WolfCity Publishing House

CSC 540 Database Management Systems
Project 1

Harini Bharata - hbharat@ncsu.edu,
Krishna Saurabh Vankadaru - kvankad@ncsu.edu,
Sri Pallavi Damuluri - sdamulu@ncsu.edu,
Subodh Thota - sthota@ncsu.edu

February 15th, 2021

1. Problem statement

We have to design a database system for WolfPubDb for the wolfcity publishing house which maintains information about the staff authors, editors, distributors, and different kinds of publications. The system must be able to perform actions such as managing publications, assigning editors, assigning authors, generating monthly reports which include payments made towards editors and authors, revenue and expenses of the publishing house, and sales statistics of each publication. The system also performs operations for distributors like adding and updating a distributor, maintaining balances, taking orders of different publications, and billing them accordingly.

The WolfPubDb deals with a large number of users involved in various activities that require concurrent access to vast data. Since a database management system offers excellent benefits in terms of efficient data storage, access, and retrieval, this can be considered as an effective solution to address the above problem statement. The transaction manager in a DBMS will help ensure that all the transactions that can occur in our publication house are atomic, consistent, isolated, and durable. Having a database management system will help achieve these properties conveniently over maintaining the records over a file system.

2. Assumptions

1. There are no special journalists, all journalists are authors
2. Each article in a publication is written by one author
3. A distributor can place an order for one publication but multiple quantities at a time.
4. A book can have more than one author
5. A distributor can only hold one account, and the current due amount will be tracked in the account.
6. A chapter is a part of exactly one book and similarly an article is part of exactly one periodical.
7. Chapters in a book cannot be written by different authors, only book has authors.
8. Payment to authors and editors will need the input of payment amount, and we will not store any salary information for the staff.
9. A payment to staff will be made on the salary date, that is when the payment is initiated.
10. Every member of the staff will have a unique staff id.
11. Every new edition of a book will have a new publication id, similarly every new periodical will have a new publication id.
12. Each article in a periodical will have a unique title.

3. Classes of users

Admin: The admin users are the ones that have responsibilities for managing the database, creating other types of users, maintaining information of publications, orders, distributors. They have the rights to perform all the operations including handling payments and generating reports.

Editors: Editors are part of the publication house and are responsible for editing the publications that are assigned to them. There are two types of editors, one which has a fixed salary staff editor, and an invited editor who gets paid for one-time work.

Authors: Authors write books, and articles in a periodic publication. They get paid for their work and have 2 types, staff authors who are paid periodically, and invited authors who get paid for their one-time work.

Distributors: Distributors issue orders to the publication house for a certain date and a publication. They hold an account with the publishing house which has the balance amount which they owe.

4. Five Main Things

1. **Staff Information**: We need to store the information about the staff of the publication house. We need information about their name, phone number, role(whether he is an author, editor or admin), and their type(if they are invited or regular staff)

2. **Publication Information**: This will hold information regarding different kinds of publications which will contain the price, topic, title, type(book or periodical). We will also need to store information regarding each book or periodical separately.

3. **Distributor Information**: This will store the information regarding distributors, with attributes like account number, name of the distributor, contact person, address, account balance, and phone number.

4. **Order Information**: This is the order information that the distributor places to the publication house, we need to store the information of the publication he/she orders, quantity, future delivery date and cost of the order.

5. **Payment Information**: We need to store the payments made to every staff member of the publication house. This will contain information about the date of the payment, staff information to which the payment is addressed, date of receipt and payment amount.

5. Realistic situations:

1. A distributor comes to the publishing house to place an order for 200 copies of 5th edition of Narnia, to be delivered by August 30th, 2022. The admin inputs and creates an order with the details. He will bill the distributor based on the quantity and price of the book, and modify the outstanding balance of the distributor by adding the new bill amount.
2. A publishing house decides to publish a new edition of Database Systems as multiple authors have introduced significant changes in the latest edition. The admin creates a new publication and adds the book details like ISBN, edition, etc. The publishing house also changes the newest edition price from 15 dollars to 20 dollars. Once the above actions are performed, the publication house assigns the publication to an editor to ensure that the book's content is good to go.

