Kingston University, BSc (Hons) (top-up)

CI6320 - Advanced Data Modelling

Draft Coursework – Subject to Moderation

Weight of this assessment is 25% of the final grade.

Coursework Cover Sheet

Part 1 - To Remain with the Assignment after Marking

Student ID: E030087/K2526504	**Student Name: Amanda Kaveesha Karunanayaka**
Module Code: CI6320	**Module Name: Advanced Data Modeling**
Assignment number: A1	**ESoft Module Leader:** Mr. W A D B C Goonatillaka
Date set:	**Date due: 25th February 2025**

Guidelines for the Submission of Coursework

1. Print this cover sheet and securely attach both pages to your assignment. You can help us ensure work is marked more quickly by submitting at the specified location for your module. You are advised to keep a copy of every assignment.

2. Coursework deadlines are strictly enforced by the University.

3. You should not leave the handing in of work until the last minute. Once an assignment has been submitted it cannot be submitted again.

Academic Misconduct: **Plagiarism** and/or **collusion** constitute **academic misconduct** under the University's Academic Regulations. Examples of academic misconduct in coursework: making available your work to other students; presenting work produced in collaboration with other students as your own (unless an explicit assessment requirement); submitting work, taken from sources that are not properly referenced, as your own. By printing and submitting this coversheet with your coursework you are confirming that the work is your own.

ESoft Office Use Only: Date stamp: work received

Kingston University, BSc (Hons) (top-up)

Coursework Cover Sheet

Part 2 – Student Feedback

Student ID:	**Student Name:**
Module Code:	**Module Name:**
Assignment number:	**ESoft Module Leader:**
Date set:	**Date due:**

Strengths (areas with well-developed answers)

Weaknesses (areas with room for improvement)

Additional Comments

ESoft Module Lecturer: **Provisional mark as %:**

ESoft Module Marker: **Date marked:**

ADVANCED DATA MODELING

COURSEWORK 01

Submitted by

Amanda Kaveesha Karunanayaka | E030087

For Advanced Data Modeling

By Mr. W A D B C Goonatillaka

On 25/02/2025

Contents

METHOD

This report adopts a streamlined approach to critically analyze and compare different data models in real-world scenarios. It examines their core principles, characteristics, merits, and limitations to arrive at a precise evaluation.

Following this, a library management system was designed using an Object-Relational Data Model and implemented in Oracle. The process involved modeling and validating the data to ensure data integrity and cohesion.

ANSWER A

A.1 Introduction

Data models are structured representations of data, used to organize and manage databases. They define the relationships between data elements and how they are stored, accessed, shared, and leveraged across organizations. Object-oriented data models, Relational data models, and Object-relational data models will be critically analyzed and compared in the document.

A.2 Relational Data Model

A.2.1 Core Principles

The Relational Data Model (RDMs) is built on a strong mathematical foundation, which provides a framework for structuring and optimizing data sets. This foundation allows the relational model to be both efficient and consistent.

The model organizes and presents data in tabular structure. These structures are known as relations. The rows represent instances of real-world entities and columns represent their unique attributes.

Further, to ensure data integrity and enable relationships between tables, each table should have a primary key to uniquely identify each row and foreign keys to establish links between different tables (Codd, 1970).

A.2.2 Characteristics

This model allows attributes to contain only atomic values and it utilizes a record-based structure where attributes must have a specific data type and fixed format.

Relationships between entities are governed by their cardinality. Cardinality represents the number of times an entity of an entity set participates in a relationship set. (geeksforgeeks, 2024)

Moreover, to ensure data integrity, ACID properties (atomicity, consistency, isolation, durability) and constraints (domain, entity integrity, referential integrity) are enforced. Furthermore, to remove inconsistencies arising from insert, delete, and update operations, a process called Normalization is used.

A.2.3 Merits and Limitations

RDMs are proven to be successful in implementing stable and clear data due to its reliability and clear structure. This model is often preferred by modern day organizations due to the widespread availability of database management systems that utilize the relational model. Further, RDMs allow concurrent and role-based user management to enhance data consistency and security.

However, RDMs find it challenging to implement complex requirements with multi-valued attributes. Such requirements lead to data inconsistencies and performance bottlenecks

A.3 Object Oriented Data Model

A.3.1 Core Principles

The foundation for Object Oriented Data Model (OODM) was created based on the principles of Object-Oriented Programming. This model stores data and their relationships in the form of an object unlike in RDMs where data is stored in tables. These objects represent real world entities, methods (behaviors), and grouped into classes based on object similarities.

OODMs utilize concepts from object-oriented programming, such as encapsulation, inheritance, and polymorphism, to model data. (Du, 2002).

A.3.2 Characteristics

The object-oriented data model allows multi valued attributes to be store. Further, this model is capable of storing nested tables and BLOBs such as images, audio and videos.

Encapsulation prevents direct access of data by external sources to ensure data integrity. Inheritance minimizes data redundancy by basing a class upon another class. Polymorphism allow objects of the same type to override methods and have customized behaviors.

OODMs promotes high maintainability due to modularity and reusability. This is achieved by breaking complex system components into smaller and manageable parts.

A.3.3 Merits and Limitations

Object oriented model is highly preferred when complex objects with large amounts of data must be stored (Leroux, Metke-Jimenez and Lawley, 2017). Data structures are represented more naturally when compared RDMs. Object oriented languages can be easily integrated alongside with OODMs to simplify application development.

However, OODMs find it hard to address simple data requirements and scenarios. Performance degrades can due to dynamic dispatch, memory allocation and garbage collection.

A.4 Object Relational Data Model

A.4.1 Core Principles

Object-relational data models are a hybrid approach that seeks to integrate the best features of both relational and object-oriented data models (Silberschatz, Korth and Sudarshan, 1996). This model integrates OOP features into tabular structures to tackle both complex and simple data structures.

A.4.2 Characteristics

This model supports traditional data types alongside with User Defined Types (UDTs). UDTs can consist of collection data types, structured data types, and large object data types.

Moreover, ORDMs utilize an extended version SQL to combine structured querying capacity with Object-Oriented features. Further, this model initiates relationships with object classes and database tables using a mechanism known as Object Relational Mapping (ORM).

A.4.3 Merits and Limitations

Due to the combination of RDMs and OODMs, both simple and complex requirements can be efficiently implemented. This combination also allows relatively better data integrity, security, consistency, and performance Availability of UDTs allow domain specific functions to easily be implemented.

However, ORDMs can be difficult to manage due to its complex structures and migration issue may arise.

A.5 Critical Analysis and Comparison

Feature	RDM	OODM	ORDM
Data Representation	Real world Entities (Relations) Rows(Instances) Columns(Attributes)	Real world Entities as Objects (contains attributes and methods)	Real world Entities (Relations) Rows(Instances) Columns(Attributes)
Relationship	Associations (Linkage via primary and foreign keys)	Association Aggregations Compositions Inheritance	Inheritance Association
Effectiveness	-Fast querying for structured data. -Complex data querying takes time.	-Complex data querying is fast	-Complex data querying is fast.
Suitability	-For application with structured data requirements	-Object-oriented Applications	-For applications seeking to combine structured data with OOP features.

Each data model possesses unique strengths and weaknesses, making them suitable for different application scenarios (Du, 2002).

In conclusion, rational Data models would be highly suitable for application with structured data requirements with well-defined relationships. Such scenarios will be guaranteed with consistent performance due to the availability of strong data rules. However, this model lacks capability to handle complex data. (Ex-: Finance and Banking systems).

Object-Oriented Data model would be highly suitable for applications with complex data structures and behaviors that aim to utilize OOP principles and languages. However, this model is not suitable to handle simple data requirements. (Ex-: Modeling software systems)

Object Relational Data models would be suitable for applications that aim to utilize structured data as well as complex data. The hybrid approach utilizing both rational and OOP features would be highly effective in maintaining flexibility and scalability. No significant downsides can be seen. (Ex-: Geographic Information system).

