College Library Database System

ECS740P – Database Systems

Group 49

Table of Contents

Full name

Nahid Topalovic

Introduction

A college library is defined as an institution providing various complementary resources for borrowing for students and staff, including books, videos, DVDs, and CDs, intended to supplement lecture materials. Generally, access to electronic resources is also made available for students.

The goal of the following report is to provide a synopsis of the design of the database in the context of a college library system. The subsequent steps taken to satisfy these requirements include:

- The conceptual schema in the form of a Unified Modeling Language (UML) diagram
- A list of all the assumptions that will be required in designing a database of a college library system
- A relational database schema derived from the UML diagram
- A normalization of the relations, up to Third Normal Form (3NF)

The requirements for the design of this system are described in the following paragraphs:

‘ *A college library provides various resources for students and staff, including books, videos, DVDs and CDs. It is common that several copies are kept of some resources, for example recommended books for courses. The usual loan period of a resource is 2 weeks, but some resources are available for short loan only (2 days) and some other resources can only be used within the library.*

The library consists of 3 floors. Resources are stored in the library on shelves. To locate a specific item in the library a combination of floor number and shelf number are used. In addition to this, a class number system is used to identify in which subject area a particular item belongs, for example all resources concerned with Database Systems will have the same class number.

Students hold library cards which identify them as valid members of the Library. Students can lone a number of different resources at one time, but the total number of resources they may borrow at a given time must never exceed 5. Staff members at the College also hold library cards, and are allowed to lone up to 10 different items at one time.

The library charges fines for resources that are loaned for longer than the time allowed for that resource. For each day a resource is overdue the member is fined one dollar. When the amount owed in fines by a member is more than 10 dollars, that member is suspended until all resources have been returned and all fines paid in full.

The system is required to maintain a record of all of the above details. Specifically, it needs to keep track of:

- *What each resource is, its class number, how many copies of it are held by the library and where these are located in the library.*

- *Student and staff members of the library.*

- *All current loans, including whether they are overdue.*
- *A record of previous loans to help in identifying popular resources. (Create a view to show previous loans NT)*
- *The details of any fines owed by members.*
- *A list of library members who have been suspended due to overdue loans or unpaid fines'* (Stockman, 2020).

Assumptions

The following assumptions add information that is supplementary to the specification mentioned in the Introduction section, yet fundamental to our design of the college library database system by filling the gaps left from the specification to reproduce a comprehensive context of how a college library operates. Various assumptions are made for each entity, based upon inference and logical deductions with respect to the physical structure and operation of a college library.

Library and **Floor**

- This database can support more than one library, and each library has its own LibraryName and Address attributes.
- Each library has several floors and each floor contains one or more shelves.
- Each library shares the standard British floor numbering system. For example, the floor that is with the ground is referred to as "G/F", and the floor above it "1/F" and so on.

Shelf

- Each library adopts its own shelf numbering system.
- A shelf can be located with its ShelfNumber and FloorNo.
- Shelves are large enough to accommodate all copies of the same resource item.

Resource

- Each resource has a unique ID (ResourceID) and copies of the same resource share the same ID. The attribute NumOfCopies indicates the total number of copies of a single resource, whereas AvailableCopies specifies the number of copies of that resource available for loan, which is a derived attribute and thus not included in the model. There is also an advantage of adopting ResourceID instead of ISBN for each resource, given the fact that some resource items such as DVDs do not possess an ISBN.
- The attribute TotalNumberOfLoans indicates the number of previous loans, which can reflect popularity of a particular resource item.
- The type of resources is defined by the ResourceType attribute whereby any future introduction of new resource types can be easily carried out.
- Different resources can have the same name but they can be of a different type.

Course

- Each resource has a corresponding subject area. One resource is limited to one subject area, i.e. each resource is assigned one CourseNo.
- A course may not necessarily require any resources that can be borrowed from the library.

LibraryCard, StudentCard and **StaffCard**

- There are only two types of library cards – Students and Staff. Other types such as "Alumni" are not considered in our design.
- The attribute IsSuspended offers information about whether a user has been suspended. It is assumed that a user who has an outstanding fine exceeding £10, they will be automatically suspended from using the library services.
- There is only one registered card holder (user) for each library card, which can only be used to access one library. Hence, the LibraryCard entity can correspond to only one LibraryID as its foreign key.

