PROJECT REPORT
FOR
WONDER LIBRARY
DATABASE DESIGN

CS 6360.004

Spring 2019

Group 13

Maleeha Koul

Shruti Agrawal

Prof. Dr. Wei Li Wu

Project Description

Wonder Library is a library for all ages. Wonder Library would like one relational database to be able to smoothly carry out their work in an organized way. The library has following modules: Person, Employee, Member, Books, Borrow details and Payment.

A Person can be an Employee or a Silver member. Details of a person such as Person ID, Name (First, Middle, Last), Address, Gender, Date of Birth (must be in year 2001 or before), and Phone number (one person can have more than one phone number) are recorded. The Person ID should have the format “PXXX” where X is a number from 0 to 9. A person with a silver membership has certain limited privileges. A person can be both an employee and a Silver member.

Each Person is issued a library card. The library card details such as card ID, date of issue, membership level and other information are stored.

Employee is further classified as Library Supervisors, Cataloging Managers or Receptionists. The start date of the employee is recorded.

A Gold member is someone who has extra privileges than a Silver member. A Gold Member can be an Employee or a Silver Member or both. Date of membership is recorded for both type of members.

A Guest log is maintained for the Gold members, which stores information such as member ID, guest ID, guest name, guest address, and guest contact information. There are temporary IDs that a person gets when they visit as a guest of a Gold member. Each guest ID is not unique and cannot be used to identify a guest in the library.

Books details such as book ID, book title and other information are stored. Books are classified as Class 1 books or Class 2 books. Silver Members can borrow up to 2 Class 1 books in a week and Gold members can borrow up to 5 Class 1 books in a week. Additionally Gold members can borrow 3 Class 2 books in a week. The cataloging manager has access to the book details.

A book is published by a publisher. A publisher can publish more than one books but a book is assumed to be published by a single publisher. The publisher details such as publisher ID and publisher name and other information are stored.

Author details such as author ID, author name and other information is stored. One book can have multiple authors and one author can write more than one book.

A receptionist maintains records. Records contain record ID and other relevant information.

Borrowed book details are stored containing information about the book borrowed, the date of issue and date of return and the details about the person borrowing the book and the record ID. Borrowed details are stored only when a person borrows a book. Information about penalty payment is also stored with the details of the book, the person who borrowed and could not return within the due period. The receptionist also tracks this late fee payment records.

Project Questions

1. Is the ability to model superclass/subclass relationships likely to be important in a library environment such as Wonder Library? Why or why not?

Yes, it helps us classify the information pertaining to a specific role played by an entity in a hierarchical manner. It helps us refine the number of attributes for each of the sub-classes and allows disjoint and overlapping properties between these entities. It also helps us to get rid of repetitive attributes and redundant information. It is a more concise way of describing the workflow in a library environment.

2. Can you think of 5 more business rules (other than the one explicitly described above) that are likely to be used in a library environment? Add your rules to the above requirement to be implemented.

1. For each unique identifier in a table like Library Card ID, Gold Member ID etc., there should be a convention to be followed like "{GM}-{X}-{X}-{X}" .
2. We can enforce benefits such as if a member has been a regular patron for a year, then he gets the extra days for each book that he has issued.
3. We can have more types of books and media available for issue such as newspaper archives, audio books etc.
4. If any of the guests of the gold member joins the library as a silver member, then the gold member gets some referral benefits.
5. Add some constraints (responsibilities) to the library supervisor such as lost and found.

3. Justify using a Relational DBMS like Oracle for this project.

Data is inherently better suited to a relational database which can host several databases on one server. Other advantages include:

• Ease of Use

• Data Security

• SQL Standard

• Data Integrity

• Performance

• Development and Support

Oracle implementation was an efficient database design.

Logical Design

Person

FirstName	MiddleName	LastName	PersonID	DateOfBirth	Address

Primary key: PersonID

Phone#

PersonID	Phone#

Primary key: PersonID

Foreign key: PersonID (from Person relation)

Employee

<u>PersonID</u>	Emp_start_date	GoldMemberID	SupervisorFlag	ReceptionistFlag	RecordID	CatalogManagerFlag	BookID

Primary key: PersonID

Foreign key: PersonID, GoldMemberID, RecordID, BookID

*First approach not used because the Library Supervisor entity (relation) is redundant in that case as it has no new attributes different from the super-class (Employee)

*Flags used because Library Supervisor would vanish otherwise.

