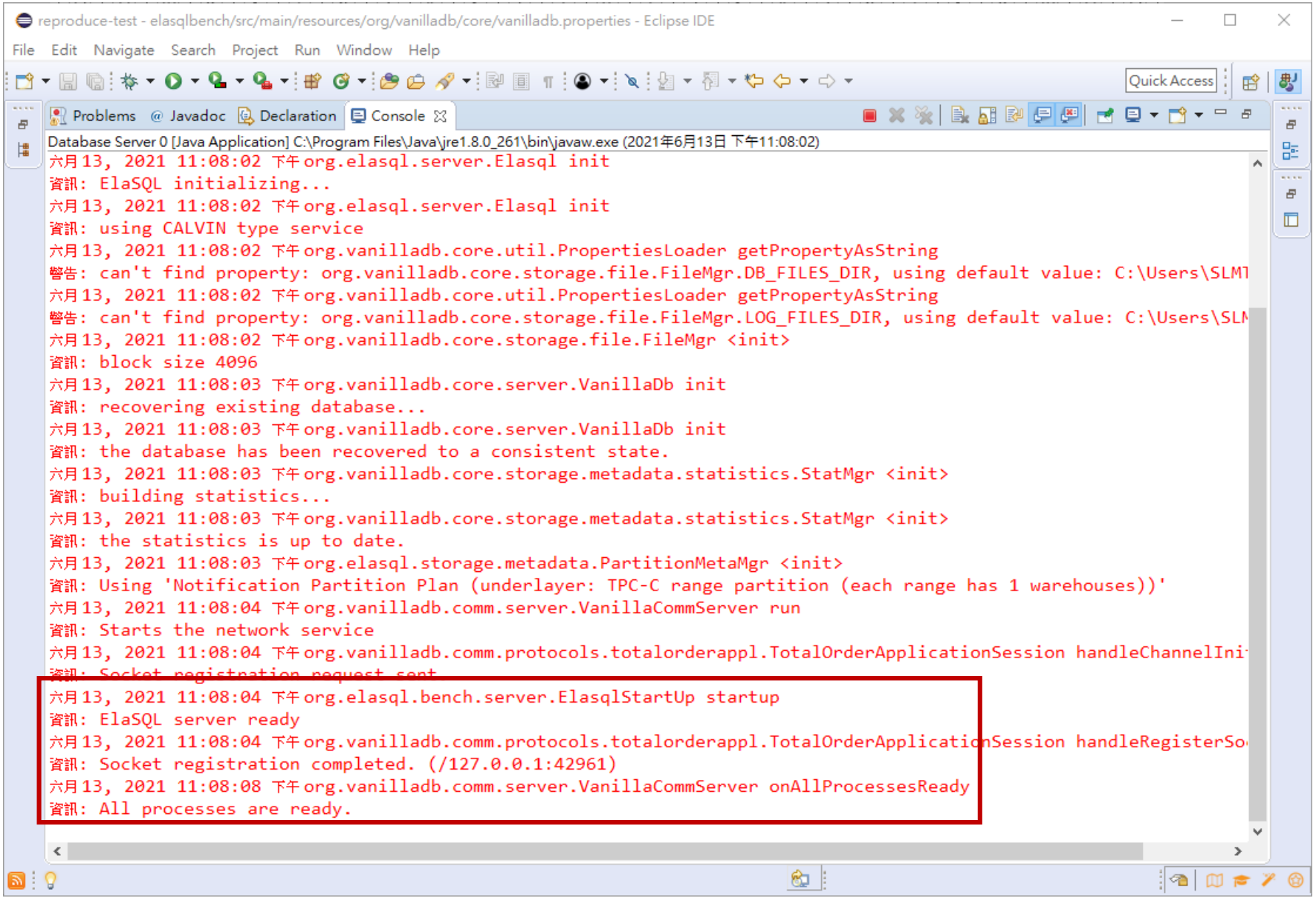
Launching A Database Server



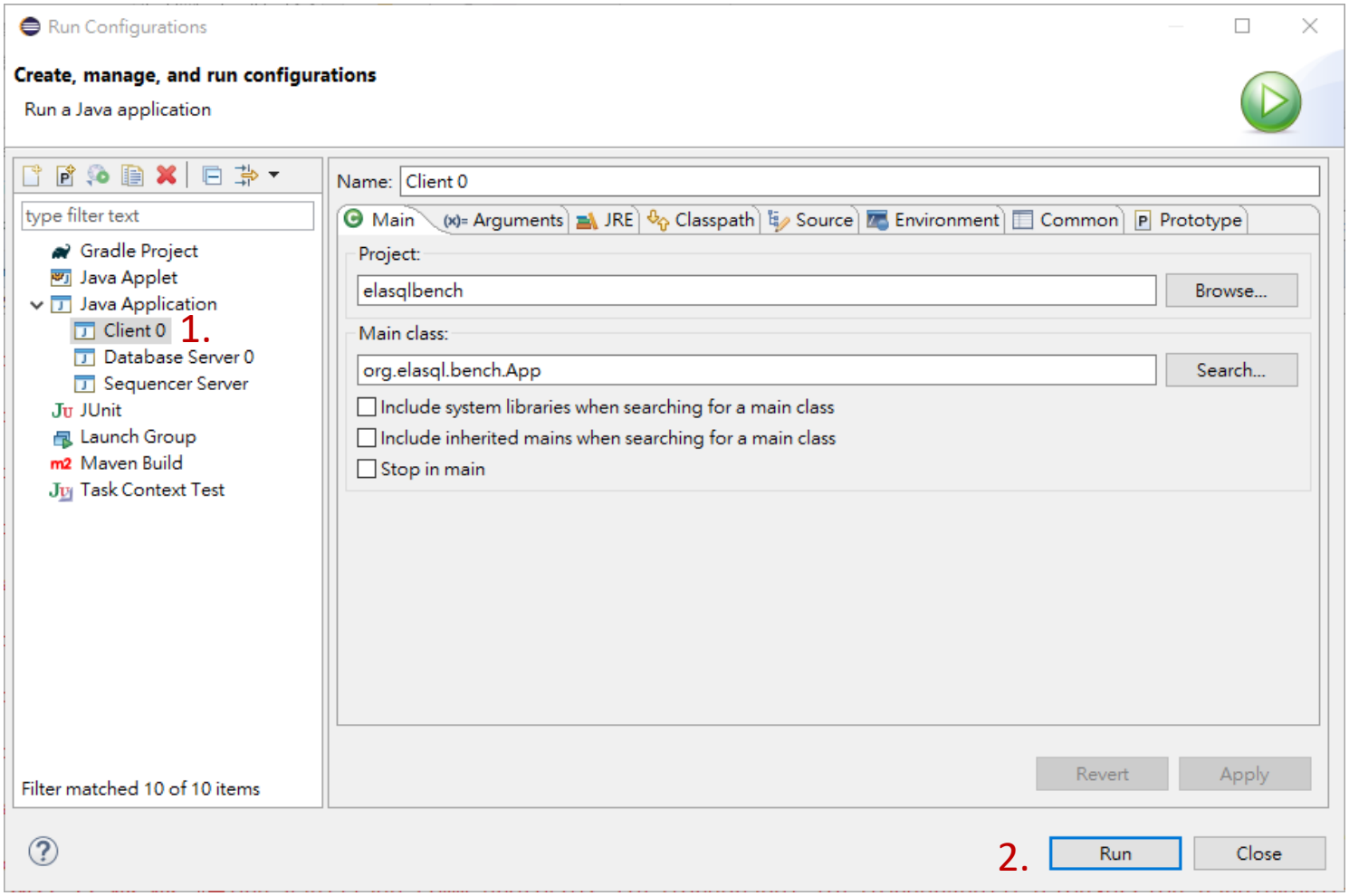

reproduce-test - elasqlbench/src/main/resources/org/vanilladb/core/vanilladb.properties - Eclipse IDE
File Edit Navigate Search Project Run Window Help
Quick Access
Problems Javadoc Declaration Console
Database Server 0 [Java Application] C:\Program Files\Java\jre1.8.0_261\bin\javaw.exe (2021年6月13日 下午11:04:05)
六月 13, 2021 11:04:05 下午 org.elasql.migration.MigrationComponentFactory <init>
資訊: using MGCRAB as migration algorithm.
六月 13, 2021 11:04:05 下午 org.elasql.server.Elasql init
資訊: ElaSQL initializing...
六月 13, 2021 11:04:05 下午 org.elasql.server.Elasql init
資訊: using CALVIN type service
六月 13, 2021 11:04:05 下午 org.vanilladb.core.util.PropertiesLoader getPropertyAsString
警告: can't find property: org.vanilladb.core.storage.file.FileMgr.DB_FILES_DIR, using default value: C:\Users\SLMT
六月 13, 2021 11:04:05 下午 org.vanilladb.core.util.PropertiesLoader getPropertyAsString
警告: can't find property: org.vanilladb.core.storage.file.FileMgr.LOG_FILES_DIR, using default value: C:\Users\SLM
六月 13, 2021 11:04:05 下午 org.vanilladb.core.storage.file.FileMgr <init>
資訊: block size 4096
六月 13, 2021 11:04:06 下午 org.vanilladb.core.server.VanillaDb init
資訊: recovering existing database...
六月 13, 2021 11:04:06 下午 org.vanilladb.core.server.VanillaDb init
資訊: the database has been recovered to a consistent state.
六月 13, 2021 11:04:06 下午 org.vanilladb.core.storage.metadata.statistics.StatMgr <init>
資訊: building statistics...
六月 13, 2021 11:04:06 下午 org.vanilladb.core.storage.metadata.statistics.StatMgr <init>
資訊: the statistics is up to date.
六月 13, 2021 11:04:06 下午 org.elasql.storage.metadata.PartitionMetaMgr <init>
資訊: Using 'Notification Partition Plan (underlayer: TPC-C range partition (each range has 1 warehouses))'
六月 13, 2021 11:04:07 下午 org.vanilladb.comm.server.VanillaCommServer run
資訊: Starts the network service
六月 13, 2021 11:04:07 下午 org.vanilladb.comm.protocols.totalorderappl.TotalOrderApplicationSession handleChannelIni
資訊: Socket registration request sent.
六月 13, 2021 11:04:07 下午 org.vanilladb.comm.protocols.totalorderappl.TotalOrderApplicationSession handleRegisterSo
資訊: Socket registration completed. (/127.0.0.1:42961)
六月 13, 2021 11:04:07 下午 org.elasql.bench.server.ElasqlStartUp startup
資訊: ElaSQL server ready

