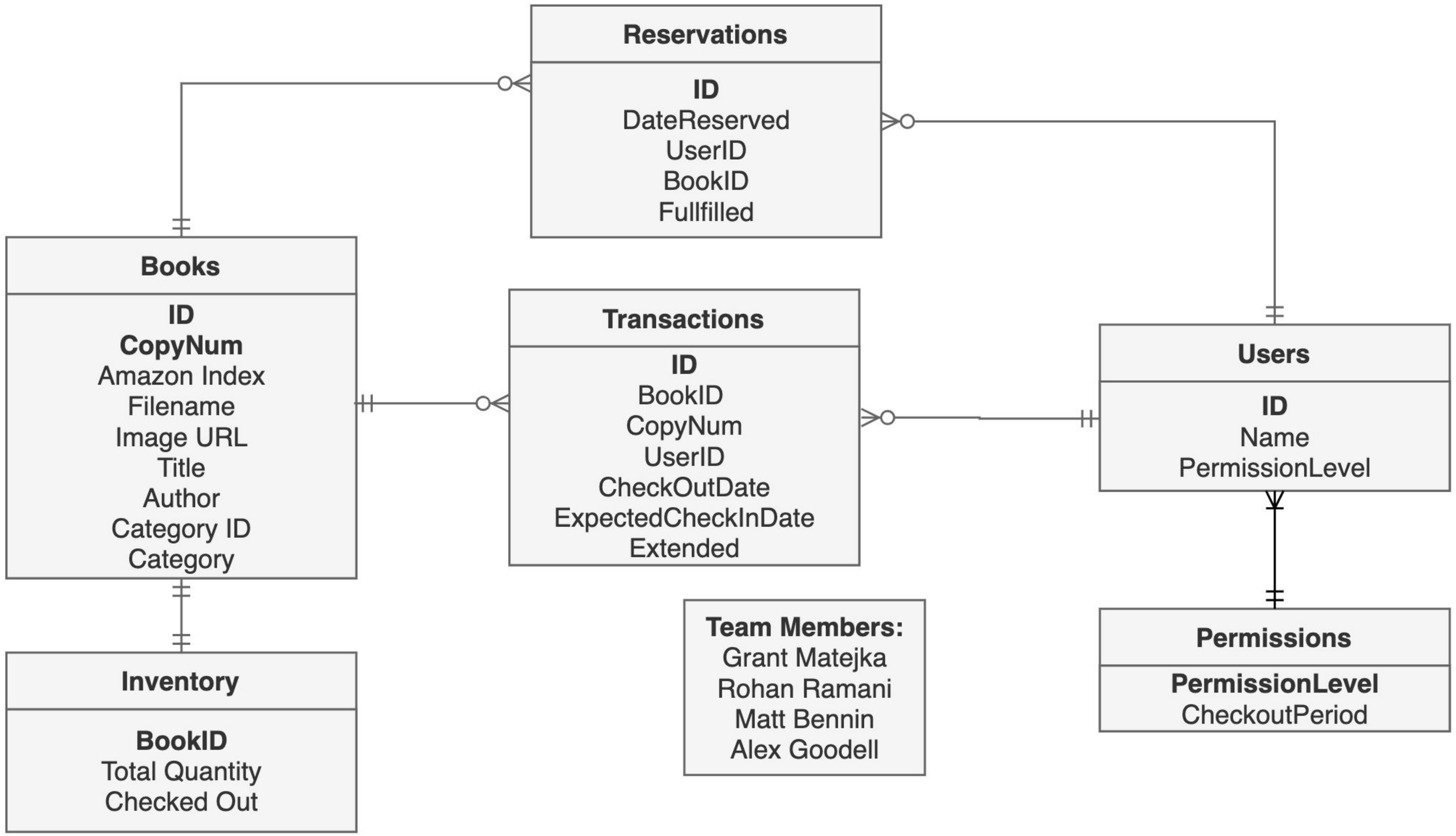
Group 09 - Library

Grant Matejka, Alex Goodell, Matthew Bennin, Rohan Ramani

ER-Diagram



Team Members:
Grant Matejka
Rohan Ramani
Matt Bennin
Alex Goodell

Create Table Statements

Books Table

```
CREATE TABLE Books (
   asin VARCHAR(15),
   copyNum INT AUTO_INCREMENT,
   filename VARCHAR(20),
   imageUrl VARCHAR(100),
   title VARCHAR(200),
   author VARCHAR(50),
   categoryId INT,
   category VARCHAR(20),
   KEY(copyNum),
   PRIMARY KEY(asin, copyNum)
);
```

Inventory Table

```
CREATE TABLE Inventory (
    bookId VARCHAR(15) REFERENCES Books(asin),
    totalQuantity INT,
    checkedOut INT,
    PRIMARY KEY(bookId)
);
```

Permissions Table

```
CREATE TABLE Permissions (
    permissionLevel VARCHAR(15),
    checkoutPeriod INT,
    PRIMARY KEY(permissionLevel)
);
```

Users Table

```sql
CREATE TABLE Users (
   id INT AUTO_INCREMENT,
   username VARCHAR(30),
   name VARCHAR(30),
   permissionLevel VARCHAR(15) REFERENCES Permissions(permissionLevel),
   PRIMARY KEY(id)
);
```

Transactions Table

```
CREATE TABLE Transactions (
    id INT AUTO_INCREMENT,
    bookId VARCHAR(15) REFERENCES Books(asin),
    copyNum INT REFERENCES Books(copyNum),
    userId INT REFERENCES Users(id),
    checkOutDate DATE,
    expectedCheckInDate DATE,
    extend BOOLEAN,
    PRIMARY KEY(id)
);
```

