Assignment 4 & Benchmarks

Introduction to Database Systems

DataLab

CS, NTHU

Outline

- Assignment 4
- Benchmarks
  - The Micro-benchmark
  - The TPC-C Benchmark
- Guidelines for Experiments
- Example Results
- Benchmarking with Scripts

Outline

- Assignment 4
- Benchmarks
  - The Micro-benchmark
  - The TPC-C Benchmark
- Guidelines for Experiments
- Example Results
- Benchmarking with Scripts

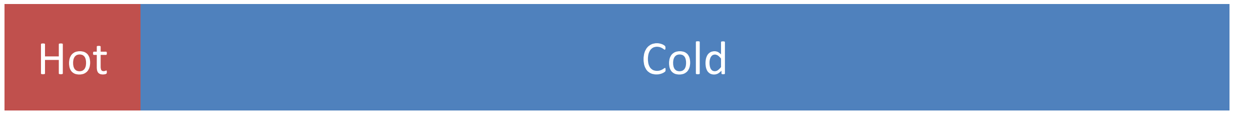
Outline

- Assignment 4
- Benchmarks
  - The Micro-benchmark
  - The TPC-C Benchmark
- Guidelines for Experiments
- Example Results
- Benchmarking with Scripts

The Micro-Benchmark

- Two types of transactions.
  - **Read-only** transaction => reads 10 records.
  - **Read-write** transaction => reads and updates 10 records.
  - The ratio is controlled by RW_TX_RATE.

- The data set is split into two parts.



  - 1 is chosen from hot set, 9 are chosen from cold set.
  - The number of hot records is control by HOT_CONFLICT_RATE.

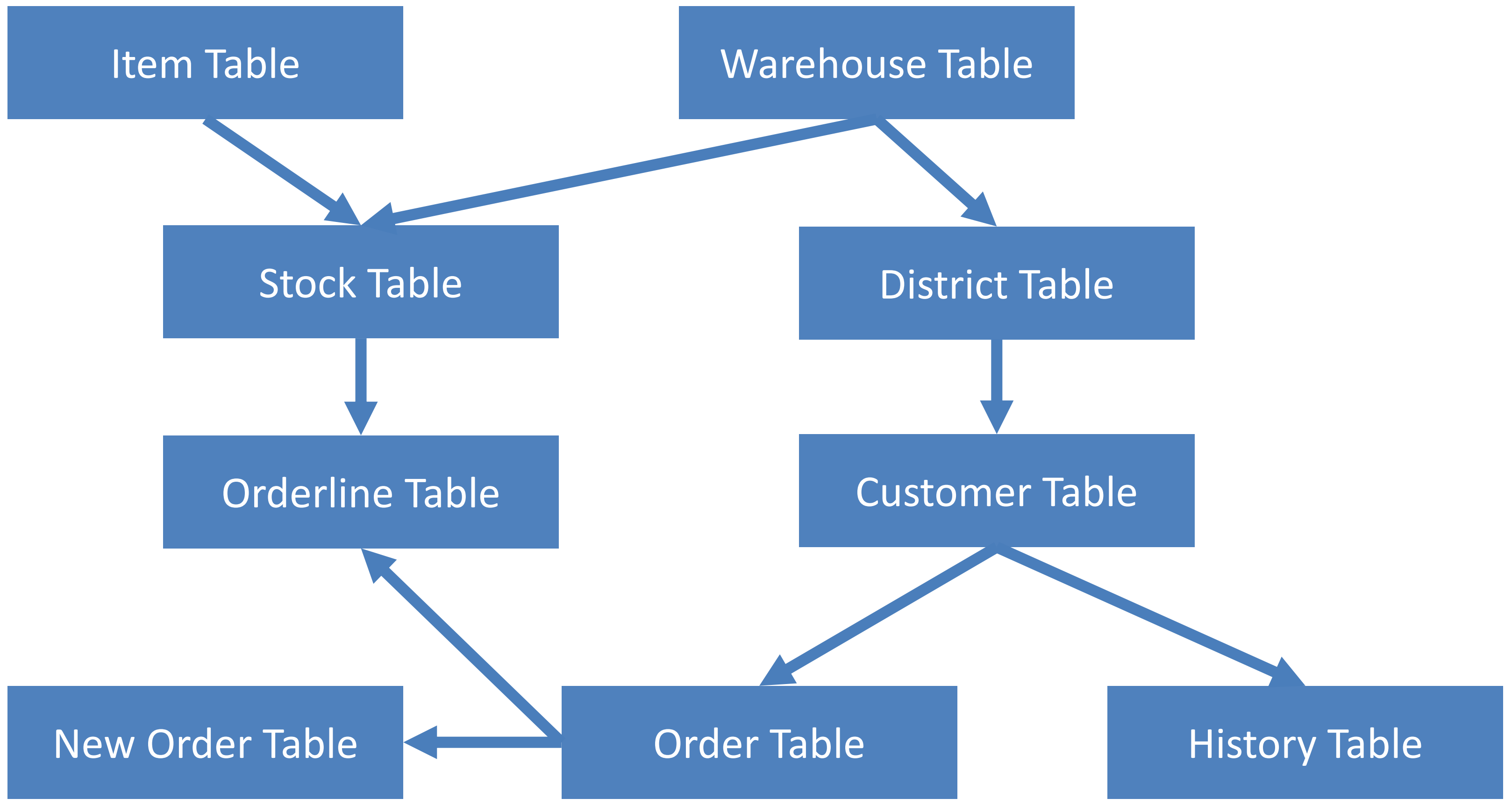
Outline

- Assignment 4
- Benchmarks
  - The Micro-benchmark
  - The TPC-C Benchmark
- Guidelines for Experiments
- Example Results
- Benchmarking with Scripts