6. Application Programming interfaces:

***Editing and publishing*.**

1. enter_new_publication(title, topic, price, type)
   Returns api for enter_new_book or enter_new_periodical depending on the type

2. edit_publication(title, topic, price)
   Returns confirmation if edit was successful

3. enter_new_book (publication_id, ISBN, publication_date, edition, list_of_authors)
   Returns confirmation if new book was created successfully.

4. Edit_book (publication_id, ISBN, publication_date, edition, list_of_authors)
   Returns confirmation if if edit was successful.

5. enter_new_periodical (publication_id, date_of_issue, periodicity)
   Returns confirmation if new periodical was created successfully.

6. edit_periodical (publication_id, date_of_issue, periodicity)
   Returns confirmation if edit periodical was successful.

7. assign_publication_to_editor(staff_id, publication_id)
   Return confirm

8. view_assigned_publications(staff_id)
   Return list of publications that an editor is assigned

Production of a book edition or of an issue of a publication.

1. new_chapter(publication_id, title, text, date)
   Return confirmation

2. edit_chapter(publication_id, title, text, date)
   Return confirmation

3. delete_chapter(publication_id, title)
   Return confirmation

4. new_article(publication_id, title, topic, text, date)
   Return confirmation

5. edit_article(publication_id, title, topic, text, date)
   Return confirmation

6. delete_article(publication_id, title)
   Return confirmation

7. delete_book(publication_id)
   Return confirmation

8. delete_periodical(publication_id)
   Return confirmation

9. find_books_by_topic(topic)
   Return list of books based on the given parameter

10. find_books_by_date(date)
    Return list of books based on the given parameter

11. find_books_by_author_name(author_name)
    Return list of books based on the given parameter

12. find_articles_by_topic(topic)
    Return list of articles based on the given parameter

13. find_articles_by_date(date)
    Return list of articles based on the given parameter

14. find_articles_by_author_name(author_name)
    Return list of articles based on the given parameter

15. add_staff_payment(staff_id, date_of_payment, date_of_collection, amount)
    Returns confirmation of successful creation

16. update_staff_payment(staff_id, date_of_payment, date_of_collection, amount)
    Returns confirmation of successful creation

17. claimPayment(staff_id, date_of_payment, date_of_collection)
    Returns confirmation of successful updation of payment record.

Distribution

1. add_new_distributor(account_number, name, contact_person, phone, city, street_address, type, balance)
    Return confirm

2. edit_distributor(account_number, name, contact_person, phone, city, street_address, type, balance)
    Return confirm

3. delete_distributor(account_number)
    Return confirmation after successful deletion

4. create_order(publication_id, account_number, order_date, order_delivery_date, shipping_cost, number_of_copies, order_cost)
    Return order_id of the new order created when successful creation of order

5. bill_distributor(account_number)
    Returns the balanced owed for a distributor

6. update_account_balance(account_number, new_amount)
   Returns confirmation after successful updation of distributor account balance

Reports

1. get_total_revenue()
   Returns the revenue of the publication house aggregated montly

2. get_revenue_per_city()
   Returns revenue aggregated per city

3. get_revenue_per_distributor()
   Returns revenue aggregated per distributer

4. get_revenue_per_location()
   Returns revenue aggregated per location

5. get_total_expenses()
   Returns total expenses by the publication house from the payments made to the staff

6. get_total_distributors()
   Returns the total number of distributors

7. Get_statistics_for_each_publication_per_distributor()
   Returns price and number of copies of publication sold aggregated by the distributor per month

8. get_total_staff_payment(from_date, to_date, work_type)
   Returns total amount of payments made to the authors and editors aggregated per time period per work type.

7. Description of views:

1. **Admin** can create users, delete users, maintain information about publications in the database, handle payments and generate reports.