*Flags will also help in efficient querying

Silver member

PersonID	<u>SilverMemberID</u>	Silver_start_date	GoldMemberID

Primary key: SilverMemberID

Foreign key: PersonID, GoldMemberID

Gold member

<u>GoldMemberID</u>	GoldStartDate

Primary key: GoldMemberID

Library Card

<u>CardID</u>	MembershipLevel	PersonID	DateOfIssue

Primary key: CardID

Foreign key: PersonID (from Person relation)

Guest Log

<u>GuestID</u>	<u>GoldMemberID</u>	GuestName	GuestAddress	GuestContact

Primary key: GuestID + GoldMemberID (Composite key)

Foreign key: GoldMemberID

Book

<u>BookID</u>	Title	ClassType

Primary key: BookID

Author

<u>AuthorID</u>	<u>BookID</u>	Name

Primary key: AuthorID + BookID (Composite key)

Foreign key: BookID

Publisher

<u>PublisherID</u>	<u>BookID</u>	PublisherName

Primary key: PublisherID + BookID (Composite key)

Foreign key: BookID

Payment

<u>PaymentID</u>	Penalty	RecordID

Primary key: PaymentID

Foreign key: RecordID

Record

<u>RecordID</u>	DateOfIssue	DateOfReturn

Primary key: RecordID

Borrow

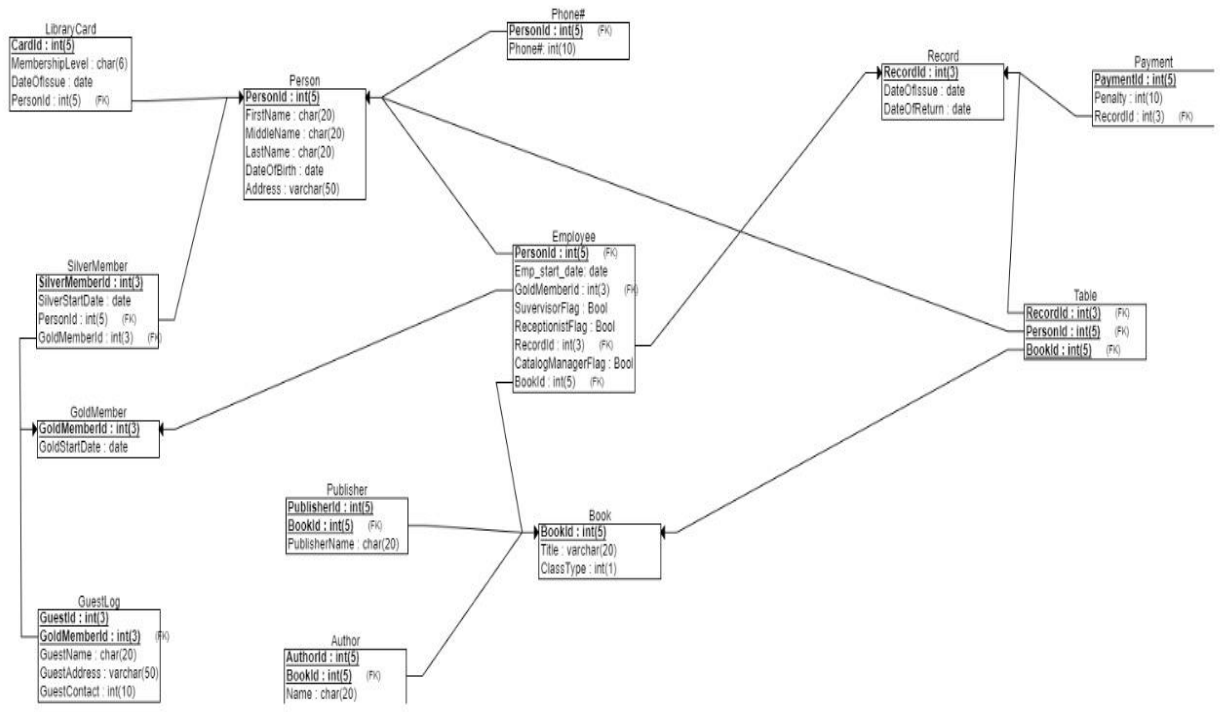
<u>PersonID</u>	<u>BookID</u>	<u>RecordID</u>

