Independent Work Project

Table of Contents

1 General idea

Most of the information systems are too big in order to understand the detailed needs of the system and implement it with one big step. Look, for instance, the database of the Moodle e-learning system: https://docs.moodle.org/dev/images_dev/5/5a/Moodle2erd.png In case of such systems we should develop the system gradually. For that purpose one has to conduct strategic analysis to find, among other things, subsystems of the system. In the Moodle example different data-centric subsystems, i.e., registers have different background color in the picture. After determining the subsystems it will be possible to start planning as to in which order, when, and by whom to develop the subsystems. In the project you have to conduct a very simplified strategic analysis and find subsystems of an information system. After that you will proceed with the detailed analysis, design, and implementation of only a part of the system (exactly one functional subsystem and data centric subsystems, i.e., registers that are needed by it).

2 Project Timeline

For the *second* practice class read the document and ask questions if something remains unclear.

For the *third* practice class form a group (1–3 persons) for making the project. The task does not depend on the size of the group. Forming of groups of 2–3 students is encouraged because working in a group is a very common way of working (in this field). Team members have to participate in different kind of activities (analysis, design, programming SQL statements), there are no specialized roles within the team, and the team as s a whole is responsible to deliver the committed delivery in time and with the defined quality. This is a common and efective way of working (see, for instance, the principles of development team in the popular agile methodology Scrum: https://www.scrum-institute.org/Scrum_Roles_The_Scrum_Team.php). It is not mandatory to follow the principles of Scrum in your project, but it is also not prohibited. The project task does not prescribe a development methodology.

In the *third* practice class takes place selection of the project topic (by using a lottery with the help of this tool: https://www.random.org/integers/). Thus, at least one member of the project must be present. Just picking a variant you want is not permitted. Possible topics are the following.

- **Variant 1**: Product management functional subsystem of an e-shop (essentially this subsystem is about managing the product list: https://www.amazon.com/s?k=SQL&ref=nb_sb_noss) **(X= Product)**
- **Variant 2**: Service management functional subsystem of a haircut salon (essentially this subsystem is about managing the price list: https://www.tuuletuka.ee/en/price-list/) **(X=Service)**
- **Variant 3**: Golf course management functional subsystem of a golf club (see https://en.wikipedia.org/wiki/Golf_course) **(X=Golf course)**
- **Variant 4**: Training specification management functional subsystem of a fitness club (Training specification is like course/subject specification in a university information system – see, for instance, https://ois.ttu.ee/subject/ITI0206) **(X=Training specification)**

Get a working version of Enterprise Architect (EA) – **ver 12**.
https://moodle.taltech.ee/mod/page/view.php?id=218265
It would be the best if you will get if for the *third* practice class. In principle you could use some other CASE tool. However, you still have to produce all the required artifacts of the project but with A LOT of extra work and trouble. For this time, please also make a preliminary decision about the database management system (DBMS) that you are going to use (you can change it later). The DBMS has to be a SQL DBMS and you must be able to demonstrate the results. MS Access is an easy option but its use is not mandatory: https://moodle.taltech.ee/mod/page/view.php?id=218268
However, if you want, I can give to you access to the server that has PostgreSQL. If you want this you have to contact me.

Question: What to do if in a *computer class computer* Enterprise Architect (EA) asks a key and is unable to open without it?

Answer: Close EA.

Choose: Control Panel => System and Security => Configuration Manager => Configurations => evaluate CB WRK Enterprise Achitect Key

After that open EA again.

In the practice classes of the weeks *3–6*, we make the project together. The lecturer explains different artifacts, their purpose, dependencies, and creation techniques. Most probably, we cannot finish the project in the classes but at least students will get a good start. The course homepage (section *Independent work*) has an example project that is based on the same template (room management of a hotel).

