| **Lecturer:** | **Javier Calle** | | |
| --- | --- | --- | --- |
| **Group:** | **286** | **Lab User** |  |
| **Student:** | **Alberto** | **NIA:** | **100495775** |
| **Student:** | **Silvia Bayo Martínez** | **NIA:** | **100495811** |
| **Student:** | [**CARMEN SERRANO PEREZ**](mailto:100495711@alumnos.uc3m.es) | **NIA:** | **100495711** |

# INDEX

[**INDEX**](#_heading=) **1**

[**1 Introduction**](#_heading=) **1**

[**2 Queries**](#_heading=) **2**

[QUERY 1: BoreBooks](#_heading=h.e8u5mxcbagz0) 2

[QUERY 2: Reports on Employees](#_heading=h.wrqn5cdx5t14) 4

[**3 Package**](#_heading=) **4**

[**4 External Design**](#_heading=) **4**

[**5 Explicitly required Triggers**](#_heading=) **5**

[**6 Concluding Remarks**](#_heading=) **5**

# Introduction

This report documents the design, implementation, and testing of different SQL tools applied to the FOUNDICU® relational database as part of the second assignment. These tools include SQL queries, procedures, views, and triggers.

The objective of this task is to consolidate and expand on the relational database created in the first assignment, by introducing practical SQL functionalities that enhance querying and operability.

The document is divided into four main sections:

* Queries: Includes relational algebra, SQL implementation, and test design.
* Packages: Procedure implementations with logic and tests.
* External Design: Implementation of user profile views and their operativity.
* Triggers: Implementation and validation of required triggers.

Each of these sections follows a consistent structure: Design → SQL Implementation → Tests.

# 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

## QUERY 1: BoreBooks

Books with at least three different language editions and no recorded loans.

Relational Algebra

| LangCount ← γ title, author; COUNT(DISTINCT language) → lang\_count (EDITIONS)  ValidBooks ← σ lang\_count ≥ 3 (LangCount)  Loaned ← π title, author (  σ l.type = 'L' (  ((EDITIONS ⨝ EDITIONS.isbn = COPIES.isbn COPIES)  ⨝ COPIES.signature = LOANS.signature LOANS)  )  )  ValidNotLoaned ← π b.title, b.author (  (BOOKS ⨝ BOOKS.title = ValidBooks.title ∧ BOOKS.author = ValidBooks.author)  ) − Loaned |
| --- |

SQL

****WITH Books\_Lang\_Count AS (

SELECT title, author, COUNT(DISTINCT language) AS lang\_count

FROM editions

GROUP BY title, author

),

Valid\_Books AS (

SELECT title, author

FROM Books\_Lang\_Count

WHERE lang\_count >= 3

),

Loaned\_Books AS (

SELECT DISTINCT e.title, e.author

FROM editions e

JOIN copies c ON e.isbn = c.isbn

JOIN loans l ON c.signature = l.signature

WHERE l.type = 'L'

)

SELECT DISTINCT b.title, b.author

FROM books b

JOIN Valid\_Books vb ON b.title = vb.title AND b.author = vb.author

WHERE NOT EXISTS (

SELECT 1

FROM Loaned\_Books lb

WHERE lb.title = b.title AND lb.author = b.author

)

ORDER BY b.title;



Tests

**Test 1 – Book with 3 editions (different languages), no loans**

****-- Insert book

INSERT INTO books (title, author, country, language, pub\_date, topic)

VALUES ('TEST1', 'Serrano, Carmen', 'Spain', 'Spanish', 2010, 'Fiction');

-- 3 editions (different languages)

INSERT INTO editions (isbn, title, author, language, national\_lib\_id)

VALUES ('ISBN1001', 'TEST1', 'Serrano, Carmen', 'Spanish', 'NL1001');

INSERT INTO editions (isbn, title, author, language, national\_lib\_id)

VALUES ('ISBN1002', 'TEST1', 'Serrano, Carmen', 'English', 'NL1002');

INSERT INTO editions (isbn, title, author, language, national\_lib\_id)

VALUES ('ISBN1003', 'TEST1', 'Serrano, Carmen', 'French', 'NL1003');

-- Copies with no loans

INSERT INTO copies (signature, isbn, condition)

VALUES ('SIG01', 'ISBN1001', 'G');

INSERT INTO copies (signature, isbn, condition)

VALUES ('SIG02', 'ISBN1002', 'G');

INSERT INTO copies (signature, isbn, condition)

VALUES ('SIG03', 'ISBN1003', 'G');

Expected result: The book “TEST1” should appear in the result.