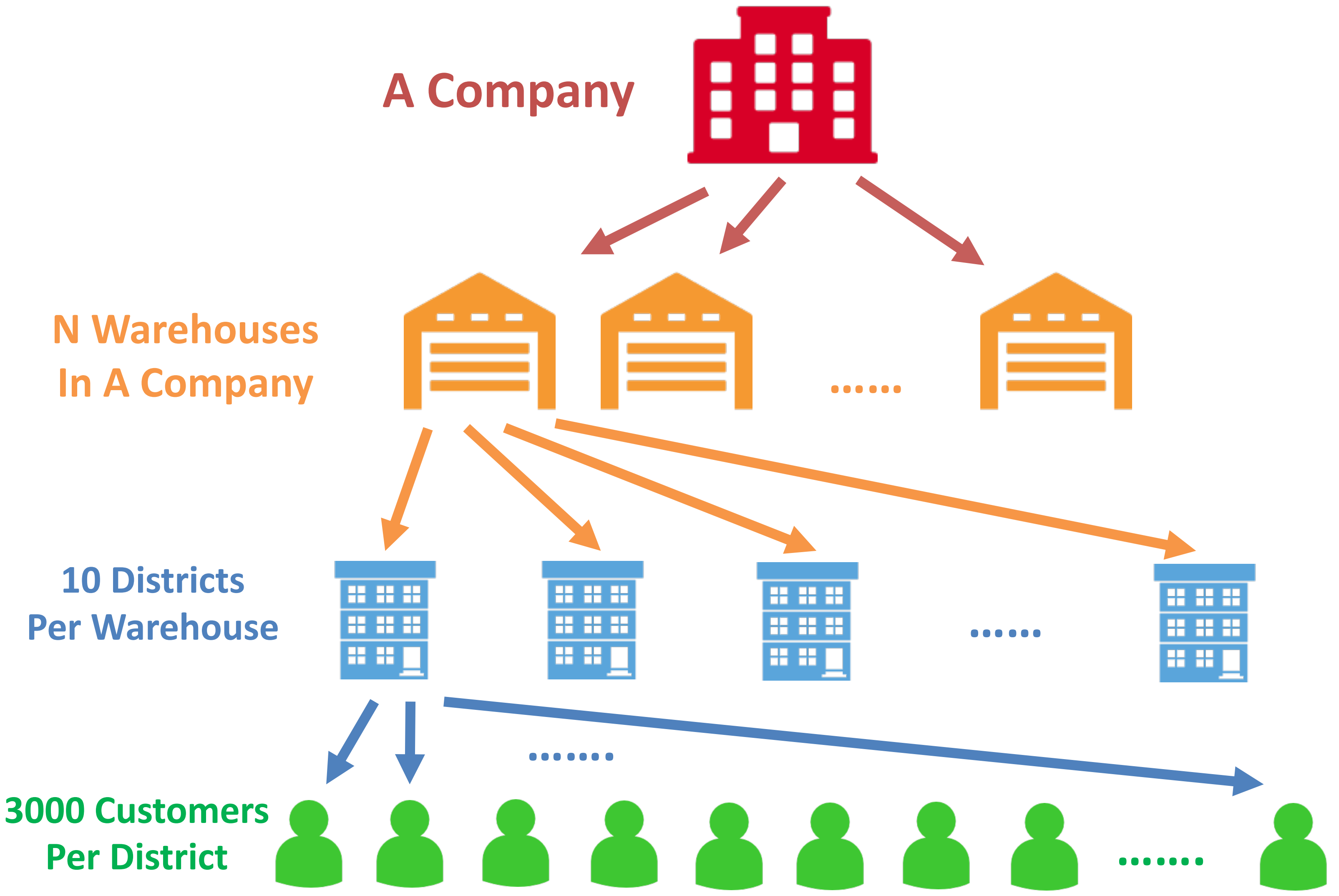
The TPC-C Benchmark

- The TPC-C benchmark is a industry-standard benchmark purposed by TPC (Transaction Processing Council).
  - There are also TPC-A, TPC-B, TPC-E, TPC-H.
- It simulates a warehouse management system.
  - Tree-structured: almost all records are related to a warehouse record.
  - Easy-to-partition: good for a distributed DBMS.