Primary key: PersonID + BookID + RecordID (Composite key)

Foreign key: PersonID (from Person relation), BookID, RecordID

All the tables are already in the **Third Normal Form**. The database design has been consistent with use of primary keys properly so there is no violation of the Third Normal Form.

EER diagram:



Data Definition:

```
CREATE database Library;
```

Data definition and dictionary:

Queries for creating all the tables *with constraints*:

1.
   ```
   CREATE table Person
   (FirstName varchar(20)  not null,
   MiddleName varchar(20),
   LastName varchar(20) not null,
   PersonId varchar(4) primary key CHECK (REGEXP_LIKE(PersonId,
   'P{1}[0-9]{1}[0-9]{1}[0-9]{1}')),
   DOB date not null CHECK(DOB < 01-Jan-2002),
   Address varchar(50) not null);
   ```

2.
   ```
   CREATE table PhoneNumber
   (PersonId varchar(4) REFERENCES Person(PersonId),
   Phone_no int ,
   CONSTRAINT PK PRIMARY KEY (PersonId, Phone_no));
   ```

3.
   ```
   CREATE table Employee
   (EmployeeId varchar(10) primary key REFERENCES Person(PersonId),
   Emp_start_date date not null,
   GoldMemId int REFERENCES GoldMember(GoldMemId),
   Supervisor_flag char(1) not null,
   Receptionist_flag char(1) not null,
   Catalog_man_flag char(1) not null,
   RecordId int REFERENCES Record(RecordId),
   BookId int REFERENCES Book(BookId));
   ```

4.
   ```
   CREATE table SilverMember
   (PersonId varchar(4)REFERENCES Person(PersonId),
   SilverMemId int PRIMARY KEY,
   Silver_start_date date not null,
   GoldMemId int REFERENCES GoldMember(GoldMemId) );
   ```

5.
   ```
   CREATE table GoldMember
   (GoldMemId int,
   Gold_start_date date not null,
   PRIMARY KEY(GoldMemId));
   ```

6.
   ```
   CREATE table LibraryCard
   (CardId int,
   Membership_level char not null,
   PersonId varchar(4),
   DOI date not null,
   PRIMARY KEY(CardId),
   FOREIGN KEY(PersonId) REFERENCES Person(PersonId));
   ```

7.
   ```
   CREATE table GuestLog
   (GuestId int,
   GoldMemId int,
   GuestName varchar(20) not null,
   GuestAddress varchar(50) not null,
   GuestContact int not null,
   PRIMARY KEY(GuestId, GoldMemId),
   FOREIGN KEY (GoldMemId) REFERENCES GoldMember(GoldMemId));
   ```

8.
   ```
   CREATE table Book
   (BookId int,
   Title varchar(50) not null,
   ClassType int not null check (ClassType =1 OR ClassType =2),
   PRIMARY KEY(BookId));
   ```

9.
   ```
   CREATE table Author
   (AuthorId int,
   BookId int,
   AuthorName varchar(20) not null,
   PRIMARY KEY(BookId, AuthorId),
   FOREIGN KEY (BookId) REFERENCES Book(BookId));
   ```

10.
    ```
    CREATE table Publisher
    (PublisherId int,
    BookId int,
    PublisherName varchar(20) not null,
    PRIMARY KEY(BookId, PublisherId),
    FOREIGN KEY (BookId) REFERENCES Book(BookId));
    ```

11.
    ```
    CREATE table Payment
    (PaymentId int,
    Penalty int not null,
    RecordId int,
    PRIMARY KEY(PaymentId),
    FOREIGN KEY (RecordId) REFERENCES Record(RecordId));
    ```