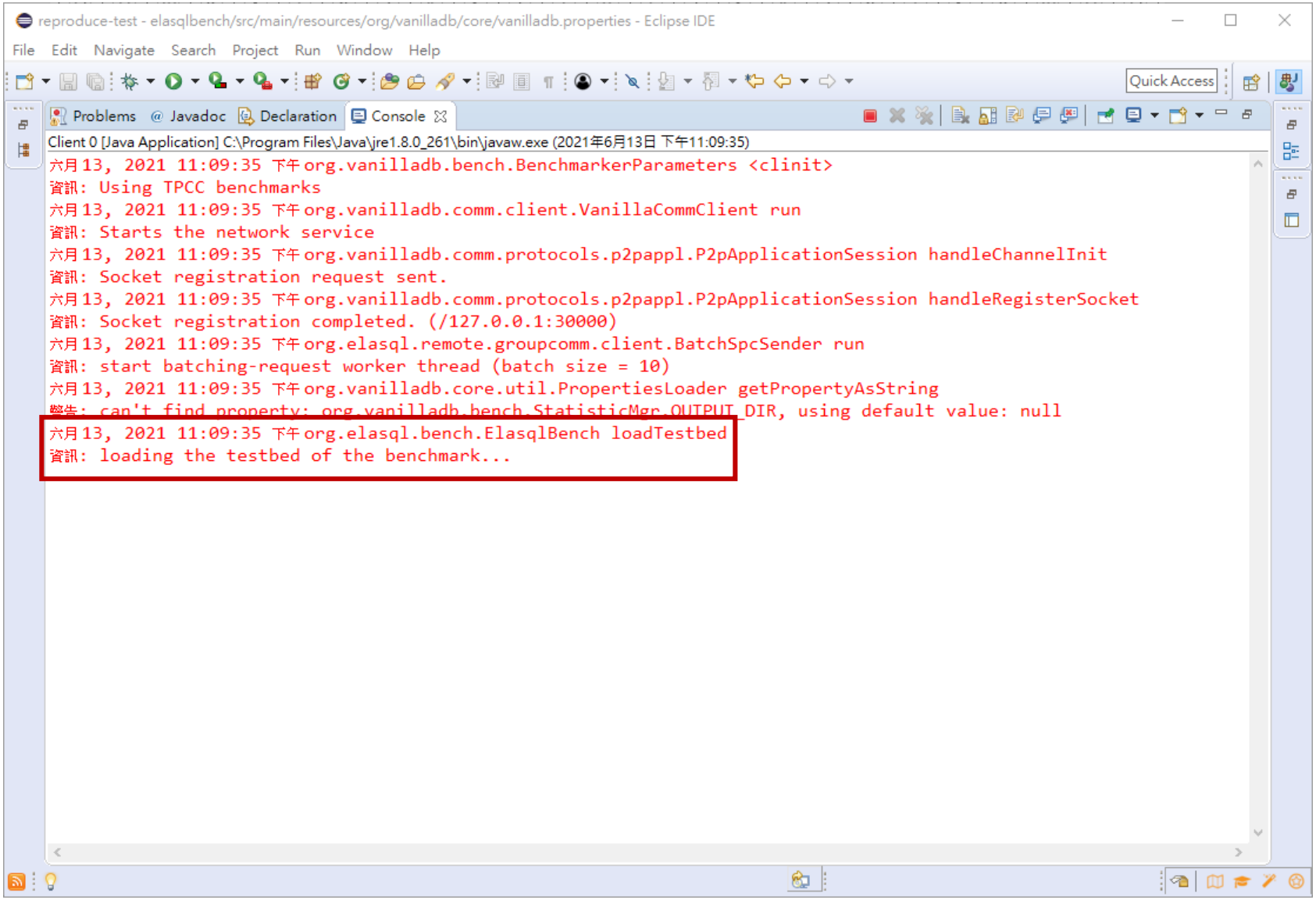
Launching The Sequencer

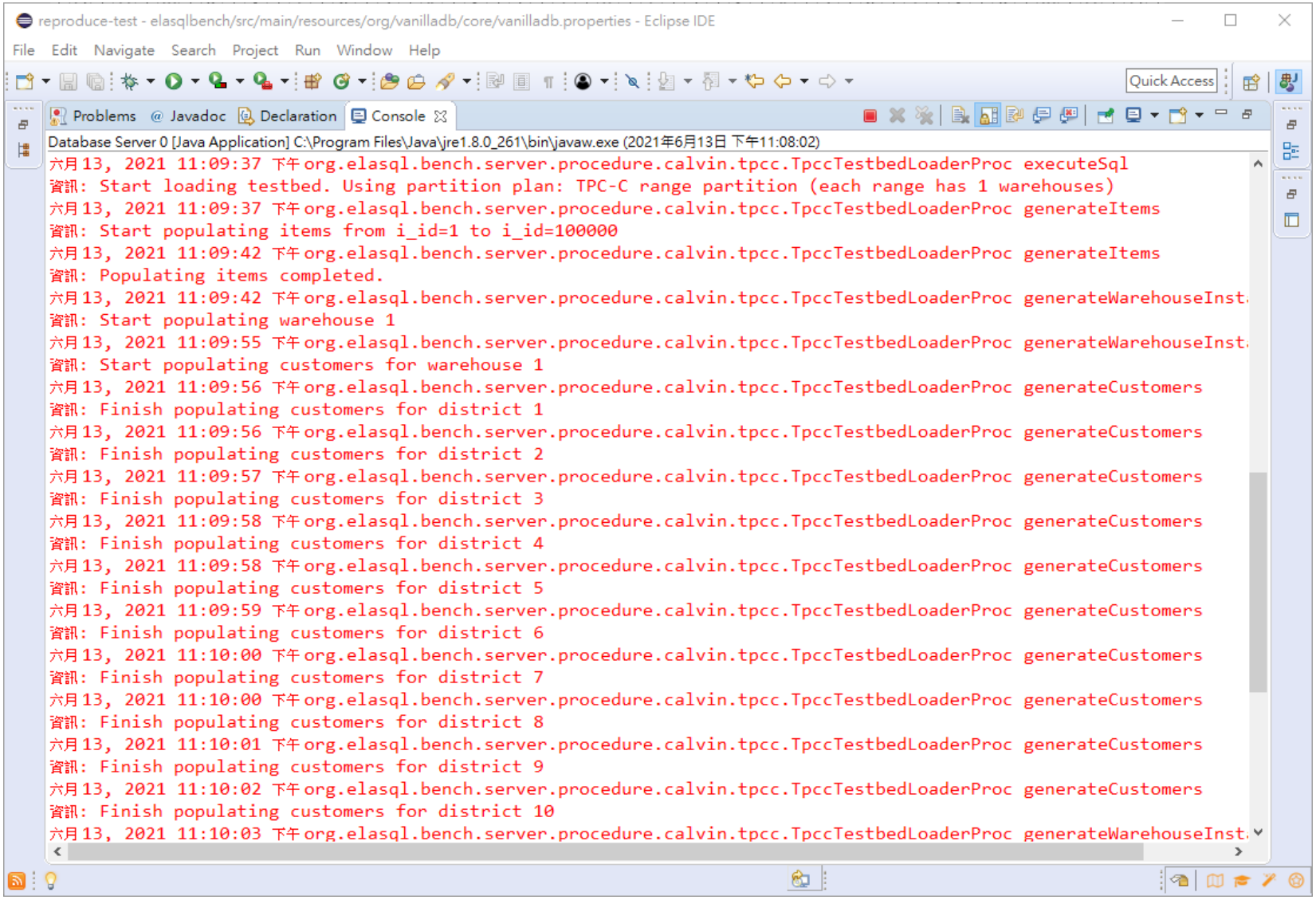



reproduce-test - elasqlbench/src/main/resources/org/vanilladb/core/vanilladb.properties - Eclipse IDE
File Edit Navigate Search Project Run Window Help
Quick Access
Problems Javadoc Declaration Console
Database Server 0 [Java Application] C:\Program Files\Java\jre1.8.0_261\bin\javaw.exe (2021年6月13日 下午11:08:02)
六月 13, 2021 11:08:02 下午 org.elasql.server.Elasql init
資訊: ElaSQL initializing...
六月 13, 2021 11:08:02 下午 org.elasql.server.Elasql init
資訊: using CALVIN type service
六月 13, 2021 11:08:02 下午 org.vanilladb.core.util.PropertiesLoader getPropertyAsString
警告: can't find property: org.vanilladb.core.storage.file.FileMgr.DB_FILES_DIR, using default value: C:\Users\SLM
六月 13, 2021 11:08:02 下午 org.vanilladb.core.util.PropertiesLoader getPropertyAsString
警告: can't find property: org.vanilladb.core.storage.file.FileMgr.LOG_FILES_DIR, using default value: C:\Users\SLM
六月 13, 2021 11:08:02 下午 org.vanilladb.core.storage.file.FileMgr <init>
資訊: block size 4096
六月 13, 2021 11:08:03 下午 org.vanilladb.core.server.VanillaDb init
資訊: recovering existing database...
六月 13, 2021 11:08:03 下午 org.vanilladb.core.server.VanillaDb init
資訊: the database has been recovered to a consistent state.
六月 13, 2021 11:08:03 下午 org.vanilladb.core.storage.metadata.statistics.StatMgr <init>
資訊: building statistics...
六月 13, 2021 11:08:03 下午 org.vanilladb.core.storage.metadata.statistics.StatMgr <init>
資訊: the statistics is up to date.
六月 13, 2021 11:08:03 下午 org.elasql.storage.metadata.PartitionMetaMgr <init>
資訊: Using 'Notification Partition Plan (underlayer: TPC-C range partition (each range has 1 warehouses))'
六月 13, 2021 11:08:04 下午 org.vanilladb.comm.server.VanillaCommServer run
資訊: Starts the network service
六月 13, 2021 11:08:04 下午 org.vanilladb.comm.protocols.totalorderappl.TotalOrderApplicationSession handleChannelIni
資訊: Socket registration request sent
六月 13, 2021 11:08:04 下午 org.elasql.bench.server.ElasqlStartUp startup
資訊: ElaSQL server ready
六月 13, 2021 11:08:04 下午 org.vanilladb.comm.protocols.totalorderappl.TotalOrderApplicationSession handleRegisterSo
資訊: Socket registration completed. (/127.0.0.1:42961)
六月 13, 2021 11:08:08 下午 org.vanilladb.comm.server.VanillaCommServer onAllProcessesReady
資訊: All processes are ready.

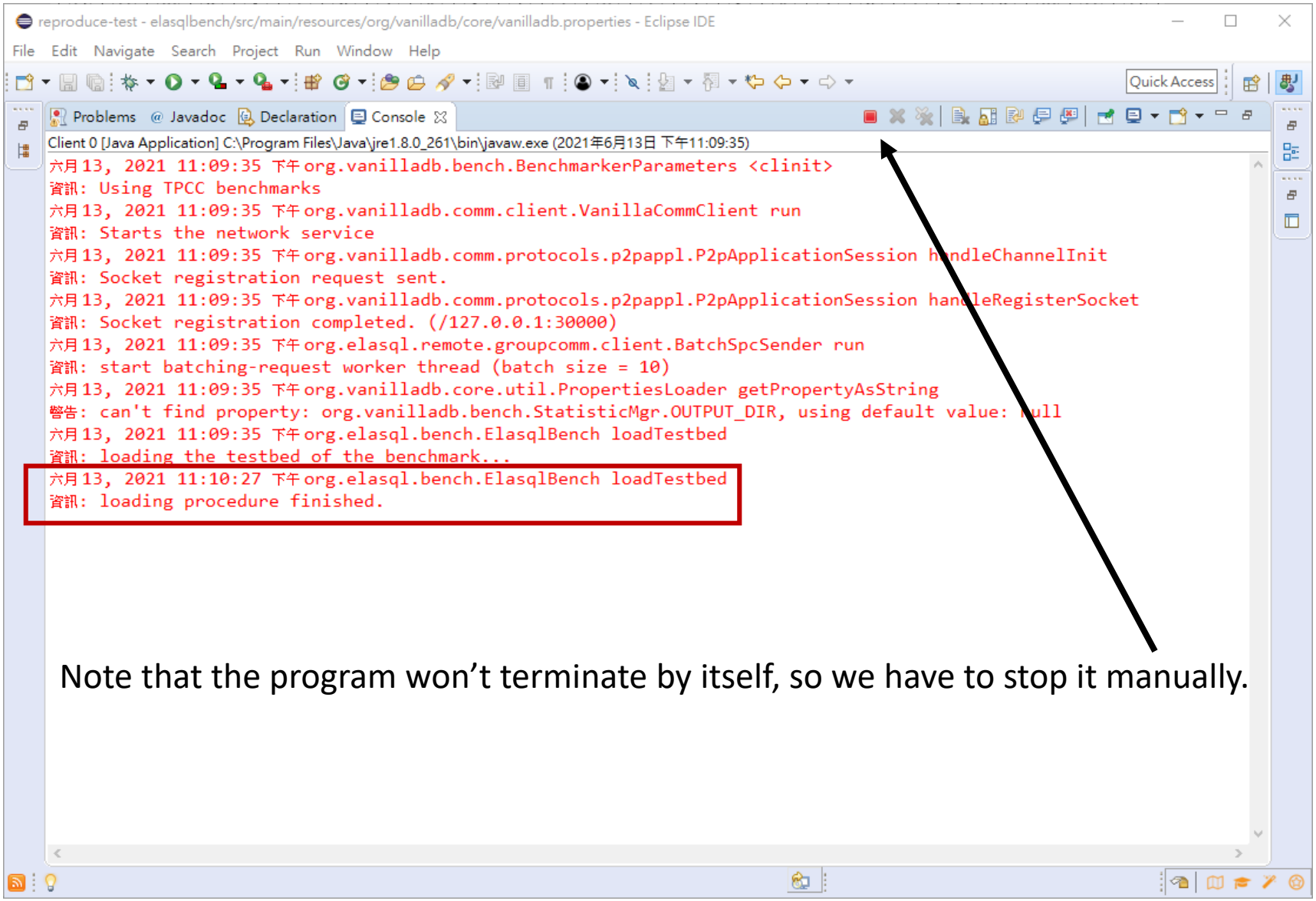
Launching a Client




reproduce-test - elasqlbench/src/main/resources/org/vanilladb/core/vanilladb.properties - Eclipse IDE
File Edit Navigate Search Project Run Window Help
Quick Access
Problems Javadoc Declaration Console
Client 0 [Java Application] C:\Program Files\Java\jre1.8.0_261\bin\javaw.exe (2021年6月13日 下午11:09:35)
六月 13, 2021 11:09:35 下午 org.vanilladb.bench.BenchmarkerParameters <clinit>
資訊: Using TPCC benchmarks
六月 13, 2021 11:09:35 下午 org.vanilladb.comm.client.VanillaCommClient run
資訊: Starts the network service
六月 13, 2021 11:09:35 下午 org.vanilladb.comm.protocols.p2pappl.P2pApplicationSession handleChannelInit
資訊: Socket registration request sent.
六月 13, 2021 11:09:35 下午 org.vanilladb.comm.protocols.p2pappl.P2pApplicationSession handleRegisterSocket
資訊: Socket registration completed. (/127.0.0.1:30000)
六月 13, 2021 11:09:35 下午 org.elasql.remote.groupcomm.client.BatchSpcSender run
資訊: start batching-request worker thread (batch size = 10)
六月 13, 2021 11:09:35 下午 org.vanilladb.core.util.PropertiesLoader getPropertyAsString
警告: can't find property: org.vanilladb.bench.StatisticMgr.OUTPUT_DIR, using default value: null
六月 13, 2021 11:09:35 下午 org.elasql.bench.ElasqlBench loadTestbed
資訊: loading the testbed of the benchmark...