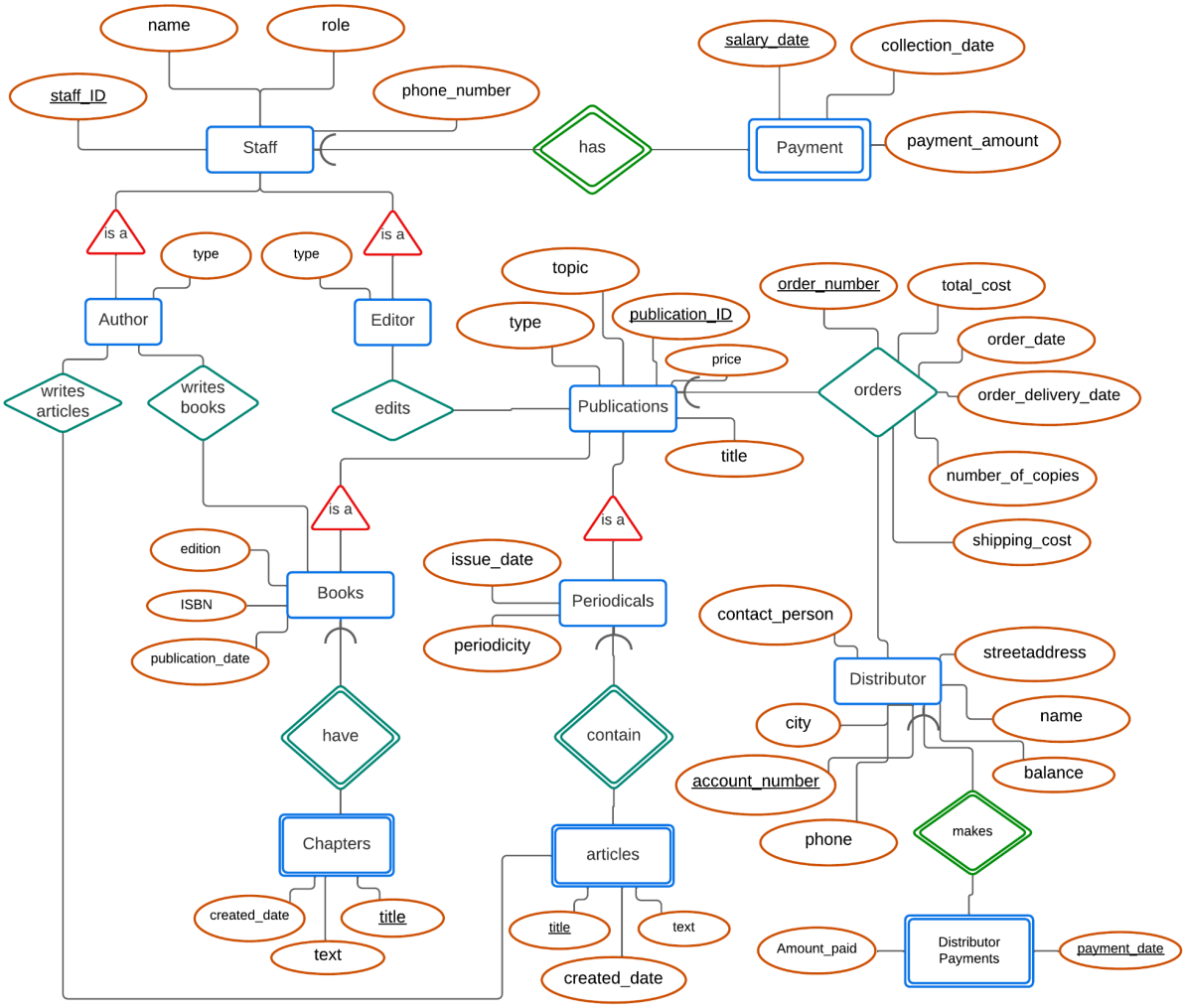
2. **Editor** is responsible for making changes to a book/article. They can add chapters, delete chapters, edit chapters, add articles, edit articles and delete articles. They should have access to all the books and publications in the publishing house. They do not have the access to the payment information and the right to delete a publication.