References

Bercich, N.H. (2003) “The Evolution of the Computerized Database,” arXiv (Cornell University) [Preprint]. Cornell University. doi:10.48550/arXiv.

Codd, E.F. (1970) “A relational model of data for large shared data banks,” Communications of the ACM. Association for Computing Machinery, p. 377. doi:10.1145/362384.362685.

Du, T.C. (2002) “Emerging Database System Architectures,” in Elsevier eBooks. Elsevier BV, p. 1. doi:10.1016/b978-012443895-8/50003-9.

Leroux, H., Metke-Jimenez, A. and Lawley, M. (2017) “Towards achieving semantic interoperability of clinical study data with FHIR,” Journal of Biomedical Semantics. BioMed Central. doi:10.1186/s13326-017-0148-7.

Silberschatz, A., Korth, H.F. and Sudarshan, S. (1996) “Data models,” ACM Computing Surveys. Association for Computing Machinery, p. 105. doi:10.1145/234313.234360.

geeksforgeeks (2024). *Cardinality in DBMS*. [online] GeeksforGeeks. Available at: https://www.geeksforgeeks.org/cardinality-in-dbms/.

ASNWER B

B.1 Introduction

The Library Management System (LMS) is an integrated solution that was modeled by clearly understanding the data requirements. The system is utilized to handle members, publications, loans, and due fines.

The system was created based on the Relational Data Model with Oracle as the Database Management System.

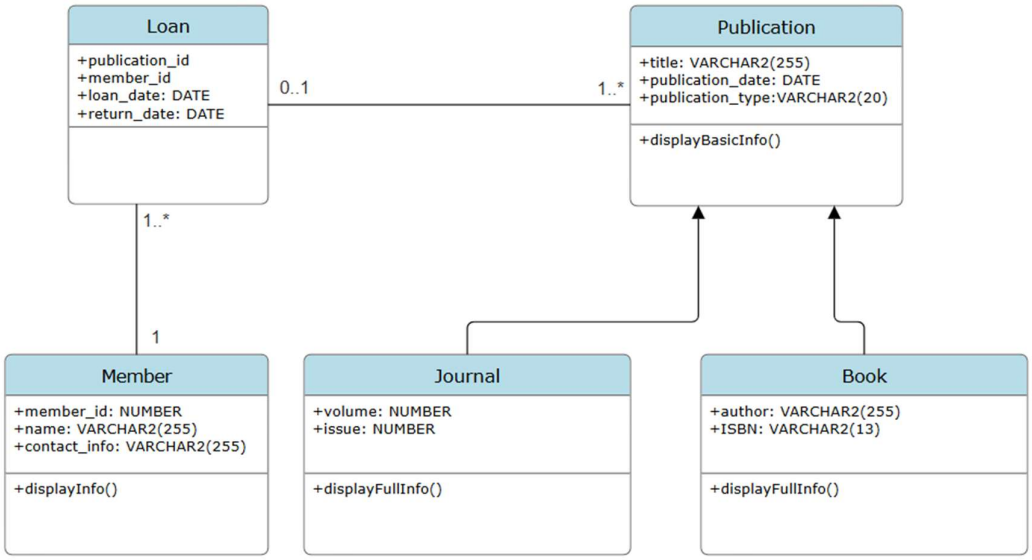


Figure 1: showcases the class diagram used to represent the Library Management System.

B.2 Object Design

B.2.1 Publication

```
SQL> -- Creation of abstract supertype for publications
SQL> CREATE OR REPLACE TYPE Publication AS OBJECT (
  2  title VARCHAR2(255),
  3  publication_date DATE,
  4  publication_type VARCHAR2(20),
  5  MEMBER FUNCTION displayBasicInfo RETURN VARCHAR2 --method to display basic info
  6  );
  7  /
```

```
SQL> -- Implementing the method "displayBasicInfo" for "Publication" type
SQL> CREATE OR REPLACE TYPE BODY Publication AS
  2  MEMBER FUNCTION displayBasicInfo RETURN VARCHAR2 IS
  3  BEGIN
  4  RETURN 'Title: ' || title || ', Publication Date: ' || TO_CHAR(publication_date, 'YYYY-MM-DD') || ', Type: ' || publication_type;
  5  END displayBasicInfo;
  6  END;
  7  /
```

```sql
-- Creation of abstract supertype for publications
CREATE OR REPLACE TYPE Publication AS OBJECT (
  title VARCHAR2(255),
  publication_date DATE,
  publication_type VARCHAR2(20),
  MEMBER FUNCTION displayBasicInfo RETURN VARCHAR2
  --method to display basic info
  );
  /

-- Implementing the method "displayBasicInfo" for "Publication" type
CREATE OR REPLACE TYPE BODY Publication AS
  MEMBER FUNCTION displayBasicInfo RETURN VARCHAR2 IS
  BEGIN
  RETURN 'Title: ' || title || ', Publication Date: '
  || TO_CHAR(publication_date, 'YYYY-MM-DD') || ', Type: '
  || publication_type;
  END displayBasicInfo;
  END;
  /
```

- The publication object type (“Publication”) is an abstract supertype that represents books and journals released by the library.
- It is declared as NOT FINAL, so that it could be further extended.

B.2.2 Book

```
SQL> -- Creation of subtype "Book" under supertype "Publication"
SQL> -- (Inheritance)
SQL> CREATE OR REPLACE TYPE Book UNDER Publication (
  2  author VARCHAR2(255),
  3  ISBN VARCHAR2(13),
  4  MEMBER FUNCTION displayFullInfo RETURN VARCHAR2 --method to display full info
  5  );
  6  /

Type created.
```

```
SQL> -- Implementing method "displayFullInfo" for "Book" type
SQL> CREATE OR REPLACE TYPE BODY Book AS
  2  MEMBER FUNCTION displayFullInfo RETURN VARCHAR2 IS
  3  BEGIN
  4   RETURN 'Title: ' || title || ', Author: ' || author || ', ISBN: ' || ISBN || ', Publication Date: ' || TO_CHAR(publication_date, 'YYYY-MM-DD');
  5  END displayFullInfo;
  6  END;
  7  /

Type body created.
```