The database server will show some messages about the loaded data.



Note that the program won't terminate by itself, so we have to stop it manually.

A Note

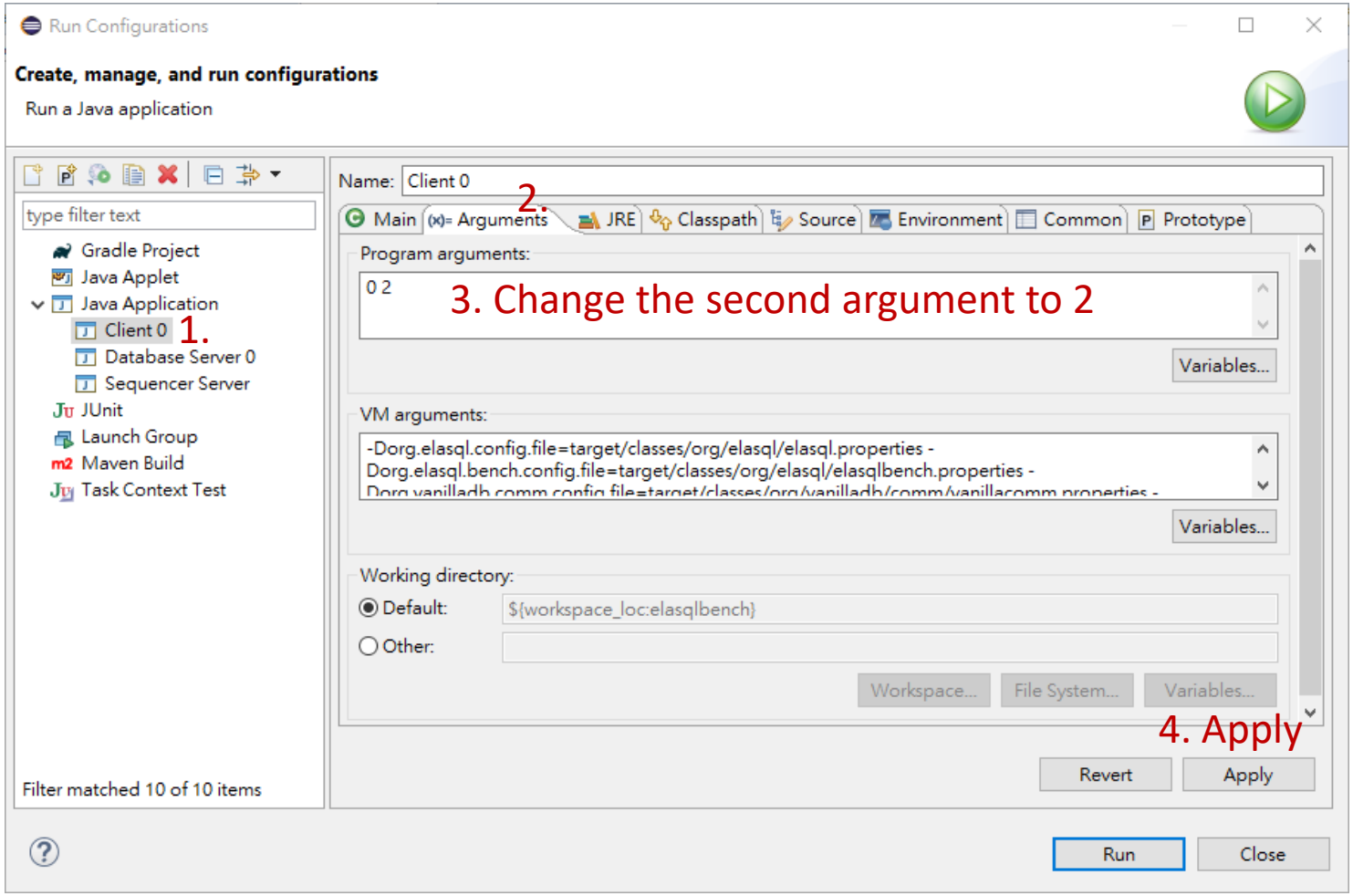
- Remember **always to terminate all the processes** before the next run.

- Since the database system may change the state of database files after each benchmarking test, in order to ensure the consistency of the benchmarking result, we suggest to
  1. Backup the database directory (usually in your home directory)
  2. Replace the database directory with the backup before each benchmarking run.

Testing inside Eclipse

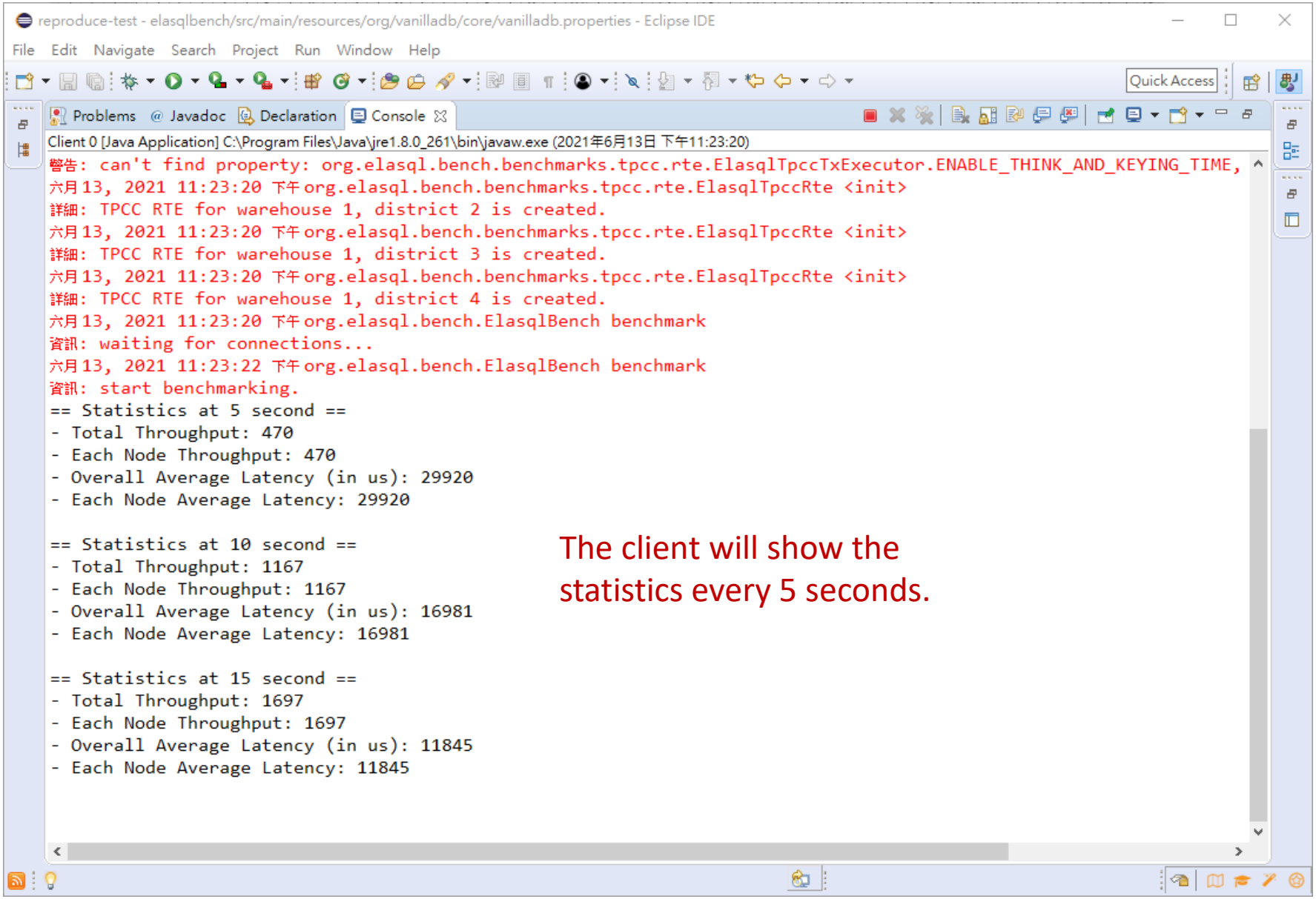
1. Setup the properties files
   - Which includes the configurations for ElaSQL and ElaSQL-Bench
2. Setup run configurations
3. Loading a testbed
   1. Launch servers
   2. Launch clients
4. Benchmarking
   1. Launch servers
   2. Launch clients