12. CREATE table Record
    (RecordId int,
    DOI date not null,
    DOR date not null,
    PRIMARY KEY(RecordId));

13. CREATE table Borrow
    (PersonId varchar(4),
    BookId int,
    RecordId int,
    PRIMARY KEY(PersonId, RecordId, BookId),
    FOREIGN KEY(PersonId) REFERENCES Person(PersonId),
    FOREIGN KEY (RecordId) REFERENCES Record(RecordId),
    FOREIGN KEY (BookId) REFERENCES Book(BookId));

VIEWS:

1. *TopGoldMember:*

```
CREATE OR REPLACE VIEW TopGoldMembers AS
SELECT P.FirstName, P.LastName, G.Gold_start_date, P.PersonId
FROM Person P, GoldMember G, Employee E
WHERE P.PersonId=E.EmployeeId AND G.GoldMemID=E.GoldMemID
AND P.PersonId IN (
                SELECT B.PersonId
                FROM Borrow B, Record R
                WHERE B.RecordId=R.RecordId AND R.DOR-R.DOI <8
AND CURRENT_DATE - DOI <366
```

```
                    HAVING COUNT(*)>5
                    GROUP BY B.PersonId);
```

2. *PopularBooks:*

```
    CREATE OR REPLACE VIEW PopularBooks AS
        SELECT
        B.BookID, B.title, B.ClassType,COUNT(B.BookId) as num_books
        FROM Book B, Borrow Br
        WHERE B.BookId = Br.BookId
        GROUP BY B.BookID, B.title, B.ClassType
        ORDER BY num_books desc
        FETCH FIRST 4 ROWS WITH TIES;
```

//Assuming the first four most borrows books are the most popular ones (reason: dealing with limited data/tuples)

3. *TopLatePaymentMembers:*

```
    CREATE OR REPLACE VIEW TopLatePaymentMembers AS
        SELECT P.FirstName, P.MiddleName, P.LastName, P.PersonId,
    P.DOB,P.Address,Count(Py.PaymentID) as num_pay
        FROM Payment Py, Person P, Borrow Br
        WHERE Py.RecordId = Br.RecordId AND Br.PersonID=P.PersonID
        GROUP BY P.FirstName, P.MiddleName, P.LastName, P.PersonId,
    P.DOB, P.Address
        ORDER BY num_pay desc;
```

4. *PotentialGoldMember:*

```
CREATE OR REPLACE VIEW  PotentialGoldMember AS
        SELECT P.FirstName,  pn.phone_no, s.silvermemid, P.personID
        FROM SilverMember S, PhoneNumber PN, Person P, Record R,
Borrow Br
        WHERE P.PersonId=PN.PersonId AND P.Personid=Pn.personID
AND S.PersonId=P.PersonId AND Br.PersonID=P.PersonId AND
Br.RecordId=R.RecordId AND R.DOR-R.DOI<8 AND
CURRENT_DATE-30< DOI
        HAVING COUNT(Br.BookId)>4
        GROUP BY P.FirstName, pn.phone_no, s.silvermemid,P.personID;
```

//Assuming the potential gold member borrows 1 book each week on an average which would be 4 books in the month

5. *TopPublisher:*

```
CREATE OR REPLACE VIEW TopPublisher As
SELECT PublisherName, Count (*) as num_books
FROM Publisher NATURAL JOIN Book
GROUP BY PublisherName
ORDER BY num_books desc;
```

QUERIES:

1. *For each employee class, list the employees belonging to that class.*

```
SELECT FirstName, LastName, Address,
  CASE
                    WHEN Supervisor_flag ='T'
          Then 'SUPERVISOR'
          WHEN Catalog_man_flag ='T'
          Then 'CATALOG MANAGER'
          WHEN Receptionist_flag ='T'
          THEN 'RECEPTIONIST'
  END as ClassType
FROM EMPLOYEE, PERSON WHERE employeeID=PersonID;
```

2. *Find the names of employees who are also a silver member and the books they have borrowed in the past month.*

```
SELECT P.FirstName, P.LastName, Bo.Title
FROM EMPLOYEE E, Person P, Borrow B, SilverMember S, Book bo, Record
R
WHERE S.PersonID = E.EMPLOYEEID AND E.EMPLOYEEID=P.PERSONID
AND B.PERSONID=E.EMPLOYEEID AND Bo.Bookid=B.bookid AND
R.RecordId=B.RecordId AND R.DOI>(CURRENT_DATE-30) ;
```

