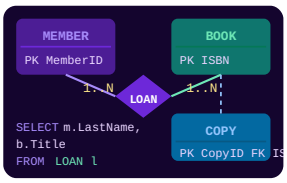
// EDUCATIONAL SAMPLE — FOR LEARNING PURPOSES ONLY

 UK LEVEL 5 — YEAR 2 UNDERGRADUATE



Database Design Project: Conceptual Modelling, Normalisation and SQL Implementation

-- System: LibraryLink University Library Management System

MODULE CODE	CSC2010
MODULE TITLE	Database Design
ACADEMIC LEVEL	UK Level 5 (Year 2 Undergraduate)
TARGET GRADE	Distinction — 70%+
WORD COUNT	Approximately 1,500 words (excl. code & references)
TECHNOLOGY STACK	SQL (ISO/IEC 9075), MySQL 8.x compatible
REFERENCING STYLE	Harvard (UK)
DATE	July 2026
PREPARED BY	AssignProSolution.com

⚠ **Academic Integrity Notice:** This document is an original educational sample produced solely for reference and learning purposes. Students must not submit this work as their own. "LibraryLink" and all data are entirely fictitious. AssignProSolution.com promotes responsible academic practice.

TABLE OF CONTENTS

```
# Approx. 1,500 words | Distinction Level | UK Level 5 | SQL / MySQL 8.x
```

1. INTRODUCTION AND SYSTEM REQUIREMENTS

This project designs and implements a relational database for **LibraryLink** — a fictitious university library management system. The system must support the core operations of a university library: cataloguing books by title, ISBN, author(s) and category; tracking physical copies of each title across library locations; registering members (students, staff and external members); and managing the complete lifecycle of a loan — from issue through to return and fine calculation.

The design follows the three-stage methodology established by Connolly and Begg (2015): *conceptual design* (Entity-Relationship modelling), *logical design* (relational schema and normalisation) and *physical design* (SQL DDL implementation). This structured approach separates data modelling concerns at each level of abstraction, ensuring that the final database is both logically correct and efficiently implementable on a real RDBMS — in this case, MySQL 8.x.

Key requirements extracted from the problem domain: (R1) A book may have multiple authors; one author may write multiple books (M:N). (R2) A title may have multiple physical copies; each copy belongs to exactly one title (1:N). (R3) A member may borrow many copies over time; each loan record links one copy to one member at a specific date (1:N on both sides via LOAN). (R4) Overdue returns incur a calculable fine stored at loan level. (R5) Books are classified by category and published by a publisher.

2. CONCEPTUAL DESIGN: ENTITY-RELATIONSHIP MODEL

The Entity-Relationship (ER) model, introduced by Chen (1976), provides a diagrammatic language for representing real-world entities,

their attributes and the relationships between them — independent of any physical implementation. Figure 1 presents the ER diagram for LibraryLink using simplified crow's foot cardinality notation (Connolly and Begg, 2015).