Loan

- One user can have multiple loans on record for each CardID, where CardID, ResourceID and LoanDate constitute the primary key for a particular loan.
- Only one resource item can belong to one record in the Loan relation.
- The Loan relation stores current and previous loans, which can be differentiated by the LoanDate attribute.

Fine

- One library card can be associated with multiple records of fines.
- Each record of fine in the Fine entity corresponds to one loan only and hence one resource.

UML Class Diagram

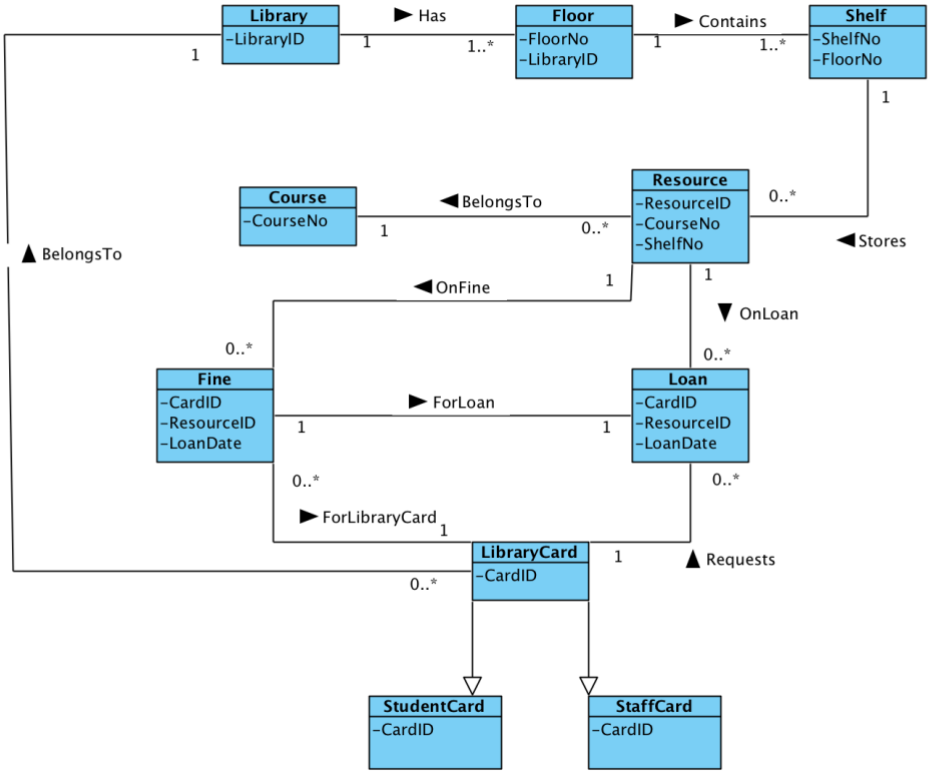

Figure 1 UML Class Diagram of College Library System

The college library database system presented here consists of ten entities (Library, Floor, Shelf, Course, Resource, Fine, Loan, LibraryCard, StudentCard and StaffCard). This model can support multiple libraries which have their own corresponding entities floor, shelf, resources and library cards. That means that libraries do not share their storage capacities, resources, library cards and other entities. Each entity is unique to one library.

Since libraries can have a different number of floors and shelves, a separate entity Floor and Shelf have been created to enable this requirement. Each floor has a primary key FloorNo and foreign key LibraryID. Given that different libraries adopt the same floor numbering system, Floor becomes a weak entity due to its existence-dependent relationship with the entity Library. Hence, FloorNo per se does not suffice as a sole key attribute type, thereby necessitating the use of LibraryID as a foreign key.

Shelf entity has its primary key ShelfNo and foreign key FloorNo. Contrary to the Floor entity, Shelf has its own key attributes on the grounds that the libraries utilise different shelf numbering systems. To ensure an accurate search of a resource item across different libraries, the attribute FloorNo is borrowed from the Floor entity, which becomes a foreign key to the Shelf entity. Modelling the location of a resource using separate entities for Floor and Shelf give us an option to list all resources that are in each floor or in each shelf. This way each resource can be located using a ShelfNo, which can be traced back to the corresponding floor and library.

