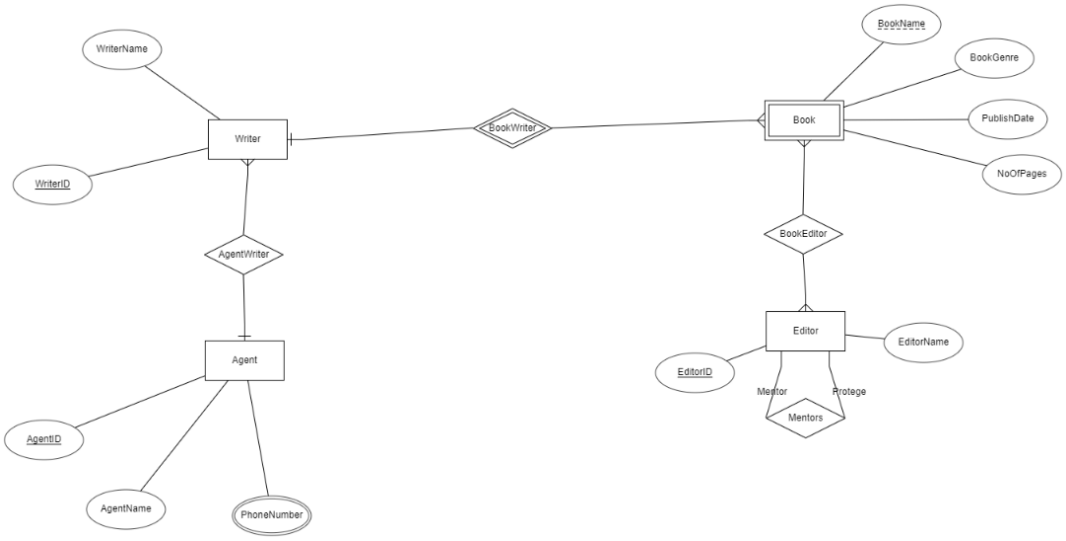
CS 370 Database Mngt Systems, Spring 2024

Assignment 03, Total Points: 100

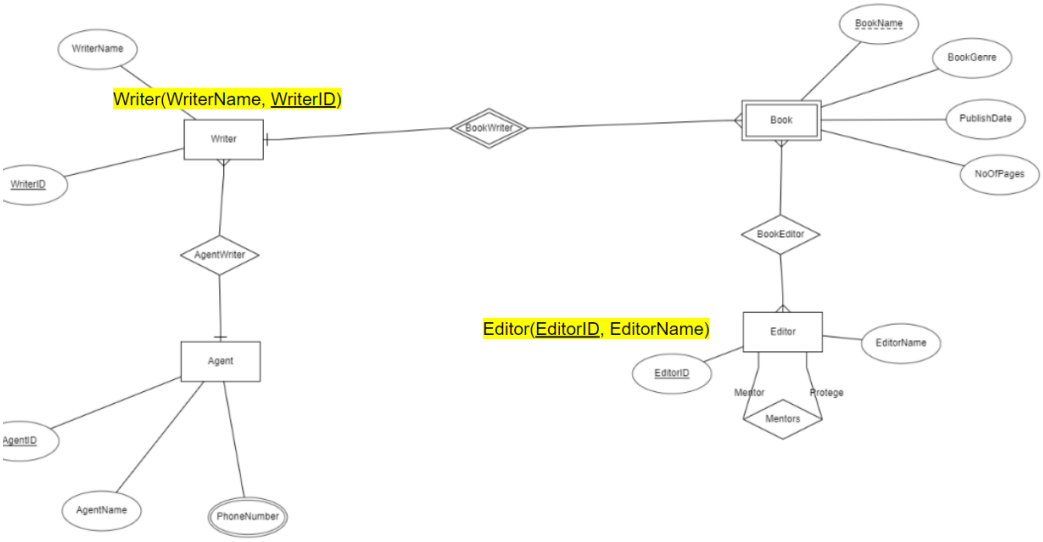
Deadline: April 30th, Tuesday, 11:59 PM

Note: A deduction of 30% of the points will apply for each day of delay in submission.