In the practice classes *14* and *15* takes place evaluation of the finished projects. In these classes it is also possible to listen presentation of projects, finish your own project, and ask questions. If you want to present your project for assessment, then you must previously register and upload the files to

Moodle. Projects can be showed earlier and later as well but in this case you have to agree with the lecturer a meeting during a consultation time. The later you get acceptance of the project, the lower is the project coefficient that will be used to calculate the final grade:
https://moodle.taltech.ee/mod/resource/view.php?id=16840

3 General Description of The Work Process

Download the project template EA file and rename it so that the new name contains all the student codes of the students who participate in the project.
https://moodle.taltech.ee/mod/resource/view.php?id=16935

The model template contains:

- a fixed part,
- a mutable part
  - X
  - ...
  - missing model elements or missing comments of model elements

In the file look over **all** the already made models and make changes/adjustments according to the selected topic. **You may not delete or modify the fixed part of the specification that is present in the provided template (except if explicitly required by the work guideline)! However, you have add things.** In this sense the template is similar to an *application framework*: https://en.wikipedia.org/wiki/Application_framework I suggest you to use the following order of work. Perhaps you can print the next pages out and use it as checklist.

The fixed part describes what is *similar* in different domains for which one can create the information system project by using the template. Your task is to describe what would be *different* in case of the different domains.

Firstly, you have to use your imagination as well as domain knowledge to determine what is the exact business of the organization, how it is conducted, and thus what parts (subsystems) the information system has to have in order to support the business. **You are going to design only one functional subsystem and its needed registers but you have to know how to find the subsystems and have to understand that in your detailed project you are actually specifying a small part of the information system as a whole.** The model file contains under the package *Analysis* document *Overview of the entire information system*. To this file you have to add the following.

- The list of actors (roles) of the entire information system. The representatives of these will somehow use the information system.
- The list of the main entity types of the entire information system.
- The list of areas of competence. There is one-to-one correspondence between the actors and the areas of competence.
- The list of functional subsystems. There is one-to-one correspondence between the main entity types and the functional subsystems.

- The list of registers. There is one-to-one correspondence between the main entity types and the registers.
- How to differentiate between different types of subsystems?
  - https://martinfowler.com/bliki/UtilityVsStrategicDichotomy.html
  - Subsystems that correspond to the main activities of the organization – strategic software.
  - Administrative subsystems – utility software.

Replace **X** with the name of the main entity type (product, service, etc.) that corresponds to the selected topic. For instance.

- Replace **X**-s in the use case diagram (*Analysis => Functional subsystems => X management functional subsystem*) and in the comments of the use cases. You also have to do it in the description of goals that is in the comment part of the package *X management functional subsystem*.
- Replace **X**-s in the activity diagram (*Analysis => Functional subsystems => X management functional subsystem => Scenarios of the use case "Permanently inactivate a X" => Activity diagram of permanently inactivating a X*).
- Replace **X**-s in the conceptual data model (*Analysis => Registers => Register of X*). You also have to do it in the names of entity types and attributes as well as in their comments.
- Replace **X**-s in the state machine of the main entity type that is the topic of your project (*Analysis => Registers => Register of X => Lifecycles of X => State machine diagram of X*)

Replace **X**-s also in the names of packages, diagrams, and textual descriptions that are associated wth the model elements.