The Resource entity is used to keep information about resources that a library holds. Different types of resources (CDs, Books, DVDs) can be specified using an attribute ResourceType. Location of a resource is stored as a foreign key ShelfNo. Number of copies attribute is used to store information about how many copies of each resource is in the library. All copies of the same resource share the same shelf. Attribute TotalNumOfLoans is used to keep track of previous loans so that it could be possible to list the most popular resources. This attribute can also be derived, but we have opted not to, so that it can be represented in the relational model.

In order to satisfy requirement that resources belong to certain subject area, the Course entity was created to store information about different courses. CourseNo is used as a foreign key in Resource entity.

The Loan entity keeps track of all current and previous loans that library users made. CardID (a foreign key from LibraryCard), ResourceID (a foreign key from Resource) and LoanDate are combined to form a composite primary key for this entity, which makes Loan a weak entity. This composite primary key allows us to uniquely identify each loan. In first instance of the UML model we used LoanID as a primary key, since that would allow us not to list all the three attributes as foreign keys to reference a particular loan in, for example, the FINE table, but after reviewing it we concluded that it was unnecessary. Besides its primary key, the Loan entity stores information about due date (DueDate) and whether a loan is overdue (IsOverdue). These two attributes can be modelled as derived attributes, where DueDate can be calculated using fields LoanPeriod from Resource table and LoanDate from Loan table. IsOverdue can be calculated based on fields DueDate and LoanDate from Loan relation. However, in this model we have stored them as regular attributes.

In order to keep information about Fine separate from the Loan entity, the Fine entity was created with foreign key fields (CardID, ResourceID, LoanDate) that uniquely identify a resource item that a card owner was fined for and the date of a loan. Fine is a weak entity because it lacks a unique primary key of its own and needs to borrow the key attributes CardID, ResourceID and LoanDate from LibraryCard, Resource and Loan respectively. Fine amount, whether the user had settled the fine and the date when the fine was paid are all stored in Fine relation as FineAmount, IsPaid, PayDate attributes.

Library uses library cards in order to identify students or staff who make the loans. For that reason entity LibraryCard was created which represents a generalisation of StudentCard and StaffCard entities. This is a case of a total specialization where library card user can be either student or staff member (disjoint specialization). LibraryCard entity stores information about Name, Email, IssueDate, whether the user is suspended (IsSuspended) and the number of resources currently loaned (NumOfResourcesLoaned). CardID is used as a primary key. StudentCard and StaffCard specializations inherit all attributes through a CardID foreign key, while they have their own attributes StudentID and StaffID. These two specializations have been created in order to satisfy different requirements of college students and staff. For example, StaffID and StudentID might use a different format for their identification and later in implementation of this database we will want to set different field value constraints.

Based on all the considerations above, the UML class diagram can be mapped into the relational model, which is displayed in the next section.

Relational Database Schema

After defining the UML diagram, the relational model is mapped from the UML diagram. The primary keys of each entity or relation come first on the list of attributes and are underlined. The foreign keys are denoted in italics and put right after the primary key(s) of a particular entity or relation. The first step was to map each entity type into a relation.

Library (LibraryID, LibraryName, Address)

As this is a simple entity we have created a relation where the primary key is LibraryID.

Floor (FloorNo, *LibraryID*)

Relation Floor has 1:N relationship type with Library entity, which represents that library can have at least one or N number of floors. Each floor uses FloorNo as a primary key and this relationship is mapped by including a foreign key in the Floor relation, LibraryID.

Shelf (ShelfNo, *FloorNo*)

Analogous to the previous relation, shelf has a 1:N relationship with the Floor entity. ShelfNo is a primary key in this relation, while FloorID is a foreign key from the Floor relation.

Resource(ResourceID, *CourseNo*, *ShelfNo*, ResourceType, ResourceName, NumOfCopies, LoanPeriod, TotalNumOfLoans, Author)

In this relation ResourceID is a primary key. In order to have an information about the location of each resource, we have included ShelfNo foreign key from Shelf relation. Because Shelf relation has 1:N relationship type with Resource table, this attribute is saved on the N side (Resource relation). CourseNo is a foreign key to Course relation and it is included in this side of the relation for the same reason as previous attribute. Other relevant attributes included in this relation are ResourceType, ResourceName, NumOfCopies, LoanPeriod, TotalNumOfLoans and Author.

Course (CourseNo, CourseName)

The primary key in this table is CourseNo.

