COMP 3005 Project Report (Fall 2022)

COMP 3005: Database Management Systems

Instructor: Ahmed El-Roby and Abdelghny Orogat

Spencer Low (101114608), Kaushik Paul (101158930)

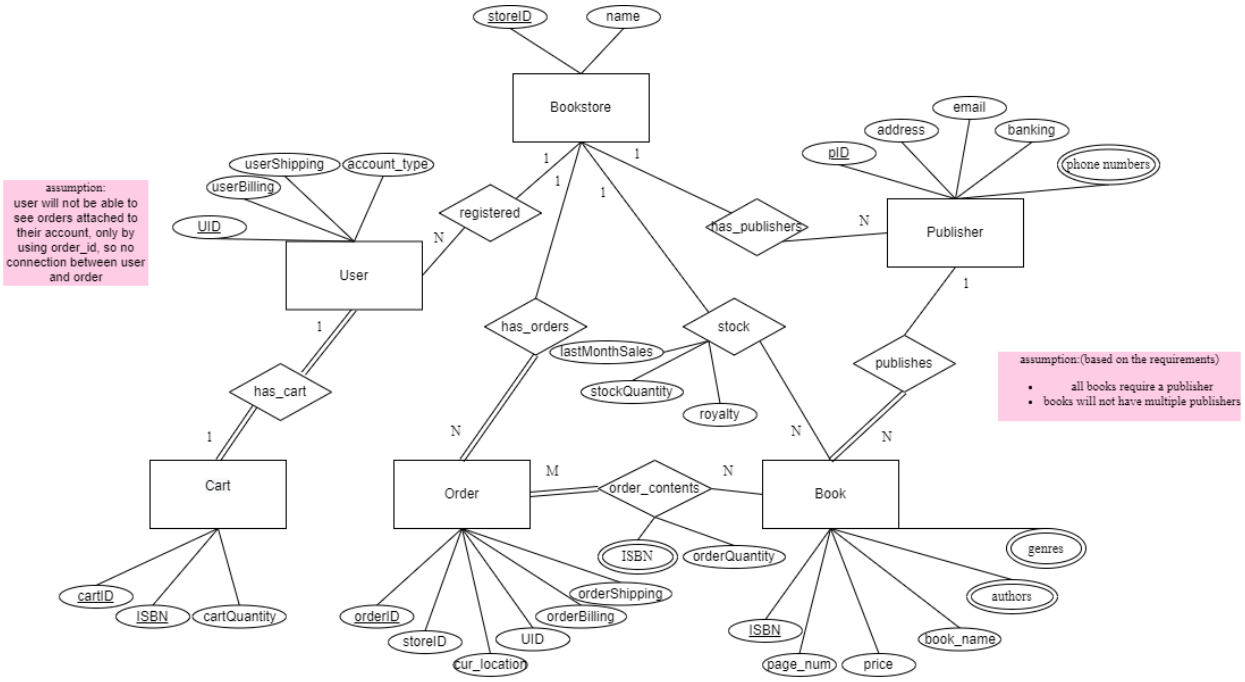
December 9th, 2022

https://github.com/spencerlow/bookstore-comp3005final

2.1 Conceptual Design

1.1 Introduction

The purpose of this project is to design and implement an online bookstore application using a database management system (DBMS). This section provides information explaining the conceptual design of the database using diagrams and justifications to any assumptions created.



1.2 Entities

This section will define all created entities within the system and justify design choice. Entities include:

a) Bookstore
   i) This is the main store entity of our system, everything will branch off of this entity.
   ii) Relation: Bookstore{storeID, name}

b) User
   i) This entity manages a user, it will comprise all necessary details a user will have upon registering with the bookstore.
   ii) Relation: User{UID, userBilling, userShipping, account_type}

c) Cart
   i) This entity manages a users cart, it will store information regarding the contents a user would like to checkout with.

ii) Relation: Cart{cartID, ISBN, cartQuantity}

d) Order
   i) This entity composes of attributes that a order will contain.
   ii) Relation: Order{orderID, storeID, cur_location, UID, orderBilling, orderShipping}

e) Publisher
   i) This entity defines a publisher, necessary information about the address, email, banking, and books published will populate this entity.
   ii) Relation: Publisher{pID, address, email, banking}

f) Book
   i) A book entity will contain information about the book.
   ii) Relation: Book{ISBN, page_num, price, book_name}

1.3 Relationships sets

This section will define our relationships and justify our design choice.

Our relationships sets are as follow:

1. Bookstore ↔ User *(relationship: registered)*
   a. This relationship set is **one-to-many** from Bookstore to User.
      a1. **Bookstore** can have **many** users,
      because there is no restriction on how many users can use a bookstore.
      a2. **users** will only have **one** bookstore,
      because the database is designed for one specific online bookstore from one specific warehouse.
   b. Bookstore is in **partial participation.**
      b1. A bookstore entity does not need to have registered users.
   c. User is in **partial participation.**
      c1. Some entity users may not be registered to a bookstore but can still browse the site.

