|  |  |  |  |
| --- | --- | --- | --- |
| **Lecturer:** |  | | |
| **Group:** |  | **Lab User** |  |
| **Student:** |  | **NIA:** |  |
| **Student:** |  | **NIA:** |  |
| **Student:** |  | **NIA:** |  |

# Introduction

It will consist of an introductory paragraph, dealing with the problem to be solved, analyzing it and fixing the goals of the work. It will also include the SQL code generated for this *labwork* (*queries*, *views,* *triggers*, …). Finally, it should describe the document structure.

# Queries

This section is devoted to describing the mathematical solution to the queries proposed in the problem description (in Relational Algebra). It should also include the SQL translation for each query, and the description and results of the tests designed to check the validity of each solution. Thus, this chapter is divided in several sections (one per query) and each section has three items:

* Relational Algebra
* SQL
* Tests

# Package

Include an introduction with the structure of the package, and a subsection for each procedure or function that it includes. For each procedure, you must describe:

1. its design (inputs, outputs, logic of the main block), and in case of having needed to make use of auxiliary elements (queries, views, other procedures/functions...) their design and implementation must also be included (unless they are trivial queries).
2. its implementation in SQL
3. tests

# External Design

Describe the views and carry out their design, implementation, and tests (in a similar way to how the queries were made in section 2 but developing their operativity completeness where required). Include a subsection for each view you develop, outlining:

1. its design in relational algebra
2. its implementation in SQL
3. Tests: notice that it must be checked that the view is properly defined (like a query), as well as the operativity of the read and write views: it is necessary to establish which operations (insertion/deletion/modification) the manager resolves itself, and which other operations it does not.

Operations on views not automatically supported by the manager must be resolved using triggers (of type *instead of*), which must also be described, implemented and tested in this section.

# Explicitly required Triggers

For each resolved trigger, include a subsection containing:

1. Description of the design: Table to which it is associated, Event or events in which it is triggered, Temporality (before, after or instead of), Granularity (by row or statement), Condition (if it has one) and Action (description in natural language).
2. Code (PL/SQL)
3. Tests

# Concluding Remarks

Firstly, you have to defend the achieved result, emphasizing the goodness of the semantic coverage, usage (comment unfeasible queries, in case), documentation, etc.

After stating your results, comment your achievement through this labwork: required effort (how much time you spent), knowledge gain, progress, etc. You can also propose improvements for further editions (size of the problem, requested items, deadlines, supporting materials, etc.).