Loan (*CardID*, *ResourceID*, *LoanDate*, DueDate, IsOverdue)

Loan relation consists of composite primary key CardID, ResourceID and LoanDate. These three can be used to uniquely identify each loan. CardID and ResourceID are foreign keys from LibraryCard and Resource table respectively. Loan table has a 1:N relationship type with Resource and Loan tables and for that reason primary key attributes from those two relations are included in Loan relation.

Fine (*CardID, ResourceID, LoanDate*, FineAmount, IsPaid, PayDate)

In order to keep information about fines separate from Loans table, we have created a FINE table. This relation has a composite primary key consisting of three foreign attributes: CardID, ResourceID and LoanDate from Resource, LibraryCard and Loan tables.

LibraryCard (CardID, *LibraryID*, Name, Email, IssueDate, IsSuspended, NumOfResourcesLoaned)

The primary key in this relation is CardID. This relation has a 1:N relationship with Library table, which means that each tuple from LibraryCard relation can correspond to only one Library and Library can have many library cards. In order to satisfy this requirement, we have included LibraryID as a foreign key. LibraryCard is a generalization of the next two tables: StudentCard and StaffCard.

StudentCard (*CardID*, StudentID)

This relation has been mapped as a specialization of LibraryCard relation. We have included a primary key CardID which is a foreign key of LibraryCard relation and StudentID attribute which is specific to this specialization.

StaffCard (*CardID*, StaffID)

Analogous to StudentCard relation, this relation is also a specialization of LibraryCard relation. The primary key is CardID and specific attribute to this specialization is StaffID.