3. *Find the average number of books borrowed by the top five gold members in the library.*

```
SELECT AVG (num_books) FROM (
          SELECT Count (B.PersonID) as num_books
          FROM Borrow B, TopGoldMembers TG
WHERE B.Personid = TG.PersonID
GROUP BY B.PersonID);
```

4. *Find the name of the publisher and the book title that is borrowed most at the library.*

```
SELECT P.PublisherName, PB.title
FROM Publisher P, PopularBooks PB
WHERE P.bookid=PB.bookid;
```

5. *Find names of books that have not been borrowed in the last 5 months.*

```
SELECT DISTINCT B.title
FROM Book B
WHERE B.Bookid NOT IN (SELECT B.BookID FROM Borrow Br, Record R,
Book B WHERE R.recordid=Br.recordid AND Br.Bookid=B.bookid AND
CURRENT_DATE-151<R.DOI);
```

6. *Find the total number of Class 1 books and number of Class 2 books borrowed till date.*

```
SELECT ClassType, Count (ClassType)
FROM Book
WHERE BookID IN (SELECT DISTINCT BookId FROM Borrow)
Group BY ClassType;
```

7. *Find the Gold Member with most number of guests.*

```
SELECT GoldMemID, COUNT (GuestID) as Numg
```

```
FROM GuestLog
GROUP BY GoldMEMID ORDER BY numg desc;
```

8. *Find the month and year with the maximum books borrowed.*

```
SELECT EXTRACT(month from R.DOI) "Month" , EXTRACT(year
from R.DOI) "Year", Count(Br.Bookid)as MaximumNumberofBooks
FROM Borrow Br, Record R
WHERE Br.RecordId=R.RecordId
Group by EXTRACT(month from R.DOI), EXTRACT(year from R.DOI)
ORDER BY MaximumNumberofBooks desc
FETCH FIRST 2 ROWS WITH TIES;

//The top two maximum tuples
```

9. *Find the names of members who borrowed the most popular books.*

```
SELECT DISTINCT P.FirstName, P.LastName
FROM Person P, PopularBooks pb, Borrow B
WHERE P.PersonID=B.PersonID AND Pb.BookID=B.bookid;
```

10. *List all the details of books issued after the most current employee was hired.*

```
SELECT  DISTINCT B.Title
FROM Book B, Record R, Borrow Br
WHERE B.BOOKID=BR.BOOKID AND R.RecordId=Br.RecordId AND
R.DOI > (SELECT max(E.Emp_start_date)FROM EMPLOYEE E) ;
```

11. *List all the members that have enrolled into Gold membership within a month of being enrolled as Silver members.*

```
SELECT *
FROM SilverMember S, GoldMember G
WHERE S.GoldmemID=G.GoldMemId AND G.Gold_start_date-
S.Silver_start_date < 32;
```

12. *Find the total amount of late fee paid in each month, for the last 3 months.*

```
SELECT EXTRACT(month from R.DOI), SUM(P.penalty)
FROM PAYMENT p, Record R
WHERE P.RecordId=R.RecordId AND (EXTRACT(month from R.DOI)
> EXTRACT(month from CURRENT_DATE)-3) AND (EXTRACT(year
from R.DOI) = EXTRACT(year from CURRENT_DATE) AND
(EXTRACT(month from R.DOI) < EXTRACT(month from
CURRENT_DATE)) )
Group by EXTRACT(month from R.DOI);
```

13. *Find the name of members who have been a silver member for over 5 years.*

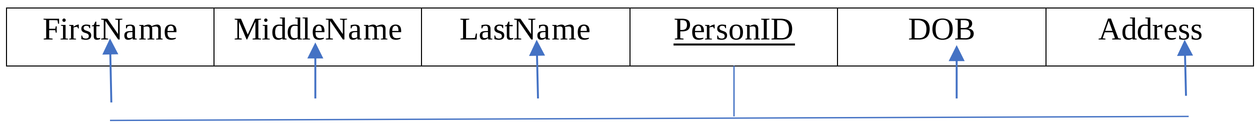
```
SELECT P.FirstName, P.LastName, s.personid
FROM Person P, SilverMember S
WHERE P.PersonID=S.PersonID AND CURRENT_DATE-
Silver_start_date > 1825;
```