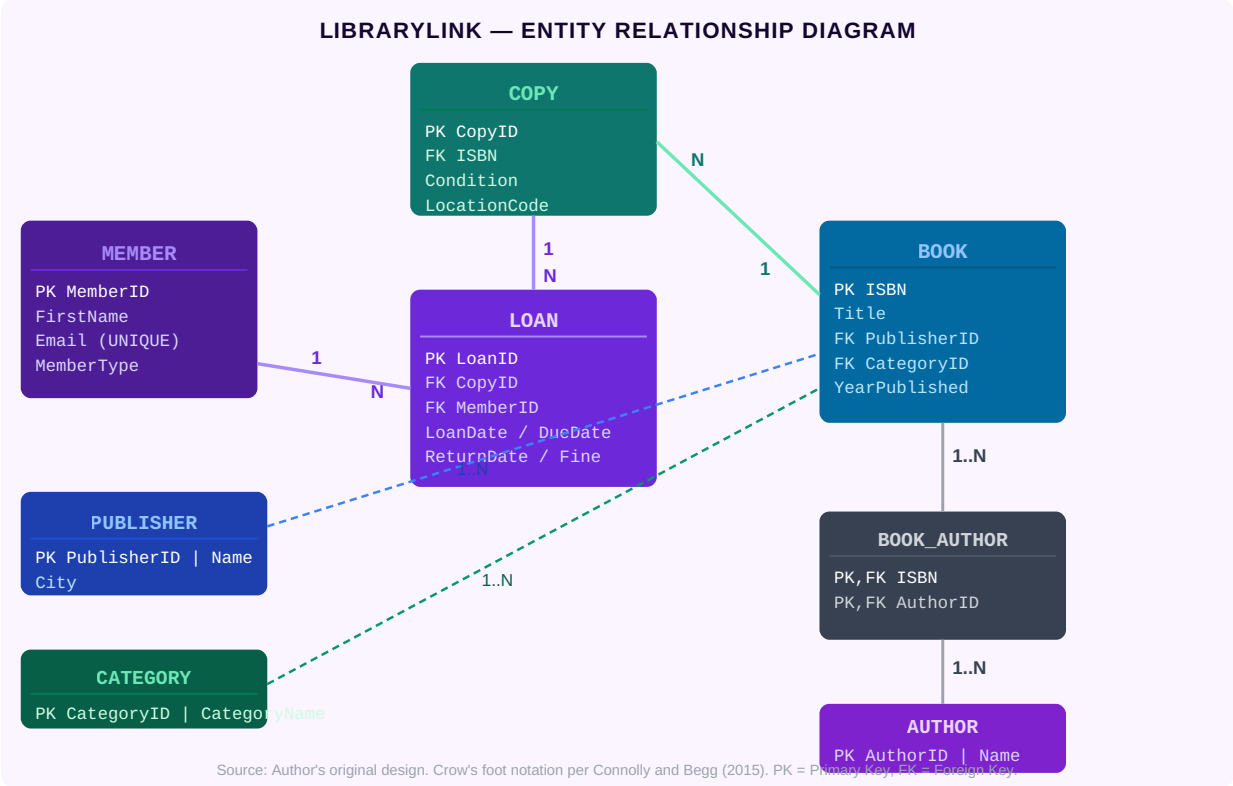


Figure 1: Entity-Relationship Diagram — LibraryLink University Library Management System (8 entities, 6 relationships). Dashed lines indicate FK look-up relationships. Source: Author's original design. Notation: Connolly and Begg (2015, p. 342).

The ER diagram identifies eight entities. The `BOOK_AUTHOR` junction table resolves the M:N relationship between `BOOK` and `AUTHOR` (Requirement R1) — a design decision essential in relational modelling since a direct M:N relationship cannot be represented in a single relation without violating First Normal Form. Each entity has a surrogate integer primary key (except `BOOK`, which uses the natural ISBN-13 key) to ensure uniqueness and immutability of row identifiers (Date, 2019).

3. LOGICAL DESIGN: RELATIONAL SCHEMA

The logical design translates the ER model into a set of relation schemas — the formal representation of tables in the relational model (Codd, 1970). Each schema lists attributes with their domain constraints; underlined attributes indicate primary keys; `FK` prefixes indicate foreign keys:

```
MEMBER (MemberID, FirstName, LastName, Email, JoinDate, MemberType)
PUBLISHER (PublisherID, PublisherName, City)
CATEGORY (CategoryID, CategoryName)
BOOK (ISBN, Title, FK PublisherID, FK CategoryID, YearPublished)
AUTHOR (AuthorID, FirstName, LastName)
BOOK_AUTHOR (ISBN, AuthorID)  ← composite PK
COPY (CopyID, FK ISBN, Condition, LocationCode)
LOAN (LoanID, FK CopyID, FK MemberID, LoanDate, DueDate, ReturnDate, Fine)
```

Referential integrity is enforced through foreign key constraints at every FK attribute: the RDBMS rejects any INSERT or UPDATE that would introduce a foreign key value with no matching primary key in the referenced table (Elmasri and Navathe, 2016). The `BOOK_AUTHOR` table carries a *composite primary key* of `(ISBN, AuthorID)` — ensuring that the same author cannot be listed twice for the same book, while permitting the same author to appear with multiple books and the same book to have multiple authors.

