Group 4:

Library Management System

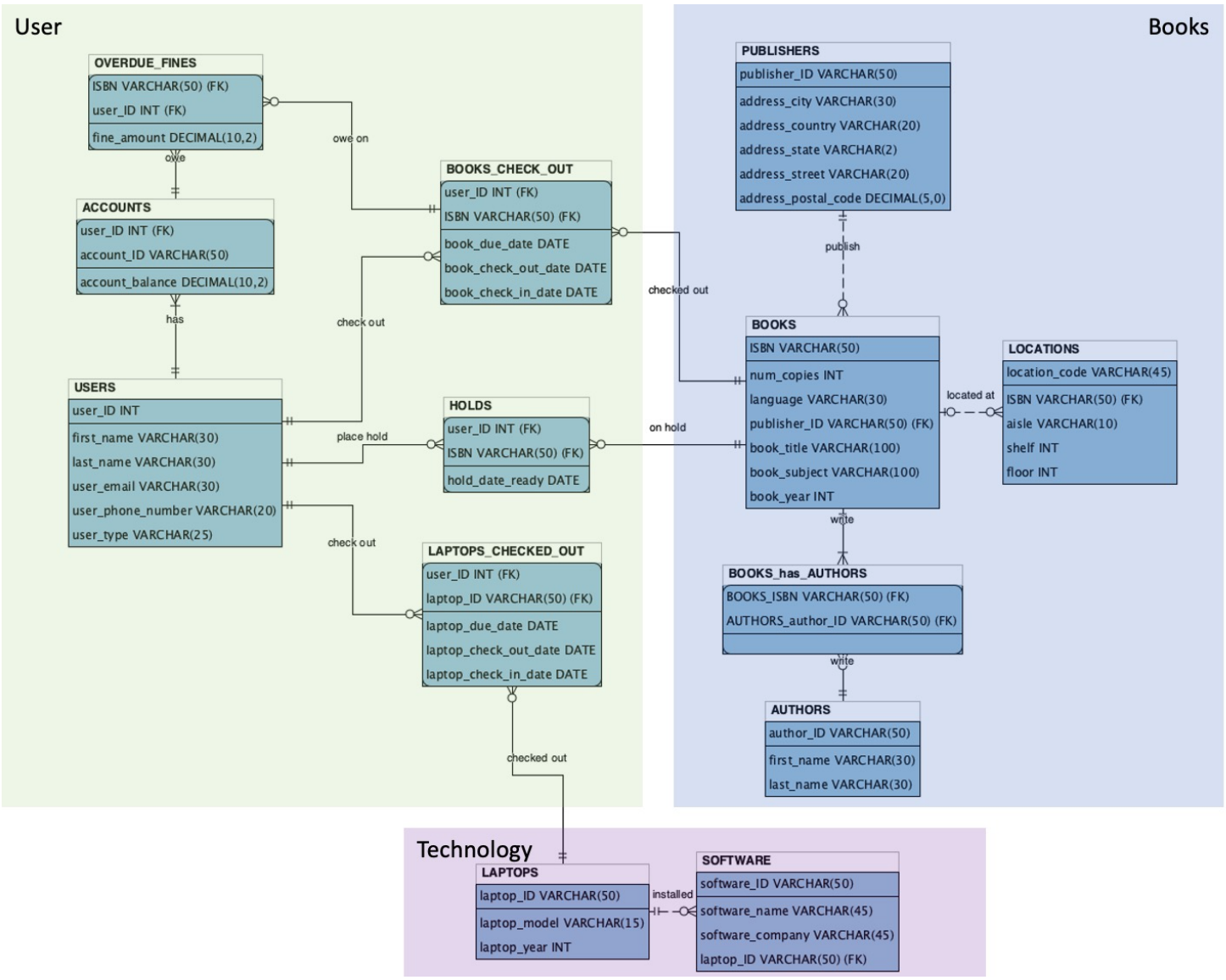
Maria Cuevas

Jordan Chan

Pui Ching Wong

Yanwen Cheng

Milestone 2: **ER Diagram**



Milestone 2: **SQL Queries**

1. Users who borrowed both book and laptop which are overdue and have not been returned

This query is for the administrator to know the information that those who borrowed both books and laptops from the library are overdue. The user ID, book ISBN, book due date, laptop id, and laptop due date will be shown.

```
USE mm_cpsc502101team04;
SELECT b.user_ID,
      b.ISBN,
      b.book_due_date,
      l.laptop_ID,
      l.laptop_due_date
FROM BOOKS_CHECK_OUT AS b
INNER JOIN LAPTOPS_CHECKED_OUT AS l ON b.user_ID = l.user_ID
WHERE book_due_date < book_check_in_date
  AND laptop_due_date < laptop_check_in_date
ORDER BY b.user_ID;
```

Data

user_ID	ISBN	book_due_date	laptop_ID	laptop_due_date
84	010808498-1	10/1/21	371-39-0386	10/31/21
87	027805238-X	9/30/21	108-89-5919	10/20/21
92	031990036-3	7/20/21	204-46-1803	9/30/21
153	054211304-X	8/30/21	412-31-5559	7/31/21

2. The books where the due day is in less than 1 week but have not been returned

This query is for the administrator to know the user id, user name and the book title that the due day is less than 1 week (2021-11-18) but still have not returned.

```
USE mm_cpsc502101team04;
SELECT c.user_ID,
    CONCAT(u.first_name, ' ', u.last_name) AS user_name,
    b.book_title,
    c.book_due_date
FROM BOOKS_CHECK_OUT AS c
JOIN USERS AS u ON u.user_ID = c.user_ID
JOIN BOOKS AS b ON c.ISBN = b.ISBN
WHERE book_check_in_date IS NULL
  AND book_due_date < '2021-11-18'