3. **Author** is responsible for writing books and articles. They should be able to view their pay checks.

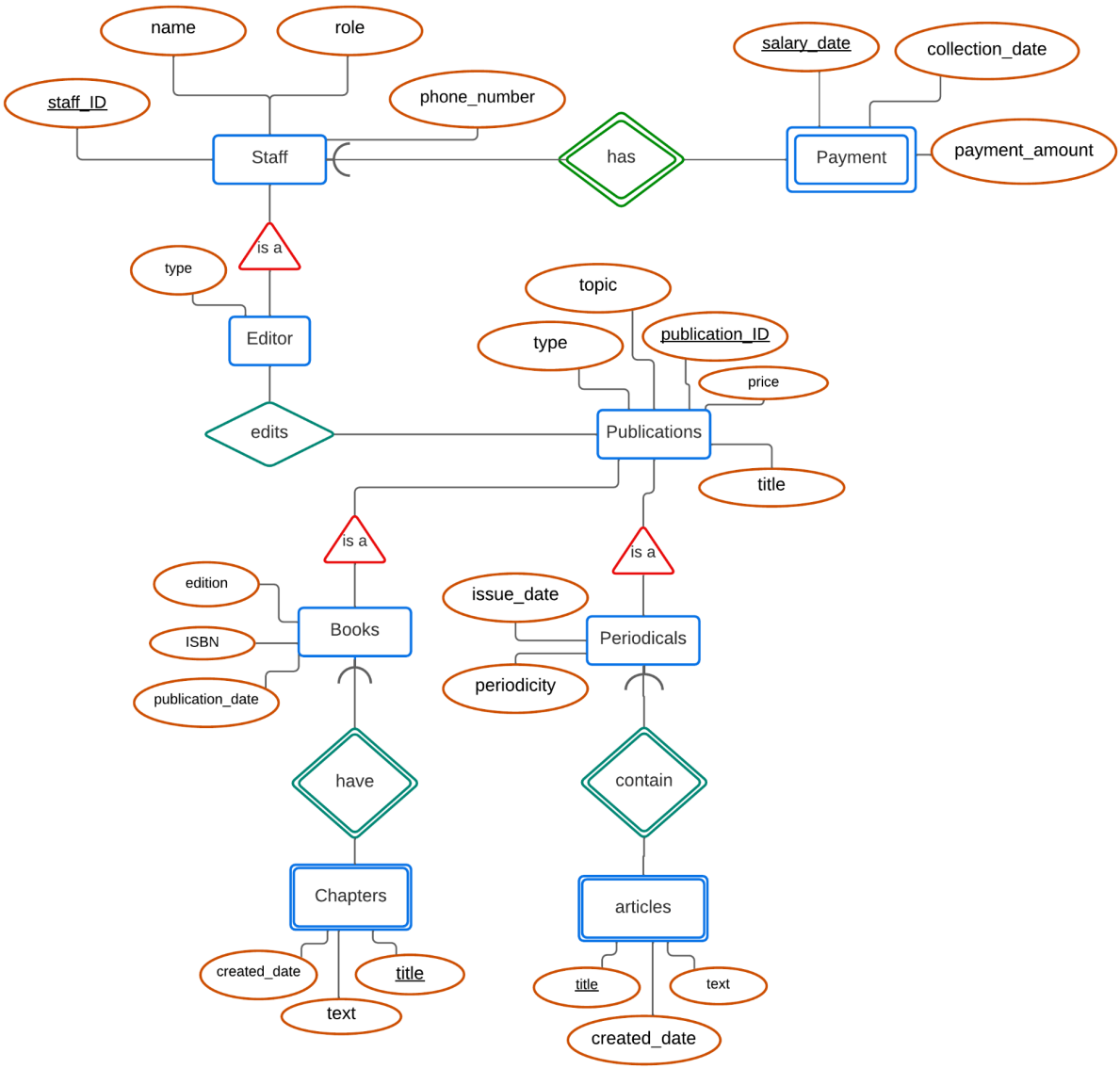
4. **Distributor** is responsible for placing orders to the publishing house. He can track the order, view the shipping cost and makes payments. Distributor should be able to view the available publications from a publishing house.