4. NORMALISATION: UNF TO THIRD NORMAL FORM

Normalisation is the systematic process of restructuring relations to reduce data redundancy and eliminate update anomalies — the pathological states (insertion, deletion and modification anomalies) that arise in poorly designed schemas (Codd, 1971). The process is guided by the theory of **functional dependencies**: attribute B is functionally dependent on attribute A (written A → B) if each value of A determines exactly one value of B (Armstrong, 1974).

Table 1 traces the normalisation of a hypothetical flat `BOOK_LOAN` relation through three normal forms:

Table 1: Normalisation Trace — BOOK_LOAN (UNF → 1NF → 2NF → 3NF)

Stage	Relation(s)	Violation Removed	Action Taken
UNF	BOOK_LOAN(LoanID, MemberName, MemberEmail, ISBN, **AuthorList**, BookTitle, CopyID, LoanDate, DueDate, Fine)	**Repeating group: AuthorList holds multiple author names in one cell**	Flatten repeating groups → one row per author per loan
1NF	BOOK_LOAN(LoanID, AuthorID, MemberName, MemberEmail, ISBN, AuthorName, BookTitle, CopyID, LoanDate, DueDate, Fine) — composite PK (LoanID, AuthorID)	**Partial dependency: BookTitle depends only on ISBN (part of key), not on full composite PK**	Extract BOOK(ISBN, Title) and AUTHOR(AuthorID, Name)
2NF	LOAN(LoanID, MemberID, CopyID, LoanDate, DueDate, Fine) MEMBER_DETAIL(MemberID, MemberName, MemberEmail) BOOK(ISBN, Title) | COPY(CopyID, ISBN)	**Transitive dependency: MemberEmail depends on MemberID, not on LoanID**	Extract MEMBER(MemberID, FirstName, LastName, Email, MemberType) as separate table
3NF ✓	Final schema: MEMBER, PUBLISHER, CATEGORY, BOOK, AUTHOR, BOOK_AUTHOR, COPY, LOAN — all in 3NF	**No further violations — every non-key attribute depends directly on the primary key only**	Schema is complete. No transitive or partial dependencies remain.

The critical transition from 2NF to 3NF eliminates the transitive dependency `LoanID → MemberID → MemberEmail`. If MemberEmail were stored in LOAN rather than in a separate MEMBER table, updating a member's email address would require modifying every loan row for that member — a modification anomaly that could leave the database in an inconsistent state if any row were missed (Connolly and Begg, 2015). The 3NF schema guarantees that any change to member contact details requires updating exactly one row in the MEMBER table, regardless of how many loans that member has.

5. PHYSICAL IMPLEMENTATION: SQL DDL

The physical design translates the 3NF relational schema into SQL Data Definition Language (DDL) statements. All tables are created in dependency order — referenced tables before referencing tables — to satisfy foreign key constraints at creation time. Constraint definitions are co-located with the table they protect, following the principle that data integrity should be enforced at the database level, not only in application code (Date, 2019).