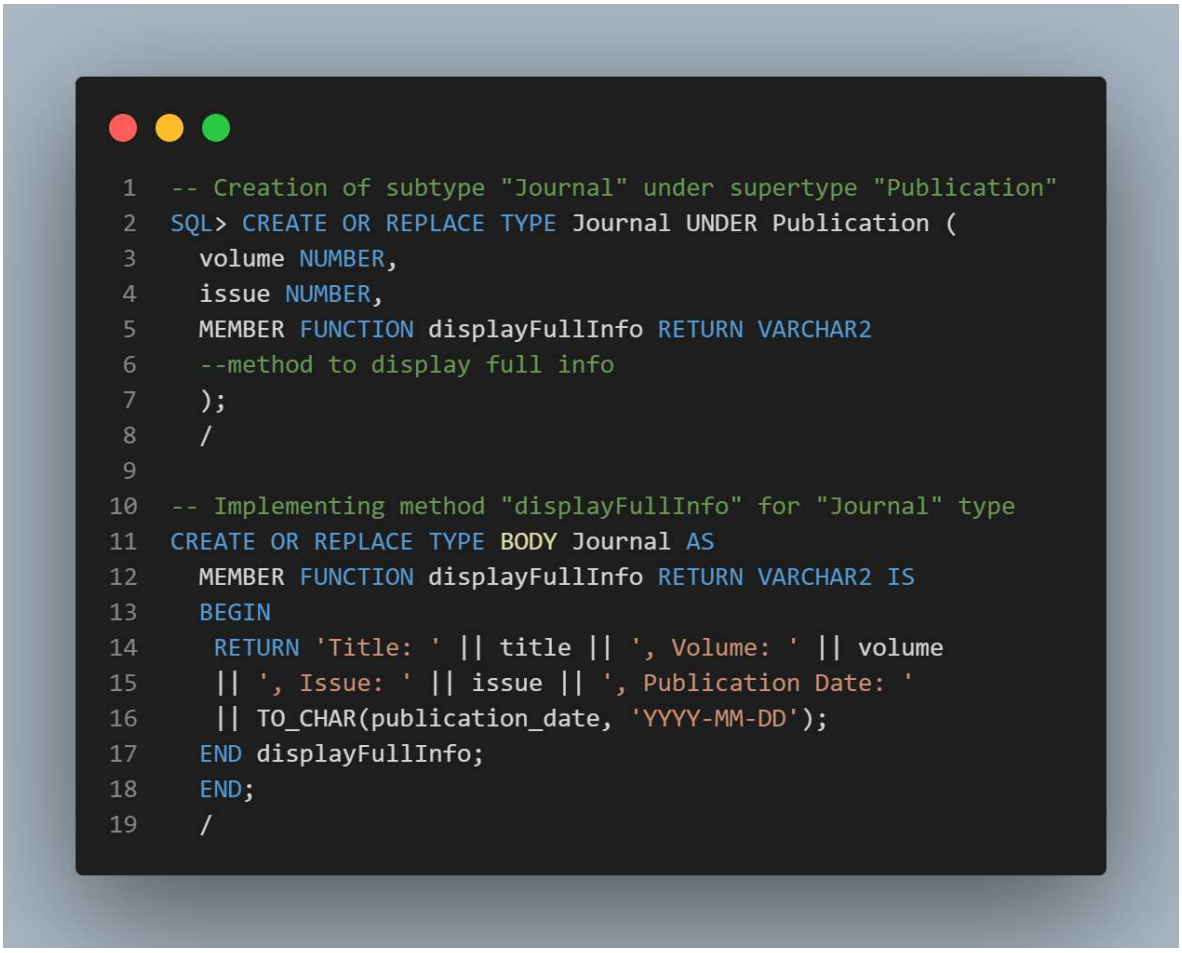
```sql
-- Creation of subtype "Book" under supertype "Publication"
-- (Inheritance)
CREATE OR REPLACE TYPE Book UNDER Publication (
  author VARCHAR2(255),
  ISBN VARCHAR2(13),
  MEMBER FUNCTION displayFullInfo RETURN VARCHAR2
  --method to display full info
  );
  /

-- Implementing method "displayFullInfo" for "Book" type
CREATE OR REPLACE TYPE BODY Book AS
  MEMBER FUNCTION displayFullInfo RETURN VARCHAR2 IS
  BEGIN
   RETURN 'Title: ' || title || ', Author: ' || author
   || ', ISBN: ' || ISBN
   || ', Publication Date: '
   || TO_CHAR(publication_date, 'YYYY-MM-DD');
  END displayFullInfo;
  END;
  /
```

- The book object type (“Book”) is a subtype that inherits attributes and methods of the publication object type (“Publication”).
- Simply, inheritance is utilized in the above scenario. It's a key object-oriented programming mechanism for code reuse. (Johnson and Foote, 1988; Holland and Lieberherr, 1996)

B.2.3 Journals

```
SQL> -- Creation of subtype "Journal" under supertype "Publication"
SQL> CREATE OR REPLACE TYPE Journal UNDER Publication (
  2  volume NUMBER,
  3  issue NUMBER,
  4  MEMBER FUNCTION displayFullInfo RETURN VARCHAR2 --method to display full info
  5  );
  6  /

Type created.
```

```
SQL> -- Implementing method "displayFullInfo" for "Journal" type
SQL> CREATE OR REPLACE TYPE BODY Journal AS
  2  MEMBER FUNCTION displayFullInfo RETURN VARCHAR2 IS
  3  BEGIN
  4   RETURN 'Title: ' || title || ', Volume: ' || volume || ', Issue: ' || issue || ', Publication Date: ' || TO_CHAR(publication_date, 'YYYY-MM-DD');
  5  END displayFullInfo;
  6  END;
  7  /

Type body created.
```



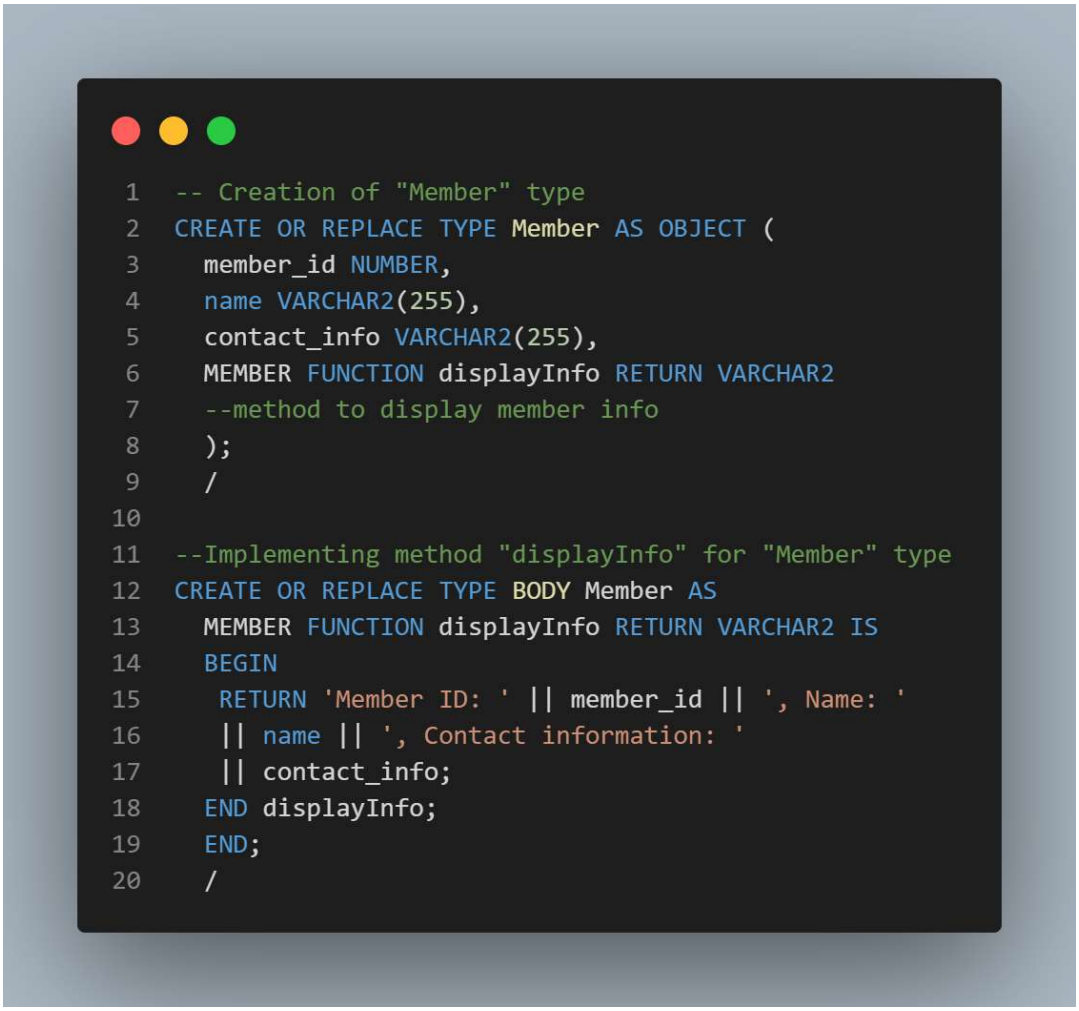
```
-- Creation of subtype "Journal" under supertype "Publication"
SQL> CREATE OR REPLACE TYPE Journal UNDER Publication (
  volume NUMBER,
  issue NUMBER,
  MEMBER FUNCTION displayFullInfo RETURN VARCHAR2
  --method to display full info
  );
  /

-- Implementing method "displayFullInfo" for "Journal" type
CREATE OR REPLACE TYPE BODY Journal AS
  MEMBER FUNCTION displayFullInfo RETURN VARCHAR2 IS
  BEGIN
   RETURN 'Title: ' || title || ', Volume: ' || volume
   || ', Issue: ' || issue || ', Publication Date: '
   || TO_CHAR(publication_date, 'YYYY-MM-DD');
  END displayFullInfo;
  END;
  /
```

- The Journal object type (“Journal”) is a subtype that inherits attributes and methods of the publication object type (“Publication”).