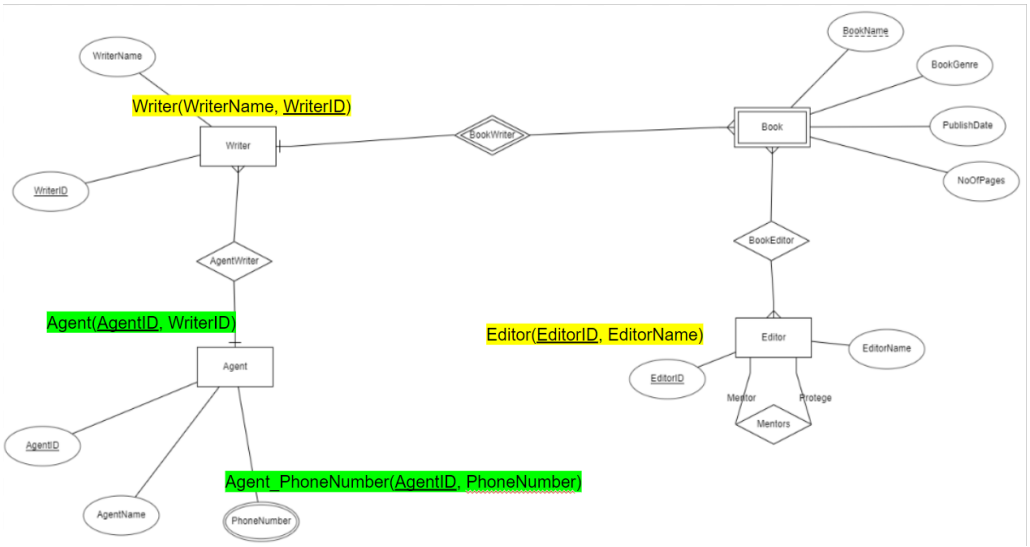
1. Convert the following ER diagram to non-redundant relation schemas.
   [15 + 10 pints]



a. Show the intermediate steps.