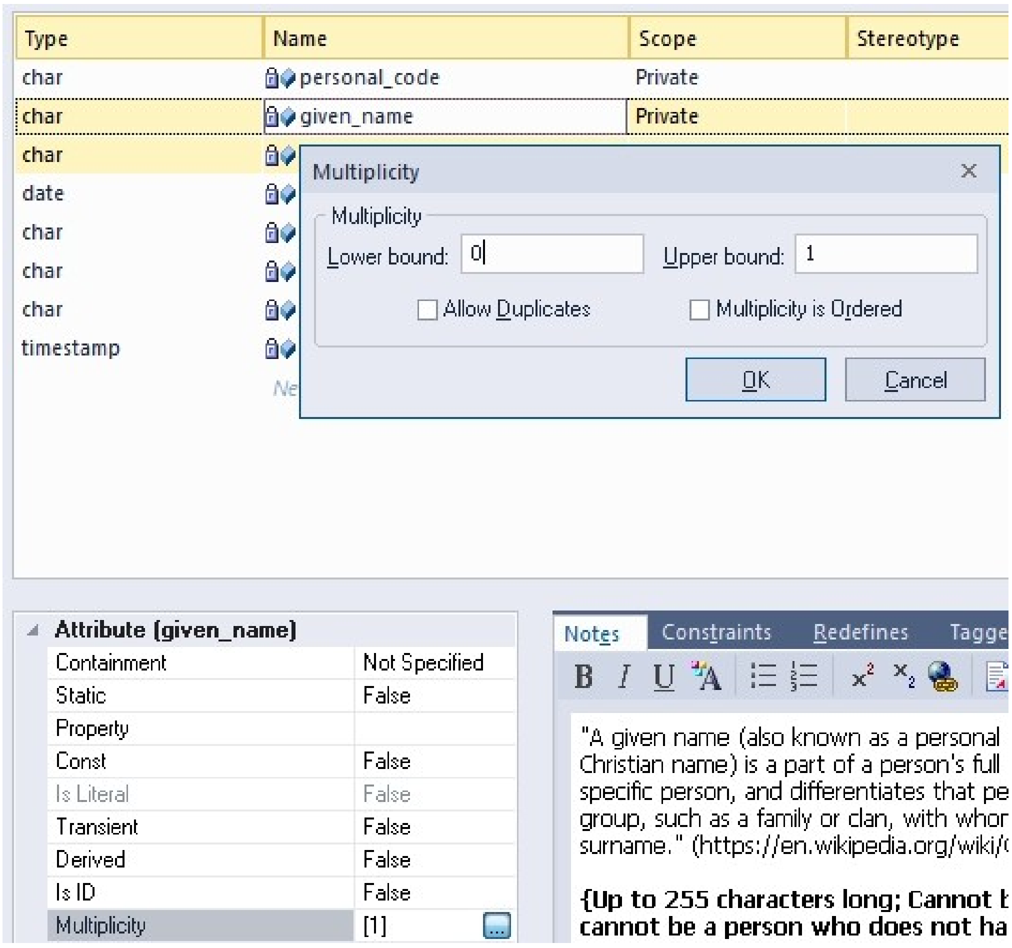
If you think that some additional actors must be connected with the existing use cases, then add the actors and their relationships with the use cases to the use case diagram. Make sure that all the added actors are also in the list of actors and areas of competence. Make also sure that the list of actors contains all the actors that are already present in the use case diagram.

Add events to the state transitions in the state machine model of the main entity type.

- The state machine model depicts all the possible lifecycles of the entities that belong to the main entity type that is the basis of your project. These lifecycles are described from the point of view of the organization that needs the information system.
- Each state transition has a corresponding use case. Make sure that the use case model and the description of events in the state machine model and are consistent (for instance, a mistake is that an event in the state machine model reffers to some other actor than is connected with the corresponding use case).

Extend the conceptual data model (the package *Analysis => Registers*) according to the predefined requirements and business rules (you will find these at the end of the document in the section that is dedicated to your variant). Do not forget to write definitions, including constraints, for every new entity type and attribute you add to the diagram. You must add the definitions as the notes of the corresponding element.

- Change the multiplicity of the following already modeled attributes to 0..1 in order to satisfy the specified constraints to the attributes.
  - Person.given_name
  - Person.surname
  - Person.current_address
- In the entity type *Person* change the order of attributes so that the attribute *e_mail* is the second. The reason is that (e_mail) is one of the unique identifiers of persons and a convention is to have such attributes at the beginning of the attribute list.