B.2.4 Member

```
SQL> -- Creation of "Member" type
SQL> CREATE OR REPLACE TYPE Member AS OBJECT (
  2  member_id NUMBER,
  3  name VARCHAR2(255),
  4  contact_info VARCHAR2(255),
  5  MEMBER FUNCTION displayInfo RETURN VARCHAR2 --method to display member info
  6  );
  7  /
```

```
SQL> --Implementing method "displayInfo" for "Member" type
SQL> CREATE OR REPLACE TYPE BODY Member AS
  2  MEMBER FUNCTION displayInfo RETURN VARCHAR2 IS
  3  BEGIN
  4    RETURN 'Member ID: ' || member_id || ', Name: ' || name || ', Contact information: ' || contact_info;
  5  END displayInfo;
  6  END;
  7  /
```



```
-- Creation of "Member" type
CREATE OR REPLACE TYPE Member AS OBJECT (
  member_id NUMBER,
  name VARCHAR2(255),
  contact_info VARCHAR2(255),
  MEMBER FUNCTION displayInfo RETURN VARCHAR2
  --method to display member info
  );
  /

--Implementing method "displayInfo" for "Member" type
CREATE OR REPLACE TYPE BODY Member AS
  MEMBER FUNCTION displayInfo RETURN VARCHAR2 IS
  BEGIN
   RETURN 'Member ID: ' || member_id || ', Name: '
   || name || ', Contact information: '
   || contact_info;
  END displayInfo;
  END;
  /
```

- Member object type (“Member”) represent members of the library and their info.

B.3 Table Design

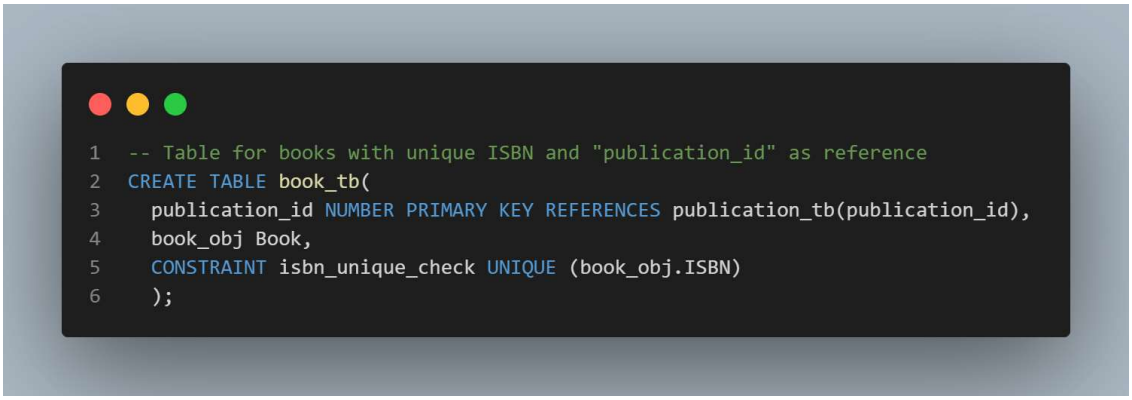
The table design of the LMS utilizes normal relational tables with object columns. However, Member table (“member_tb”) is an exception as it is made using the object table-based approach.

B.3.1 Publication

```
SQL> -- Table For publications with "Publication_ID" as primary key
SQL> CREATE TABLE publication_tb(
  2  publication_id NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
  3  publication_obj Publication
  4  );

Table created.
```

B.3.2 Book

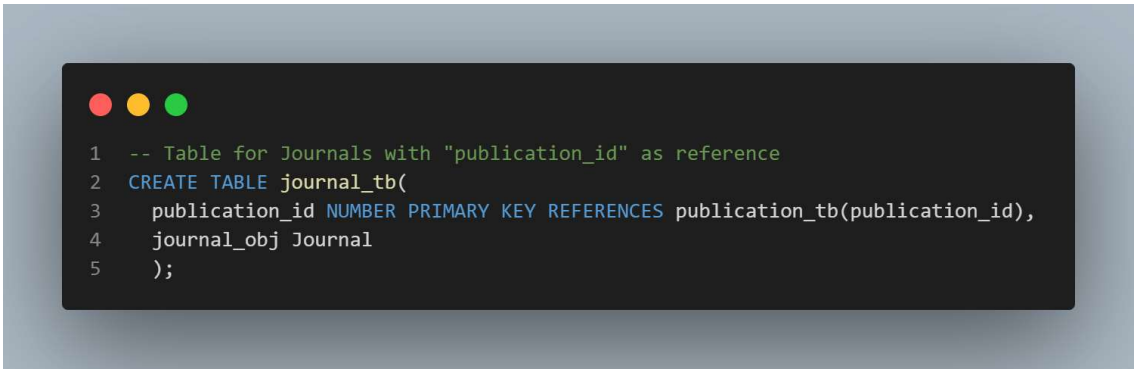
```
SQL> -- Table for books with unique ISBN and "publication_id" as reference
SQL> CREATE TABLE book_tb(
  2  publication_id NUMBER PRIMARY KEY REFERENCES publication_tb(publication_id),
  3  book_obj Book,
  4  CONSTRAINT isbn_unique_check UNIQUE (book_obj.ISBN)
  5  );

Table created.
```



```sql
-- Table for books with unique ISBN and "publication_id" as reference
CREATE TABLE book_tb(
  publication_id NUMBER PRIMARY KEY REFERENCES publication_tb(publication_id),
  book_obj Book,
  CONSTRAINT isbn_unique_check UNIQUE (book_obj.ISBN)
  );
```

B.3.3 Journal

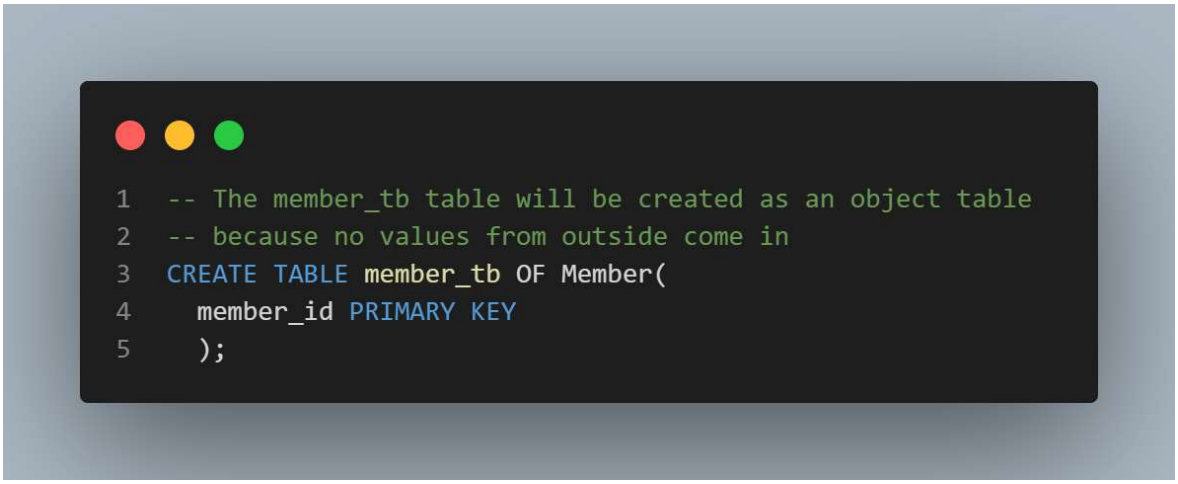
```
SQL> -- Table for Journals with "publication_id" as reference
SQL> CREATE TABLE journal_tb(
  2  publication_id NUMBER PRIMARY KEY REFERENCES publication_tb(publication_id),
  3  journal_obj Journal
  4  );

Table created.
```



```sql
-- Table for Journals with "publication_id" as reference
CREATE TABLE journal_tb(
  publication_id NUMBER PRIMARY KEY REFERENCES publication_tb(publication_id),
  journal_obj Journal
  );
```