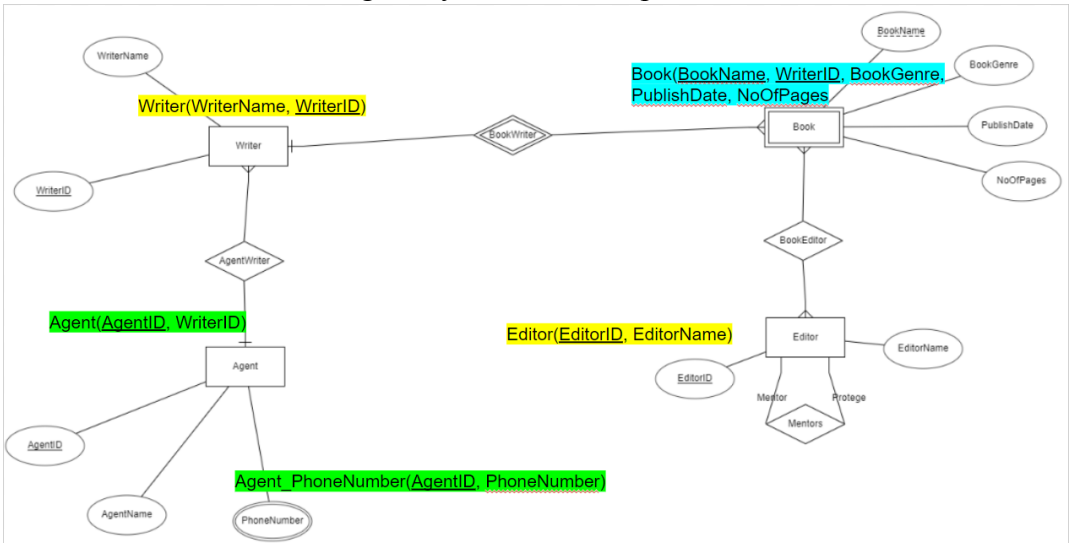


1. Label strong entities with strong relations

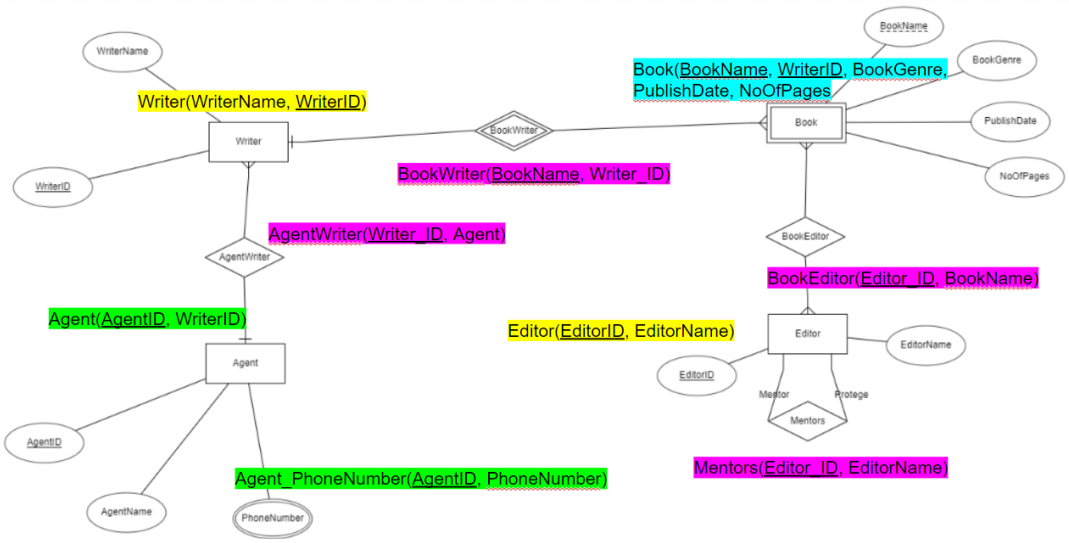


*Note - I messed up and put WriterID instead of AgentName, so just think of it As AgentName. I fix this in the last schema

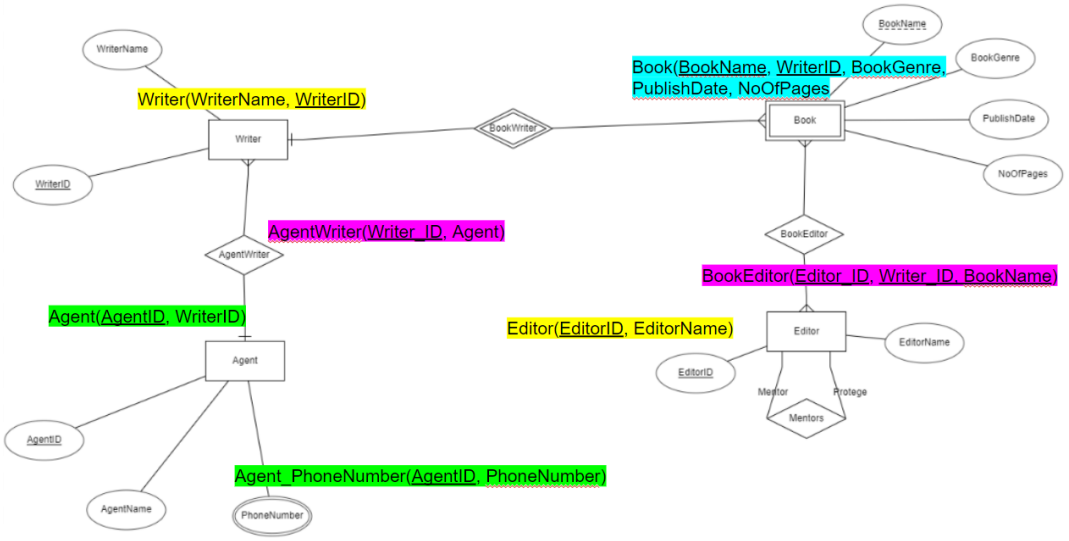
2. Strong entity sets with complex attributes