  AND book_due_date > '2021-11-11'
ORDER BY c.user_ID;
```

Data

user_ID	user_name	book_title	book_due_date
203	Alfons Kenewell	Inserts	11/17/21
206	Alf Macieiczyk	Hedda Gabler	11/12/21
211	Maurits Le Brom	Maid in Manhattan	11/15/21
220	Corby St. Aubyn	Corn on the Cop	11/13/21
300	Christian Clyburn	Evil Under the Sun	11/14/21

3. The popularity of laptop model being used by Students

This query counts each laptop model's borrowing frequency by Students. The query orders the output from the most popular laptop models to the least laptop models.

```
USE mm_cpsc502101team04;
SELECT l.laptop_model,
    COUNT(*) laptop_count_by_student
FROM LAPTOPS_CHECKED_OUT AS c
JOIN LAPTOPS AS l ON c.laptop_id = l.laptop_id
JOIN USERS AS u ON u.user_ID = c.user_ID
WHERE u.user_type = 'Student'
GROUP BY l.laptop_model
ORDER BY laptop_count_by_student DESC;
```

Data

laptop_model	laptop_count_by_student
Lenovo	3
Apple	3
Dell	1

4. The popularity of book subject in both students and faculties

Count the books that are borrowed by students and faculties in each subject. Show the top 10 subjects in all the users.

```
USE mm_cpsc502101team04;
SELECT b.book_subject,
      COUNT(*) bookCount,
      u.user_type
FROM BOOKS_CHECK_OUT AS c
JOIN BOOKS AS b ON b.ISBN = c.ISBN
JOIN USERS AS u ON u.user_ID = c.user_ID
GROUP BY b.book_subject,
        u.user_type
ORDER BY bookCount DESC
LIMIT 10;
```

Data

book_subject	bookCount	user_type
Literary Fiction	10	Faculty
Historical Fiction	9	Faculty
Fiction	6	Faculty
Fiction	6	Student
Horror	6	Student
Fantasy	5	Faculty
Mystery	5	Student
Horror	4	Faculty