SQL (MySQL 8.x) librarylink_schema.sql — Core DDL (selected tables)

```sql
-- ── LibraryLink Schema — CSC2010 Database Design Project ──

CREATE TABLE MEMBER (
    MemberID   INT          PRIMARY KEY AUTO_INCREMENT,
    FirstName  VARCHAR(50)  NOT NULL,
    LastName   VARCHAR(50)  NOT NULL,
    Email      VARCHAR(120) NOT NULL UNIQUE,
    JoinDate   DATE         NOT NULL DEFAULT (CURRENT_DATE),
    MemberType CHAR(1)      NOT NULL
                            CHECK (MemberType IN ('S','F','E'))
               -- S=Student | F=Faculty | E=External
);

CREATE TABLE BOOK (
    ISBN          CHAR(13)     PRIMARY KEY,
    Title         VARCHAR(200) NOT NULL,
    PublisherID   INT          NOT NULL,
    CategoryID    INT,
    YearPublished SMALLINT
                  CHECK (YearPublished BETWEEN 1800 AND 2100),
    FOREIGN KEY (PublisherID) REFERENCES PUBLISHER(PublisherID),
    FOREIGN KEY (CategoryID)  REFERENCES CATEGORY(CategoryID)
              ON DELETE SET NULL  -- preserve book if category removed
);

CREATE TABLE BOOK_AUTHOR (
    ISBN     CHAR(13) NOT NULL,
    AuthorID INT      NOT NULL,
    PRIMARY KEY (ISBN, AuthorID),    -- composite PK
    FOREIGN KEY (ISBN)     REFERENCES BOOK(ISBN)   ON DELETE CASCADE,
    FOREIGN KEY (AuthorID) REFERENCES AUTHOR(AuthorID) ON DELETE CASCADE
);

CREATE TABLE LOAN (
    LoanID     INT          PRIMARY KEY AUTO_INCREMENT,
    CopyID     INT          NOT NULL,
```

```
    MemberID   INT          NOT NULL,
    LoanDate   DATE         NOT NULL DEFAULT (CURRENT_DATE),
    DueDate    DATE         NOT NULL,
    ReturnDate DATE,        -- NULL until returned
    Fine       DECIMAL(6,2) DEFAULT 0.00
               CHECK (Fine >= 0.00),
    FOREIGN KEY (CopyID)   REFERENCES COPY(CopyID),
    FOREIGN KEY (MemberID) REFERENCES MEMBER(MemberID)
);
```

Three constraint types warrant critical discussion. `NOT NULL` prevents missing required data at the database level — applying it to `LoanDate` and `DueDate` ensures every loan has mandatory temporal records, regardless of how the application layer behaves. `CHECK` constraints enforce domain integrity inline: `MemberType IN ('S','F','E')` prevents invalid category values being inserted, removing the need for application-side validation of that column. `ON DELETE CASCADE` on `BOOK_AUTHOR` ensures that deleting a book also removes its authorship records — avoiding orphaned foreign key rows. The alternative `ON DELETE SET NULL` on the Book–Category relationship preserves book records if a category is deleted, a deliberate design choice reflecting the relative importance of the two entities.

6. QUERY DESIGN

The following SQL `SELECT` query retrieves all currently overdue loans — those where `DueDate` has passed but `ReturnDate` is still NULL — joining across four tables to produce a human-readable report for library staff:

SQL — SELECT Query — Overdue loans report

```
SELECT
    m.LastName   AS 'Member',
    m.Email      AS 'Email',
    b.Title      AS 'Book Title',
    l.DueDate    AS 'Due',
    DATEDIFF(CURRENT_DATE, l.DueDate) AS 'Days Overdue',
    ROUND(DATEDIFF(CURRENT_DATE, l.DueDate) * 0.20, 2)
                 AS 'Fine (£)'    -- £0.20/day fine rate
FROM       LOAN   l
INNER JOIN MEMBER m ON l.MemberID = m.MemberID
INNER JOIN COPY   c ON l.CopyID   = c.CopyID
INNER JOIN BOOK   b ON c.ISBN     = b.ISBN
WHERE l.ReturnDate IS NULL
  AND   l.DueDate   < CURRENT_DATE
ORDER BY 'Days Overdue' DESC;
```

7. TRANSACTION INTEGRITY: ACID PROPERTIES

A *transaction* is a logical unit of work consisting of one or more SQL operations that must execute as an atomic, consistent, isolated and durable unit (Haerder and Reuter, 1983). Returning a book in LibraryLink involves multiple operations — setting `ReturnDate`, calculating and writing the `Fine` — that must succeed or fail together:

A

ATOMICITY

If the fine calculation fails mid-transaction, the ReturnDate update is rolled back. Either the entire return is recorded or nothing is.

C

CONSISTENCY

The DB moves from one valid state to another — all FK constraints, NOT NULL and CHECK constraints remain satisfied after each commit.