Type	Name	Scope	Stereotype
char	personal_code	Private	
char	given_name	Private	
char			
date			
char			
char			
char			
timestamp			

Multiplicity

Multiplicity

Lower bound: 0 Upper bound: 1

Allow Duplicates ☐ Multiplicity is Ordered ☐

Attribute (given_name)

Containment	Not Specified
Static	False
Property	
Const	False
Is Literal	False
Transient	False
Derived	False
Is ID	False
Multiplicity	[1]

"A given name (also known as a personal Christian name) is a part of a person's full specific person, and differentiates that pe group, such as a family or clan, with whor surname." (https://en.wikipedia.org/wiki/(

{Up to 255 characters long; Cannot b cannot be a person who does not ha

Change the view to the conceptual data model by representing entity types/attributes/relationship types at multiple diagrams, each of which is dedicated to a specific register.

- The diagrams should be focused. If C belongs to the register of X and A and B do not belong to it, then the following diagram of X

  [A]-1-----0..*-[B]-1-----0..*-[C],

  should be replaced with the diagram **[B]-1-------0..*-[C]**
- Each attribute should be represented on exactly one diagram. (http://wiki.c2.com/?OnceAndOnlyOnce)
- Each entity type should be represented on one or more diagrams. The representation of the same entity type at different diagrams helps the reader to mentally join different diagrams. However, for each entity

type there should be only one diagram where its attributes are visible. (http://wiki.c2.com/?DontRepeatYourself)

- Look the Moodle example at the beginning of the task – we cannot have much details on a too big diagram and the big number of relationships makes it very hard to read.

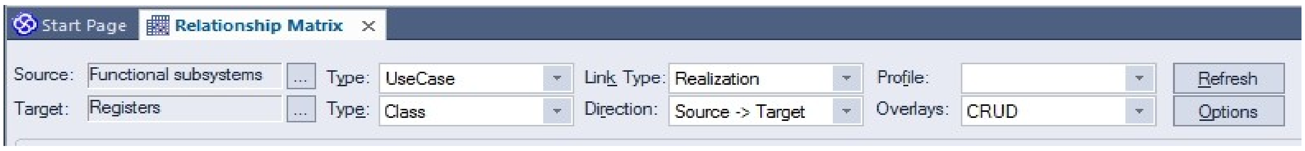
Modify contracts of database operations (*Analysis* => *Registers* => *Contracts of data modification operations*) according to the extended conceptual data model.

- Do not forget to define correctly pre- and postconditions of the OP4.
- Please note that the operation OP6 does not change the state of X because there are other operations for that. The operation also does not change registration time and information about the person who registered X because the system should not allow us to falsify data.

Add references to the database operations to the transitions in the state machine model where these are missing.

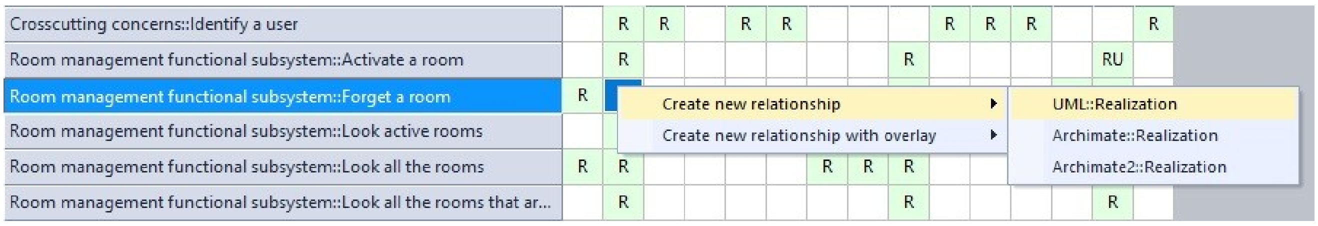
Modify the CRUD matrix (*Tools* => *Relationship Matrix*) to make sure that there are all the entity types and all the operations have been correctly defined.

- Do not forget to add correct depedncies for the use case "Forget X".
- To see the correct CRUD matrix, you have to make the following selections.