8. Local E/R Diagrams:

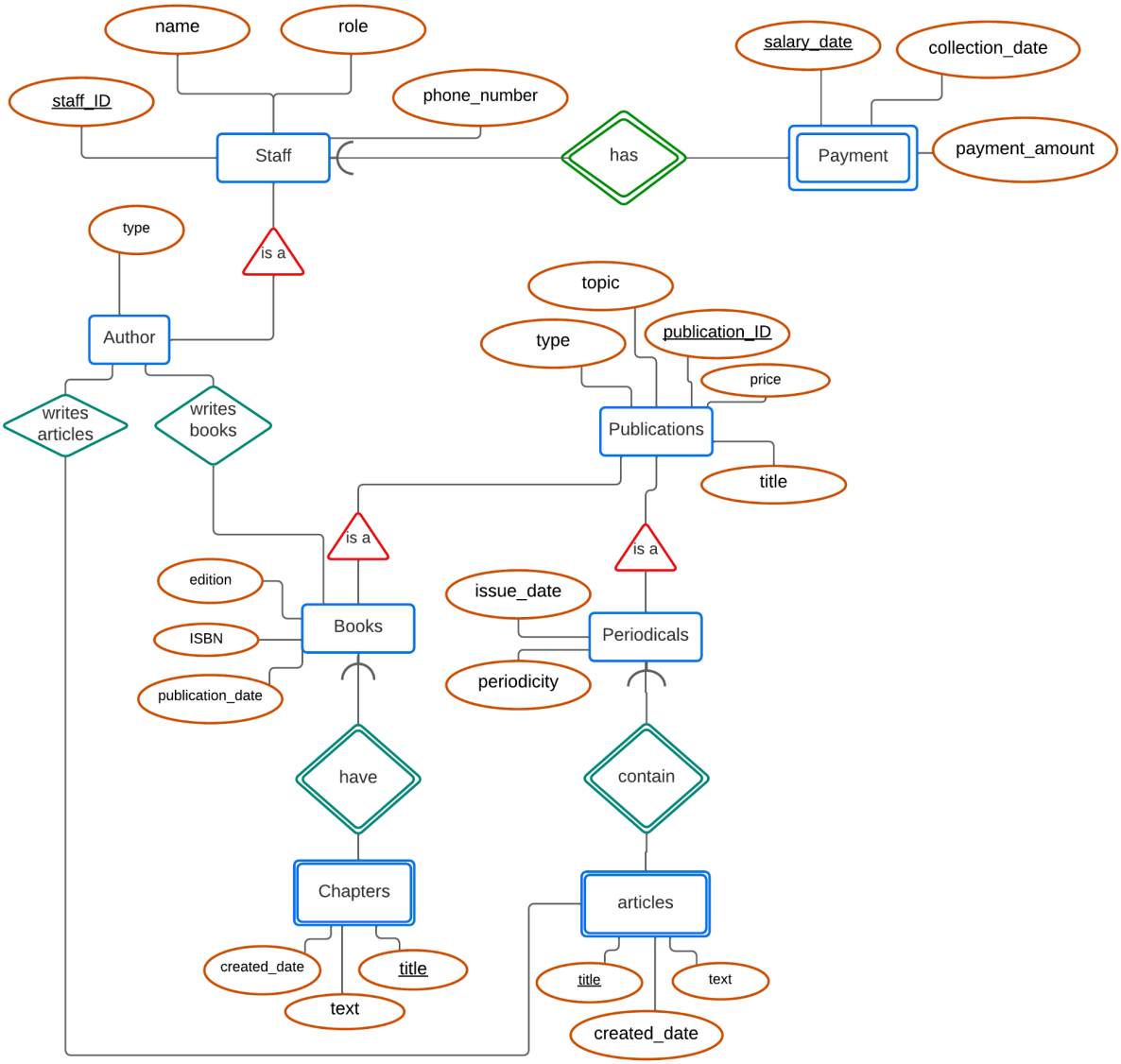
Admin View



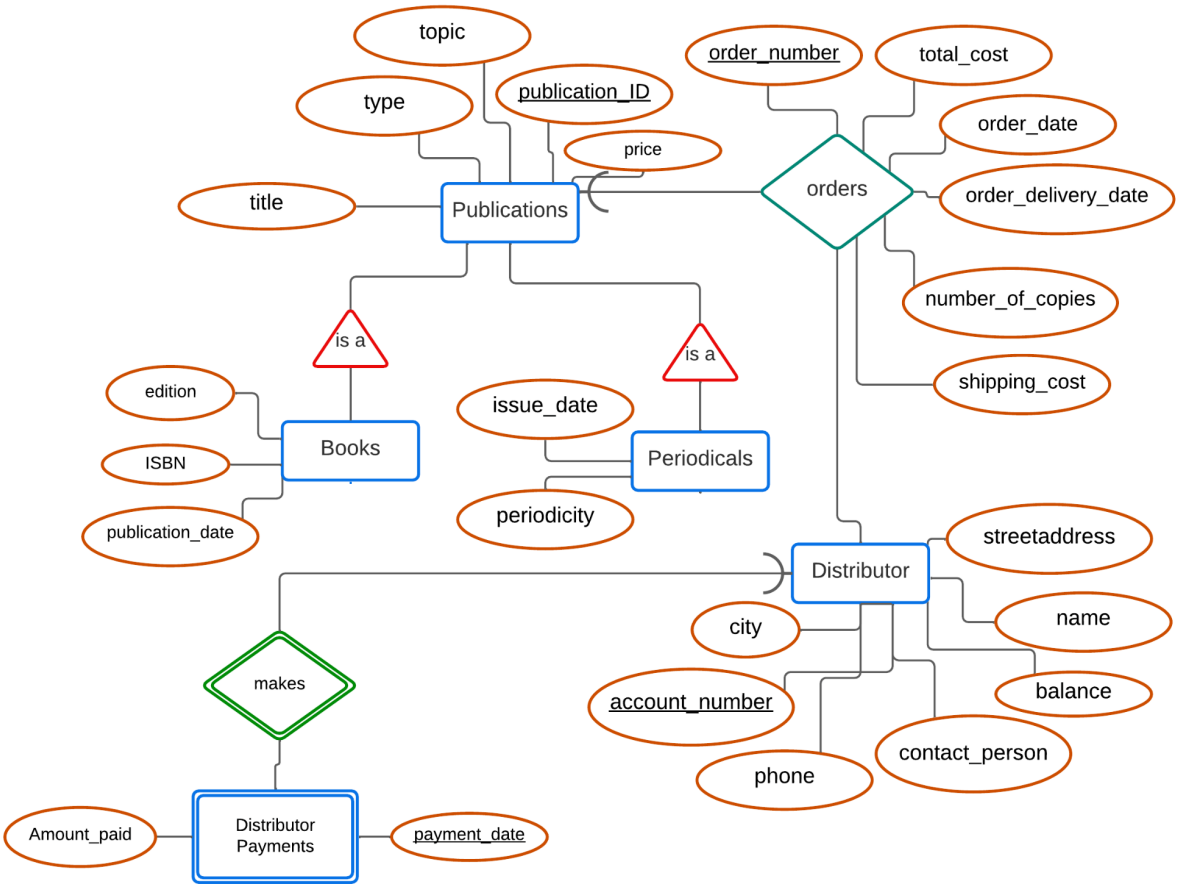
Editor View


name
role
staff_ID
phone_number
Staff
has
salary_date
collection_date
Payment
payment_amount
is a
type
Editor
topic
type
publication_ID
price
edits
Publications
title
is a
is a
edition
ISBN
publication_date
Books
issue_date
periodicity
Periodicals
have
contain
Chapters
articles
created_date
title
text
title
text
created_date

