

Introduction to Databases – Guidelines for the Exam Project

— Draft – Check Online for Latest Version —

The final mark of the *Introduction to Databases* exam is computed as a weighted average of the *written exam mark* (70%) and the *project mark* (30%). Moreover, to be admitted to the written exam, the student must have completed the project discussion, and the project must have received a positive evaluation. In other words, **the written exam cannot be taken without first passing the project.**

1. Interaction with the Lecturer/TA Regarding the Project

All interactions with the lecturer/TA concerning the project must take place **directly**, either:

- in person, after lectures/labs or during office hours; or, when not possible in person
- online, through a video call arranged with the lecturer/TA.

In addition, each student or group must create a group chat in MS Teams involving the lecturer, the TA, and the student(s) developing the project, and rename it to:

“IDB Project – $\langle \text{Surname St1} \rangle \langle \text{First Name St1} \rangle$ [, $\langle \text{Surname St2} \rangle \langle \text{First Name St2} \rangle$]”

The MS Teams chat must be used to **submit files for evaluation, ask questions, and arrange meetings with the lecturer/TA**. However, questions concerning the project will **not be answered in writing**. They will instead be discussed directly with the student(s), either in person or during a video call. Students may therefore use the chat to indicate the questions or issues they wish to discuss, but should not expect written explanations or solutions.

Due to the large number of emails received by the lecturer and TA, messages concerning the project sent by email may not be read or answered. Email should therefore not be used for project-related communication or, more generally, for matters concerning the course. MS Teams is the designated channel for written communication, file submission, and arranging direct interactions with the lecturer/TA.

2. Rules for the Project

- a) The project may be developed individually or by a group of two students. In the latter case, the students may decide how to divide the work; however, both must be fully familiar with every part of the project.
- b) The project involves designing a simple database based on a **specification of the requirements** developed by the student(s) and **discussed and agreed upon with the lecturer/TA**. As a starting point for the discussion, the student(s) should ideally provide an initial written description of the application domain (the specification of the requirements) and a draft of the ER schema that is as complete as possible with respect to the specification. Here, “draft” means a schema containing the entities, relationships, and only the main attributes, typically the identifiers. The diagram can be drawn by hand and need not be elaborate, but it should be readable. The diagram is typically revised during the initial discussion and design phases, so it is not worth spending too much time initially on its layout or graphical appearance. Alternatively, student(s) may present only the draft ER diagram and describe the application domain during the meeting. However, writing the specification of the application domain is a useful exercise for better understanding what information the database should store. In any case, this specification is required as part of the project documentation.
- c) The **ER diagram** must be specified **in the notation that is adopted in the course**. This is not merely for aesthetic reasons: alternative notations sometimes used (e.g., in textbooks or by specific tools) often cannot represent important information about the schema. Small variations in the notation are acceptable only if they allow one to represent (in a straightforward way, without requiring additional explanations) all of the following elements: possibly multiple identifiers consisting of multiple attributes and/or roles; external identifiers of entities; identifiers of relationships; primary identifiers; minimum and maximum cardinalities on roles and on attributes; attributes both of entities and of relationships; optional and multivalued attributes; composite attributes; complete and non-complete generalization hierarchies. If the student(s) intend to

use an alternative notation in their project, they must carefully verify how it represents each of the above elements. If the notation does not support these elements, it must be discarded.

In addition, to ensure readability, the diagram must be drawn on a plain white background, avoiding inverted colors, graph paper, lined paper, and other elements that reduce contrast. If it is drawn on a computer, the font size must ensure that the text remains readable when the entire diagram is shown on a laptop screen; if it is drawn by hand, the handwriting must be legible.