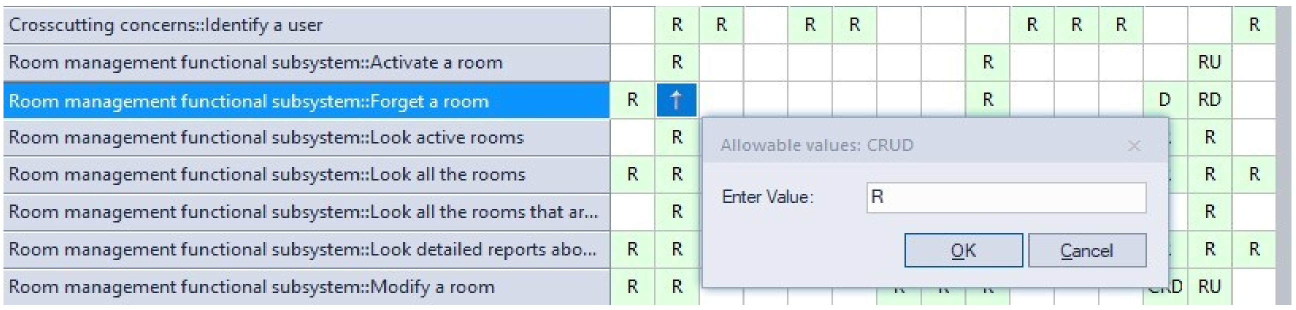


Source: Functional subsystems
Target: Registers
Type: UseCase
Type: Class
Link Type: Realization
Direction: Source -> Target
Profile:
Overlays: CRUD

How to add letters to the matrix?

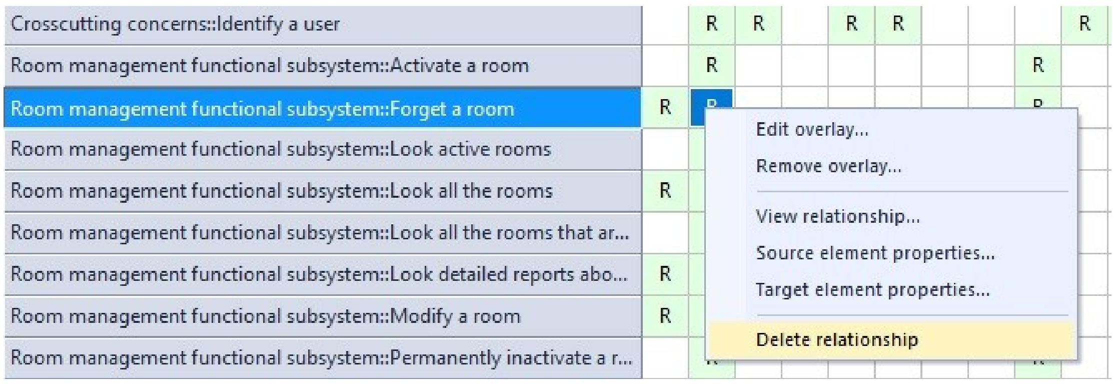


Crosscutting concerns::Identify a user		R	R		R	R				R	R	R			R
Room management functional subsystem::Activate a room		R							R					RU	
Room management functional subsystem::Forget a room	R		[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Room management functional subsystem::Look active rooms		[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Room management functional subsystem::Look all the rooms	R	R					R	R	R		[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
Room management functional subsystem::Look all the rooms that ar...		R							R					R	



Crosscutting concerns::Identify a user		R	R		R	R				R	R	R			R
Room management functional subsystem::Activate a room		R							R					RU	
Room management functional subsystem::Forget a room	R	↑							R				D	RD	
Room management functional subsystem::Look active rooms		R	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	R	
Room management functional subsystem::Look all the rooms	R	R	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	R	R
Room management functional subsystem::Look all the rooms that ar...		R	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	R	
Room management functional subsystem::Look detailed reports abo...	R	R	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	R	R
Room management functional subsystem::Modify a room	R	R	[illegible]	[illegible]	[illegible]	[illegible]	R	R	R	[illegible]	[illegible]	[illegible]	CRD	RU	

How to delete letters from the matrix if one made an incorrect modification?