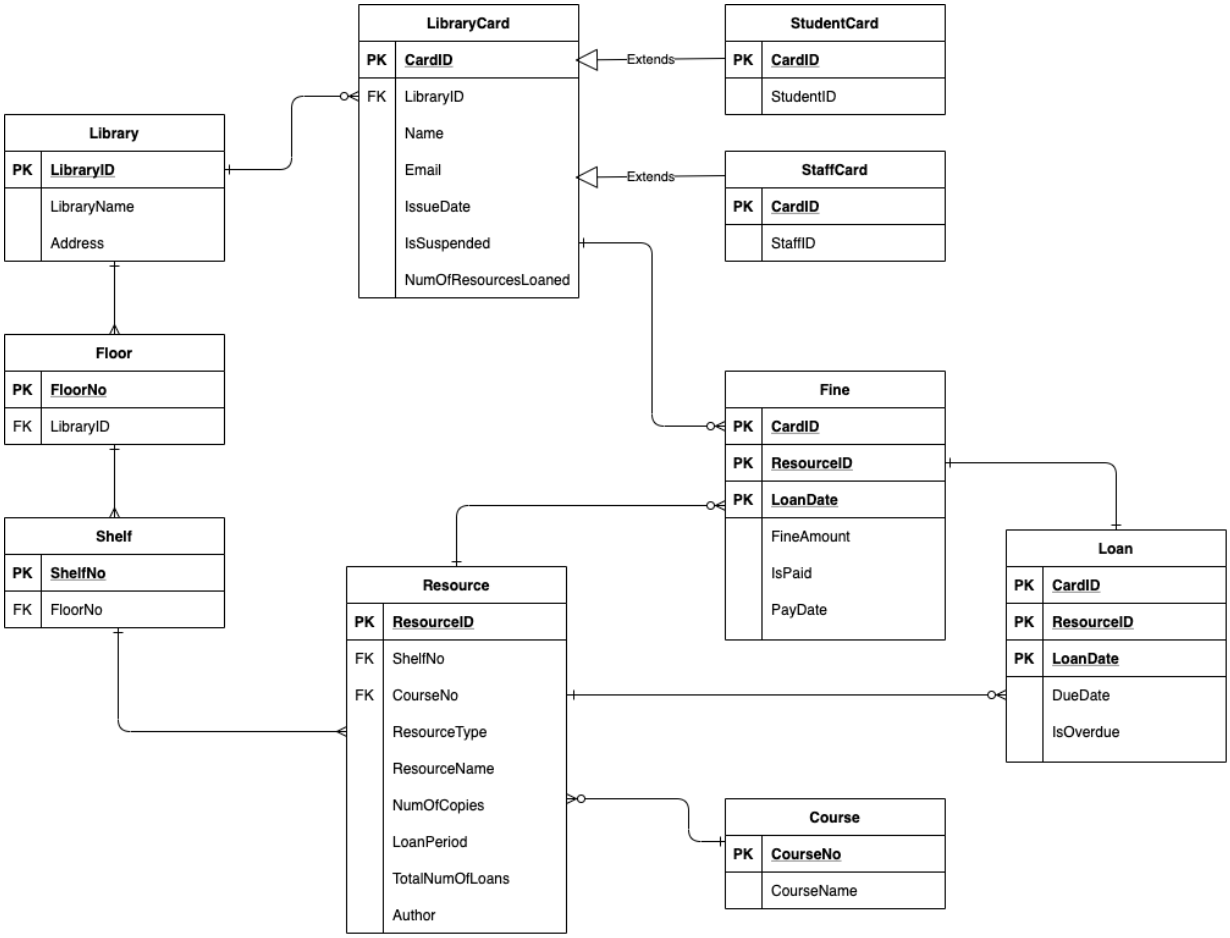

Figure 2 Relational Schema Diagram of College Library System

Normalization

Normalization refers to a database design technique used for improving the flexibility of database designs by eliminating data redundancy. The aim of normalization is to eliminate repetitive or redundant data by further ensuring that data is stored in a logical manner. Edgar Codd (1971) proposed the theory of Normalization by introducing the First Normal Form. The theory was extended with introducing further the Second and Third Normal Form. Higher forms of Normalization such as Fourth and Fifth Normal Form and various other normal forms were also proposed in the database research literature; however, most designs are adopting Third Normal Form as their standard level of Normalization.

The aim of this section is to generate the successive normal forms required by the specification that our design should fulfill the Third Normal Form (3 NF).

First Form Normalization (1 NF):

The requirements for a First Normal Form are as follows:

1. The values are atomic containing single values, i.e. small strings and numbers.
2. The other part of the process of getting something into first normal form is to remove any repeating groups.

In order to satisfy first normal form requirements, the following changes are made:

1. Address column in Library gets separated to AddressLine1, Postcode, City
2. Name in LibraryCard gets separated to FirstName and LastName.

Library (LibraryID, LibraryName, AddressLine, PostCode, City)

LibraryCard (CardID, *LibraryID*, FirstName, LastName, IssueDate, Email, IsSuspended, NumOfResources)

Therefore, after changing the Library and LibraryCard entities, the following entities are in the first normal form as there are no attributes with multiple values.

Library (LibraryID, LibraryName, AddressLine, PostCode, City)

Floor (FloorNo, *LibraryID*)

Shelf (ShelfNo, *FloorNo*)

Resource
(ResourceID, *CourseNo*, *ShelfNo*, ResourceType, ResourceName, NumOfCopies, LoanPeriod, TotalNumOfLoans, Author)

Course (CourseNo, CourseName)

Loan (*CardID, ResourceID, LoanDate*, DueDate, IsOverdue)

Fine (*CardID, ResourceID, LoanDate*, FineAmount, IsPaid, PayDate)

LibraryCard (CardID, *LibraryID*, FirstName, LastName, IssueDate, Email, IsSuspended, NumOfResources)

StudentCard (*CardID*, StudentID)

StaffCard (*CardID*, StaffID)

Second Form Normalization (2 NF):

Second form normalization is defined as the following:

'A relation that is in first normal form and every non-primary-key attribute is fully functionally dependent on the primary key' (Connelly & Begg, 2015: 578)

By analyzing the entities created from the specification in first normal form, it can be seen that:

Library (LibraryID, LibraryName, AddressLine, PostCode, City)

This relation is already in second normal form as all the attributes are functionaly dependent on the LibraryID.

Floor (FloorNo, *LibraryID*)

This relation is already in second normal form as all the attributes are functionaly dependent on the primary key FloorNo.

Shelf (ShelfNo, *FloorNo*)

This relation is already in second normal form as all the attributes are functionaly dependent on the primary key ShelfNo.

Resource
(ResourceID, *CourseNo*, *ShelfNo*, ResourceType, ResourceName, NumOfCopies, LoanPeriod, TotalNumOfLoans, Author)

This relation is already in second normal form as all the attributes are functionaly dependent on the primary key ResourceID.

Course (CourseNo, CourseName)

This relation is already in second normal form as all the attributes are functionaly dependent on the primary key CourseNo.