B.3.4 Member

```
SQL> -- The member_tb table will be created as an object table because no values from outside come in
SQL> CREATE TABLE member_tb OF Member(
  2  member_id PRIMARY KEY
  3  );

Table created.
```



B.3.5 Loan

```
SQL> -- Loan table, will be created as a linking table
SQL> CREATE TABLE loan_tb(
  2  publication_id NUMBER,
  3  member_id NUMBER,
  4  loan_date DATE,
  5  return_date DATE,
  6  CONSTRAINT loan_pk PRIMARY KEY (member_id, publication_id, loan_date),
  7  CONSTRAINT fk_loan_member FOREIGN KEY (member_id) REFERENCES member_tb (member_id),
  8  CONSTRAINT fk_loan_publication FOREIGN KEY (publication_id) REFERENCES publication_tb (publication_id)
  9  );

Table created.
```

To Make sure that the loan date "loan_date" is not in the future, a trigger was activated.

```
SQL> CREATE OR REPLACE TRIGGER check_loan_date
  2  BEFORE INSERT OR UPDATE ON loan_tb
  3  FOR EACH ROW
  4  BEGIN
  5  IF :NEW.loan_date > SYSDATE THEN
  6  RAISE_APPLICATION_ERROR(-20001, 'Loan date does not exist');
  7  END IF;
  8  END;
  9  /

Trigger created.
```

```
-- Loan table, will be created as a linking table
CREATE TABLE loan_tb(
  publication_id NUMBER,
  member_id NUMBER,
  loan_date DATE,
  return_date DATE,
  CONSTRAINT loan_pk PRIMARY KEY (member_id, publication_id, loan_date),
  CONSTRAINT fk_loan_member FOREIGN KEY (member_id) REFERENCES member_tb (member_id),
  CONSTRAINT fk_loan_publication FOREIGN KEY (publication_id) REFERENCES publication_tb (publication_id)
  );

-- TRIGGER CREATION --
CREATE OR REPLACE TRIGGER check_loan_date
  BEFORE INSERT OR UPDATE ON loan_tb
  FOR EACH ROW
  BEGIN
  IF :NEW.loan_date > SYSDATE THEN
  RAISE_APPLICATION_ERROR(-20001, 'Loan date does not exist');
  END IF;
  END;
  /
```

B.4 Testing

B.4.1 Insertion of values for testing

B.4.1.1 Publications

```
SQL> -- SAMPLE INSERTION --
SQL> -- PUBLICATION SAMPLES --
SQL> INSERT INTO publication_tb (publication_obj) VALUES (Publication('The Power of Positive Thinking', TO_DATE('1952-10-01', 'YYYY-MM-DD'), 'Book'));

1 row created.
```

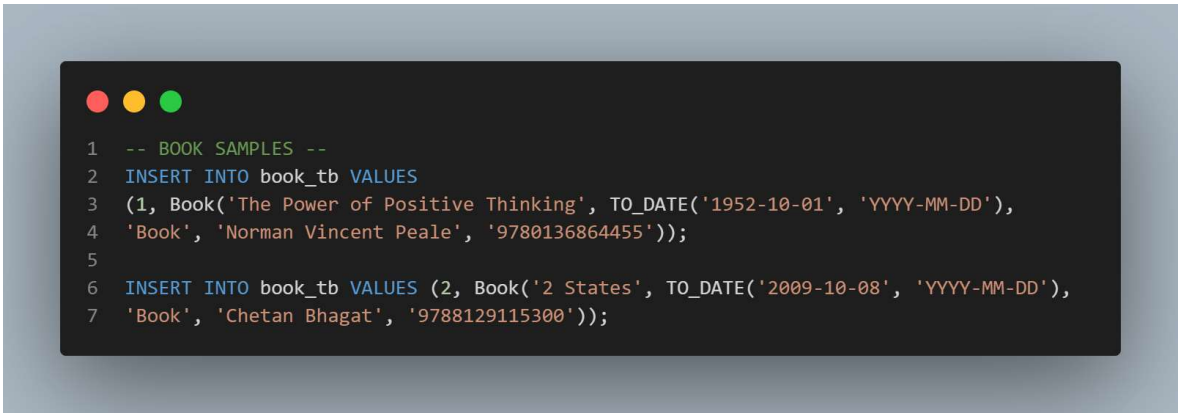
```
SQL> INSERT INTO publication_tb (publication_obj) VALUES (Publication('2 States', TO_DATE('2009-10-08', 'YYYY-MM-DD'), 'Book'));

1 row created.
```

```
SQL> INSERT INTO publication_tb (publication_obj) VALUES (Publication('A relational model of data for large shared data banks', TO_DATE('1970-06-01', 'YYYY-MM-DD'), 'Journal'));

1 row created.

SQL> INSERT INTO publication_tb (publication_obj) VALUES (Publication('Data Science Monthly', TO_DATE('2023-03-10', 'YYYY-MM-DD'), 'Journal'));

1 row created.
```

```sql
-- SAMPLE INSERTION --
-- PUBLICATION SAMPLES --

--BOOKS--
INSERT INTO publication_tb (publication_obj) VALUES
(Publication('The Power of Positive Thinking',
TO_DATE('1952-10-01', 'YYYY-MM-DD'), 'Book'));

INSERT INTO publication_tb (publication_obj) VALUES
(Publication('2 States',
TO_DATE('2009-10-08', 'YYYY-MM-DD'), 'Book'));

--JOURNALS--
INSERT INTO publication_tb (publication_obj) VALUES
(Publication('A relational model of data for large shared data banks', T
O_DATE('1970-06-01', 'YYYY-MM-DD'), 'Journal'));

INSERT INTO publication_tb (publication_obj) VALUES
(Publication('Data Science Monthly',
TO_DATE('2023-03-10', 'YYYY-MM-DD'), 'Journal'));
```

B.4.1.2 Books

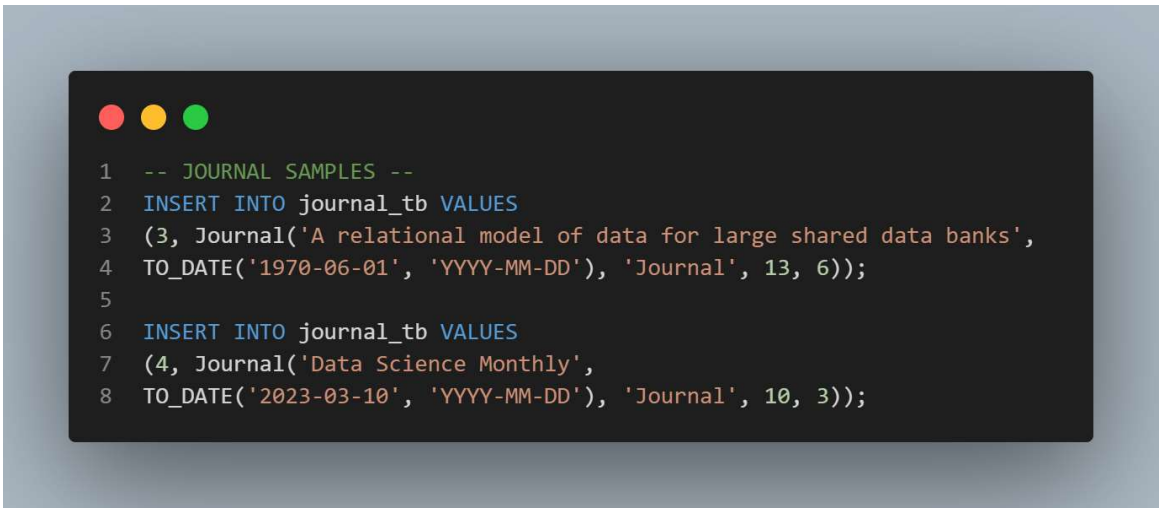
```
SQL> -- BOOK SAMPLES --
SQL> INSERT INTO book_tb VALUES (1, Book('The Power of Positive Thinking', TO_DATE('1952-10-01', 'YYYY-MM-DD'), 'Book', 'Norman Vincent Peale', '9780136864455'));

1 row created.

SQL> INSERT INTO book_tb VALUES (2, Book('2 States', TO_DATE('2009-10-08', 'YYYY-MM-DD'), 'Book', 'Chetan Bhagat', '9788129115300'));

1 row created.
```



```sql
-- BOOK SAMPLES --
INSERT INTO book_tb VALUES
(1, Book('The Power of Positive Thinking', TO_DATE('1952-10-01', 'YYYY-MM-DD'),
'Book', 'Norman Vincent Peale', '9780136864455'));

INSERT INTO book_tb VALUES (2, Book('2 States', TO_DATE('2009-10-08', 'YYYY-MM-DD'),
'Book', 'Chetan Bhagat', '9788129115300'));
```