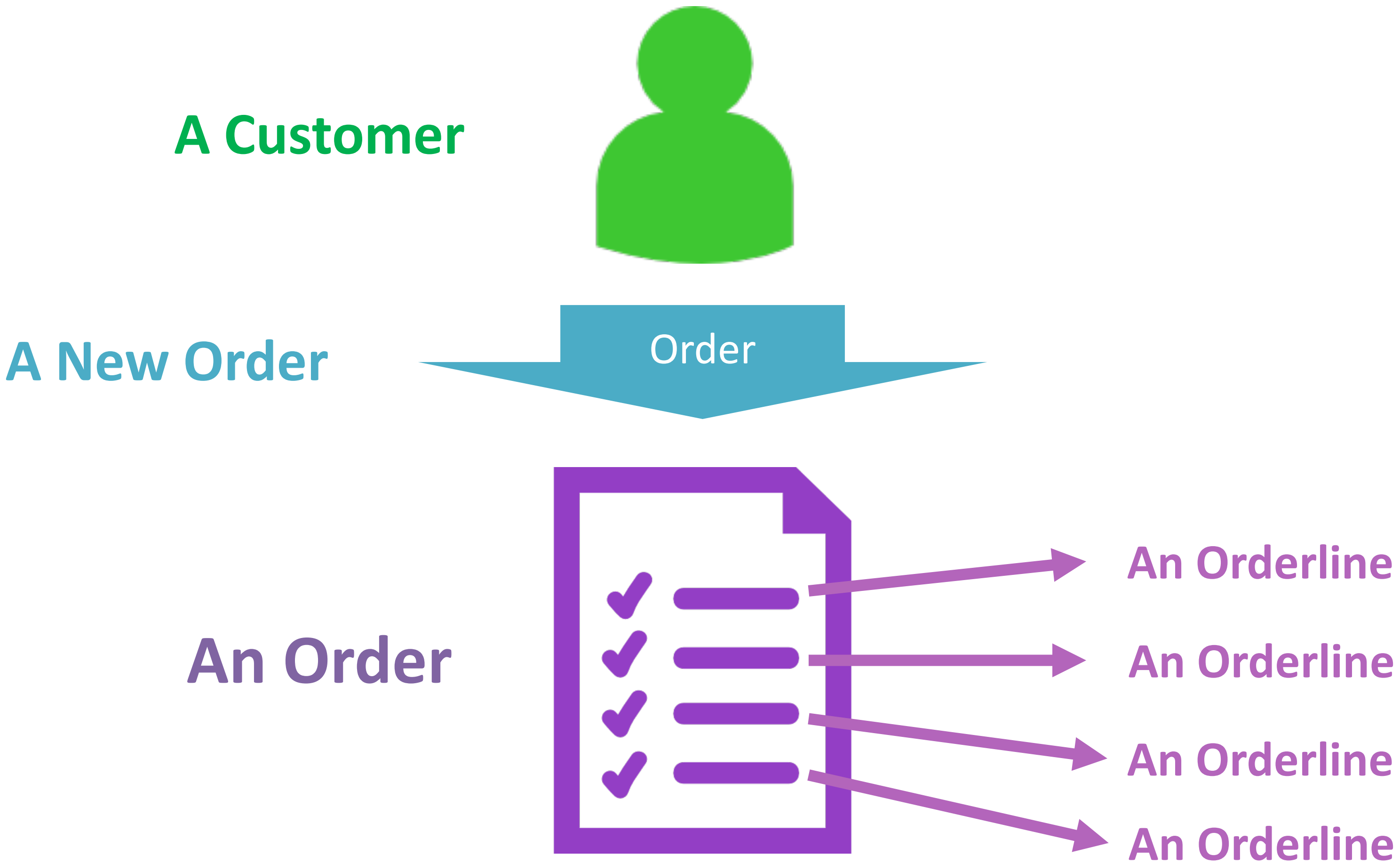
Database Architecture



Warehouses (Tree-Structured)


A Company
N Warehouses In A Company
……
10 Districts Per Warehouse
……
3000 Customers Per District
…….
…….

Orders


A Customer
A New Order
Order
An Order
An Orderline
An Orderline
An Orderline
An Orderline

Types of Transactions

- New Order
  - 23 reads, 11 updates, 12 inserts in average.
- Payment
  - 4 reads, 3 updates, 1 insert.
- Stock Level
- Order Status
- Delivery

Outline

- Assignment 4
- Benchmarks
  - The Micro-benchmark
  - The TPC-C Benchmark
- Guidelines for Experiments
- Example Results
- Benchmarking with Scripts

Guidelines for Experiments

- Think about what settings can highlight your improvement.
- Make sure there is no other CPU-intensive programs running on the testing machines.
- Put the server and the client on different machine if you can.
- Use stored procedures.
- Using a fresh database every time.
- Find best # of RTEs before real experiments.
  - Which give you highest throughput.
- Throughput is a more important indicator for concurrency than latency.
- Draw you results as line plots or histograms in the report.

Outline

- Assignment 4
- Benchmarks
  - The Micro-benchmark
  - The TPC-C Benchmark
- Guidelines for Experiments
- Example Results
- Benchmarking with Scripts

Example Results for the Micro-benchmarks

- Settings
  - RTE = 10
  - RW Tx Rate = 0.5
  - Conflict Rate = 0.001
- Throughputs (txs/min)

Buffer Size	Basic Version	Optimized Version	Speed Up
100000	111558	174521	56%
100	39285	75164	91%

Outline

- Assignment 4
- Benchmarks
  - The Micro-benchmark
  - The TPC-C Benchmark
- Guidelines for Experiments
- Example Results
- Benchmarking with Scripts

Why Do We Need Scripts?

1. To setup the system quickly.
2. To deploy and benchmark the system in different machines.
3. The environment may not have Eclipse!