Author View


name
role
staff_ID
phone_number
Staff
has
salary_date
collection_date
Payment
payment_amount
type
is a
Author
writes articles
writes books
topic
type
publication_ID
price
Publications
title
is a
is a
edition
ISBN
publication_date
Books
issue_date
periodicity
Periodicals
have
contain
Chapters
articles
created_date
title
text
title
text
created_date

Distributor View



9. Description of Local ER Diagrams:

1. Editors and authors are the staff working for the publishing house.
2. The staff entity has staff_ID as the primary key, which is used to identify the users uniquely. We also included the role field which differentiates the author from editor.
3. Editors and authors are subclasses of the staff entity. Hence, they inherit all the properties of staff. Authors and Editors have a field called type, which is used to differentiate between a permanent employee and an invited guest.
4. We have an entity called publications which is used to store the attributes of a all the publications that belong to the publishing house. This contains attributes like publication_id (which is the key), type of the publication (book or a periodical), topic (genre) and price of the publication.
5. Books are written by authors. An author can have multiple books under their name, to capture which, we introduced the "writes books" relation. A book is a subclass of publication. It can be uniquely identified using a publication_ID which is inherited from the superclass. Book also has an ISBN, which is unique to each book.
6. We have introduced an entity called periodicals which contains articles. Authors also write periodicals. A periodical is also a subclass of Publication. An author can write multiple publications to capture which we included the "writes articles" relation. The publication_ID is used as a key to uniquely identify the periodical.
7. Editor is a subclass of staff. Editor has access to edit and view all the publications including books, periodicals, chapters and articles. To capture this behavior we have "edits" as a relation from editor to publication. Since an editor can edit multiple publications and a publication can have multiple editors, we have a many to many relationship between these entities.
8. A chapter is a weak entity set as a chapter doesn't exist without a book. Each book can have multiple chapters but a chapter can belong to only a single book. To capture this relation, we introduced the have relation. Each chapter in the book can be identified using its title and it's book's publication_ID. It has other attributes like created_date and text (content in the chapter).
9. An article is a weak entity set as an article cannot exist without a periodical. Every periodical can have multiple articles but an article can belong to only one periodical. To capture this behavior we introduced the contain relation. Each article in the periodical can be identified using its title and it's periodicals' publication_ID. It has other attributes like created_date and text (content in the article).
10. Payment is a weak entity set since the payment cannot exist without staff. Payments are associated with the staff. Each Staff member can have multiple payments, and each payment can be identified using the salary date field. We made the salary_date field the key to uniquely identify the payments made to the staff. Since each payment is unique to

a particular staff member, the staff ID and the salary date together can uniquely identify the payment record.

11. A distributor is not a member of the staff since they are not part of the publishing house. Each distributor is given a unique account_number, which is the key for the distributor entity. A distributor can place multiple orders to the publishing house which is why we included a relation called **orders**. However, since in a single order, a distributor can only order a single publication, we have a one to many relationship between the two entities. We identify each order uniquely using the order_id. Order also includes other fields such as total_cost (order cost), number_of_copies, order_delivery_date etc. These values are used to store details about all the orders placed. Each distributor can make multiple payments to the publishing house on different dates but each distributor_payment can only belong to a single distributor which is why there is a one to many relationship between the distributor and distributor_payments.
12. A distributor does not not need access to chapters in the book and articles in a periodical as he is only responsible for placing orders to the publications.
13. Distributor payment is a weak entity set since it cannot exist without a distributor. It can be uniquely identified using the payment date and distributor account number. A distributor makes payments to the publishing house, to capture which, we included a relation calles makes.