B.4.1.3 Journals

```
SQL> -- JOURNAL SAMPLES --
SQL> INSERT INTO journal_tb VALUES (3, Journal('A relational model of data for large shared data banks', TO_DATE('1970-06-01', 'YYYY-MM-DD'), 'Journal', 13, 6));

1 row created.

SQL> INSERT INTO journal_tb VALUES (4, Journal('Data Science Monthly', TO_DATE('2023-03-10', 'YYYY-MM-DD'), 'Journal', 10, 3));

1 row created.
```



```sql
-- JOURNAL SAMPLES --
INSERT INTO journal_tb VALUES
(3, Journal('A relational model of data for large shared data banks',
TO_DATE('1970-06-01', 'YYYY-MM-DD'), 'Journal', 13, 6));

INSERT INTO journal_tb VALUES
(4, Journal('Data Science Monthly',
TO_DATE('2023-03-10', 'YYYY-MM-DD'), 'Journal', 10, 3));
```

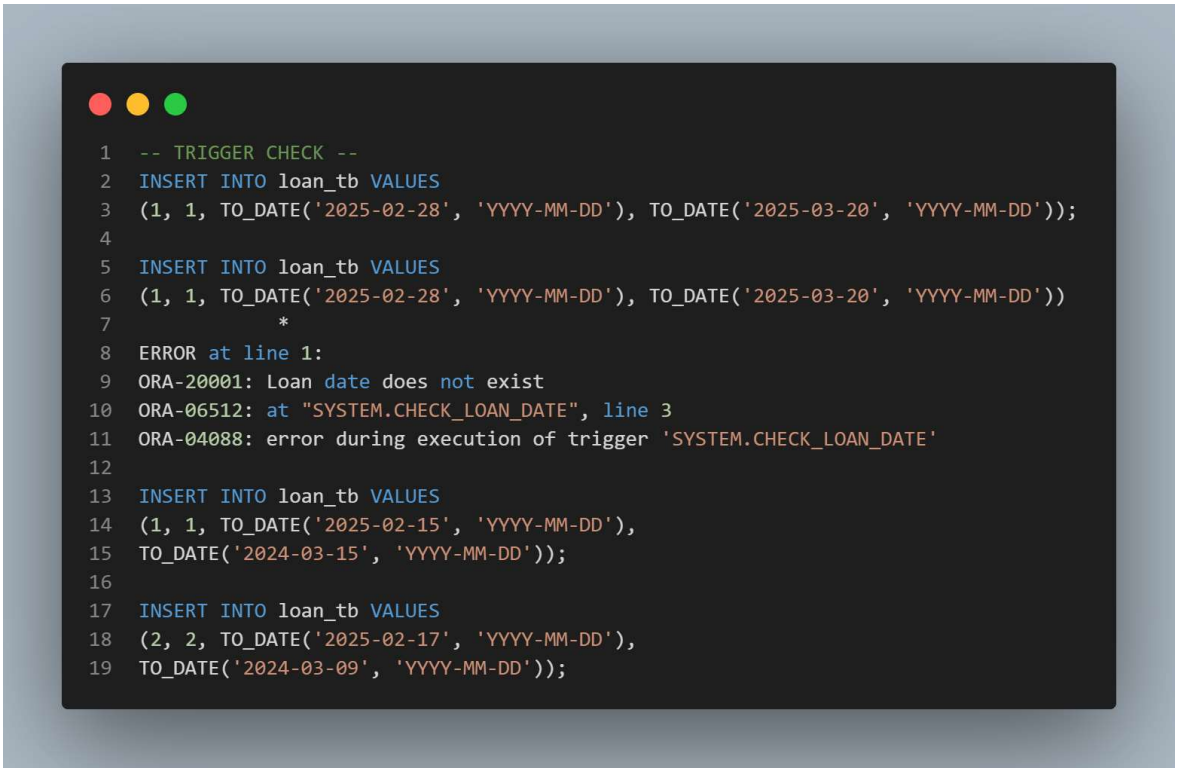
B.4.1.4 Member

```
SQL> --  MEMBER SAMPLES --
SQL> INSERT INTO member_tb VALUES (Member(1, 'Amanda Kaveesha', 'amanda.kaveesha@gmail.com'));

1 row created.

SQL> INSERT INTO member_tb VALUES (Member(2, 'Sarath Gunasekara', 'sarath123@outlook.com'));

1 row created.
```

B.4.1.5 Loan

```
SQL> INSERT INTO loan_tb VALUES (1, 1, TO_DATE('2025-02-15', 'YYYY-MM-DD'), TO_DATE('2024-03-15', 'YYYY-MM-DD'));

1 row created.

SQL> INSERT INTO loan_tb VALUES (2, 2, TO_DATE('2025-02-17', 'YYYY-MM-DD'), TO_DATE('2024-03-09', 'YYYY-MM-DD'));

1 row created.
```

```
SQL> -- TRIGGER CHECK --
SQL> INSERT INTO loan_tb VALUES (1, 1, TO_DATE('2025-02-28', 'YYYY-MM-DD'), TO_DATE('2025-03-20', 'YYYY-MM-DD'));
INSERT INTO loan_tb VALUES (1, 1, TO_DATE('2025-02-28', 'YYYY-MM-DD'), TO_DATE('2025-03-20', 'YYYY-MM-DD'))
            *
ERROR at line 1:
ORA-20001: Loan date does not exist
ORA-06512: at "SYSTEM.CHECK_LOAN_DATE", line 3
ORA-04088: error during execution of trigger 'SYSTEM.CHECK_LOAN_DATE'
```



```
1  -- TRIGGER CHECK --
2  INSERT INTO loan_tb VALUES
3  (1, 1, TO_DATE('2025-02-28', 'YYYY-MM-DD'), TO_DATE('2025-03-20', 'YYYY-MM-DD'));
4
5  INSERT INTO loan_tb VALUES
6  (1, 1, TO_DATE('2025-02-28', 'YYYY-MM-DD'), TO_DATE('2025-03-20', 'YYYY-MM-DD'))
7              *
8  ERROR at line 1:
9  ORA-20001: Loan date does not exist
10 ORA-06512: at "SYSTEM.CHECK_LOAN_DATE", line 3
11 ORA-04088: error during execution of trigger 'SYSTEM.CHECK_LOAN_DATE'
12
13 INSERT INTO loan_tb VALUES
14 (1, 1, TO_DATE('2025-02-15', 'YYYY-MM-DD'),
15 TO_DATE('2024-03-15', 'YYYY-MM-DD'));
16
17 INSERT INTO loan_tb VALUES
18 (2, 2, TO_DATE('2025-02-17', 'YYYY-MM-DD'),
19 TO_DATE('2024-03-09', 'YYYY-MM-DD'));
```

Figure 2: Shows how "days of the future" are not inserted. This is because of the trigger function that was implemented in the table design phase.