- d) Students are also **required to confirm the final version of the ER schema with the lecturer/TA** (diagram and constraints) before they start working on the subsequent phases of the database design. They are also encouraged to use office hours to clarify any doubts they might have about the various design phases. This helps prevent serious design problems from being discovered only when the project is discussed as part of the exam. Such problems could lead to a negative project evaluation, which would prevent the student from taking the written exam.
- e) Roughly speaking, the specification of the requirements and the corresponding ER schema should satisfy the following criteria:
1. It should be based on a domain containing **between 5 and 10 main conceptual entities** (i.e., excluding sub-entities introduced through ISAs or generalizations). When the project is developed by a **group of two**, these **bounds are doubled**.
 2. There should be some structure in the set of entities, i.e., the ER schema should contain **at least one ISA** and **at least one generalization**.
 3. There should be sufficient structure in the relationships, which usually means that the representation of **the ER schema** as a graph (with entities and relationships as nodes and the participation of entities in relationships as edges) **should contain some cycles**.
 4. The schema should contain all of the following elements:
 - at least **one $n:n$ relationship**, i.e., a relationship in which the maximum cardinality on two different roles is n ;
 - cardinality constraints on the participation of entities in relationships that differ from the default $(0, n)$. In particular, **all of the cardinalities** $(0, n)$, $(1, n)$, $(0, 1)$, and $(1, 1)$ **on a role** should be present in the schema;
 - **more than one entity with a primary identifier consisting of multiple attributes**, and, in addition, at least one entity with a **primary external identifier**, i.e., an identifier that involves both attributes and roles;
 - at least **one optional attribute**, at least **one multi-valued attribute**, and at least **one composite attribute**;
 - at least **one relationship with an attribute**;
 - all entities should have at least one “regular” attribute, i.e., an attribute that is neither multi-valued, nor optional, nor part of the primary identifier.
 5. There should be **some external constraints**, i.e., constraints that cannot be represented in the ER model. Remember that a proper external constraint should impose some *condition on the information represented in the database*, i.e., it should be possible to construct a database instance that violates the constraint and is therefore ruled out.
 6. The specification should include an indication of the **volumes for the various entities and relationships** (pay attention to the consistency between the volumes and the cardinality constraints of the ER schema).
 7. The specification should include a workload of the most common **operations** (between 6 and 10) relevant to the modeled domain, including queries and updates, with an indication of their frequency. **At least 3** of the operations should be **queries**, and **at least 3** should be **updates** on the database (insertions, deletions, or updates). When the project is developed by a **group of two**, these **bounds are doubled**.
- f) The student should design the database according to the requirements and the methodology presented in the course, which consists of the following phases:

1. *Conceptual design*, producing the following documentation:
 - a) structured and organized requirements;
 - b) glossary of terms (typically optional, but required for this project);
 - c) diagram of the conceptual schema;
 - d) data dictionary of the conceptual schema (listing the entities, relationships, and external constraints – the data dictionary for the attributes can be omitted);
 - e) table of volumes;
 - f) table of operations, according to the specified application load.
 2. *Restructuring of the conceptual schema*, producing the following documentation:
 - a) restructured conceptual schema (diagram and data dictionary);
 - b) updated table of volumes (only for the entities and relationships that have changed);
 - c) access tables to assess the cost of the operations in the workload. To ease the correction, before providing each access table, you should repeat both the number and the specification of the operation.
 3. *Direct translation* to the relational model, producing the relational schema with external constraints.
 4. *Restructuring of the relational schema* taking into account the application load. **The reasons for the restructuring steps applied to the relational schema, the tables and constraints affected by each change, and the resulting tables and constraints should be stated explicitly**, with reference to the operations. In addition, the specification of the **complete restructured relational schema with external constraints** should be provided, since such a complete schema is the one that must be implemented in SQL.
 5. *Implementation of the database in SQL*, by defining relations with suitable constraints (capturing keys, foreign keys, and tuple constraints) and stored procedures with triggers to implement additional constraints (e.g., inclusions, disjointness, external constraints).
- g) The student should also develop a simple application (typically in Java) that supports interaction with the database through a textual or graphical user interface. The application should provide the following functionality:
- allow the user to execute the queries that are part of the workload (typically by instantiating some parameters with values provided as input);
 - allow the user to perform the update operations (insertions, deletions, updates) that are part of the workload, asking the user for the necessary input, and guaranteeing that the consistency of the data in the database is preserved;
 - handle JDBC exceptions caused by errors in the interaction with the database, so that the application does not terminate prematurely.

3. Rules for Submitting the Project

- a) The project submission should include:
 1. A single PDF document consisting of a textual description of the requirements (of roughly one page) and the documentation for Phases 1–4 above.
The document should have a title page containing at least the name and student number of each student, the title of the project, the reference to the *Introduction to Databases* course, and the date of completion. The pages of the PDF document should be numbered, so that they can be referenced in evaluation comments.
 2. One or more SQL files containing the specification of the database (Phase 5 above); if your project includes more than one SQL file, place all SQL files in a subfolder called “sql”.
 3. One or more files (typically in Java) containing the application that interacts with the database; if your project includes more than one Java file, place all Java files in a subfolder called “java”.
- b) The student should **not** include in the project folder any files other than those listed above. In particular, they should not include any library files, JAR files, POM files, IDE configuration files, or other files or folders

generated by the IDE. The purpose of the submission is not to provide an executable version of the project, but to provide the source files for a manual review of the implementation. The application's functionality will be reviewed during the project discussion.