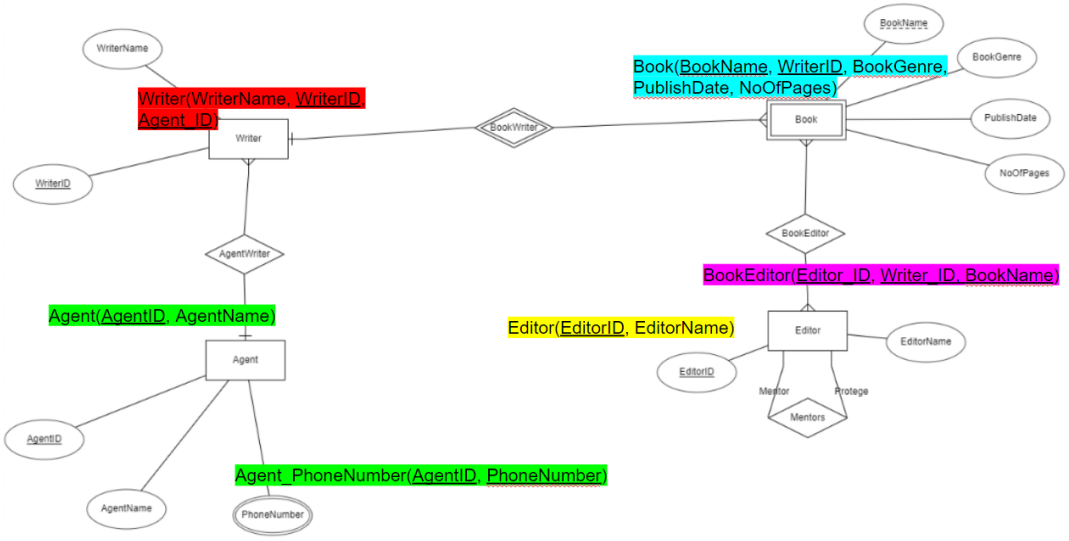
3. Weak entity sets with attributes



4. Representation of Relationship sets

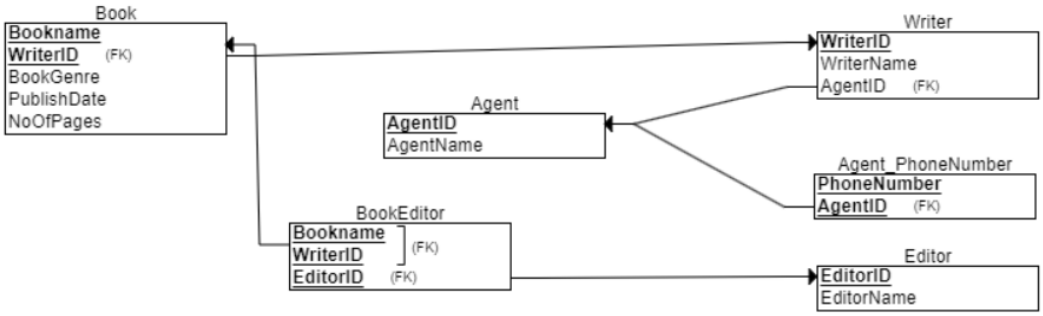


5. Removal of Redundant Data sets



6. Combining Schemas

b. Draw the Relational Schema Diagram



2. Draw a relation (table) of at least 5 columns and put some data there (at least 6 rows). Then normalize it up to 3nf (1nf,2nf and 3nf) **[7 points for each. Max 20 Points]**

Sample relation (Table)
Un-normalized Table

Column1	Column2	Coulmn3	Column4	Column5	Column6
Order ID	Customer Name	Customer Address	Product ID	Product Name	Product Price
1	Alice	123 Main St	101	Chair	50
2	Bob	456 Oak St	102	Table	100
3	Charlie	789 Pine St	101	Chair	50
1	Alice	123 Main St	201	Lamp	25
4	David	321 Elm St	103	Desk	150
5	Eve	654 Birch St	201	Lamp	25

1nf:
Customer Table:

Customer ID	Customer Name	Customer Email
1	Alice	alice@example.com
2	Bob	bob@example.com
3	Charlie	charlie@example.com
4	David	david@example.com

Products Table:

Product ID	Product Name	Product Category	Price
101	Laptop	Electronics	1200
102	Smartphone	Electronics	800
103	Headphones	Electronics	100
104	Shirt	Clothing	30
105	Shoes	Footwear	50
106	Watch	Accessories	150

Orders Table:

Order ID	Customer ID	Product ID
1	1	101
2	2	102
3	1	103
4	3	104
5	4	105
6	1	106

2nf: All non-key attributes are functionally dependent on the primary key in each table, so no decomposition is needed.

