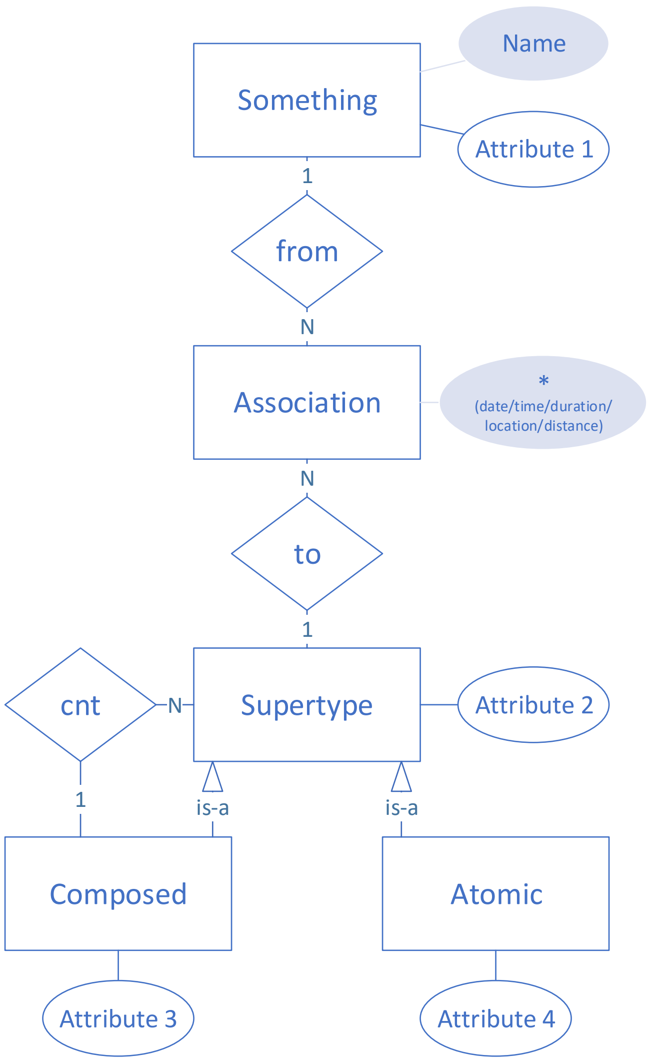
Project X

Task description for an individual project

This document describes your tasks for the project to be presented as part of the oral exam for the data management course (w.3DM).

Generic Conceptual Data Model

The following figure shows a generic conceptual data model using the entity relationship notation (ER). You get your own mapping of each generic entity type, attribute, and relationship to the ones you should use within your individual project. This mapping will be sent to you bey email.



Work Packages

WP 1: Design

Draw your personalized conceptual data model by replacing the generic entity types `Something`, `Association`, `Supertype`, `Composed` and `Atomic` and relationships `from`, `to` and `cnt` with the ones you received in the email.
Find meaningful attribute names for the generic attributes `Attribute 1`, `Attribute 2`, `Attribute 3`, `Attribute 4` as well as the attribute marked with an * (asterisk). This attribute marked with an asterisk should be chosen such that it represents one of the following: date, time, date-time, duration, location, spatial distance.
Make sure you keep the relationship cardinality constraints (one to many) and the type hierarchy (inheritance).

Output: ER Diagram describing your personalized application domain including your own entity types, attributes and relationships.

WP 2: Implement

Implement the back-end component for an application following your personalized application domain.

- **Java Classes as JPA Entities**
  Make sure the Java classes follow the entity types as specified in your personalized application domain, including their names, attributes and relationships.
- **One Repository for each JPA Entity**
  In this step you may leave these repositories empty and do with the default queries.
- **One REST Controller for each JPA Entity**
  Implement two endpoints for each controller: one returning all instances and another one expecting a URL parameter containing an ID and returning the instance referenced by this ID. Both of these endpoints should be made available as GET methods.

Output: Code written by you specifying the classes and interfaces in the three packages of the Spring Boot project `entities`, `repositories` and `controllers`.

WP 3: Test Data (Optional Alternative: see WP 0)

Use Mockaroo (https://www.mockaroo.com/) to generate random (but meaningful) test data. Make sure you have (at least) 100 instances including attribute values for the three JPA Entities representing the entity types `Something`, `Association` and `Supertype`[1], and (at least) 100 "connections" for each relationship. Insert this test data into the database and make sure the data is returned by the REST API endpoints when you access them (e.g., using your browser, or postman or equivalent).