Changing the Client to Benchmarking Mode

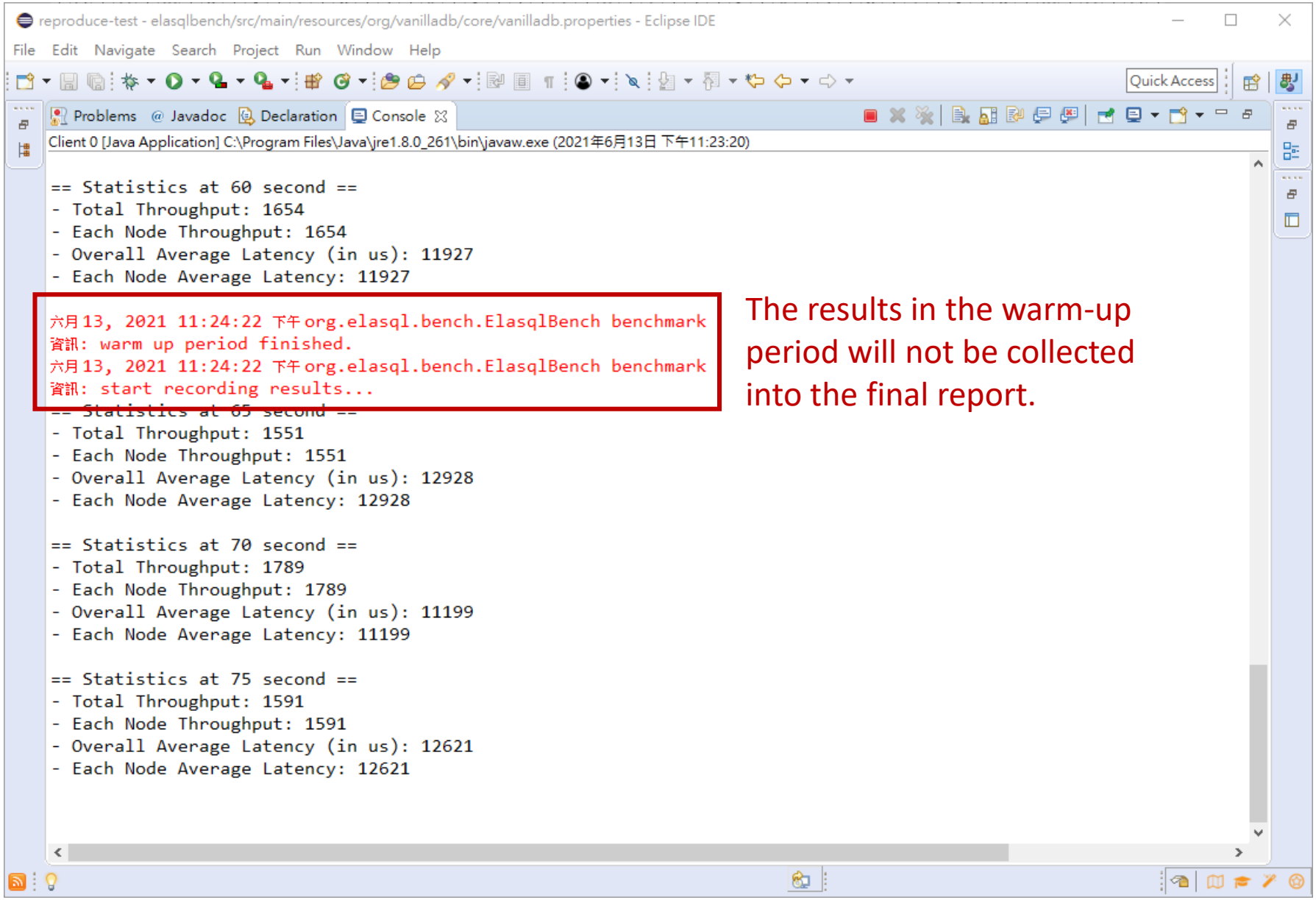


Launching the Servers and the Client

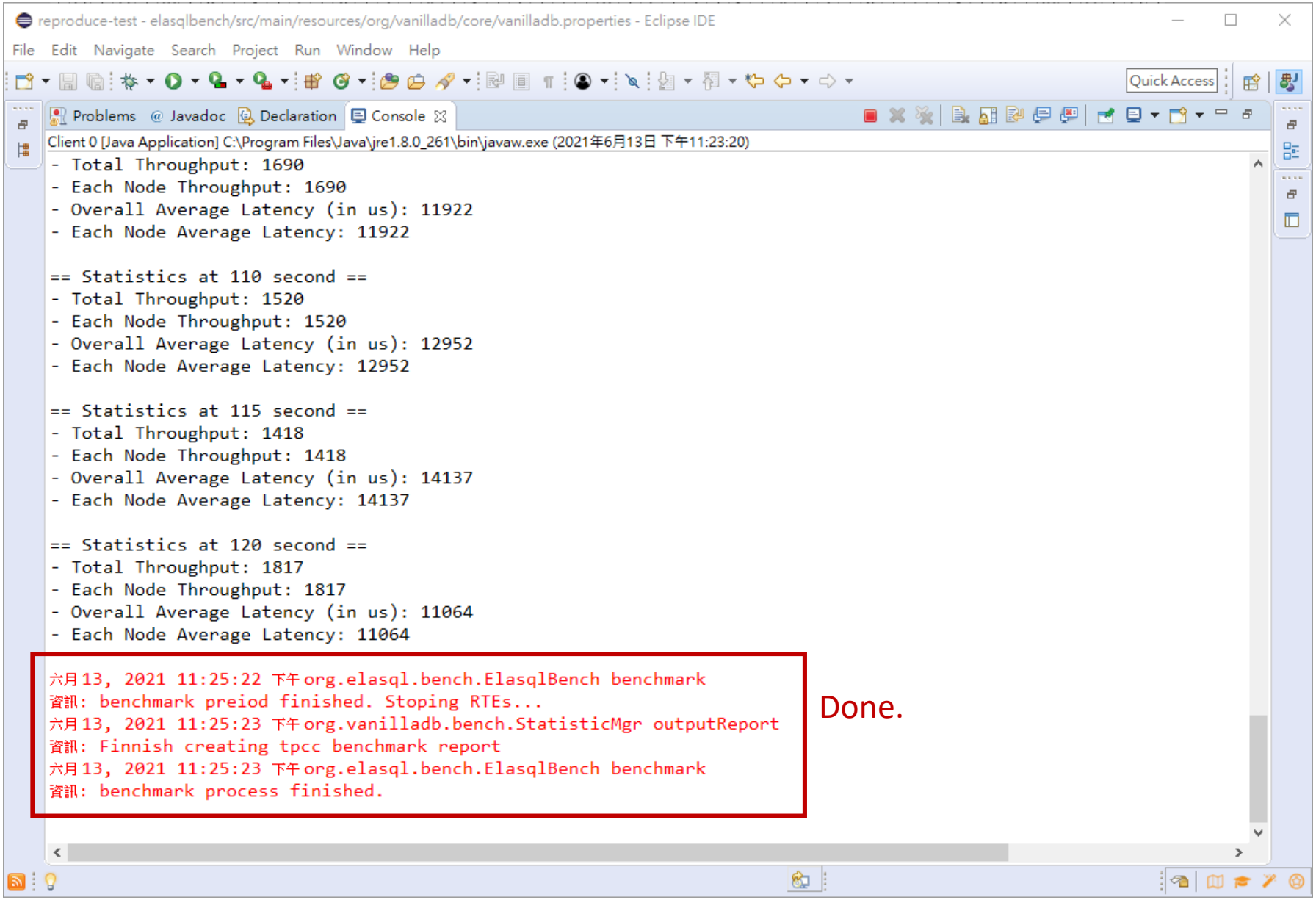
- Just follow the same launch procedure as loading a testbed.
  1. Launch the database server
  2. Launch the sequencer server
  3. Wait for the server ready
  4. Launch the client



The client will show the statistics every 5 seconds.



The results in the warm-up period will not be collected into the final report.

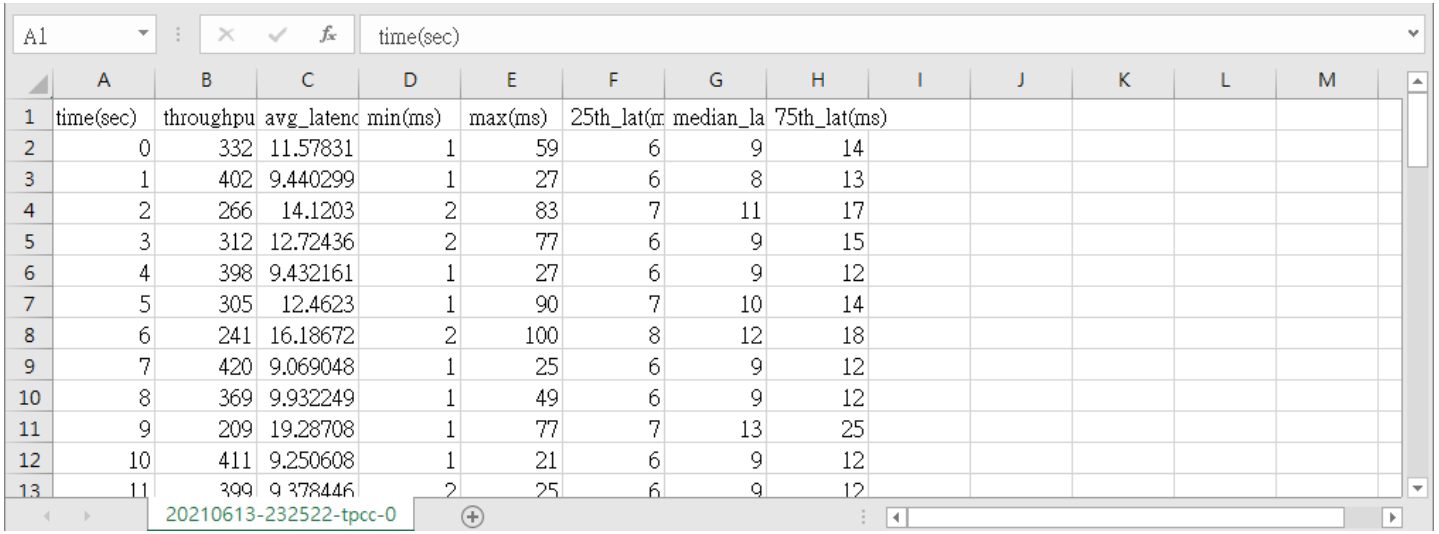

reproduce-test - elasqlbench/src/main/resources/org/vanilladb/core/vanilladb.properties - Eclipse IDE
File Edit Navigate Search Project Run Window Help
Quick Access
Problems Javadoc Declaration Console
Client 0 [Java Application] C:\Program Files\Java\jre1.8.0_261\bin\javaw.exe (2021年6月13日 下午11:23:20)
- Total Throughput: 1690
- Each Node Throughput: 1690
- Overall Average Latency (in us): 11922
- Each Node Average Latency: 11922
== Statistics at 110 second ==
- Total Throughput: 1520
- Each Node Throughput: 1520
- Overall Average Latency (in us): 12952
- Each Node Average Latency: 12952
== Statistics at 115 second ==
- Total Throughput: 1418
- Each Node Throughput: 1418
- Overall Average Latency (in us): 14137
- Each Node Average Latency: 14137
== Statistics at 120 second ==
- Total Throughput: 1817
- Each Node Throughput: 1817
- Overall Average Latency (in us): 11064
- Each Node Average Latency: 11064
六月 13, 2021 11:25:22 下午 org.elasql.bench.ElasqlBench benchmark
資訊: benchmark preiod finished. Stoping RTEs...
六月 13, 2021 11:25:23 下午 org.vanilladb.bench.StatisticMgr outputReport
資訊: Finnish creating tpcc benchmark report
六月 13, 2021 11:25:23 下午 org.elasql.bench.ElasqlBench benchmark
資訊: benchmark process finished.
Done.

Reports

- The report will be put in "$HOME$/benchmark_results" by default.
  - You can change this in vanillabench.properties

- There are two report will be generated:
  - [Datetime]-[Benchmark Name]-[Client ID].csv
    - Record the timeline of system performance
  - [Datetime]-[Benchmark Name]-[Client ID].txt
    - Summary the result for each transaction type.

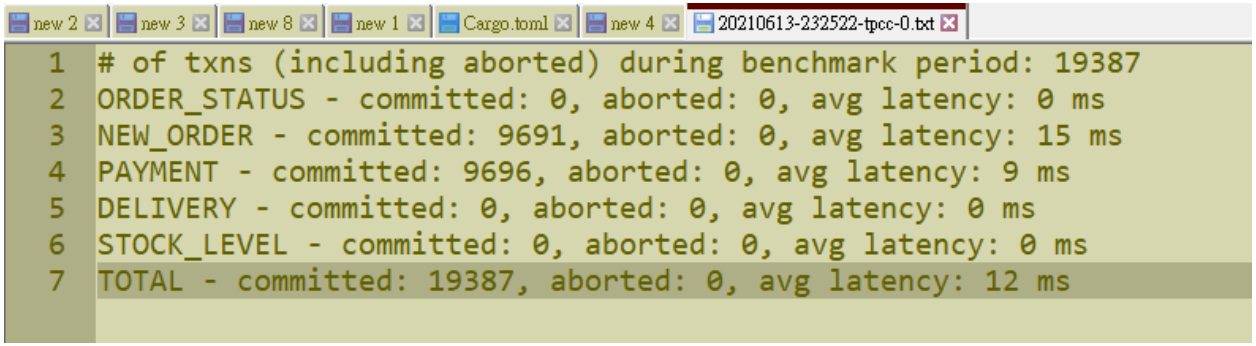
Examples of Reports



A1 | time(sec)

	A	B	C	D	E	F	G	H
1	time(sec)	throughpu	avg_latenc	min(ms)	max(ms)	25th_lat(m	median_la	75th_lat(ms)
2	0	332	11.57831	1	59	6	9	14
3	1	402	9.440299	1	27	6	8	13
4	2	266	14.1203	2	83	7	11	17
5	3	312	12.72436	2	77	6	9	15
6	4	398	9.432161	1	27	6	9	12
7	5	305	12.4623	1	90	7	10	14
8	6	241	16.18672	2	100	8	12	18
9	7	420	9.069048	1	25	6	9	12
10	8	369	9.932249	1	49	6	9	12
11	9	209	19.28708	1	77	7	13	25
12	10	411	9.250608	1	21	6	9	12
13	11	399	9.378446	2	25	6	9	12

20210613-232522-tpcc-0



new 2 | new 3 | new 8 | new 1 | Cargo.toml | new 4 | 20210613-232522-tpcc-0.txt

```
1  # of txns (including aborted) during benchmark period: 19387
2  ORDER_STATUS - committed: 0, aborted: 0, avg latency: 0 ms
3  NEW_ORDER - committed: 9691, aborted: 0, avg latency: 15 ms
4  PAYMENT - committed: 9696, aborted: 0, avg latency: 9 ms
5  DELIVERY - committed: 0, aborted: 0, avg latency: 0 ms
6  STOCK_LEVEL - committed: 0, aborted: 0, avg latency: 0 ms
7  TOTAL - committed: 19387, aborted: 0, avg latency: 12 ms
```

Outline

- Introduction to ElaSQL project
- How to test/benchmark the system?
  - Let's meet ElaSQL-Bench
  - Setting up development environment
  - Testing inside a Java IDE
  - Testing with runnable JARs
  - Testing in a cluster

Testing with Runnable JARs

- In most of time, you may want to test ElaSQL in clean environments without interfere, so running with an IDE may not be a proper way.