5. Average book renting per person in both Students and Faculties

Create a view to know the average number of books being rented by each person in both students and faculties. Get the total number of students and faculties, and the total books being borrowed by students and faculties separately. Using the number of books being borrowed by students divided by the total number of students to get the average number of books being rented by each student. Same logic for getting the number of books being rented by each faculty.

```
USE mm_cpsc502101team04;
SELECT C.user_type,
      C.count AS num_of_people,
      B.count AS book_count,
      B.count/C.count AS average_book_rent_by_person
FROM
  ( SELECT u.user_type,
         COUNT(*) COUNT
   FROM USERS AS u
   GROUP BY u.user_type) AS C
JOIN
  ( SELECT u.user_type,
         COUNT(*) COUNT
   FROM BOOKS_CHECK_OUT AS c
   JOIN BOOKS AS b ON b.ISBN = c.ISBN
   JOIN USERS AS u ON u.user_ID = c.user_ID
   GROUP BY u.user_type) AS B ON C.user_type = B.user_type;
```

Data

user_type	num_of_people	book_count	average_book_rent_by_person
Faculty	166	65	0.3916
Student	130	54	0.4154

6. All users with holds

This query shows all users with holds. It shows all the user's information including user ID, first name, last name and whether the user is a student or faculty. It allows the users to see the book they placed on hold and when the book is ready for check out. First 10 results are shown in data.

```
USE mm_cpsc502101team04;
SELECT
    HOLDS.user_ID,
    USERS.first_name,
    USERS.last_name,
    USERS.user_type AS 'Student/ Faculty',
    HOLDS.ISBN,
    HOLDS.hold_date_ready
FROM
    HOLDS
        INNER JOIN
    USERS ON HOLDS.user_ID = USERS.user_ID
ORDER BY hold_date_ready , last_name
LIMIT 10;
```

Data

user_ID	first_name	last_name	Student/ Faculty	ISBN	hold_date_ready
137	Miguela	Pourvoieur	Student	818659561-9	10/21/21
116	Caroljean	Scahill	Faculty	795089665-5	10/21/21
232	Zonnya	Spleving	Student	575207749-4	10/22/21
281	Diahann	Prestland	Faculty	384318629-4	10/23/21
46	Alejandrina	Bisterfeld	Faculty	194762146-7	10/24/21
6	Hermann	Blondell	Faculty	264564313-2	10/28/21
284	Sauncho	Cumberledge	Student	405501278-0	10/28/21
230	Gene	McNeilley	Student	150683038-2	10/28/21
294	Eugenius	Orring	Faculty	336313002-3	10/30/21
73	Laurie	Guilloud	Student	048860603-9	11/1/21

7. All books in French

This query shows all books in French. It allows the users to see a list of books other than English (e.g., French). The result is sorted in the order of the year the book was published.

```
USE mm_cpsc502101team04;
SELECT
    ISBN,
    book_title,
    book_year,
    Language
FROM
    BOOKS
WHERE
    Language = 'French'
ORDER BY book_year;
```

Data

ISBN	book_title	book_year	Language
735576168-4	Devil's Doorway	1972	French
002870871-7	Introduction to Physics, An	1995	French
167066681-6	My Man (Mon homme)	1996	French
001433194-2	Song of Sparrows, The (Avaze gonjeshk-ha)	2001	French
785484432-1	Ivan the Terrible, Part Two (Ivan Groznyy II: Boyarsky zagovor)	2002	French
292864933-6	Taking Care of Business	2004	French
755492798-1	Sucker, The (Corniaud, Le)	2005	French
178705862-X	Seven Angry Men	2006	French
633423242-8	Brainstorm	2008	French
613763844-8	Rick	2009	French
871735052-2	Quiet Man, The	2009	French
861421519-3	Invasion U.S.A.	2010	French
119419593-8	Immortals, The	2012	French

8. Books checked out by a given user that have not yet been returned

This query creates a view that lists books checked out for a given user that have not yet been returned. This shows the user's ID, first and last name, the title of the book, as well as check out and check in dates.

```
USE mm_cpsc502101team04;
SELECT u.user_id, u.first_name, u.last_name, b.book_title, c.book_check_out_date,
c.book_check_in_date
FROM USERS u INNER JOIN BOOKS_CHECK_OUT c
ON u.user_id = c.user_id
INNER JOIN BOOKS b
ON c.isbn = b.isbn
WHERE c.book_check_in_date IS NULL
ORDER BY u.user_id;
```