- c) All documents should be placed inside a folder with the following name:

“ $\langle SurnameSt1 \rangle - \langle FirstNameSt1 \rangle - \langle IDSt1 \rangle [+ \langle SurnameSt2 \rangle - \langle FirstNameSt2 \rangle - \langle IDSt2 \rangle] - \langle ProjectName \rangle$ ”

The folder should be compressed into a single ZIP file with the same name as the folder and the extension .zip.

- d) The ZIP file must be uploaded to the “Introduction to Databases Project” assignment on MS Teams.
e) Unless announced otherwise, the deadline for submitting the project is 23:59 two days before the scheduled project discussion (see below).

4. Rules for the Project Discussion before the Exam

- a) The IDB project discussions typically take place one or two days before the written exam. Their schedule will be announced on MS Teams before each exam session.
b) One or two days before the project discussion, we will post on MS Teams a schedule assigning a specific time slot to each project. Projects will be discussed in the order of their most recent submission time on MS Teams, with earlier submissions discussed first. Each project discussion will last between 15 and 30 minutes.

- c) The project discussion serves three purposes:

1. To review the various phases of database design carried out in the project and verify the correctness of the corresponding documentation.
2. To verify the functionality of the application developed for the project and demonstrate how its update operations affect the data stored in the database.
3. To assess each student's familiarity with **all phases** of database design and implementation. Students must therefore be prepared to answer questions about every part of the project and all submitted material. When a project is developed by a group of two, both students must be fully familiar with the entire project, regardless of how the workload was divided between them.

- d) For an in-person discussion (the default unless specific circumstances require otherwise), students should bring a laptop containing the project documentation, a running version of the application developed for the project, and a running version of pgAdmin connected to their database.

For an online discussion via MS Teams (used only in exceptional circumstances), students should be ready to share their screen and display the project documentation, a running version of the application developed for the project, and a running version of pgAdmin connected to their database.

- e) Students should be ready to display on their laptop (or share on screen via MS Teams) the different parts of the project documentation:

- the requirements,
- the conceptual schema (diagram and constraints),
- the restructured conceptual schema (diagram and constraints),
- the relational schema (with constraints) resulting from the direct translation,
- the restructured relational schema (with constraints).

These can simply be the respective pages of the project document, if the information is clearly readable on screen, or separate PDF or image files (e.g., if the resolution in the project document is too low).

- f) To discuss the various phases of the database design and illustrate how the output of one phase is used to generate that of the next, it is useful to display both outputs simultaneously. Therefore, for the discussion, there should be two separate copies of the PDF document, so that two different parts can be displayed simultaneously (e.g., the restructured ER schema and the relational schema resulting from the direct translation).

- g) Note that when the project discussion takes place in the lecturer's office, the submitted project documentation will be displayed on the lecturer's screen.

By contrast, the functionality of the developed application and its effects on the database must always be demonstrated on the student's laptop (or shared on screen via MS Teams).

5. Use of Generative AI

Students may use generative AI tools (e.g., ChatGPT, Claude, Copilot) in the development of the project. However, their use is subject to the following requirements:

Declaration of use. Any use of generative AI must be explicitly declared in the project documentation. The declaration should indicate which tools were used and briefly describe how they were used (e.g., for brainstorming, improving text, generating or debugging code, or supporting database design).

Student responsibility. The use of generative AI does not transfer responsibility for the submitted work to the AI tool. Students remain fully responsible for the correctness, quality, and originality of all submitted material, including content generated or suggested by AI.

Full understanding. Students must fully understand all parts of the submitted project, regardless of whether generative AI was used to produce or refine them. During the project discussion, students may be asked to explain, justify, modify, or extend any part of their work.

Group projects. For projects developed by two students, both students must be familiar with the entire project, including parts primarily developed by the other student or with the assistance of generative AI.

No substitution for learning. Generative AI may be used as a supporting tool, but not as a substitute for acquiring the knowledge and skills that the project is intended to assess. Inability to explain or justify submitted material will therefore affect the evaluation, irrespective of whether that material was produced with AI assistance.

Lecturer/TA guidance. Explanations and guidance provided orally by the lecturer or TA are intended to support the students' own understanding and should not be recorded, transcribed, or used as input to generative AI tools.