3nf: removal of transitive dependencies by further splitting the tables

Products table:

Product ID	Product Name	Price
101	Laptop	1200
102	Smartphone	800
103	Headphones	100
104	Shirt	30
105	Shoes	50
106	Watch	150

Product Categories Table:

Product ID	Product Category
101	Electronics
102	Electronics
103	Electronics

104	Clothing
105	Footwear
106	Accessories

Customers Table:

Customer ID	Customer Name	Customer Email
1	Alice	alice@example.com
2	Bob	bob@example.com
3	Charlie	charlie@example.com
4	David	david@example.com

Orders table:

Order ID	Customer ID	Product ID
1	1	101
2	2	102
3	1	103
4	3	104
5	4	105
6	1	106

3. What is an SQL injection attack? Explain how it works, and what precautions must be taken to prevent SQL injection attacks. **[10 Points]**

   An SQL injection attack is an attack that targets the vulnerabilities in the SQL software. It works by inputting SQL code where the database takes something in, like a login or search bar. The database then takes this code as commands rather than normal data. After this is done, the injector is able to modify the commands of the database, meaning that all of the data inside of it is compromised.

   A way to avoid these attacks is to use parameterized queries. This means that instead of taking in commands which could be hidden as normal text, the inputs are bound to the query parameters. A few other ways to avoid it are input validations to make sure after taking in the input that it is not a malicious command. Another way is to run checks on the database to make sure that commands haven't been run on it that were not meant to be run.

4. Indices speed query processing, but it is usually a bad idea to create indices on every attribute, and every combination of attributes, that is a potential search keys. Explain why? **[10 points]**

It is usually a bad idea to create too many indices because of a few factors. The first one being that they take up additional storage space, and increasing the space in the database slows down operations because it has to hold more data. Whenever you update the database, the database itself needs to update all the relevant indices as well. This slows down the operation of minute to minute use of the database. The Indices need to be maintained as well when you are updating the database, and if they break there has to be maintenance done on them to get the database working again. The final reason why is that if multiple instances of modifying the same data occur, conflicts can arise and result in a slower database. The overuse of Indices cause the database to slow down and break, but using them sparingly can result in a faster and healthier database.

5. Query: Find the names of all customers who have an account at some branch located in Peoria. Draw the Relational algebra expressions and query-evaluation plans. Mark the most efficient query-evaluation plan between your plans. [Hints: Relations(tables): Branch, account, depositor. Needs join operation] **[20 points]**

Assume the following tables:

Branch:

BranchName

BranchCity

Assets

Account:

AccountNumber

BranchName

Balance

Depositor:

CustomerName

AccountNumber

1. $\pi CustomerName,AccountNumber(Depositor \bowtie Depositor.BranchName=Branch.BranchNameBranch)$

2. $\pi CustomerName((\pi CustomerName,AccountNumber(Depositor \bowtie Depositor.BranchName=Branch.BranchNameBranch)) \bowtie Account)$

Query-Evaluation Plans:

1. Perform the first join between Branch and Depositor, then join the result with Account.

2. Perform the first join between Depositor and Account, then join the result with Branch.

The most efficient query-evaluation plan would be one that minimizes intermediate results. Since the size of the Depositor table might be smaller than the Account table, the first plan is likely more efficient as it filters out unnecessary rows early on in the process.

6. What are the ACID Properties of a Transaction. Provide a scenario and describe the acid properties based on the scenario**. [15 points]**

The ACID properties of a transaction are Atomicity, Consistency, Isolation, and Durability. Atomicity is treating the transaction as a single unit of work where either the whole transaction is complete or it doesn't register. Consistency is making sure the database remains consistent after the transaction to verify integrity of the system. The Isolation is so that each transaction is done separately, so that there is no overlap in data that could cause an error. Durability is so that when the transaction is completed, the effects are permanently applied to the database and changes will remain even if the system were to go down.

With all of this in mind, a scenario of this would be somebody sending money to somebody else using a software like venmo. The Atomicity treats the transaction as an individual request, both the sending and receiving are treated as one. The transaction starts and the money request from one person's account is sent to the other in isolation of other transactions at the time. The Consistency makes sure that the guidelines of the transaction were followed, and the Durability updates the accounts permanently so that they do not accidentally roll back. With this scenario, ACID properties successfully transfer funds over to the other person.