B.4.2 Test Cases

B.4.2.1 Publication Test Cases

B.4.2.1.1 Insert Test Case

```
SQL> SELECT * FROM publication_tb;

PUBLICATION_ID
--------------
PUBLICATION_OBJ(TITLE, PUBLICATION_DATE, PUBLICATION_TYPE)
--------------------------------------------------------------------------------
             1
PUBLICATION('The Power of Positive Thinking', '01-OCT-52', 'Book')

             2
PUBLICATION('2 States', '08-OCT-09', 'Book')

             3
PUBLICATION('A relational model of data for large shared data banks', '01-JUN-70
', 'Journal')

PUBLICATION_ID
--------------
PUBLICATION_OBJ(TITLE, PUBLICATION_DATE, PUBLICATION_TYPE)
--------------------------------------------------------------------------------

             4
PUBLICATION('Data Science Monthly', '10-MAR-23', 'Journal')
```

Test case	Description	Expected result	Actual result	Status
1.1	Data insertion to publication table.	Data being inserted	Data was inserted.	Pass

B.4.2.1.2 Method Execution Test Case

```
SQL> -- Invoking Method for publication --
SQL> DECLARE
  2  pub Publication;
  3  result VARCHAR2(4000);
  4  BEGIN
  5  SELECT publication_obj INTO pub FROM publication_tb WHERE publication_id = 1;
  6  result := pub.displayBasicInfo();
  7  DBMS_OUTPUT.PUT_LINE(result);
  8  END;
  9  /
Title: The Power of Positive Thinking, Publication Date: 1952-10-01, Type: Book

PL/SQL procedure successfully completed.
```

Test case	Description	Expected result	Actual result	Status
1.2	Executing displayBasicInfo() Method	Output with basic info	Information was outputted	Pass

B.4.2.2 Book Test Cases

B.4.2.2.1 Insert Test Case

```
SQL> SELECT * FROM book_tb;

PUBLICATION_ID
--------------
BOOK_OBJ(TITLE, PUBLICATION_DATE, PUBLICATION_TYPE, AUTHOR, ISBN)
--------------------------------------------------------------------------------
             1
BOOK('The Power of Positive Thinking', '01-OCT-52', 'Book', 'Norman Vincent Peal
e', '9780136864455')

             2
BOOK('2 States', '08-OCT-09', 'Book', 'Chetan Bhagat', '9788129115300')
```

Test case	Description	Expected result	Actual result	Status
1.3	Data insertion to books table.	Data being inserted	Data was inserted.	Pass

B.4.2.2.2 Method Execution Test Case

```
SQL> -- Invking Method of Book type --
SQL> DECLARE
  2  b_ks Book;
  3  result VARCHAR2(4000);
  4  BEGIN
  5  SELECT book_obj INTO b_ks FROM book_tb WHERE publication_id = 1;
  6  result := b_ks.displayFullInfo();
  7  DBMS_OUTPUT.PUT_LINE(result);
  8  END;
  9  /
Title: The Power of Positive Thinking, Author: Norman Vincent Peale, ISBN:
9780136864455, Publication Date: 1952-10-01

PL/SQL procedure successfully completed.
```

Test case	Description	Expected result	Actual result	Status
1.4	Executing displayFullInfo() Method in books type	Output with complete info	Information was outputted	Pass

B.4.2.3 Journal Test Cases

B.4.2.3.1 Insert Test Case

```
SQL>  SELECT * FROM journal_tb;

PUBLICATION_ID
--------------
JOURNAL_OBJ(TITLE, PUBLICATION_DATE, PUBLICATION_TYPE, VOLUME, ISSUE)
--------------------------------------------------------------------------------
             3
JOURNAL('A relational model of data for large shared data banks', '01-JUN-70', '
Journal', 13, 6)

             4
JOURNAL('Data Science Monthly', '10-MAR-23', 'Journal', 10, 3)
```

Test case	Description	Expected result	Actual result	Status
1.5	Data insertion to journals table.	Data being inserted	Data was inserted.	Pass

B.4.2.3.2 Method Execution Test Case

```
SQL> -- Invoking Method of Journal type --
SQL> DECLARE
  2  journ_l Journal;
  3  result VARCHAR2(4000);
  4  BEGIN
  5  SELECT journal_obj INTO journ_l FROM journal_tb WHERE publication_id = 3;
  6  result := journ_l.displayFullInfo();
  7  DBMS_OUTPUT.PUT_LINE(result);
  8  END;
  9  /
Title: A relational model of data for large shared data banks, Volume: 13,
Issue: 6, Publication Date: 1970-06-01

PL/SQL procedure successfully completed.
```

Test case	Description	Expected result	Actual result	Status
1.6	Executing displayFullInfo() Method in Journals type	Output with complete info	Information was outputted	Pass

B.4.2.4 Member Test Cases

B.4.2.4.1 Insert Test Case

```
SQL> SELECT * FROM member_tb;

 MEMBER_ID
----------
NAME
--------------------------------------------------------------------------------
CONTACT_INFO
--------------------------------------------------------------------------------
         1
Amanda Kaveesha
amanda.kaveesha@gmail.com

         2
Sarath Gunasekara
sarath123@outlook.com

 MEMBER_ID
----------
NAME
--------------------------------------------------------------------------------
CONTACT_INFO
--------------------------------------------------------------------------------
```

Test case	Description	Expected result	Actual result	Status
1.7	Data insertion to members table.	Data being inserted	Data was inserted.	Pass

B.4.2.4.2 Method Execution Test Case

```
SQL> -- Invoking Method of Member Type --
SQL> DECLARE
  2  mem_b Member;
  3  result VARCHAR2(4000);
  4  BEGIN
  5  SELECT VALUE(m) INTO mem_b FROM member_tb m WHERE member_id = 1;
  6  result := mem_b.displayInfo();
  7  DBMS_OUTPUT.PUT_LINE(result);
  8  END;
  9  /
Member ID: 1, Name: Amanda Kaveesha, Contact information:
amanda.kaveesha@gmail.com

PL/SQL procedure successfully completed.
```

Test case	Description	Expected result	Actual result	Status
1.8	Executing displayInfo() Method in Journals type	Output with complete info	Information was outputted	Pass

B.4.2.5 Loan Test Cases

B.4.2.5.1 Insert Test Case

```
SQL> INSERT INTO loan_tb VALUES (1, 1, TO_DATE('2025-02-15', 'YYYY-MM-DD'), TO_DATE('2024-03-15', 'YYYY-MM-DD'));

1 row created.

SQL> INSERT INTO loan_tb VALUES (2, 2, TO_DATE('2025-02-17', 'YYYY-MM-DD'), TO_DATE('2024-03-09', 'YYYY-MM-DD'));

1 row created.
```

Test case	Description	Expected result	Actual result	Status
1.9	Data insertion to loans table.	Data being inserted	Data was inserted.	Pass

B.4.2.6 Functionality Test Cases

B.4.2.6.1 Display of all publications sorted by title

Functionality Implementation

```
SQL> -- CODE TO DISPLAY  ALL PUBLICATIONS SORTED BY TITLE --
SQL> SELECT p.publication_id, p.publication_obj.title, p.publication_obj.publication_date, p.publication_obj.publication_type
  2  FROM publication_tb p
  3  ORDER BY p.publication_obj.title;
```

```
SELECT p.publication_id, p.publication_obj.title,
p.publication_obj.publication_date, p.publication_obj.publication_type
FROM publication_tb p
ORDER BY p.publication_obj.title;
```