Crosscutting concerns::Identify a user
Room management functional subsystem::Activate a room
Room management functional subsystem::Forget a room
Room management functional subsystem::Look active rooms
Room management functional subsystem::Look all the rooms
Room management functional subsystem::Look all the rooms that ar...
Room management functional subsystem::Look detailed reports abo...
Room management functional subsystem::Modify a room
Room management functional subsystem::Permanently inactivate a r...
Edit overlay...
Remove overlay...
View relationship...
Source element properties...
Target element properties...
Delete relationship

Generate database physical design model and modify it according to the business rules from the domain as well as best practices of database design.

Generate SQL statements from the model.

Execute the generated SQL statements to create base tables in the database.

Make sure that your tables have all the necessary CHECK constraints (according to the business rules and requirements derived from the rules). If you use MS Access, then you have to add these in the table design view as validation rules (**see practice class exercise 1**).
https://moodle.taltech.ee/mod/resource/view.php?id=16878

If you use MySQL, PostgreSQL, etc., then you must define the CHECK constraints in the design model and generate SQL code that already contains these CHECK constraints.

Add at least one row of test data to each created base table. In case of state classifier *X_state_type* you have to register all its values based on the state machine diagram of X. Please note that if an X is forgotten, then its data is deleted from the database and thus there is no row about this state in the table *X_state_type*.

Implement the workplace of *General manager* by implementing queries and data modification statements that are needed in case of its use cases:

- Permanently inactivate X.
- Look detailed reports of X.

All the queries have to be implemented as views (named SELECT statements). In case of use case “Look detailed reports of X”, you have to look the document section that is dedicated to your variant to find out what queries you have to implement.

SQL statements for all the groups

In case of use case “Permanently inactivate X”, you have to implement the following statements.

- Query that finds the list of X instances that are in the state "Active" or "Temporarily inactivated". The result must contain X_code, X_name, and the name of the current state of the X instance.
- Parameterized UPDATE statement (if you use MS Access) or a function/procedure (if you use a SQL DBMS that supports these) that implements OP5.

In the end you have to present to the lecturer.
https://moodle.taltech.ee/mod/assign/view.php?id=16923

- Updated model file.
- Database with all the base tables, test data, and required views/data modification statements.
- Text file that contains SQL statements for creating tables according to the project.
- If you use MS Access, then the queries must be saved as query objects in the database file. Otherwise you must submit a text file with the queries. It must be possible to execute these to check their correctness.

See also: https://moodle.taltech.ee/mod/page/view.php?id=218236

All the project participants must be present while presenting. All the participants must know all the parts of the project and have ability to explain. If there are mistakes, then these must be corrected before the project can be accepted.

Next, I will explain each possible domain of the project. I will describe additional business rules and requirements to the ones already depicted in the project template. I will list queries that you will have to implement in the context of the use case “Look detailed reports of X”. I do not describe business rules and requirements that are already presented in the template (but you still have to take them into account while designing the database).

NB! In case of all the variants one must follow the *data protection by default principle*: https://ec.europa.eu/info/law/law-topic/data-protection/reform/rules-business-and-organisations/obligations/what-does-data-protection-design-and-default-mean_en according to the European General Data Protection Regulation: https://en.wikipedia.org/wiki/General_Data_Protection_Regulation Thus, in case of clients, one must assume that they *do not* agree with using their contact information for sending advertisements and offerings. Hence, the default value of the column must be FALSE.

Product (variant 1)

Statements with additional requirements and business rules.

- Each product has exactly one brand.
- Each brand characterizes zero or more products.
- Each brand is a classifier.
- Each product has exactly one list price (in Euros).
- Each product has exactly one minimum net price (in Euros).
- Each product has zero or one picture address.
- Each product has zero or one descriptions for the public.
- Each list price must be equal or bigger than minimum net price.
- Each minimum net price must be zero or more.
- Each picture address must contain a dot (.).
- Each client is a person.
- There could be persons who are not clients and workers.
- The same person could be a client and a worker at the same time.
- Each client has to provide (TRUE/FALSE) information as to whether he/she agrees that his/her contact information is used to send him/her advertisements and offerings.
- Each client has exactly one client state type as its current state.
- Each client state type characterizes zero or more clients.
- Each client state type is a classifier.