Check Your Environment

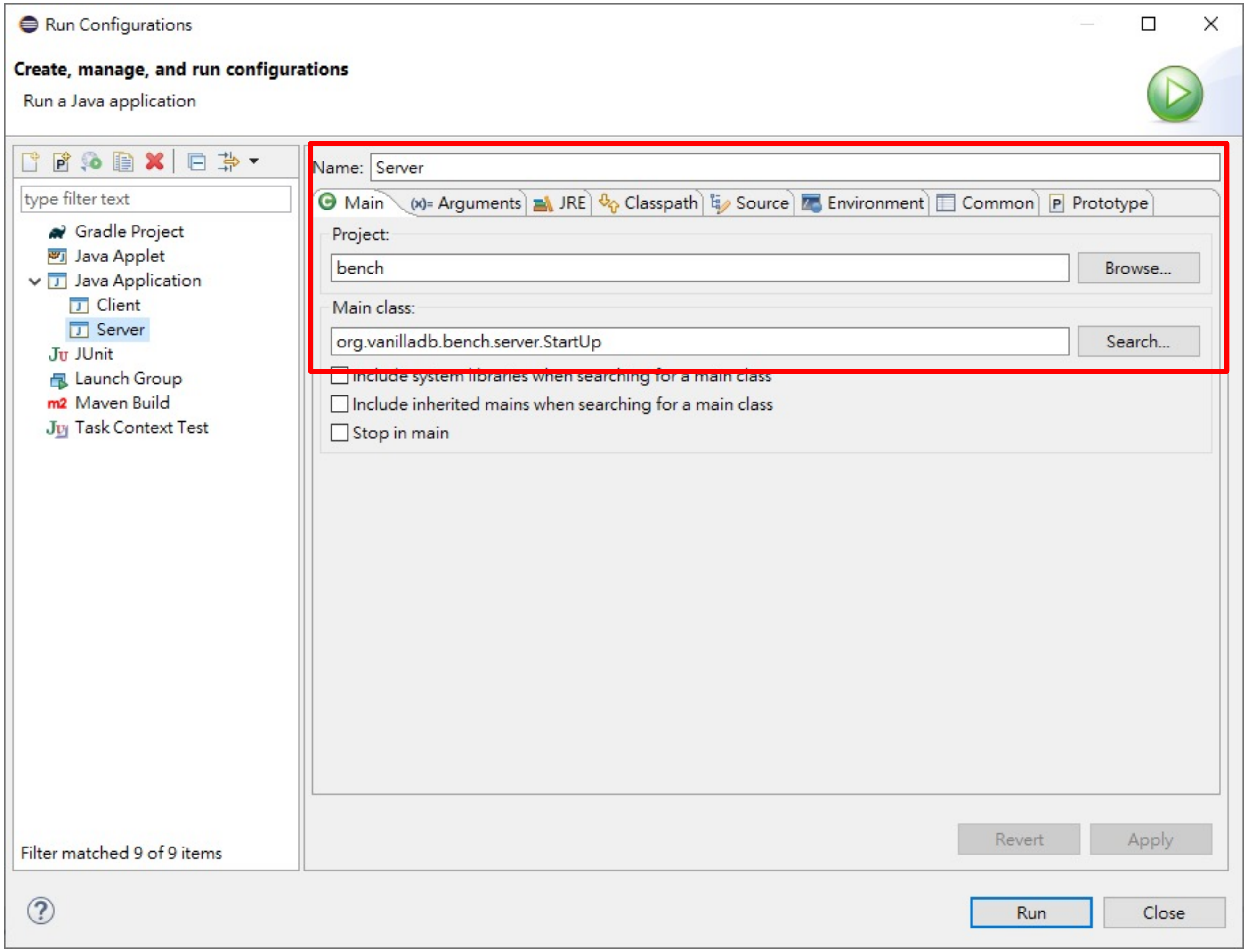
- Requirements
  - Bash
    - Which you may have had if you are using Unix, Unix-like systems or have installed Git on Windows.
  - Java in your system path

```
> java -version
```

Package Your Code

- We use Eclipse built-in tools.
- Steps
  1. Setup run configurations for jars.
  2. Export the project.