14. Find the names and number of books borrowed by the potential gold members in the last year.

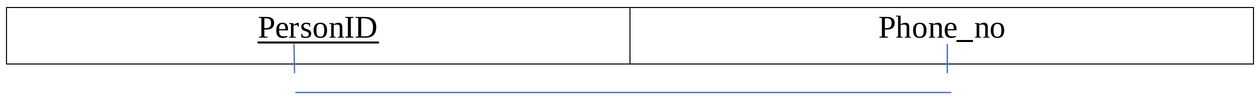
```
SELECT PG.FirstName, Count(Br.BookID)
FROM Borrow Br, PotentialGoldMember PG
WHERE Br.Personid=PG.Personid
GROUP By PG.firstname;
```

Dependency Diagram

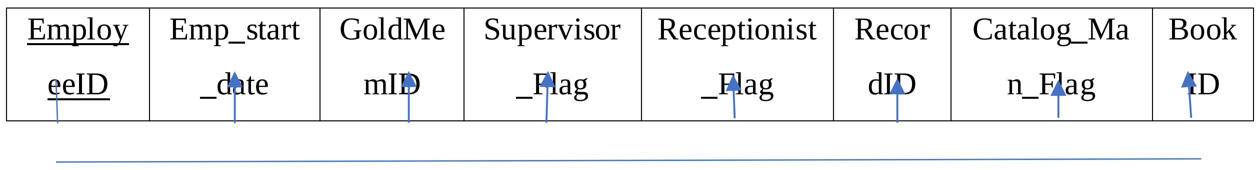
Person:



PhoneNumber:



Employee:



SilverMember:

PersonID	SilverMemID	Silver_start_date	GoldMemID

GoldMember:

GoldMemID	Gold_Start_Date

LibraryCard:

CardID	Membership_Level	PersonID	DOI

GuestLog:

GuestID	GoldMemID	GuestName	GuestAddress	GuestContact

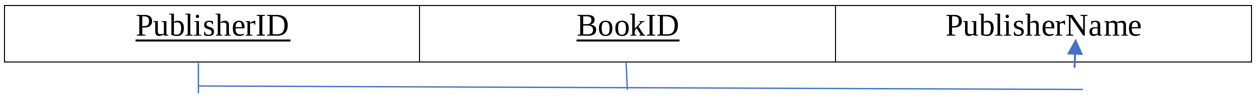
Book:

BookID	Title	ClassType

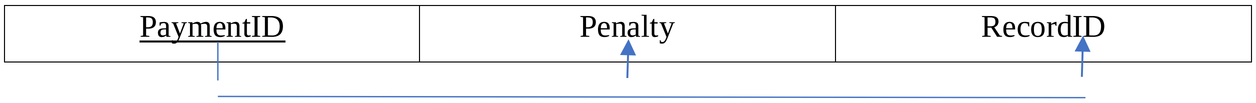
Author:

AuthorID	BookID	AuthorName

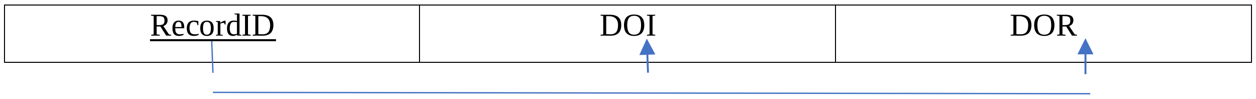
Publisher:



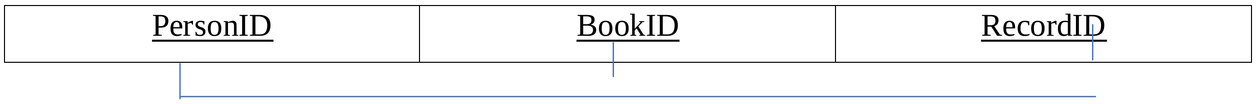
Payment:



Record:



Borrow:


PersonID
BookID
RecordID