Output: All SQL insert statements (as downloaded from Mockaroo) and the JSON data shown in your browser (or postman or equivalent)

[1] Note that each instance of `Supertype` must also be an instance of either `Composed` or `Atomic`. While you may decide on how to distribute these instances, you need to make sure you end up having instances of both subtypes.

WP 4: Queries for a Graph

Specify two queries that meet the following requirements[2].

- Query 1 (list of nodes) returns a list of instances of **`Something`** along with an ID and the Attribute **`Name`**.
- Query 2 (list of edges) returns a list of pairs of IDs referring to instances of **`Something`**. Each pair corresponds to two instances of **`Something`** having associations to one instance of **`Supertype`** in common. The associations occur along the composed relationship consisting of **`from`**, **`Association`** and **`to`**.

Implement these queries in a repository interface and make them available with two additional endpoints in your REST API. The HTTP-method of these endpoints should be GET.

Make sure your endpoints return the expected results given the test data in your database (e.g., using your browser or postman or equivalent).

Output: The queries as two SQL select statements (NOT JPQL) as well as the code you wrote to implement it (e.g., as JPQL queries in a repository and endpoints implemented in a controller).

WP 5: XSLT (Optional Alternative: see WP 0)

In Workbench, execute a query that returns all instances of `Supertype`. Export the result in an XML format. Write an XSLT that transforms this XML into an HTML document where the instances are listed in a table. Make sure the table has at least one column for the attribute declared by `Supertype`.

Output: XML and XSLT

[2] The requirements to the queries are specified using the generic entity types, attributes and relationship shown in the figure above. You need to do the mapping to your personalized application domain.

WP 0: Optional Alternatives and Extensions for Bonus Points

WP 3: Test Data

Find or generate test data other than using Mockaroo. For example, you may find a data set on Kaggle[3] that suits your application domain. Alternatively, you may find data available using another REST API[4], or by importing data from a file. In order to get bonus points, you need to have code for

a) accessing (e.g. from File, REST, Web, …) and
b) transforming (e.g. from CSV, JSON, XML, HTML, …)

the data.

Output: In addition to the output required for WP 3, show the data source as well as your code accessing and transforming the data

WP 5: XSLT

Export instances of additional entity types associated to `Supertype`. Using XSLT, follow the references (e.g. using the key() function or by generating modal windows) to display all data in the resulting HTML document.

Output: In addition to the output required for WP 5, show the extensions to the XML document, the XSLT document, and the resulting HTML document.

Front-End

Design and implement a front-end for your back-end. The front-end consists of HTML document(s) styled with CSS (e.g. Bootstrap) and including JavaScript code (e.g. svelte) interacting with your REST API. The front-end should support a user journey (sequence of actor-system interactions) outlining the core idea of your application (MVP).

Note: It is forbidden to have the same artefacts evaluated (e.g., graded or pass/fail) multiple times in the context of different courses. The front-end you develop as part of this optional extension must differ substantially from any front-end you handed in for evaluation as part of the w.3PT course.

Output: Front-End components including HTML, CSS, and JavaScript

[3] Data sets at Kaggle: https://www.kaggle.com/datasets

[4] For example, check out https://www.programmableweb.com/category/all/apis and https://www.google.com/publicdata/directory.

Presentation

As part of your exam, you must be prepared to present and discuss the following:

- WP 1: Your personalized application domain (show ERD figure)
- WP 2: Your code (in Eclipse or equivalent)
- WP 3: The SQL Insert statements (in a text editor or in Workbench or equivalent)
- WP 3: Show your test data as JSON data in your browser (or postman or equivalent)
- WP 4: The SQL select statements and its execution in Workbench (or equivalent) including the results
- WP 4: Show the result as JSON data in your browser (or postman or equivalent) when accessing the endpoints
- WP 5: Show the XML and XSLT (in a text editor or equivalent) and the resulting HTML document in a browser.
- WP 0 (Optional Alternatives): Show the output listed for each alternative you chose to do.

Evaluation

Each of the items in the list above (Section "Presentation") yields 2 Points if the expectations according to the work package descriptions are met. As part of the oral exam, we may ask questions to make sure you know and understand what you did.

For each item, points will be given as follows.

- 2 Points if everything is done according to the requirements, the outputs presented successfully, and you can answer related questions.
- 1 Point if requirements are met partially only, the presentation of the outputs does not work as expected, or you are unable to answer some of the related questions.
- 0 Point if item is missing or your answers to questions show a significant lack of understanding.

The number of points you collect determines your grade. We may grant individual bonus points if your results are outstanding. However, you may not collect indefinite amounts of bonus points. Required artefacts that are missing cannot be replaced entirely by means of bonus points.