Setup Run Configurations - Server



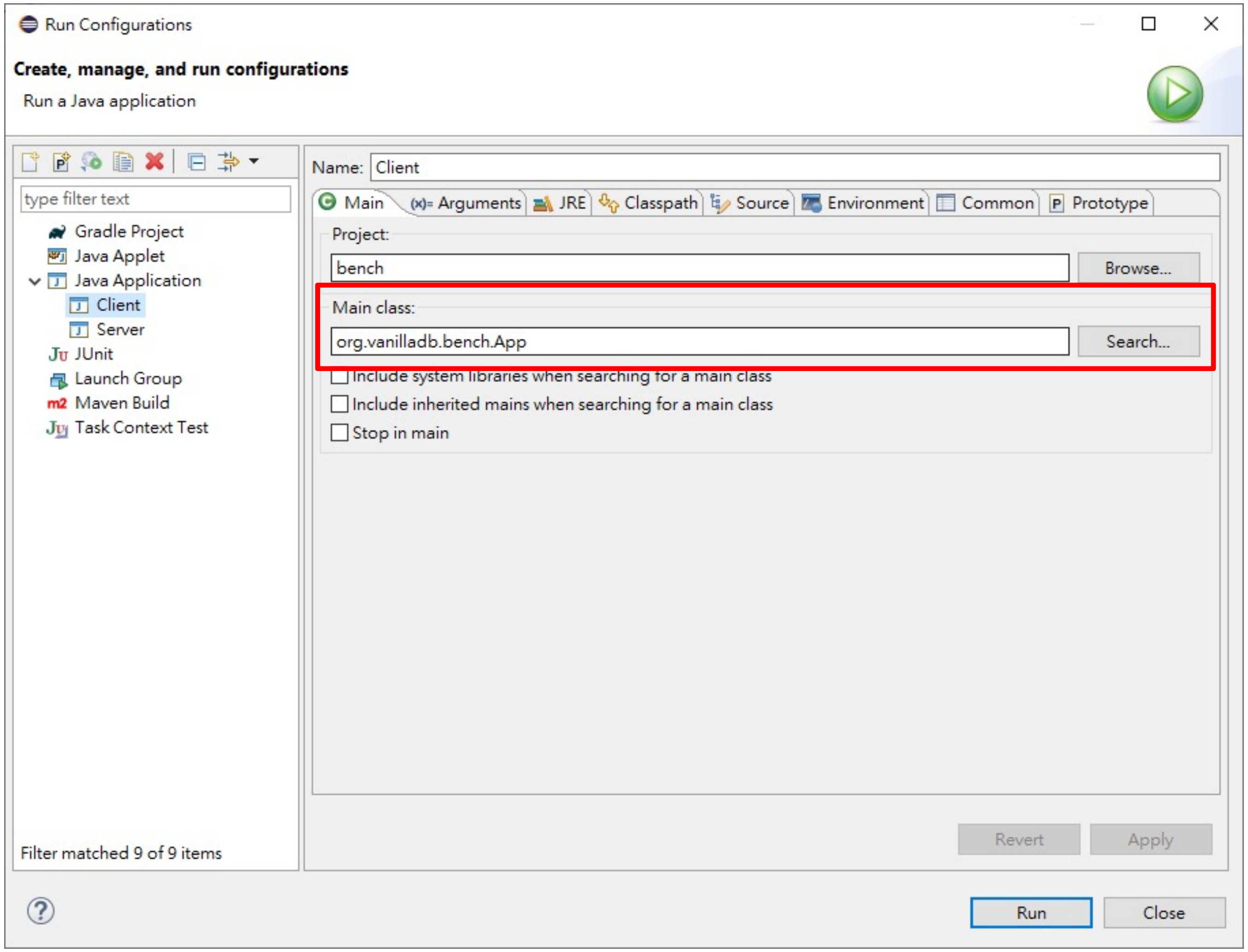
Name: Server

Project: bench

Main class: org.vanilladb.bench.server.StartUp

☐ Include system libraries when searching for a main class
☐ Include inherited mains when searching for a main class
☐ Stop in main