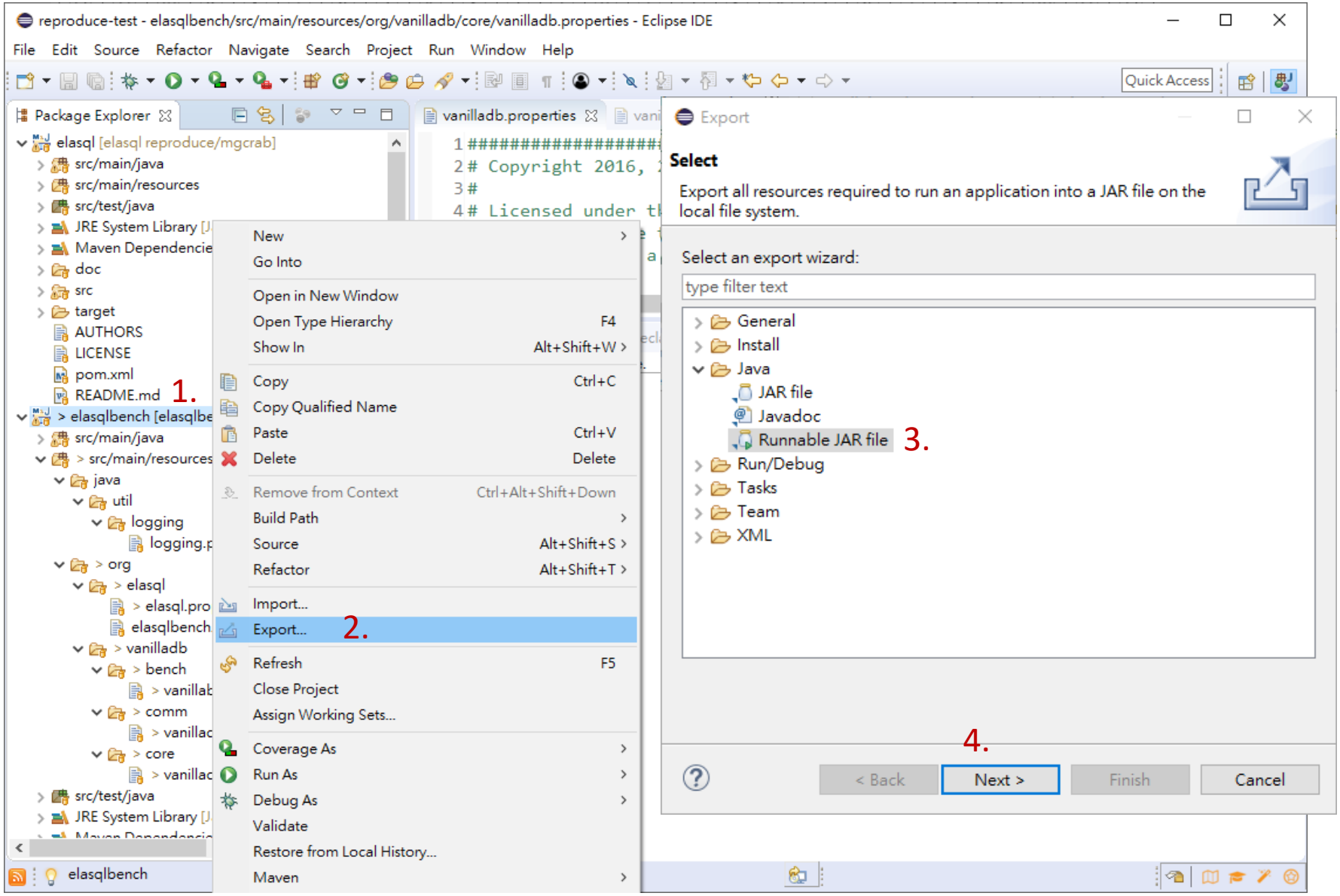
- In that case, we export the projects as runnable JARs and run with scripts.

Steps to Run with Runnable JARs

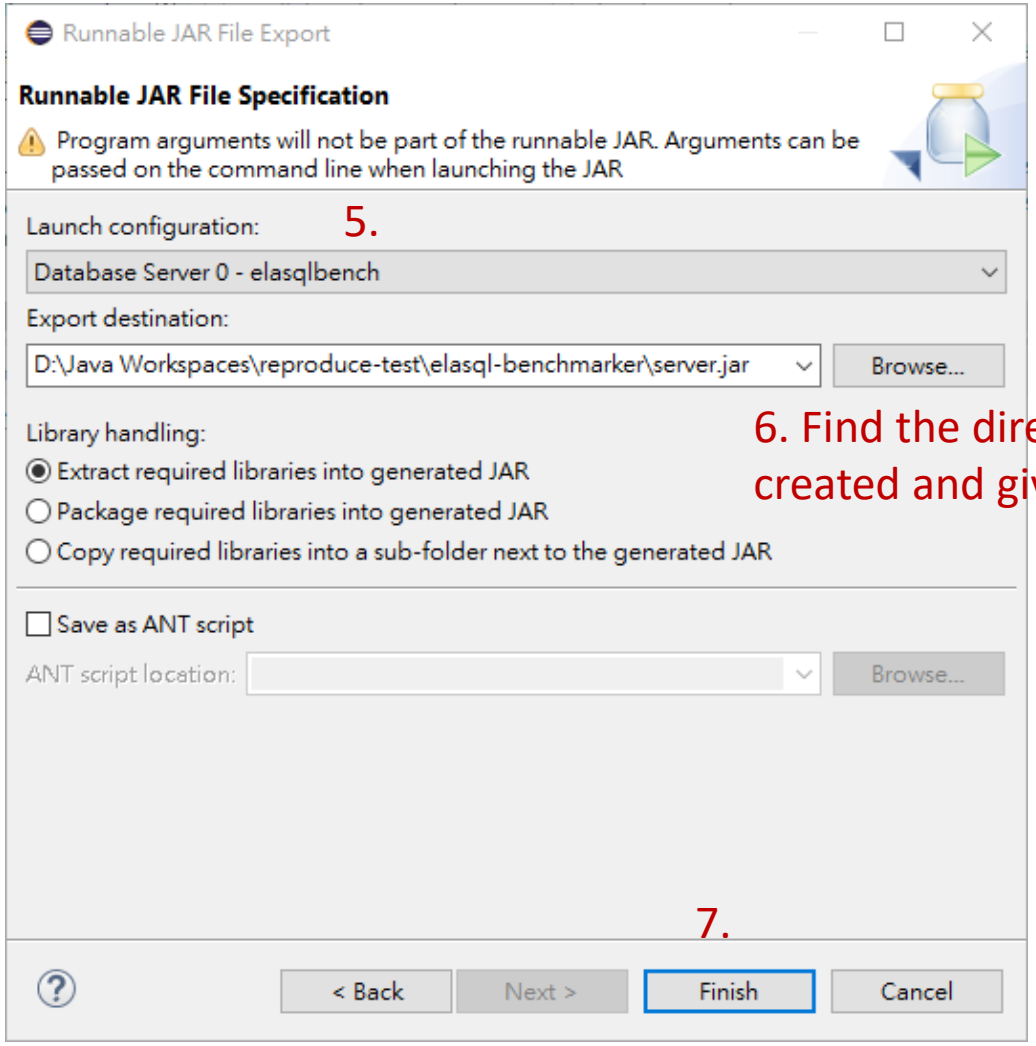
1. Create a directory to put all things together
2. Export the projects into runnable JARs
   - One for servers (including the sequencer) and one for clients
3. Copy the properties files
4. Writing scripts
5. Run with scripts!

Steps to Run with Runnable JARs

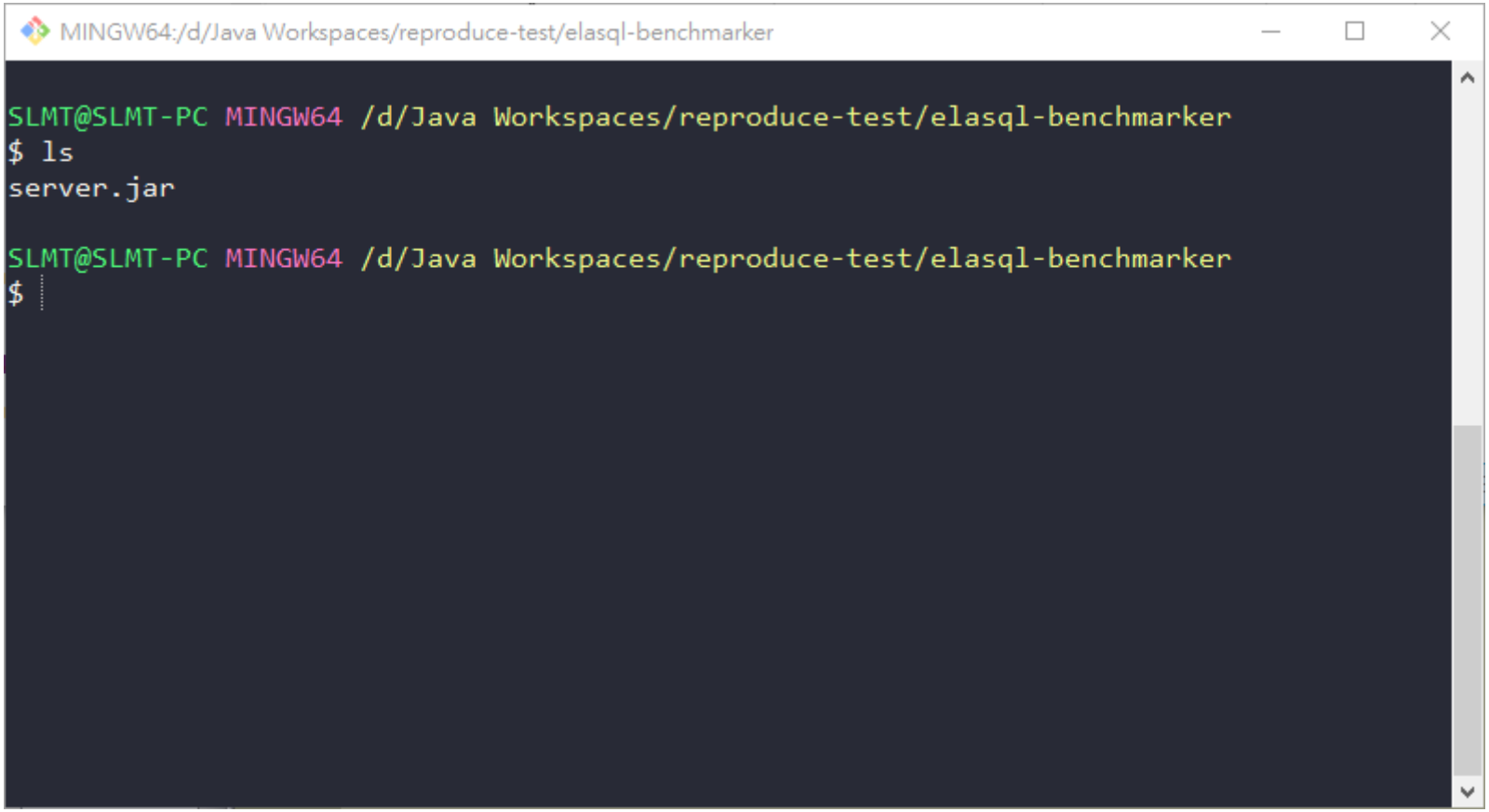
1. Create a directory to put all things together
2. Export the projects into runnable JARs
   - One for servers (including the sequencer) and one for clients
3. Copy the properties files
4. Writing scripts
5. Run with scripts!


reproduce-test - elasqlbench/src/main/resources/org/vanilladb/core/vanilladb.properties - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Package Explorer
elasql [elasql reproduce/mgcrab]
src/main/java
src/main/resources
src/test/java
JRE System Library [J
Maven Dependencie
doc
src
target
AUTHORS
LICENSE
pom.xml
README.md
1.
> elasqlbench [elasqlbe
src/main/java
> src/main/resources
java
util
logging
logging.p
> org
> elasql
> elasql.pro
elasqlbench
> vanilladb
> bench
> vanillab
> comm
> vanillac
> core
> vanillac
src/test/java
JRE System Library [J
elasqlbench
vanilladb.properties
1 ##################
2 # Copyright 2016,
3 #
4 # Licensed under th
New
Go Into
Open in New Window
Open Type Hierarchy F4
Show In Alt+Shift+W >
Copy Ctrl+C
Copy Qualified Name
Paste Ctrl+V
Delete Delete
Remove from Context Ctrl+Alt+Shift+Down
Build Path
Source Alt+Shift+S >
Refactor Alt+Shift+T >
Import...
Export...
2.
Refresh F5
Close Project
Assign Working Sets...
Coverage As
Run As
Debug As
Validate
Restore from Local History...
Maven
Export
Select
Export all resources required to run an application into a JAR file on the local file system.
Select an export wizard:
type filter text
General
Install
Java
JAR file
Javadoc
Runnable JAR file
3.
Run/Debug
Tasks
Team
XML
4.
< Back
Next >
Finish
Cancel