Queries that you have to implement in case of the use case "Look detailed reports about producs".

SQL statements for the specific group

- Find the list of all the products. For each product present the product code, name, list price, difference between the list price and minimum net price (in the column *price_difference*), year of its registration (in the column *registration_year*), name of its current state, and name of its brand. In addition, present for each product the concatenated given name, surname, and e-mail address of the person who registered the product (in the column *person_who_registered*). The names of the products must be in uppercase in the result.
- Find for each product state type the number of products in this state. Present the state name (in the column *product_state_name* and the number of products in the state (in the column *product_count*). If for some state there are no products, then the number must be zero. Sort the result in descending order according to the number of products. If there are multiple state types with the same number, then sort these based on the name of the product state in the alphabetical order.
- Find the number of products that have been registered in the current year. The current year must be found by using a function. The query must return a table with one row and column (in the column *cnt*).
- Find the products that belong to the biggest number of categories together with the number of categories they belong to. Present product number, product name, and the number of categories (in the column *category_cnt*).
- Find the product categories that have no associated products. For each such category present code, name, and type name (in the column *category_type*).

Service (variant 2)

Statements with additional requirements and business rules.

- Each service has exactly one difficulty level.
- Each difficulty level charaterizes zero or more services.
- Each difficulty level is a classifier.
- Each service has exactly one list price (in Euros).
- Each service has exactly one minimum net price (in Euros).
- Each service has exactly one expected time of completion (in minutes).
- Each service has zero or one descriptions for the public.
- Each list price must be equal or bigger than minimum net price.
- Each minimum net price must be bigger than zero.
- Each expected time of completion must be between 10 and 1000 (end points included).
- Each client is a person.
- There could be persons who are not clients and workers.
- The same person could be a client and a worker at the same time.
- Each client has to provide (TRUE/FALSE) information as to whether he/she agrees that his/her contact information is used to send him/her advertisements and offerings.
- Each client has exactly one client state type as its current state.
- Each client state type characterizes zero or more clients.
- Each client state type is a classifier.

Queries that you have to implement in case of the use case "Look detailed reports about services".

SQL statements for the specific group

- Find the list of services. For each service present the service code, name, list price, expected time of completion in *hours* (in the column *completion_in_hours*), name of its current state, and name of its difficulty level. In addition, present for each service the concatenated given name, surname, and e-mail address of the person who registered the service (in the column *person_who_registered*). The names of the services must be in uppercase in the result.
- Find for each service state type the number of services in this state. Present the state name (in the column *service_state_name* and the number of services in the state (in the column *service_count*). If for some state there are no services, then the number must be zero. Sort the result in descending order according to the number of services. If there are multiple state types with the same number, then sort these based on the name of the service state in the alphabetical order.
- Find the number of services that have been registered in the current year. The current year must be found by using a function. The query must return a table with one row and column (in the column *cnt*).
- Find the services that belong to the biggest number of categories together with the number of categories they belong to. Present service number, service name, and the number of categories (in the column *category_cnt*).
- Find the service categories that have no associated services. For each such category present code, name, and type name (in the column *category_type*).

Golf course (variant 3)

Statements with additional requirements and business rules.

- Each golf course has exactly one difficulty level.
- Each difficulty level charaterizes zero or more golf courses.
- Each difficulty level is a classifier.
- Each golf course has exactly one length (in meters).
- Each golf course has exactly one number of holes.
- Each golf course has exactly one number of water hazards.
- Each golf course has zero or one descriptions for the public.
- Each length must be bigger than zero.
- Each number of holes must be between 9 and 36 (end points included).
- Each number of water hazards must be zero or more.
- Each number of holes must be bigger than the number of water hazards.
- Each client is a person.
- There could be persons who are not clients and workers.
- The same person could be a client and a worker at the same time.
- Each client has to provide (TRUE/FALSE) information as to whether he/she agrees that his/her contact information is used to send him/her advertisements and offerings.
- Each client has exactly one client state type as its current state.
- Each client state type characterizes zero or more clients.
- Each client state type is a classifier.

