# Final Exam

**Databases**

Pierre Senellart

1 July 2026

You have 2 hours to answer this exam. You are allowed 10 A4 sheets (i.e., 20 pages) with the content of your choice. No electronic devices or any other material is allowed. Do not forget to add your name on all your answer sheets and to number them.

The exam consists of seven parts (A–G); it is graded out of 20 points. No relational schema is imposed: in Parts B, C, D and F you query the domain over the relational schema you design in Part A, and you should make your attribute names explicit whenever it is needed. Part A therefore conditions those parts: they are *not* independent of it. Parts E and G are self-contained. The number of points is indicated for each question. Write readable SQL: minor syntactic slips are tolerated as long as the intent is unambiguous.

*BiblioCampus* is the lending service of a university library. Members borrow books, keep them for a while, and return them. The service wants a relational database to manage its members, authors, books, and loans. The domain is described as follows:

- A *member* has a unique id, a name, an email, a membership category (for instance `'graduate'` or `'faculty'`), and a date on which they joined.
- An *author* has a unique id, a name, and a country.
- A *book* has a unique id, a title, a publication year, and a number of pages. Every book has exactly one (principal) *author*; an author has written many books.
- A *loan* records that a member borrowed a book, with a checkout date, a due date, and a return date (the return date is left empty until the book is actually returned). A member can make many loans; each loan concerns exactly one member and one book.

## A. Modelling and Design (3.5 points)

1. (2 points) Draw an Entity–Relationship diagram for this domain. Show the entity sets, their attributes (underline the key of each entity set), the relationships, and the cardinality constraints.
2. (1 point) Translate your ER diagram into a relational schema. For each relation, underline the primary key and indicate the foreign keys.
3. (0.5 point) Give one integrity constraint of this domain that *cannot* be enforced by keys and foreign keys alone, and say briefly how a DBMS could still enforce it.

## B. Relational Algebra (2 points)

Write a relational-algebra expression for each of the following queries.

4. (0.5 point) The titles of books that have at least 600 pages or were published before 1950.
5. (1.5 points) The titles of books that have never been borrowed by any member in the `'faculty'` category.

## C. Relational Calculus (1 point)

Use *domain* relational calculus with positional predicates over the relations of the schema you designed in Part A.

6. (1 point) The names of members who have borrowed at least one book written by an author from France (that is, whose country is `'France'`).

## D. SQL (6 points)

Write an SQL query for each of the following.

7. (1.5 points) The titles of the books that are currently on loan, that is, borrowed but not yet returned.
8. (1.5 points) The names of authors whose books have been borrowed by more than 20 distinct members.
9. (1.5 points) The 5 members who have borrowed the most books, together with the number of books they borrowed, most active first.
10. (1.5 points) The names of members who have *never* returned a book late, that is, who have no loan whose return date is later than its due date. Use `NOT EXISTS`.

## E. Functional Dependencies and Normalization (3 points)

To compute late fees, an intern proposes a single relation that logs, for each (member, book) borrowing, the fee information

$$\texttt{FEELOG}(\underline{\texttt{member}}, \underline{\texttt{book}}, \texttt{email}, \texttt{category}, \texttt{daily\_rate}, \texttt{title}, \texttt{days\_overdue})$$

whose only key is {member, book} (a member borrows a given book at most once), with the functional dependencies

$$\texttt{member} \to \texttt{email}, \texttt{category} \qquad \texttt{category} \to \texttt{daily\_rate}$$
$$\texttt{book} \to \texttt{title} \qquad \texttt{member}, \texttt{book} \to \texttt{days\_overdue}.$$

11. (1 point) Is FEELOG in 2NF? Justify your answer, and if it is not, exhibit a functional dependency that violates the condition.

12. (1.5 points) Give a decomposition of FEELOG into BCNF.

13. (0.5 point) Is your decomposition dependency-preserving? Justify briefly.

## F. Storage, Indexing, and Query Optimization (2.5 points)

14. (1 point) The loan table has $10^7$ rows and is not sorted on disk. A very frequent query retrieves all loans checked out within a given range of dates. (a) Which kind of index would you build to answer it efficiently, and why is a hash index *not* suitable here? (b) Would making that index *clustered* help? Explain.

15. (1.5 points) Let $R(\underline{k}, a)$ and $S(\underline{\ell}, k, c)$ be two relations, where attribute $c$ belongs to $S$ only. Consider the algebra expression

$$\sigma_{c='x'}\big(R \bowtie_{R.k=S.k} S\big).$$

Give an equivalent but cheaper-to-evaluate expression, name the equivalence rule you used, and explain in one sentence why it is faster.

## G. Databases from Python and the Web (2 points)

16. (1 point) For this question, set aside the schema you designed in Part A and assume instead that a book carries a status column. To register a loan, an application runs two statements: it inserts a row into loan, then marks the borrowed book as out.

```
cur.execute("INSERT INTO loan(member, book, checkout_date) "
            "VALUES (%s, %s, %s)", (m, b, today))
cur.execute("UPDATE book SET status = 'out' WHERE id = %s", (b,))
conn.commit()
```

(a) Why should these two statements run in a single transaction? (b) What is the effect of conn.rollback(), and when should it be called here?

17. (1 point) A member's profile page renders the free-text biography stored in the database with the Jinja expression

```
<p>{{ member.bio | safe }}</p>
```

(a) What security vulnerability does the |safe filter introduce here? (b) How would you fix it?