Setup Run Configurations - Client



Run Configurations

Create, manage, and run configurations

Run a Java application

Name: Client

Project: bench

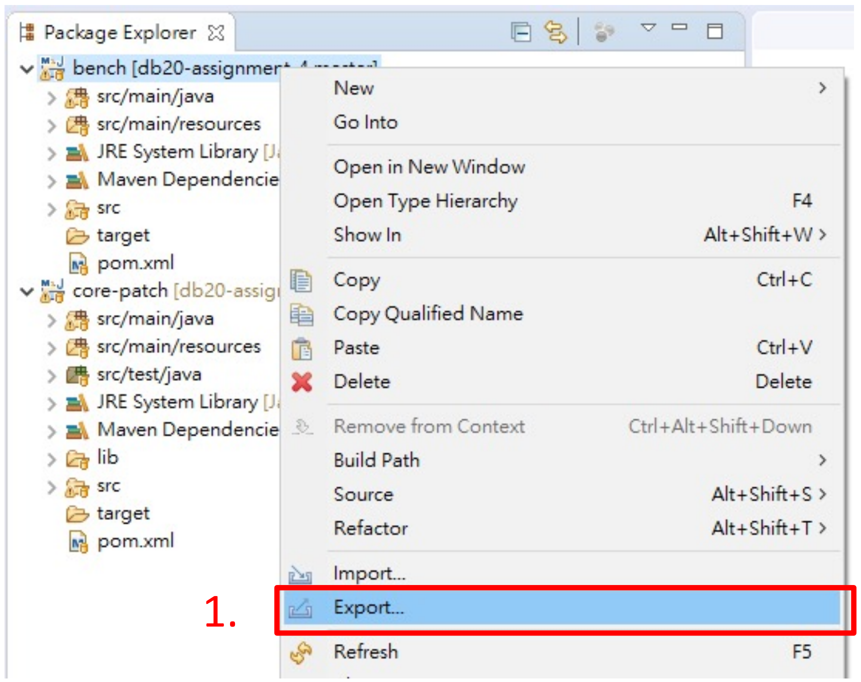
Main class: org.vanilladb.bench.App

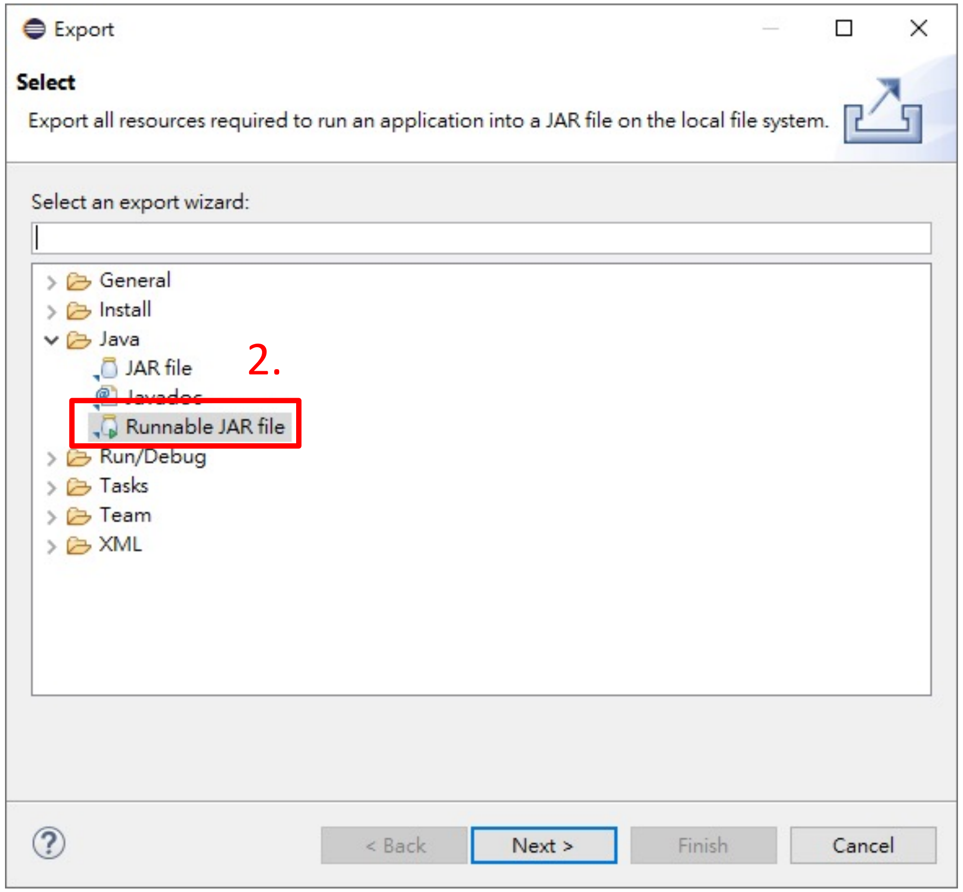
☐ Include system libraries when searching for a main class

☐ Include inherited mains when searching for a main class

☐ Stop in main

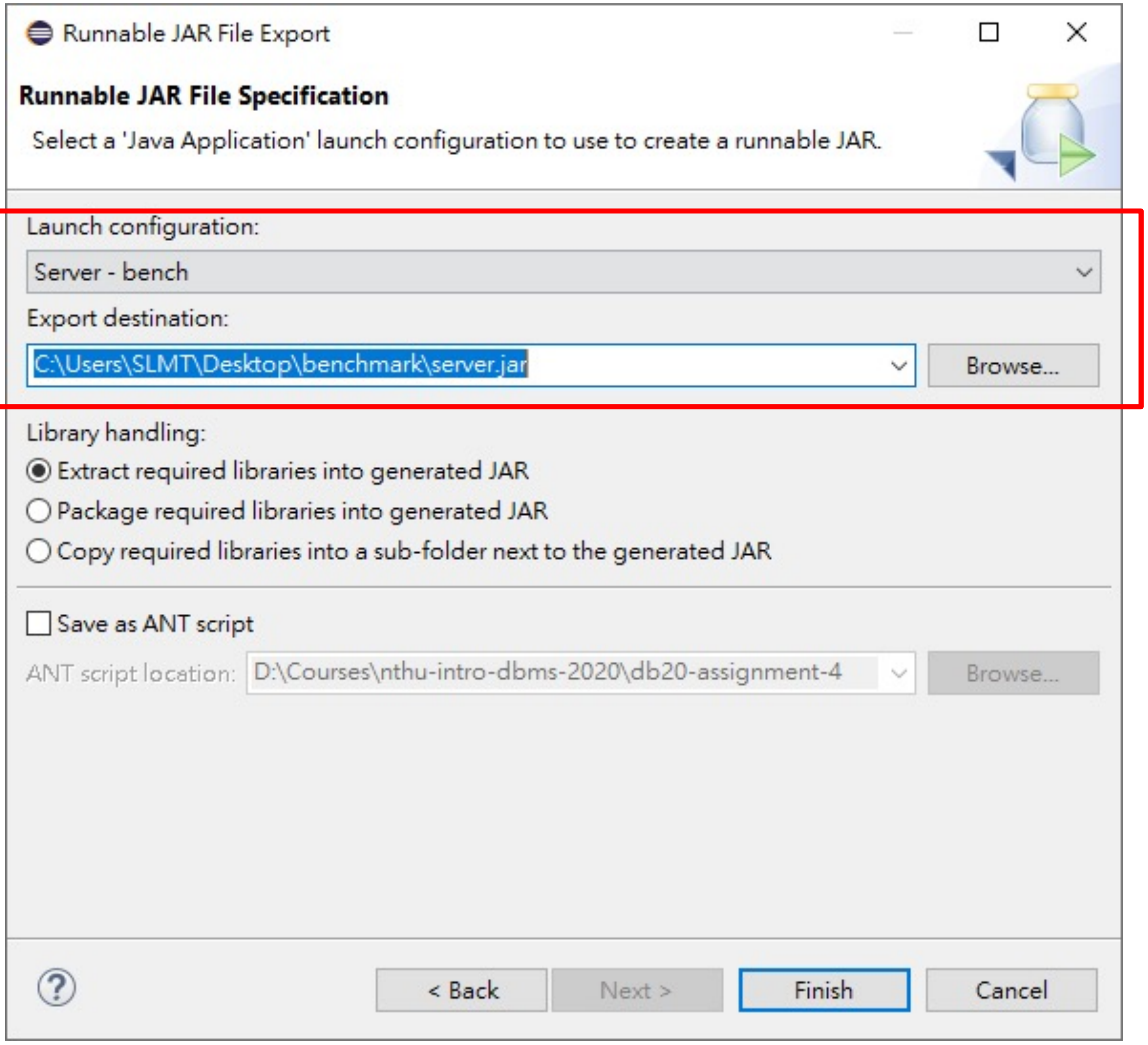
Export Runnable Jars





Export all resources required to run an application into a JAR file on the local file system.