I

ISOLATION

Concurrent loan issuances for the same copy cannot both commit — the second sees the first's lock and waits or fails.

D

DURABILITY

Once a COMMIT is issued, the return is persisted to disk even if the server crashes immediately afterwards.

8. EVALUATION AND CONCLUSION

The LibraryLink database design successfully addresses all five stated requirements through a coherent three-stage methodology. The ER model correctly identifies the M:N Book–Author relationship and resolves it via the `BOOK_AUTHOR` junction table — the only architecturally correct relational solution. Normalisation to 3NF eliminates all identified insertion, deletion and modification anomalies. The SQL DDL implements comprehensive integrity constraints that shift data quality enforcement to the database engine rather than relying on application-layer validation alone.

Three limitations exist. First, the current schema has no `RESERVATION` entity — students cannot place a hold on a copy currently on loan. Extending the schema would require a RESERVATION table linking MEMBER and COPY with a status attribute. Second, fine calculation is currently delegated to the query layer — a CHECK trigger or stored procedure would enforce the £0.20/day rule at the database level, making it impossible for application code to accidentally write an incorrect fine value. Third, the schema stores no historical price information for categories or publishers — should the library need to track acquisition costs, a PURCHASE table linked to COPY would be required. These extensions represent planned future iterations rather

than design failures: the current schema prioritises correctness and simplicity at Version 1.0.

// KEY DESIGN DECISIONS

- **ISBN as natural PK for BOOK:** ISBN-13 is globally unique, externally assigned and stable — meeting all criteria for a natural primary key (Date, 2019). Surrogate keys are used for all other entities where no suitable natural key exists.
- **BOOK_AUTHOR as explicit junction table:** Avoids the M:N relationship that cannot be represented in 1NF without violating atomicity of values.
- **Fine stored at LOAN level:** Fine is a fact of the loan event, not of the member or the copy. Storing it at LOAN level is consistent with the entity it describes and avoids the need for a separate FINE table for this use case.
- **ON DELETE CASCADE on BOOK_AUTHOR:** Ensures orphaned authorship records cannot accumulate if a book is removed from the catalogue — maintaining referential integrity automatically without requiring multi-step application-level deletes.

REFERENCES

Armstrong, W.W. (1974) 'Dependency structures of data base relationships', in *Proceedings of IFIP Congress 74*. Amsterdam: North-Holland, pp. 580–583.

Chen, P.P.-S. (1976) 'The entity-relationship model — toward a unified view of data', *ACM Transactions on Database Systems*, 1(1), pp. 9–36. https://doi.org/10.1145/320434.320440

Codd, E.F. (1970) 'A relational model of data for large shared data banks', *Communications of the ACM*, 13(6), pp. 377–387. https://doi.org/10.1145/362384.362685

Codd, E.F. (1971) 'Further normalisation of the data base relational model', in Rustin, R. (ed.) *Data Base Systems: Courant Computer Science Symposia Series 6*. Englewood Cliffs, NJ: Prentice-Hall, pp. 33–64.

Connolly, T. and Begg, C. (2015) *Database Systems: A Practical Approach to Design, Implementation and Management*. 6th edn. Harlow: Pearson.

Date, C.J. (2019) *An Introduction to Database Systems*. 8th edn. Reading, MA: Addison-Wesley.

Elmasri, R. and Navathe, S.B. (2016) *Fundamentals of Database Systems*. 7th edn. Harlow: Pearson.

Haerder, T. and Reuter, A. (1983) 'Principles of transaction-oriented database recovery', *ACM Computing Surveys*, 15(4), pp. 287–317. https://doi.org/10.1145/289.291

ISO/IEC (2023) *ISO/IEC 9075-1:2023 — Information Technology — Database Languages — SQL — Part 1: Framework (SQL/Framework)*. Geneva: International Organization for Standardization.

Ramakrishnan, R. and Gehrke, J. (2016) *Database Management Systems*. 3rd edn. New York: McGraw-Hill.

Silberschatz, A., Korth, H.F. and Sudarshan, S. (2020) *Database System Concepts*. 7th edn. New York: McGraw-Hill.