Loan (*CardID, ResourceID, LoanDate*, DueDate, IsOverdue)

In Loan relation, DueDate and IsOverdue have partial functional dependancy. Both of these attributes do not functionally depend on the CardID attribute of the composite primary key, since the loan period is dependent of the resource and the date it was loaned.

For that reason the Loan relation was modified and a new table (LoanDue) was added in order to satisfy second form normalization:

Loan (*CardID, ResourceID, LoanDate)*

LoanDue (*ResourceID, LoanDate*, DueDate, IsOverdue)

Moving on to the next relation:

Fine (*CardID, ResourceID, LoanDate*, FineAmount, IsPaid, PayDate)

This relation is already in second normal form as all the attributes are functionaly dependent on the composite primary key CardID, ResourceID, LoanDate.

LibraryCard (CardID, *LibraryID*, FirstName, LastName, IssueDate, Email, IsSuspended, NumOfResources)

This relation is already in second normal form as all the attributes are functionaly dependent on the primary key LibraryID.

StudentCard (*CardID*, StudentID)

StaffCard (*CardID*, StaffID)

Both the StudentCard and StaffCard relations are in second normal form as all the attributes are functionaly dependent on the primary key CardID.

Third Form Normalization (3 NF):

The requirement for the third form normalization is:
'A relation is in the 3 NF if it satisfies 2 NF and no non-prime attribute type of relation is transitively dependent on the primary key.' (Lemahieu, 2018)

Following first and second form normalizations, all relations of the college library database system are already in the third normal form.

However, it could be debatable that the Resource table was not in the third normalized form.

Resource
(ResourceID, *CourseNo, ShelfNo*, ResourceType, ResourceName, NumOfCopies, LoanPeriod, TotalNumOfLoans, Author)

If the ResourceName attribute in the Resource relation was unique, then the attributes Author and ResourceType would functionally depend only on ResourceName, and thus need to form a new table in order to satisfy third form normalization. Nonetheless, this would not be the case since one of our assumptions states that:

Different resources can have the same name but they can be of a different type.

Conclusion

In summary, our report details step-by-step procedures undertaken to create an appropriate design of the database system for college libraries. The first step is to come up with essential entities and their attributes while simultaneously making assumptions that are consistent with the context of college library. A UML class diagram is created to identify the types of and relationships between the entities and is subsequently mapped into a relational model, in which the database manifests itself in a collection of relations. Normalization is conducted to review these relations and eliminate data redundancy. Fourth and Fifth Form Normalizations have not been carried out as all entities are adequately presented in their normalized forms after Third Form Normalization.

After undergoing normalization the entities in the normalized relational model remain largely the same when compared to the original relational database schema:

Library (LibraryID, LibraryName, AddressLine, PostCode, City)
Floor (FloorNo, *LibraryID*)
Shelf (ShelfNo, *FloorNo*)
Resource
(ResourceID, *CourseNo*, *ShelfNo*, ResourceType, ResourceName, NumOfCopies, LoanPeriod, TotalNumOfLoans, Author)
Course (CourseNo, CourseName)
Loan (*CardID, ResourceID, LoanDate)*
LoanDue (*ResourceID, LoanDate*, DueDate, IsOverdue)
Fine (*CardID, ResourceID, LoanDate*, FineAmount, IsPaid, PayDate)
LibraryCard (LibraryID, FirstName, LastName, IssueDate, Email, IsSuspended, NumOfResources)
StudentCard (*CardID*, StudentID)
StaffCard (*CardID*, StaffID)

The key differences between the original relational schema and the third normalized form include the decomposition of attributes Address of the Library entity and Name of the LibraryCard entity, as well as the introduction of the new entity LoanDue.

The current normalized relational model is presumed sufficient to satisfy the requirements entailed by the specification. Nonetheless, similar to how we refined some of the assumptions while creating the UML diagram, this model is open to minor changes wherever appropriate during the process of constructing the actual database.

References

Antony Stockman (2020*). ECS740P Database Systems Coursework Specification Task,* https://qmplus.qmul.ac.uk/mod/resource/view.php?id=1393341

Edgar Codd (1971). Normalized data base structure: a brief tutorial. *Access and Control*, https://dl.acm.org/doi/proceedings/10.1145/1734714

Thomas Connelly & Carolyn Begg (2015). Database Systems: A practical approach to Design Implementation and Management 6th Edition.

Wilfred Lemahieu (2018). Principles of database management.