Exporting a Server JAR

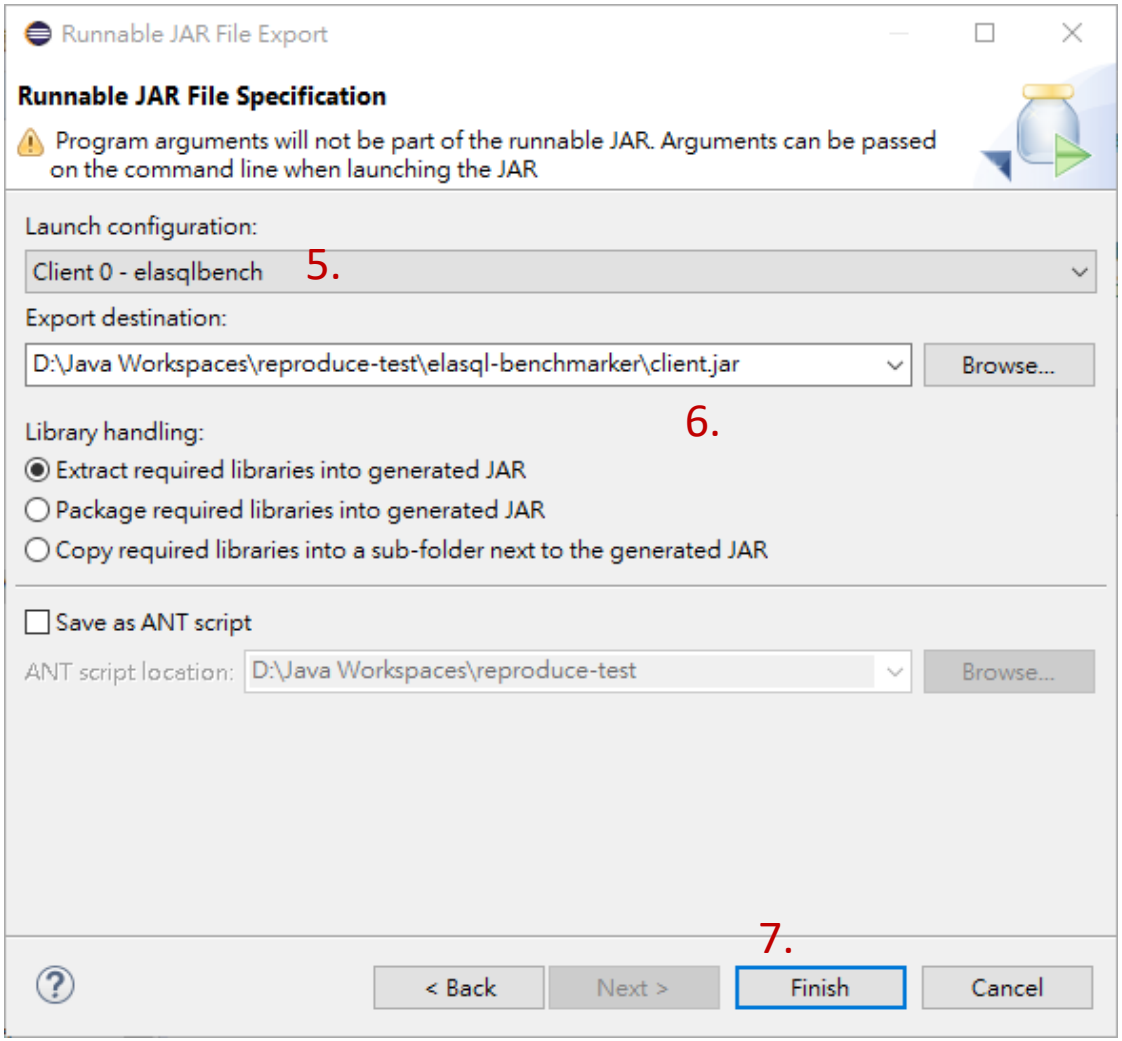


Server JAR

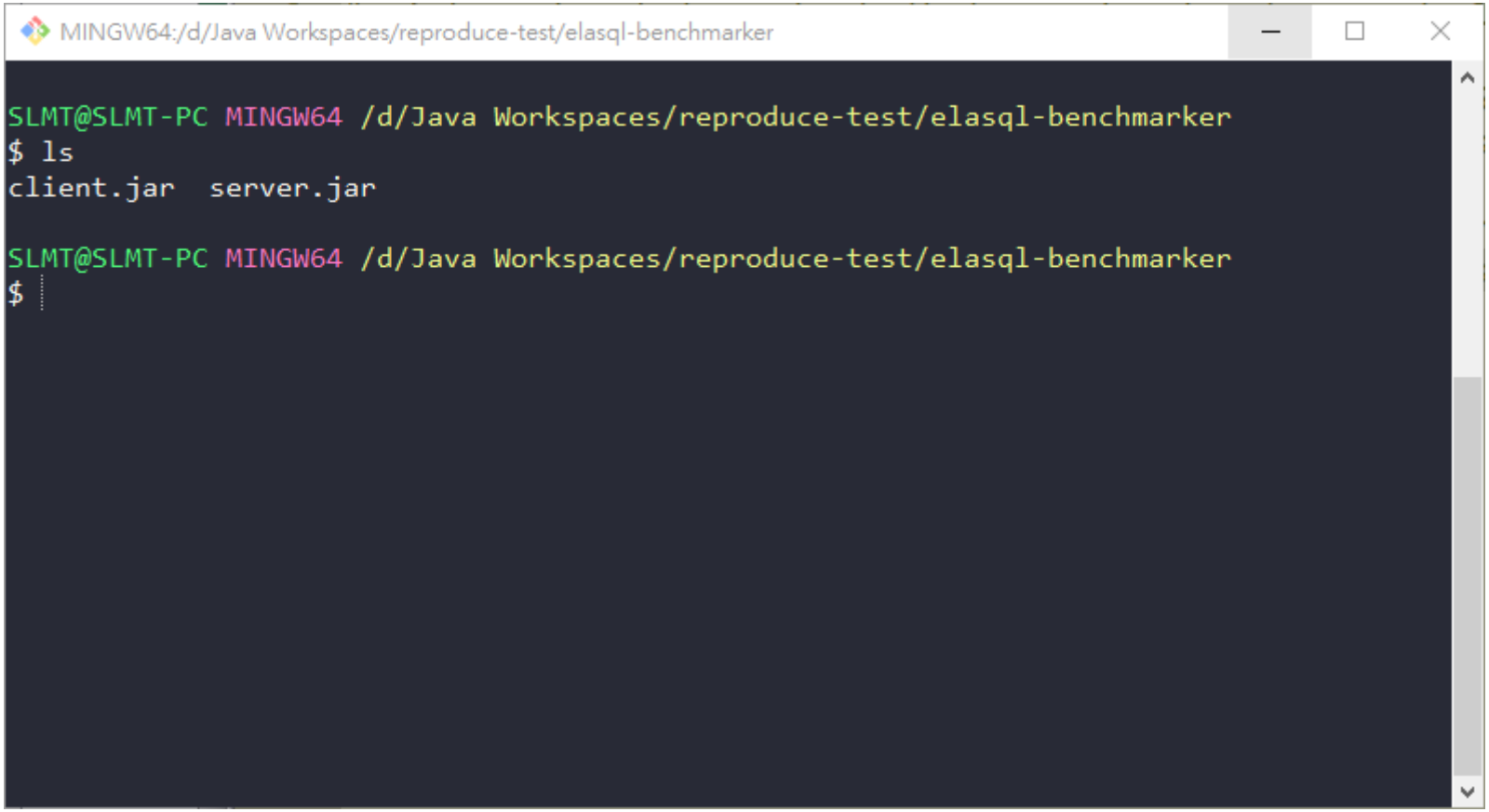

MINGW64:/d/Java Workspaces/reproduce-test/elasql-benchmarker
SLMT@SLMT-PC MINGW64 /d/Java Workspaces/reproduce-test/elasql-benchmarker
$ ls
server.jar
SLMT@SLMT-PC MINGW64 /d/Java Workspaces/reproduce-test/elasql-benchmarker
$