SQL statements for the specific group

Queries that you have to implement in case of the use case "Look detailed reports about golf courses".

- Find the list of golf courses. For each golf course present the golf course code, name, number of holes, number of water hazards, length in kilometers with at most two numbers after the comma (in the column *length_km*), name of its current state, and name of its difficulty level. In addition, present for each golf course the concatenated given name, surname, and e-mail address of the person who registered the golf course (in the column *person_who_registered*). The names of the golf courses must be in uppercase in the result.
- Find for each golf course state type the number of golf courses in this state. Present the state name (in the column *golf course_state_name* and the number of golf courses in the state (in the column *golf_course_count*). If for some state there are no golf courses, then the number must be zero. Sort the result in descending order according to the number of golf courses. If there are multiple state types with the same number, then sort these based on the name of the golf course state in the alphabetical order.
- Find the number of golf courses that have been registered in the current year. The current year must be found by using a function. The query must return a table with one row and column (in the column *cnt*).
- Find the golf courses that belong to the biggest number of categories together with the number of categories they belong to. Present golf course number, golf course name, and the number of categories (in the column *category_cnt*).

- Find the golf course categories that have no associated golf courses. For each such category present code, name, and type name (in the column *category_type*).

Training specification (variant 4)

Statements with additional requirements and business rules.

- Each training specification has exactly one type of sport.
- Each type of sport charaterizes zero or more training specifications.
- Each type of sport is a classifier.
- Each training specification has exactly one name.
- Each training specification has exactly one description for the public.
- Each training specification has exactly one number that shows how many calories will be burned (in average) with one training session.
- Each training specification has exactly one number that shows what is the expected length of a training session in minutes.
- Each number of burned calories must be bigger than zero.
- Each length of a training session must be between 5 and 480 minutes (end points included).
- Each length of the description (the number of characters) of a training specification must be at least twice as big as the length of the name of the training specification.
- Each client is a person.
- There could be persons who are not clients and workers.
- The same person could be a client and a worker at the same time.
- Each client has to provide (TRUE/FALSE) information as to whether he/she agrees that his/her contact information is used to send him/her advertisements and offerings.
- Each client has exactly one client state type as its current state.
- Each client state type characterizes zero or more clients.
- Each client state type is a classifier.

SQL statements for the specific group

Queries that you have to implement in case of the use case "Look detailed reports about training specifications".

- Find the list of training specifications. For each training specification present the training specification code, name, description, number of burned calories, length of a session, number that shows how many calories are in average burned during one session minute with at most two numbers after the comma (in the column *calories_per_minute*), name of its current state, and name of its sport type. In addition, present for each training specification the concatenated given name, surname, and e-mail address of the person who registered the training specification (in the column *person_who_registered*). The names of the training specifications must be in uppercase in the result.
- Find for each training specification state type the number of training specifications in this state. Present the state name (in the column *training_spec_state_name* and the number of training specifications in the state (in the column *training_spec_count*). If for some state there are no training specifications, then the number must be zero. Sort the result in descending order according to the number of training specifications. If there are multiple state types with the same number, then sort these based on the name of the training specification state in the alphabetical order.
- Find the number of training specifications that have been registered in the previous year. The previous year must be found by using a

function. The query must return a table with one row and column (in the column *cnt*).

- Find the training specifications that belong to the biggest number of categories together with the number of categories they belong to. Present training specification number, training specification name, and the number of categories (in the column *category_cnt*).
- Find the training specification categories that have no associated training specifications. For each such category present code, name, and type name (in the column *category_type*).