Export Runnable Jars - Server



Runnable JAR File Export

Runnable JAR File Specification

Select a 'Java Application' launch configuration to use to create a runnable JAR.

Launch configuration:

Server - bench

Export destination:

C:\Users\SLMT\Desktop\benchmark\server.jar

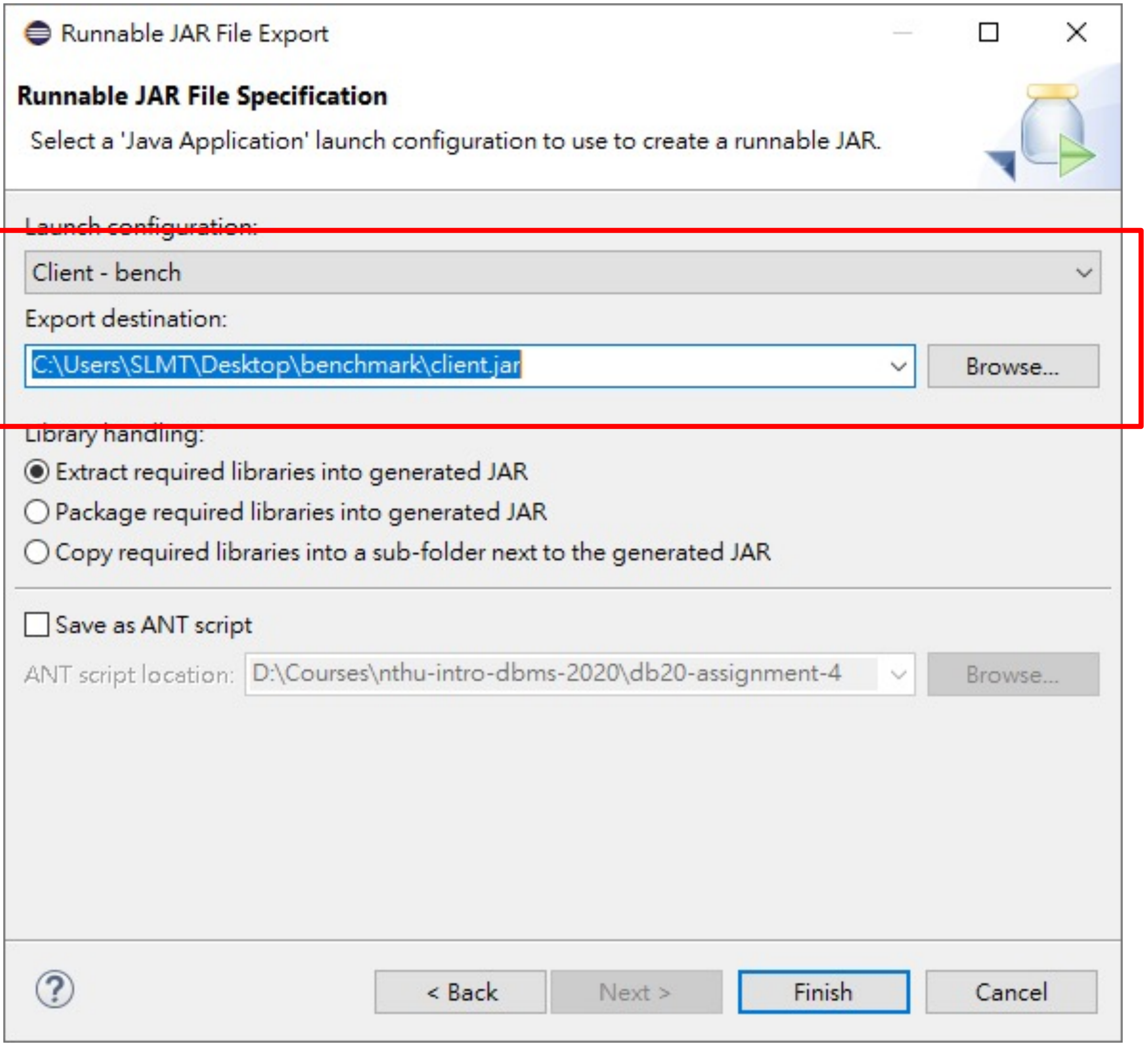
Library handling:

- [✓ Extract required libraries into generated JAR]
- Package required libraries into generated JAR
- Copy required libraries into a sub-folder next to the generated JAR

☐ Save as ANT script

ANT script location: D:\Courses\nthu-intro-dbms-2020\db20-assignment-4

Export Runnable Jars - Client



Runnable JAR File Export

Runnable JAR File Specification

Select a 'Java Application' launch configuration to use to create a runnable JAR.