Exporting a Client JAR

Step 1~4 are same

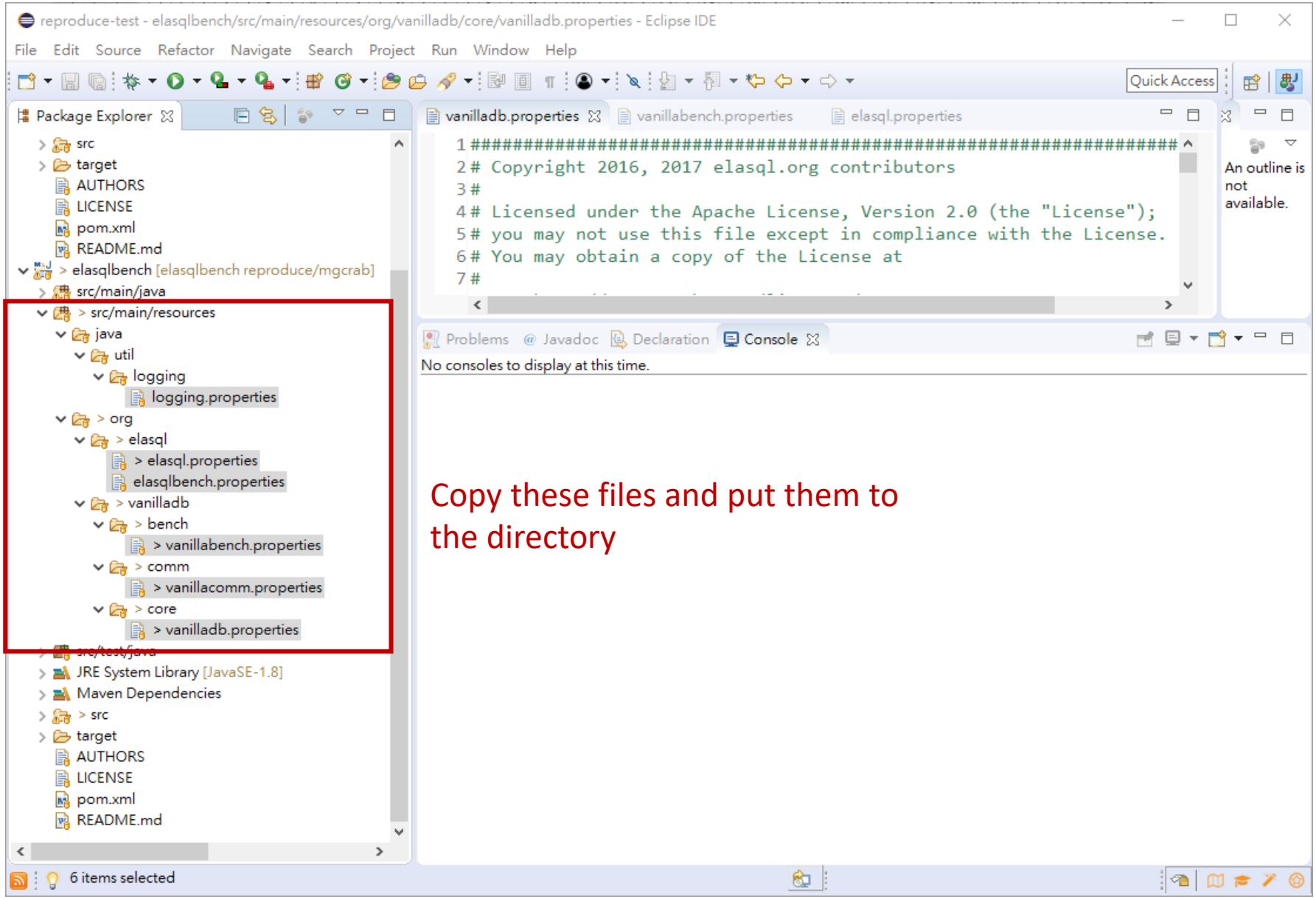


Server & Client JARs

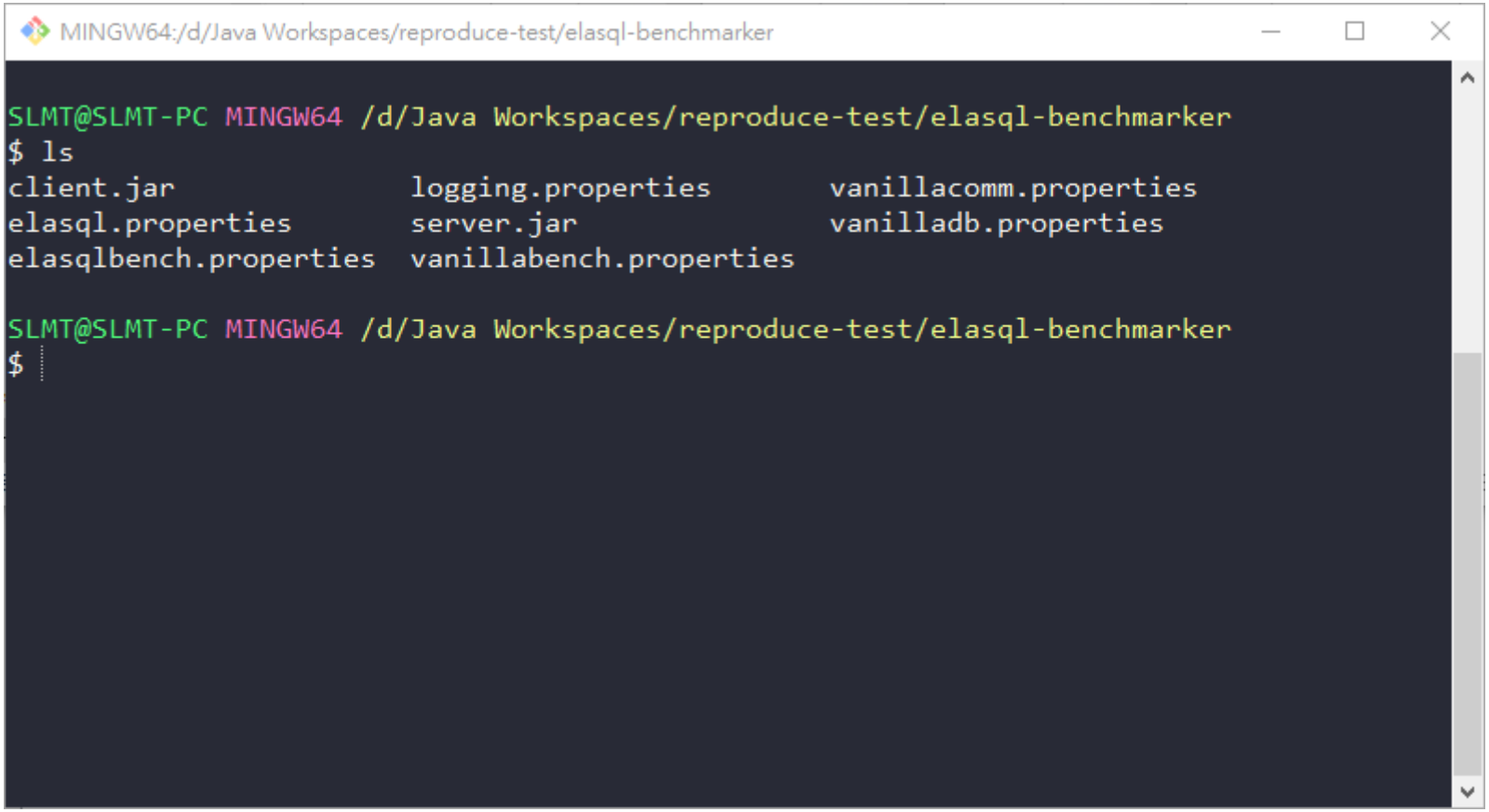


Steps to Run with Runnable JARs

1. Create a directory to put all things together
2. Export the projects into runnable JARs
   - One for servers (including the sequencer) and one for clients
3. Copy the properties files
4. Writing scripts
5. Run with scripts!


reproduce-test - elasqlbench/src/main/resources/org/vanilladb/core/vanilladb.properties - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Package Explorer
src
target
AUTHORS
LICENSE
pom.xml
README.md
elasqlbench [elasqlbench reproduce/mgcrab]
src/main/java
src/main/resources
java
util
logging
logging.properties
org
elasql
elasql.properties
elasqlbench.properties
vanilladb
bench
vanillabench.properties
comm
vanillacomm.properties
core
vanilladb.properties
JRE System Library [JavaSE-1.8]
Maven Dependencies
src
target
AUTHORS
LICENSE
pom.xml
README.md
6 items selected
vanilladb.properties
vanillabench.properties
elasql.properties
1 ##
2 # Copyright 2016, 2017 elasql.org contributors
3 #
4 # Licensed under the Apache License, Version 2.0 (the "License");
5 # you may not use this file except in compliance with the License.
6 # You may obtain a copy of the License at
7 #
An outline is not available.
Problems Javadoc Declaration Console
No consoles to display at this time.
Copy these files and put them to the directory

Server & Client JARs + Properties Files



Steps to Run with Runnable JARs

1. Create a directory to put all things together
2. Export the projects into runnable JARs
   - One for servers (including the sequencer) and one for clients
3. Copy the properties files
4. Writing scripts
5. Run with scripts!

Writing a Script for Servers (Including the Sequencer)

- Copy the script below and save it as server.sh

```
java \
-Dorg.elasql.config.file=elasql.properties \
-Dorg.elasql.bench.config.file=elasqlbench.properties \
-Dorg.vanilladb.comm.config.file=vanillacomm.properties \
-Dorg.vanilladb.bench.config.file=vanillabench.properties \
-Dorg.vanilladb.core.config.file=vanilladb.properties \
-Djava.util.logging.config.file=logging.properties \
-jar server.jar \
$1 \
$2 \
$3 \
```

You can also copy the content of the script from here.

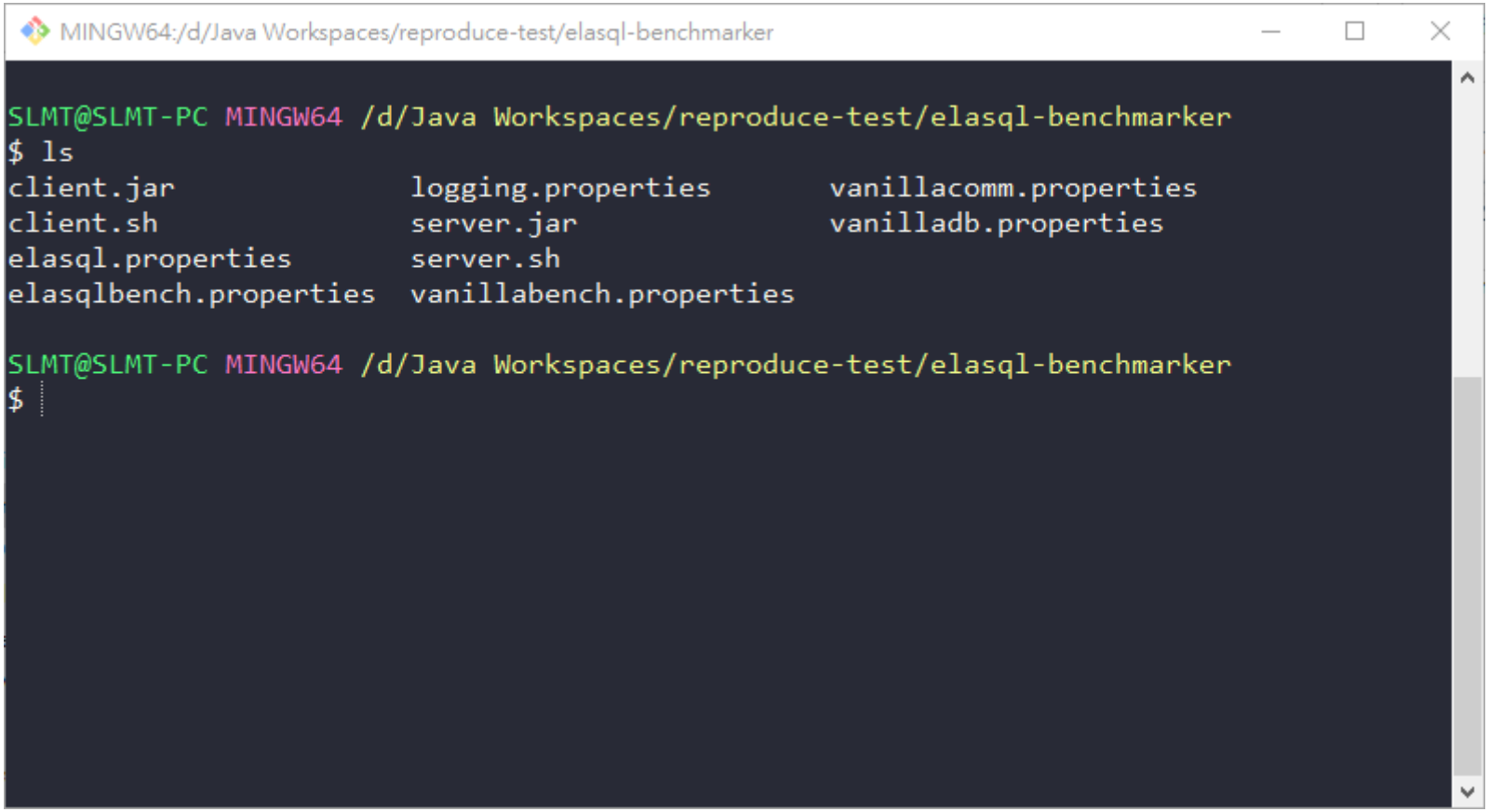
Writing a Script for Clients

- Copy the script below and save it as server.sh

```
java \
-Dorg.elasql.config.file=elasql.properties \
-Dorg.elasql.bench.config.file=elasqlbench.properties \
-Dorg.vanilladb.comm.config.file=vanillacomm.properties \
-Dorg.vanilladb.bench.config.file=vanillabench.properties \
-Dorg.vanilladb.core.config.file=vanilladb.properties \
-Djava.util.logging.config.file=logging.properties \
-jar client.jar \
$1 \
$2 \
```

You can also copy the content of the script from here.

All the Things We Need Are Now In Place



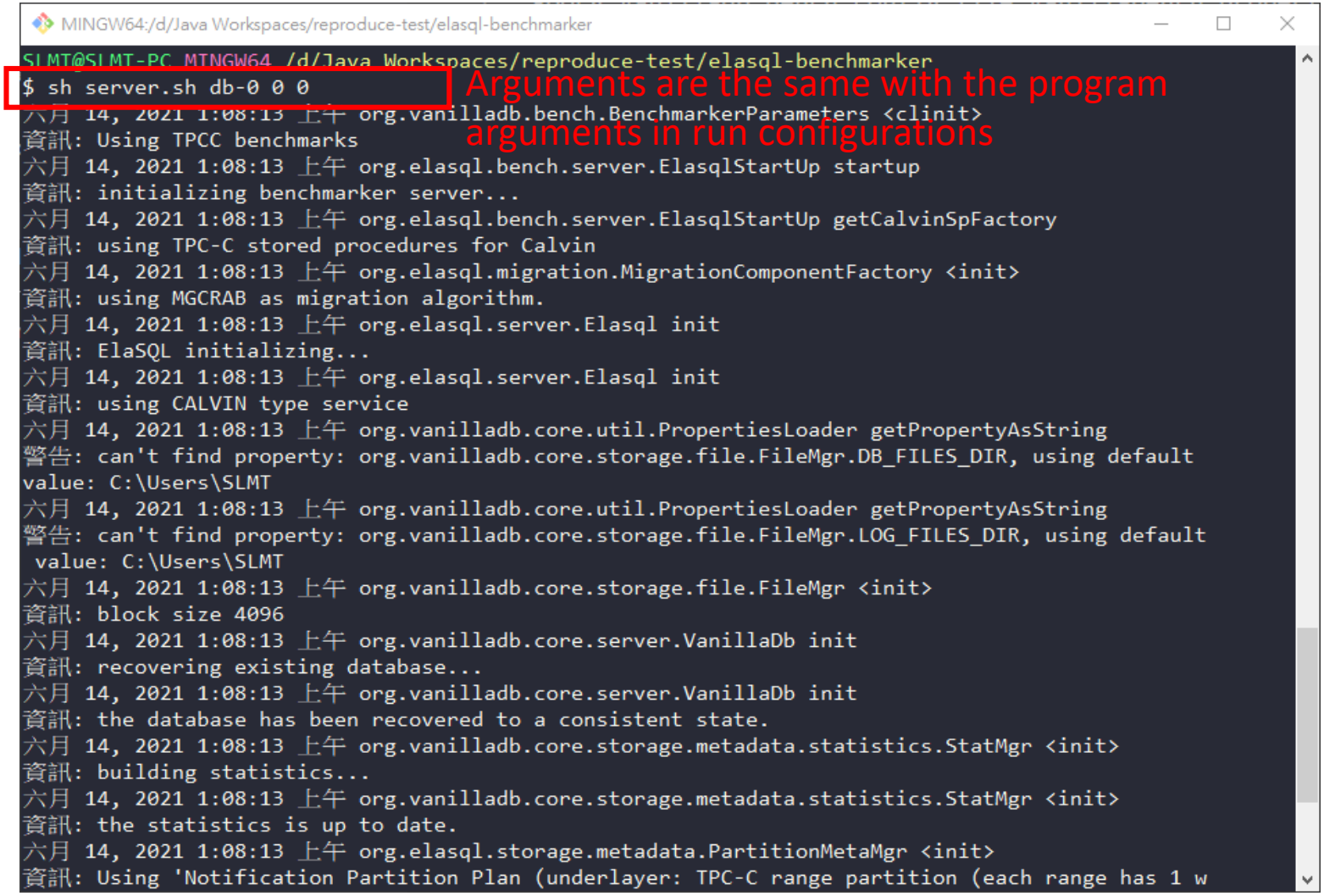
Steps to Run with Runnable JARs

1. Create a directory to put all things together
2. Export the projects into runnable JARs
   - One for servers (including the sequencer) and one for clients
3. Copy the properties files
4. Writing scripts
5. Run with scripts!