Result:

**Test 2 – Add a loan to the previously created copy**

****-- Insert user

INSERT INTO users (

user\_id, id\_card, name, surname1, surname2,

birthdate, town, province, address, email,

phone, type, ban\_up2

)

VALUES (

'USR0001', '11223344A00000001', 'Lucía', 'Ortega', 'Salas',

TO\_DATE('1999-09-09', 'YYYY-MM-DD'),

'Villaverde', 'Madrid',

'Av. de los Libros 15', 'lucia@books.com',

600123456, 'R', NULL

);

-- Insert book and editions (3 languages)

INSERT INTO books (title, author, country, language, pub\_date, topic)

VALUES ('TEST1', 'Serrano, Carmen', 'Spain', 'Spanish', 2010, 'Fiction');

INSERT INTO editions (isbn, title, author, language, national\_lib\_id)

VALUES ('RISBN001', 'TEST1', 'Serrano, Carmen', 'Spanish', 'NL101');

INSERT INTO editions (isbn, title, author, language, national\_lib\_id)

VALUES ('RISBN002', 'TEST1', 'Serrano, Carmen', 'English', 'NL102');

INSERT INTO editions (isbn, title, author, language, national\_lib\_id)

VALUES ('RISBN003', 'TEST1', 'Serrano, Carmen', 'French', 'NL103');

-- Insert copies

INSERT INTO copies (signature, isbn, condition)

VALUES ('RSIG1', 'RISBN001', 'G');

INSERT INTO copies (signature, isbn, condition)

VALUES ('RSIG2', 'RISBN002', 'G');

INSERT INTO copies (signature, isbn, condition)

VALUES ('RSIG3', 'RISBN003', 'G');

-- Insert a loan (signature RSIG1)

INSERT INTO loans (signature, user\_id, stopdate, town, province, type, time)

VALUES (

'RSIG1',

'USR0001',

TO\_DATE('22-11-2024', 'DD-MM-YYYY'),

'Sotomelones de las Caballerizas',

'Albacete',

'L',

20

);

Expected result: “TEST1” should no longer appear in the result.

Result:



**Test 3 – Book with only 2 languajes**

****-- Book with just 2 languajes

INSERT INTO books (title, author, country, language, pub\_date, topic)

VALUES ('TEST2', 'Carlos Ruiz', 'Spain', 'Spanish', 2012, 'Psychology');

INSERT INTO editions (isbn, title, author, language, national\_lib\_id)

VALUES ('ISBN2001', 'TEST2', 'Carlos Ruiz', 'Spanish', 'NL2001');

INSERT INTO editions (isbn, title, author, language, national\_lib\_id)

VALUES ('ISBN2002', 'TEST2', 'Carlos Ruiz', 'Catalan', 'NL2002');

INSERT INTO copies (signature, isbn, condition)

VALUES ('SIG04', 'ISBN2001', 'G');

INSERT INTO copies (signature, isbn, condition)

VALUES ('SIG05', 'ISBN2002', 'G');

Expected result: “TEST2” should not appear in the result.

Result:



## 

## QUERY 2: Reports on Employees

# P nackage

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

## Trigger2: Deregistration When Copy Becomes Deteriorated

Description of the Design

* Table: COPIES
* Event: UPDATE of the column condition
* Temporality: BEFORE
* Granularity: FOR EACH ROW
* Condition: The new value of :NEW.condition must be 'D' (for “Deteriorated”)
* Action: This trigger ensures that every time a copy of a book is marked as deteriorated (condition = 'D'), the system automatically sets the deregistered column to the current system date (SYSDATE). This provides a reliable way to track when copies were removed from circulation due to poor condition.

Code (PL/SQL)

## Trigger2: Deregistration When Copy Becomes Deteriorated

Description of the Design

* Table: COPIES
* Event: UPDATE of the column condition
* Temporality: BEFORE
* Granularity: FOR EACH ROW
* Condition: The new value of :NEW.condition must be 'D' (for “Deteriorated”)
* Action: This trigger ensures that every time a copy of a book is marked as deteriorated (condition = 'D'), the system automatically sets the deregistered column to the current system date (SYSDATE). This provides a reliable way to track when copies were removed from circulation due to poor condition.

Code (PL/SQL)

CREATE OR REPLACE TRIGGER trg\_deregistration\_on\_deteriorated

BEFORE UPDATE OF condition ON copies

FOR EACH ROW

WHEN (NEW.condition = 'D')

BEGIN

:NEW.deregistered := SYSDATE;

END;

/



Tests

**Test 1 – When a copy is deteriorated, the deregistration date should be set**

* Initial condition: Copy with signature 'FI430' exists with condition = 'G' and deregistered IS NULL.

Action:

****UPDATE copies

SET condition = 'D'

WHERE signature = 'FI430';

Verification:

****SELECT signature, condition, deregistered

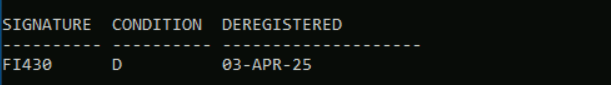
FROM copies

WHERE signature = 'FI430';



Expected Result: The deregistered field should be automatically updated with the current system date and time.

Result:



**Test 2 – When condition is changed to another value, the deregistration date must remain NULL**

Action:

****UPDATE copies

SET condition = 'G', deregistered = NULL

WHERE signature = 'FI430';

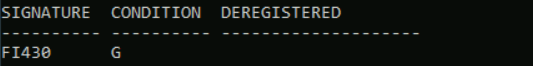
Verification:

****SELECT signature, condition, deregistered

FROM copies

WHERE signature = 'FI430';

Expected Result: The deregistered field must remain unchanged (NULL), confirming the trigger was not activated.



# 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.).