Result

```
PUBLICATION_ID
--------------
PUBLICATION_OBJ.TITLE
--------------------------------------------------------------------------------
PUBLICATI PUBLICATION_OBJ.PUBL
--------- --------------------
               2
2 States
08-OCT-09 Book

               3
A relational model of data for large shared data banks
01-JUN-70 Journal

PUBLICATION_ID
--------------
PUBLICATION_OBJ.TITLE
--------------------------------------------------------------------------------
PUBLICATI PUBLICATION_OBJ.PUBL
--------- --------------------

               4
Data Science Monthly
10-MAR-23 Journal

               1
The Power of Positive Thinking

PUBLICATION_ID
--------------
PUBLICATION_OBJ.TITLE
--------------------------------------------------------------------------------
PUBLICATI PUBLICATION_OBJ.PUBL
--------- --------------------
01-OCT-52 Book
```

Test case	Description	Expected result	Actual result	Status
2.0	Function implemented to display publications sorted by title	Publications sorted by title	Publications sorted by title	Pass

B.4.2.6.2 Display of all borrowed publications by a specific member

Functionality Implementation

```
SQL> -- CODE TO DISPLAY ALL BORROWED PUBLICATION BY A SPECIFIC MEMEBER --
SQL> SELECT l.member_id, m.name, p.publication_obj.title, l.loan_date, l.return_date
  2  FROM loan_tb l
  3  JOIN member_tb m ON l.member_id = m.member_id
  4  JOIN publication_tb p ON l.publication_id = p.publication_id
  5  WHERE l.member_id = 1;
```

Result

```
 MEMBER_ID
----------
NAME
--------------------------------------------------------------------------------
PUBLICATION_OBJ.TITLE
--------------------------------------------------------------------------------
LOAN_DATE RETURN_DA
--------- ---------
         1
Amanda Kaveesha
The Power of Positive Thinking
15-FEB-25 15-MAR-24
```

Test case	Description	Expected result	Actual result	Status
2.1	Function implemented to display all borrowed publications by one member	All borrowed publications by one member	All borrowed publications by one member	Pass

B.4.2.6.3 A simple search functionality based on title

Functionality Implementation

```
SQL> CREATE OR REPLACE PROCEDURE search_books_author(author_name IN VARCHAR2)
  2  IS
  3  BEGIN
  4  FOR book_rec IN(
  5  SELECT bt.book_obj
  6  FROM book_tb bt
  7  WHERE bt.book_obj.author = author_name
  8  )
  9  LOOP
 10  DBMS_OUTPUT.PUT_LINE('Title: ' || book_rec.book_obj.title || ', Author: ' || book_rec.book_obj.author);
 11  END LOOP;
 12  END;
 13  /

Procedure created.

SQL> SET SERVEROUTPUT ON;
SQL> BEGIN
  2  search_books_author('Chetan Bhagat');
  3  END;
  4  /
```

```
1  -- SIMPLE function to search for publications by author --
2  CREATE OR REPLACE PROCEDURE search_books_author(author_name IN VARCHAR2)
3    IS
4    BEGIN
5    FOR book_rec IN(
6    SELECT bt.book_obj
7    FROM book_tb bt
8    WHERE bt.book_obj.author = author_name
9    )
10   LOOP
11   DBMS_OUTPUT.PUT_LINE('Title: ' || book_rec.book_obj.title || ', Author: ' || book_rec.book_obj.author);
12   END LOOP;
13   END;
14   /
15
16 SET SERVEROUTPUT ON;
17 BEGIN
18   search_books_author('Chetan Bhagat');
19   END;
20   /
```

Result

```
Title: 2 States, Author: Chetan Bhagat

PL/SQL procedure successfully completed.
```

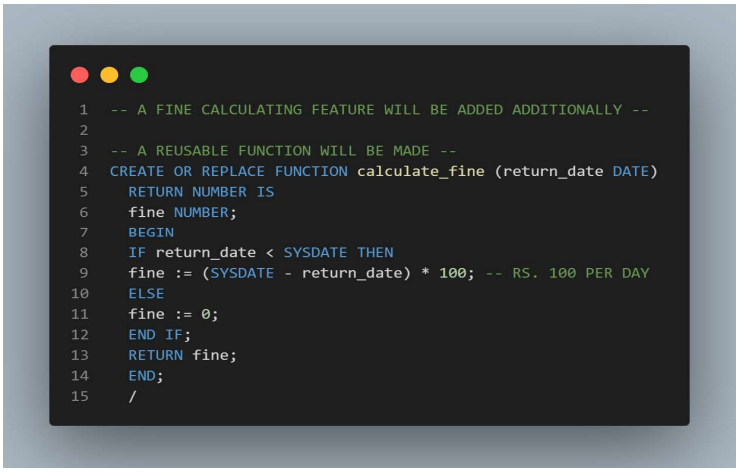
Test case	Description	Expected result	Actual result	Status
2.2	Function to search for publications(book) by author.	The author's relevant publications(books)	The author's relevant publications(books)	Pass

B.5 Additional Features

B.5.1 Functionality to calculate Fine amount

Functionality Implementation

```
SQL> -- A FINE CALCULATING FEATURE WILL BE ADDED ADDITIONALLY --
SQL>
SQL> -- A REUSABLE FUNCTION WILL BE MADE --
SQL> CREATE OR REPLACE FUNCTION calculate_fine (return_date DATE)
  2  RETURN NUMBER IS
  3  fine NUMBER;
  4  BEGIN
  5  IF return_date < SYSDATE THEN
  6  fine := (SYSDATE - return_date) * 100; -- RS. 100 PER DAY
  7  ELSE
  8  fine := 0;
  9  END IF;
 10  RETURN fine;
 11  END;
 12  /

Function created.
```



```
-- A FINE CALCULATING FEATURE WILL BE ADDED ADDITIONALLY --

-- A REUSABLE FUNCTION WILL BE MADE --
CREATE OR REPLACE FUNCTION calculate_fine (return_date DATE)
  RETURN NUMBER IS
  fine NUMBER;
  BEGIN
  IF return_date < SYSDATE THEN
  fine := (SYSDATE - return_date) * 100; -- RS. 100 PER DAY
  ELSE
  fine := 0;
  END IF;
  RETURN fine;
  END;
  /
```

Result

```
SQL> -- fine calc --
SQL> SELECT l.member_id, m.name, p.publication_obj.title, l.loan_date, l.return_date,
  2  calculate_fine(l.return_date) AS fine_total
  3  FROM loan_tb l
  4  JOIN member_tb m ON l.member_id = m.member_id
  5  JOIN publication_tb p ON l.publication_id = p.publication_id
  6  WHERE calculate_fine(l.return_date) > 0;

 MEMBER_ID
----------
NAME
--------------------------------------------------------------------------------
PUBLICATION_OBJ.TITLE
--------------------------------------------------------------------------------
LOAN_DATE RETURN_DA FINE_TOTAL
--------- --------- ----------
         1
Amanda Kaveesha
The Power of Positive Thinking
15-FEB-25 23-FEB-25  226.09375
```

Test case	Description	Expected result	Actual result	Status
2.3	Function implemented to calculate fine if return date of a publications is over.	Calculation of the relevant fine amount	Calculation of the relevant fine amount	Pass

Conclusion

The system covers essential aspects of a library management system and functionalities are utilized to deliver enhanced performance. Further, Reusability is also promoted in this system.

As a result, various aspects of object relational data modeling were covered and explored, leading to a comprehensive understanding of the concept.

To sum up, through section A, core principles, characteristic, merits, and limitations of 3 major data models were identified. This allowed to gain a structured and deep understanding about data modeling.

Section B provided practical understand on Object Relational Databases. This offered hands-on experience with different database concepts and technologies.

References

Holland, I.M. and Lieberherr, K. (1996) “Object-oriented design,” ACM Computing Surveys. Association for Computing Machinery, p. 273. doi:10.1145/234313.234421.
Johnson, R.E. and Foote, B. (1988) “Designing reusable classes,” p. 22.