Reservations Table

```
CREATE TABLE Reservations (
   id INT AUTO_INCREMENT,
   dateReserved DATE,
   userId INT REFERENCES Users(id),
   bookId VARCHAR(15) REFERENCES Books(asin),
   fullFilled BOOLEAN,
   PRIMARY KEY(id)
);
```

Triggers

Checkout/in Count Triggers

```
DELIMITER $
CREATE TRIGGER `checkout_count_trigger` AFTER INSERT ON `Transactions`
FOR EACH ROW BEGIN

SET @count = (SELECT checkedOut FROM group09.Inventory WHERE NEW.bookId = bookId);
UPDATE Inventory i SET i.checkedOut = count + 1 WHERE i.bookId = NEW.bookId;

END$ DELIMITER ;
```

```
DELIMITER $ CREATE TRIGGER `checked_in_trigger` AFTER UPDATE ON `Transactions`
FOR EACH ROW BEGIN

SET @count = (SELECT checkedOut FROM group09.Inventory WHERE old.bookId = bookId);
IF old.checkedIn <> new.checkedIn
THEN
        UPDATE Inventory i SET i.checkedOut = count - 1 WHERE i.bookId = old.bookId;
END IF;

END$
DELIMITER ;
```

Ability to extend trigger

```
DELIMITER $
CREATE TRIGGER `check_extend_trigger` BEFORE UPDATE ON `Transactions`
FOR EACH ROW
BEGIN


IF new.extend = 1 AND old.extend = 0
THEN

      SET @reserved = (SELECT * FROM Reservations WHERE NEW.bookId = bookId AND fullFilled = 0);
   IF reserved.fullFilled = 0 AND reserved IS NOT NULL
   THEN
             SIGNAL SQLSTATE '12345'
             SET MESSAGE_TEXT = 'Book reserved by someone else';
   END IF;
END IF;


END$
DELIMITER ;
```

Alter After Extend Trigger

```
DELIMITER $
CREATE TRIGGER `update_extend_trigger` AFTER UPDATE ON `Transactions`
FOR EACH ROW BEGIN

IF new.extend = 1 AND old.extend = 0
THEN
   IF "GR" = (SELECT permissionLevel FROM group09.Users u WHERE NEW.userId = u.id)
   THEN
             UPDATE Transaction t SET t.expectedCheckInDate = t.expectedCheckInDate + 14 WHERE NEW.userId =
t.userId AND NEW.bookId = t.bookId;

   END IF;

   IF "UG" = (SELECT permissionLevel FROM group09.Users u WHERE NEW.userId = u.id)
   THEN
             UPDATE Transaction t SET t.expectedCheckInDate = t.expectedCheckInDate + 7 WHERE NEW.userId =
t.userId AND NEW.bookId = t.bookId;
   END IF;
END IF;

END$
DELIMITER ;
```

Ability to Checkout Trigger

GENERAL

```
SET @stock = (SELECT checkedOut FROM group09.Inventory i WHERE i.bookId = NEW.bookId);
SET @reserved = (SELECT COUNT(fullFilled) FROM group09.Reservations r WHERE r.bookId = NEW.bookId AND fullFilled = 0);
IF @stock < 1 OR (stock = reserved) THEN
       SIGNAL SQLSTATE '41900'
       SET MESSAGE_TEXT = 'No copies of book can be checked out, none available';
END IF;
```

UNDERGRAD

```
IF "UG" = (SELECT permissionLevel FROM group09.Users u WHERE NEW.userId = u.id)
THEN SET @count = (SELECT COUNT(c.userId) FROM group09.Transactions c WHERE c.userId = NEW.userId AND checkedIn = 0);
    IF @count > 3 THEN SIGNAL SQLSTATE '42000' SET MESSAGE_TEXT = 'Undergraduate student has too many books checked out';
        END IF; END IF;
```

GRAD

```
IF "GR" = (SELECT permissionLevel FROM group09.Users u WHERE NEW.userId = u.id)
THEN SET @count = (SELECT COUNT(c.userId) FROM group09.Transactions c WHERE c.userId = NEW.userId AND checkedIn = 0);
    IF @count > 5
        THEN SIGNAL SQLSTATE '42100' SET MESSAGE_TEXT = 'Graduate student has too many books checked out';
        END IF; END IF;
```

Calculate CheckInDate Trigger

```
DELIMITER $
CREATE TRIGGER `checkout_period` AFTER INSERT ON `Transactions`
FOR EACH ROW
BEGIN
     UPDATE Transactions SET expectedCheckInDate = checkOutDate +
          (
            SELECT p.checkoutPeriod
            FROM Permissions p
            JOIN Users u ON u.permissionLevel = p.permissionLevel
            WHERE u.is = NEW.userId
          )
     WHERE id = NEW.id AND expectedCheckInDate IS NULL;
END$ DELIMITER;
```

Protect against injection

Prepared Statements

```
preparedStatement =
    this.conn.prepareStatement
        ("INSERT INTO Reservations
        (dateReserved, userId, bookId) VALUES (?, ?, ?)");
```

Protect against transaction conflict

```
try {
   preparedStatement =
            this.conn.prepareStatement("INSERT INTO Reservations (dateReserved, userId, bookId)
           VALUES (?, ?, ?)");

   preparedStatement.setDate(1, date);
   preparedStatement.setInt(2, userId);
   preparedStatement.setString(3, bookId);

   this.conn = dm.getTransConnection();

   try{
       this.conn.setAutoCommit(false);

       Boolean returnValue = preparedStatement.execute();

       this.conn.commit();

   } catch(Exception e){
```

Demo