Data

user_id	first_name	last_name	book_title	book_check_out_date	book_check_in_date
6	Hermann	Blondell	Coco Chanel	5/30/21	NULL
8	Letty	Tether	Mishen (Target)	6/5/21	NULL
12	Farly	Steketee	Coronado	5/5/21	NULL
202	Cornell	Altamirano	Kamikaze Girls (Shimotsuma monogatari)	8/30/21	NULL
203	Alfons	Kenewell	Inserts	9/17/21	NULL
206	Alf	Macieiczyk	Hedda Gabler	9/12/21	NULL
211	Maurits	Le Brom	Maid in Manhattan	9/15/21	NULL
220	Corby	St. Aubyn	Corn on the Cop	9/13/21	NULL
300	Christian	Clyburn	Evil Under the Sun	9/14/21	NULL

9. All fiction books with only one copy available

This query creates a view displaying Fiction books with only one copy available. This view lists the book titles alphabetically and includes the ISBN. This is an example of a certain genre of books that may need more copies ordered-- a search that a librarian may have use of.

```
USE mm_cpsc502101team04;
SELECT book_title, book_subject, isbn, num_copies
FROM BOOKS
WHERE num_copies = 1 AND book_subject = 'Fiction'
ORDER BY book_title;
```

Data

book_title	book_subject	isbn	num_copies
Ask Me Anything	Fiction	726575766-1	1
Brute, The (Bruto, El)	Fiction	637022376-X	1
City Lights	Fiction	193600870-X	1
Double Dynamite	Fiction	545913383-1	1
Fred: The Movie	Fiction	631892766-2	1
Go Tigers!	Fiction	321747521-6	1
Holy Man, The (Mahapurush)	Fiction	262644299-2	1
Tlt tullaan, elm!	Fiction	431815306-1	1
Wallace & Gromit in The Curse of the Were-Rabbit	Fiction	530704519-0	1
You Are God (Jestes Bogiem)	Fiction	146358104-1	1

10. All student users with fines due

This query creates a view that lists all student users with fines due, in order of user ID. The table includes their email address and corresponding fine amount.

```
USE mm_cpsc502101team04;
SELECT u.user_id, u.user_type, u.user_email, f.fine_amount
FROM USERS u INNER JOIN OVERDUE_FINES f
ON u.user_id = f.user_id
WHERE f.fine_amount > 0.00 AND u.user_type = 'Student'
ORDER BY u.user_id;
```

Data

user_id	user_type	user_email	fine_amount
11	Student	ebrobyna@vinaora.com	6.7
47	Student	rmarshal1a@apple.com	6.7
68	Student	egreenroad1v@sfgate.com	5.6
72	Student	swike1z@mapy.cz	4.55
74	Student	jkittel21@ning.com	7.36
77	Student	mwolland24@ftc.gov	6.97
83	Student	tpehrsson2a@statcounter.com	7.7
94	Student	adenisovich2l@vistaprint.com	9.7
95	Student	nclutterbuck2m@4shared.com	0.63
97	Student	kelmhurst2o@nydailynews.com	6.6
105	Student	ecaseborne2w@microsoft.com	2.52
118	Student	rcruise39@auda.org.au	2.06
133	Student	ivanhesteren3o@storify.com	6.47
155	Student	mogg4a@prweb.com	6.7
159	Student	tcorneil4e@home.pl	6.5
160	Student	rvolett4f@blogger.com	3.4
167	Student	jguillot4m@cornell.edu	7.71
180	Student	mabrahart4z@photobucket.com	3.48
181	Student	rcourtier50@ibm.com	4.75
190	Student	kshyram59@scribd.com	9.2

11. All holds that were ready for pick up before the current day and have not yet been checked out

This query displays all holds that are ready for pick up as of the current date but that have not yet been checked out by the user. The query displays the user's id number, the book's ISBN, and the date the hold was ready for pick up.

```
USE mm_cpsc502101team04;
SELECT H.user_id, H.ISBN, hold_date_ready
FROM HOLDS as H JOIN BOOKS_CHECK_OUT as B USING (ISBN)
WHERE NOT EXISTS
        (SELECT ISBN, user_id
        FROM BOOKS_CHECK_OUT
        WHERE H.user_id = B.user_id)
AND (NOW() > hold_date_ready)
ORDER BY user_id;
```