10. Local Relational Schemas:

Admin View:

Staff(staff_ID, name, role, phone_number)

Editors(staff_ID, type)

Authors(staff_ID, type)

Edits(staff_id, publication_id)

Publications(publication_id, title, topic, type, price)

Books(publication_id, ISBN, Edition, publication_date)

Periodicals(publication_id, issue_date, periodicity)

Chapters(publication_id, title, text)

Articles(publication_id, title, text, creation_date, topic)

Payment(staff_ID, salary_date, payment_amount, collection_date)

WritesBook(staff_id, publication_id)

WritesArticles(staff_id, publication_id, title)

Payment(staff_ID, salary_date, payment_amount, collection_date)

Distributor(account_number, phone, city, street_address, type, name, balance, contact_person)

Distributorpayments(account_number, payment_date, amount_paid)

Orders(order_number,publication_id, order_date, order_delivery_date, total_cost, shipping_cost)

Editor View

Staff(staff_ID, name, role, phone_number)

Publications(publication_ID, title, topic, type, price)

Books(publication_id, ISBN, Edition, publication_date)

Periodicals(publication_id, issue_date, periodicity)

Chapters(publication_id, title, text)

Articles(publication_id, title, text, creation_date, topic)

Payment(staff_ID, salary_date, payment_amount, collection_date)

Author View

Staff(staff_ID, name, role, phone_number)

Publications(publication_ID, title, topic, type, price)

Books(publication_id, ISBN, Edition, publication_date)

Periodicals(publication_id, issue_date, periodicity)

Chapters(publication_id, title, text)

Articles(publication_id, title, text, creation_date, topic)

WritesBook(staff_id, publication_id)

WritesArticles(staff_id, publication_id, title)

Payment(staff_ID, salary_date, payment_amount, collection_date)

Distributor View

Distributor(account_number, phone, city, street_address, type, name, balance, contact_person)

Distributorpayments(account_number, payment_date, amount_paid)

Orders(order_number,publication_id, order_date, order_delivery_date, total_cost, shipping_cost)

Publications(publication_ID, title, topic, type, price)

Books(publication_id, ISBN, Edition, publication_date)

Periodicals(publication_id, issue_date, periodicity)

11. Local Schema Documentation

1. In order to reduce redundancy, we created an entity called staff and inherited entities author and editor. Since, these entities inherit all the properties of the staff class, there is no need to define these fields on them again.

2. Payment can be an attribute in the Staff relation itself, instead we have decided to introduce a new payment entity with salary_date as a key which can help us identify a unique payment record.

3. We are providing a flexibility for a writer to be an author or a journalist, so we have created one single entity 'Author' which has two different relations 'write articles' and 'writes books' to articles and books entities respectively. By doing this we are removing redundancy by avoiding additional entity for journalist.

4. We have introduced a 'Distributor Payment' entity to capture multiple payments made by the distributor to the publishing house. We have used payment_date as key because a distributor can make multiple payments on different dates.

.

5. The 'Books' entity doesn't have any key because it inherits the 'publication_id' from 'Publication' entity which can be used as a key. The 'ISBN' attribute can have a unique constraint on it instead of itself being a key.

6. An 'Editor' can edit both 'Books' and 'Articles'. For this reason, we have one 'edits' relation between the 'Editor' and 'Publication', instead of having two separate relations. Since both 'Books' and 'Articles' come from 'Publication', having one single relation between 'Editor' and 'Publication' removes redundancy.

7. Every staff member (either invited or permanent staff) can have at least one payment. For this reason we have used the cardinality property between 'Staff' and 'Payment' with a rounded arrow towards 'Staff' entity.

8. Every Distributor can make at least one payment. For this reason, we have used the cardinality property between 'Distributor' and 'Distributor Payments' with a rounded arrow towards 'Distributor' entity.

9. To track the orders made by the 'Distributor', we maintain a relation 'orders', that can help us store details about the orders placed. Since a distributor can place an order for a single publication only, but a publication can be ordered by multiple distributors, we have a many to one relation between the two entities.