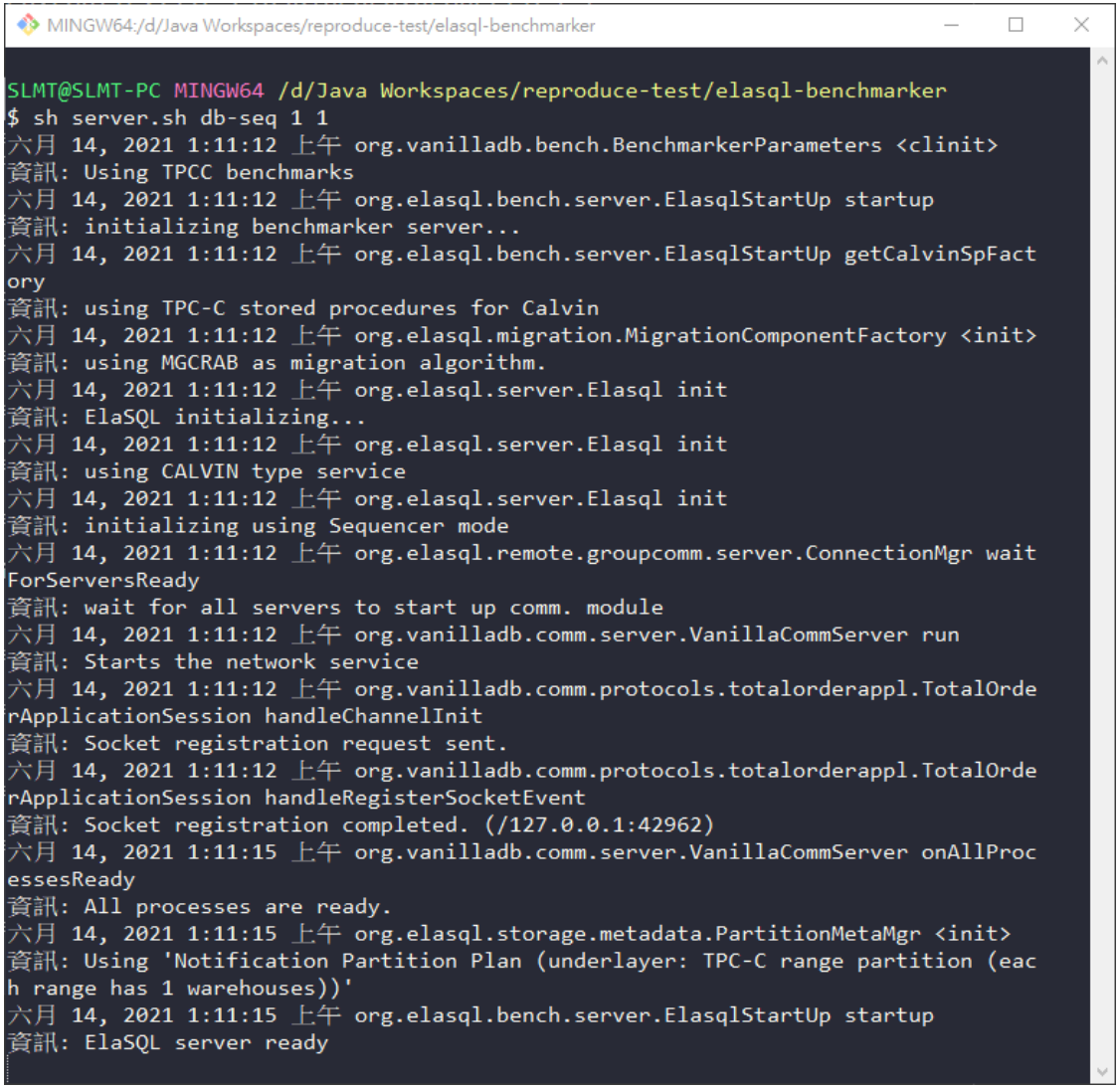
Running the Servers & Clients

- The procedure to run the servers and clients are identical with running in Eclipse.
- The only difference is that we start processes with scripts.

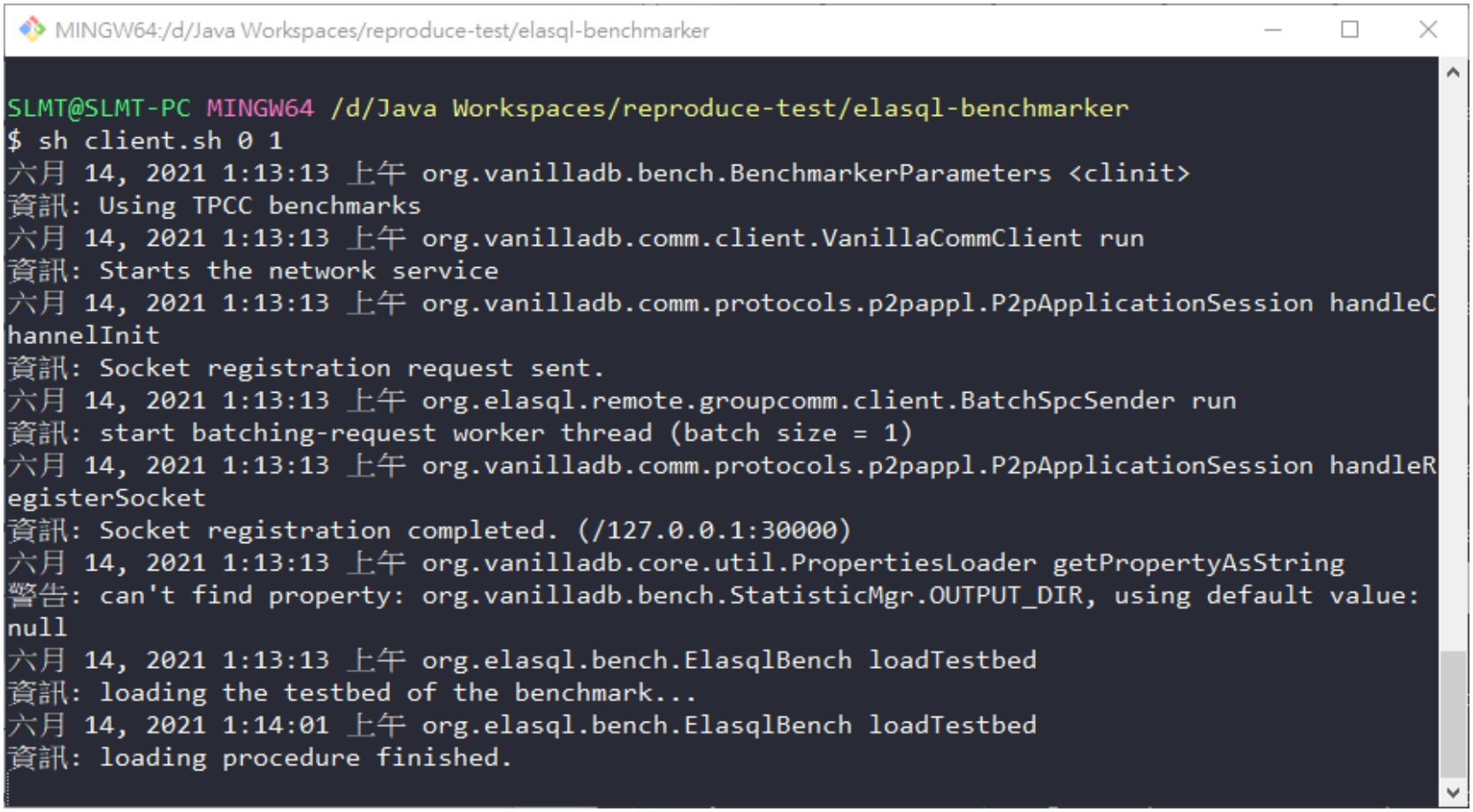
Starting a Database Server



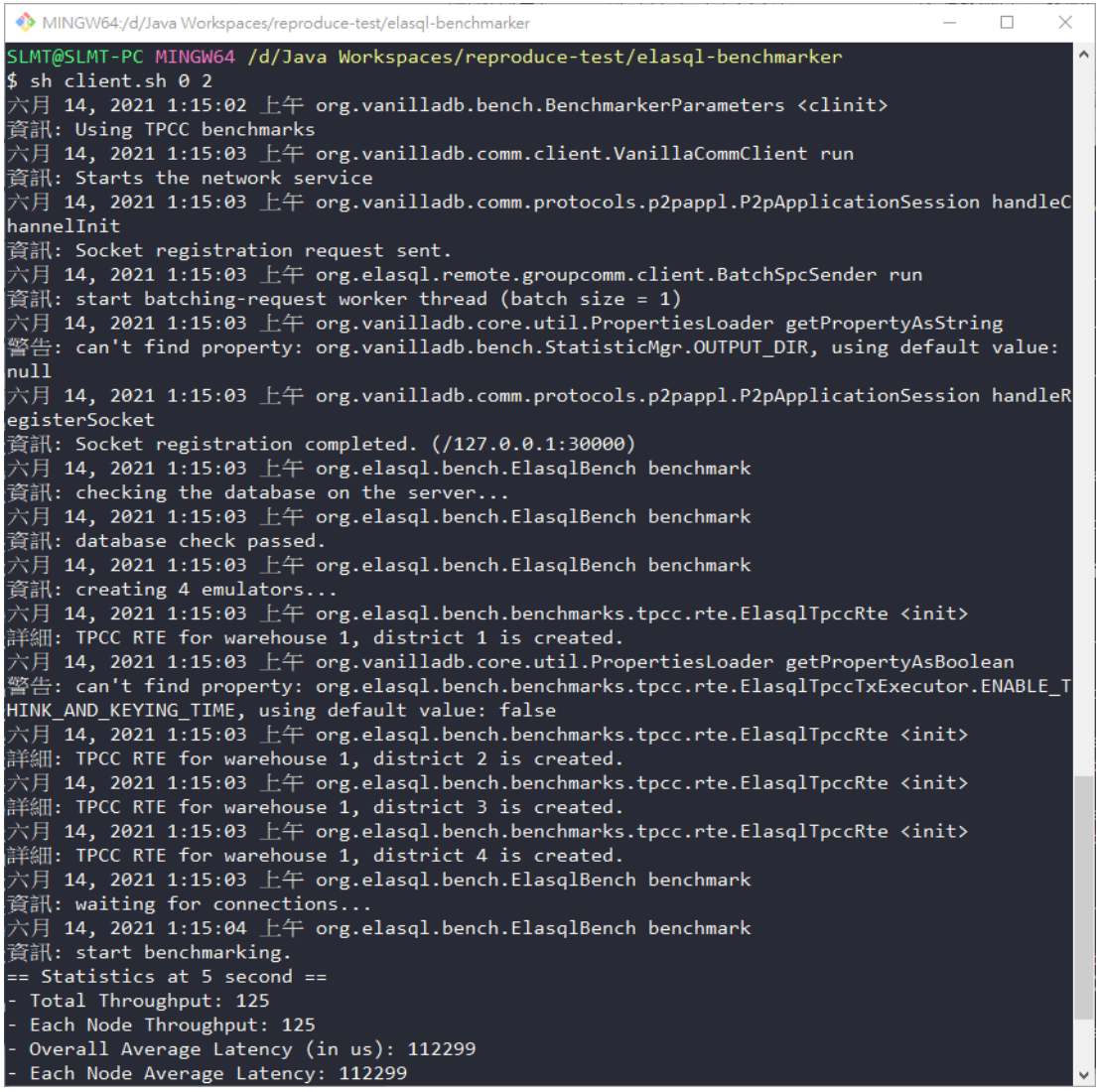
Starting a Sequencer Server



Starting a Client for Loading



Starting a Client for Benchmarking



Outline

- Introduction to ElaSQL project
- How to test/benchmark the system?
  - Let's meet ElaSQL-Bench
  - Setting up development environment
  - Testing inside a Java IDE
  - Testing with runnable JARs
  - Testing in a cluster

Testing in a Cluster

- It is not hard to manually start a few servers and clients on some machines.
- However, things get mess when there are tens of servers and clients to run on a cluster.
  - Imagine to run a scalability experiment with 20 servers, 1 sequencer, and 20 clients.
- Since we have known how to run the projects with scripts, you can write your own scripts to deal with the large scale experiments.

Auto-Bencher

- Or, you can just use the one we created:
  - https://github.com/SLMT/auto-bencher
- Key Features
  - Setting up testing environments on a clean machine.
  - Deploying ElaSQL-Bench JARs to testing machines.
  - Backing up testbeds.
  - Organizing different parameters into a test set.
  - Collecting and summaries the reports from clients.
- Note: we are currently working on migrating this project to JavaScript, so the above one may get outdated soon.

Have Fun!