Data

user_id	user_name	ISBN	book_title	hold_date_ready
46	Alejandrina Bisterfeld	194762146-7	Hum Aapke Hain Koun...!	10/24/21
73	Laurie Guilloud	048860603-9	BURN-E	11/1/21
81	Vassily Rucklesse	092274394-0	Blondie of the Follies	11/4/21
89	Dave Fero	324371478-8	The Living Magoroku	11/7/21
176	Dugald Lindwasser	664977505-3	Early Summer (Bakush)	11/3/21
189	Fairleigh Markova	185716012-6	Hen Hop	11/7/21
198	Caroljean Christofle	044307846-7	Wisdom of Crocodiles, The (a.k.a. Immortality)	11/8/21

* Table output may look different depending on the date the query is run because the query uses the current date

Milestone 2: **Stored Procedures**

1. Check Out a Book to a Given User

This procedure allows a book to be checked out to a given user by entering the book's ISBN and the user's id number. The book associated with the ISBN provided will be looked up and added to the books_check_out table along with the user's id.

```
CREATE DEFINER=`mm_cpsc502101team04`@`%` PROCEDURE `check_out_book`(
IN user_num int, IN book_num varchar(30))
BEGIN
	IF NOT EXISTS
		(SELECT *
		FROM USERS
		WHERE USERS.user_id = user_num)
	THEN
		SELECT 'Error: Library user not found.' AS output;

  ELSE
		IF NOT EXISTS
			(SELECT *
			FROM BOOKS
			WHERE BOOKS.ISBN = book_num)
		THEN
			SELECT 'Error: Book not found.' AS output;
		ELSE
			IF (
				SELECT num_copies
				FROM BOOKS
				WHERE BOOKS.ISBN = book_num) <= 0
			THEN
				SELECT 'Error: Not enough copies.' AS output;

			ELSE INSERT INTO BOOKS_CHECK_OUT
				VALUES(user_num, book_num, DATE_ADD(NOW(),
				INTERVAL 60 DAY), NOW(), NULL);

				SELECT concat((
					SELECT book_title
					FROM BOOKS
```

```
                                    WHERE BOOKS.ISBN = book_num),
                                     ' has been checked out to ',
                                    (SELECT first_name
                                    FROM USERS
                                    WHERE USERS.user_id = user_num))
                                    AS 'Book checked out:';
                        END IF;
                END IF;
        END IF;
END
```

Test

Test information for a book not currently checked out and an existing user id:
CALL mm_cpsc502101team04.check_out_book('73','161124213-4');
OUTPUT: 101 Dalmatians has been checked out to Laurie

2. Check In a Given Book for a Given User

This procedure allows a book that is marked as checked out to the given user to be marked as checked in by adding the current date as the check in date to the books_check_out table.

```
CREATE DEFINER=`mm_cpsc502101team04`@`%` PROCEDURE `check_in_book`(
user_num int, book_num varchar(30))

BEGIN
IF NOT EXISTS
	(SELECT *
	FROM BOOKS_CHECK_OUT
	WHERE BOOKS_CHECK_OUT.ISBN = book_num
		AND BOOKS_CHECK_OUT.user_id = user_num)
THEN
	SELECT 'Error: Check out record not found.' AS output;

ELSE
	IF
		(SELECT book_check_in_date
		FROM BOOKS_CHECK_OUT
		WHERE BOOKS_CHECK_OUT.ISBN = book_num
			AND BOOKS_CHECK_OUT.user_id = user_num) IS NOT NULL
	THEN
		SELECT 'Error: Book already checked in.' AS output;

	ELSE UPDATE BOOKS_CHECK_OUT
		SET book_check_in_date = NOW()
		WHERE BOOKS_CHECK_OUT.ISBN = book_num
			AND BOOKS_CHECK_OUT.user_id = user_num;

		SELECT concat((
			SELECT book_title
			FROM BOOKS
			WHERE BOOKS.ISBN = book_num),
			' has been checked in.')
			AS output;
	END IF;
END IF;
END
```

Test

Test information for a book checked out to the provided user:
CALL mm_cpsc502101team04.check_in_book('73','161124213-4');
OUTPUT: 101 Dalmatians has been checked in.