Launch configuration:

Client - bench

Export destination:

C:\Users\SLMT\Desktop\benchmark\client.jar

Library handling:

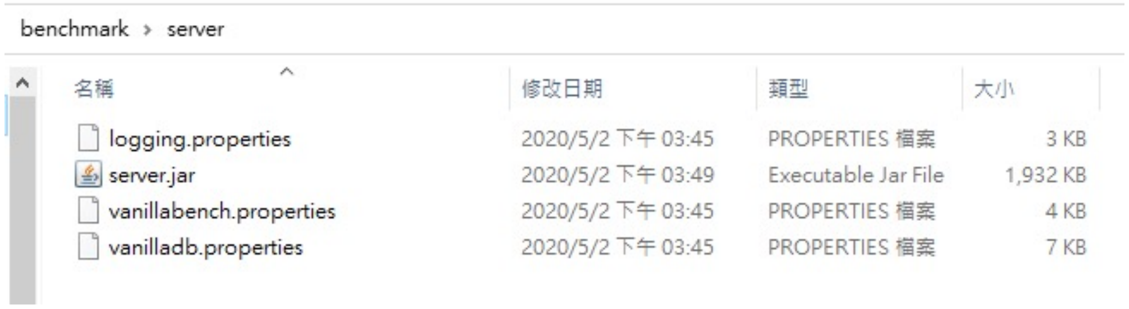
- (●) Extract required libraries into generated JAR
- ( ) Package required libraries into generated JAR
- ( ) Copy required libraries into a sub-folder next to the generated JAR

☐ Save as ANT script

ANT script location: D:\Courses\nthu-intro-dbms-2020\db20-assignment-4

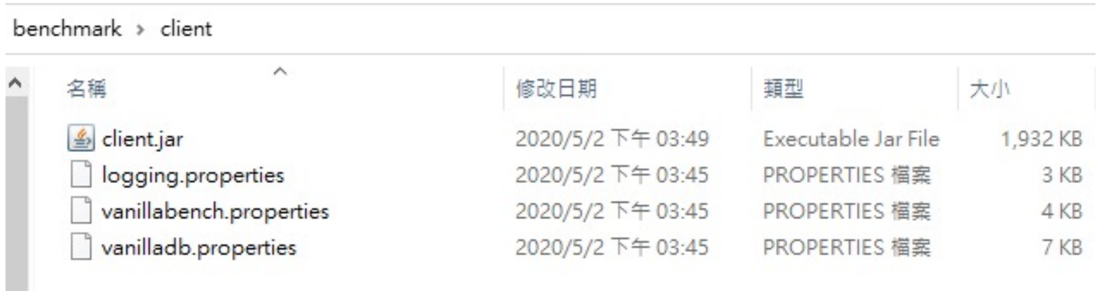
Setup Working Directory

- The next step is to setup you working directory.
- Contents
  - Server
    - server.jar
    - Properties
    - Scripts
  - Client
    - client.jar
    - Properties
    - Scripts



benchmark › server

名稱	修改日期	類型	大小
logging.properties	2020/5/2 下午 03:45	PROPERTIES 檔案	3 KB
server.jar	2020/5/2 下午 03:49	Executable Jar File	1,932 KB
vanillabench.properties	2020/5/2 下午 03:45	PROPERTIES 檔案	4 KB
vanilladb.properties	2020/5/2 下午 03:45	PROPERTIES 檔案	7 KB



benchmark › client

名稱	修改日期	類型	大小
client.jar	2020/5/2 下午 03:49	Executable Jar File	1,932 KB
logging.properties	2020/5/2 下午 03:45	PROPERTIES 檔案	3 KB
vanillabench.properties	2020/5/2 下午 03:45	PROPERTIES 檔案	4 KB
vanilladb.properties	2020/5/2 下午 03:45	PROPERTIES 檔案	7 KB

Scripts

- Now we are going to write scripts for running client and servers
- Scripts
  - Server
    - server.sh
    - copy-db.sh/reset-db.sh
  - Client
    - client-load.sh
    - client-bench.sh

Execution Scripts

- server.sh

```
java -Djava.util.logging.config.file=logging.properties -
Dorg.vanilladb.bench.config.file=vanillabench.properties -
Dorg.vanilladb.core.config.file=vanilladb.properties -jar server.jar [DB Name]
```

- client-load.sh

```
java -Djava.util.logging.config.file=logging.properties -
Dorg.vanilladb.bench.config.file=vanillabench.properties -
Dorg.vanilladb.core.config.file=vanilladb.properties -jar client.jar 1
```

- client-bench.sh

```
java -Djava.util.logging.config.file=logging.properties -
Dorg.vanilladb.bench.config.file=vanillabench.properties -
Dorg.vanilladb.core.config.file=vanilladb.properties -jar client.jar 2
```

Backup Databases

- To ensure the consistency of experiments, we usually backup the database and reset it before each experiment.
- copy-db.sh

```
DB_DIR="[DB Path]"
cp -r $DB_DIR $DB_DIR-backup
```

- reset-db.sh

```
DB_DIR="[DB Path]"
rm -r $DB_DIR
cp -r $DB_DIR-backup $DB_DIR
```

The Workflow of Benchmarking (1/2)

1. Load DB
    1. Setup properties
    2. Run server.sh
    3. Run client-load.sh
    4. Wait for loading
    5. Shut down the server (by stopping the script)
    6. Run copy-db.sh

The Workflow of Benchmarking (2/2)

2. Benchmark
    1. Setup properties
    2. Run reset-db.sh
    3. Run server.sh
    4. Run client-bench.sh
    5. Wait for benchmarking
    6. Shut down the server (by stopping the script)