2. Bookstore ↔ Order *(relationship: has_orders)*
   a. This relationship set is **one-to-many** from Bookstore to Order.
      a1. **Bookstore** can have **many** orders,
      because there is no restriction on how many orders a bookstore may have.
      a2. An **Order** can only have **one** bookstore,
      because an order cannot be from multiple bookstores.
   b. Bookstore is in **partial participation**
      b1. A bookstore entity does not need to have orders.
   c. Order is in **total participation**
      c1. An order must be handled by a bookstore.

3. Bookstore ↔ Book *(relationship: stock)*
   a. This relationship set is **one-to-many** from Bookstore to Book.
      a1. **Bookstore** can have **many** books,
      Because the store isnt restricted to only selling one book to customers.
      a2. A **book** will only have **one** bookstore,
      because the database is designed for one specific online bookstore from one specific warehouse.
   b. Bookstore is in **partial participation**
      b1. A bookstore entity does not need to have books in stock.
   c. Book is in **partial participation**
      c1. A book does not need to be stocked by a bookstore

4. Bookstore ↔ Publisher *(relationship: has_publishers)*
   a. This relationship set is **one-to-many** from Bookstore to Publisher.
      a1. **Bookstore** can have **many** publishers,
          because there is no restriction on how many publishers a bookstore chooses to sell for.
      a2. **Publisher** can have only **one** bookstore,
          because based on the requirements and design, there can only be one bookstore, thus a publisher will only be connected to one bookstore.
   b. Bookstore is in **partial participation**
      b1. A bookstore entity does not need to have publishers .
   c. Publisher is in **partial participation**
      c1. A publisher entity does not need to have a bookstore displaying its works.

5. Publisher ↔ Book *(relationship: publishes)*
   a. This relationship set is **one-to-many** from Publisher to Book.
      a1. **Publisher** can have **many** books published,
          nothing restricts a publisher to only publish one book.
      a2. **Book** can only have **one** publisher,
          this is based on the requirements and assumption that each book has one publisher.
   b. Publisher is in **partial participation**
      b1. A publisher does not need to publish books.
   c. Book is in **total participation**
      c1. All books need to have a publisher.

6. Order ↔ Book *(relationship: order_contents)*
   a. This relationship set is **many-to-many** from Order to Book.
      a1. An **Order** can have **many** books to deliver,
          Because an order can consist of several different ISBN books
      a2. A **Book** can have **many** orders from which it is requested,
          Because several orders can have the same book, once, to deliver
   b. Order is in **total participation**
      b1. An order entity must consist of items, otherwise nothing is request and there is no order
   c. Book is in **partial participation**
      c1. A book entity does not need to be in any orders, and can remain unrequested

7. User ↔ Cart *(relationship: has_cart)*
   a. This relationship set is **one-to-one** from User to Cart.
      a1. **User** can only have **one** cart,

design of the database only allows to track one cart per user.

a2. **Cart** can only have **one** user,
multiple users can not share the same cart.

b. Useris in **total participation**

b1. A user is required to have a cart, even if empty.

c. Cartis in **total participation**

c1. All carts need to be attached to a user, can't have filled carts with a null user.

1.4 Assumptions

Assumption	Justification
"Registered" relation	Asumming a user is not registered, this table is created to prevent NULL values in the 'User' relation.
Reports are requested and calculated with the request, not stored as a entity.	The requirements do not state to have a separate relation to store the history of all types of sale data, thus no new relations will be made, and reports will be given through subqueries and aggregate functions.
All books require a publisher and will not have multiple publishers.	This is based off of the problem statement, but explicitly stated here. We are assuming that all books require a publisher.
Users will only view orders through the order number, so we dont have to have an attribute that ties user with their order.	Problem statement: "A user can use this order number to track where the order is currently. Although shipping is carried out by a third-party shipping service, the online bookstore should have the tracking information available for when the user inquires about an order using the order number."

2.2 Reduction to Relation Schemas

User={UID, userBilling, userShipping, account_type}
Stock={storeID, ISBN, stockQuantity, royalty, lastMonthSales}
Cart={cardID, ISBN, cartQuantity}
has_Cart={cartID, UID}
book={ISBN, page_num, price, book_name}
Bookstore={storeID, name}
Registered={storeID, UID}
Order_contents={ISBN, orderID, orderQuantity}
Order={orderID, storeID, cur_location, UID, orderBilling, orderShipping}
publisher={pID, address, email, banking}
has_authors={ISBN, author}
Has_genres={ISBN, genre}
publishes={ISBN, pID}
has_numbers={pID, phoneNumber}

2.3 Normalization of Relation Schemas

Functional Dependencies

F={
UID → userBilling,
UID → userShipping,
UID → account_type,
UID → cartID,
UID → storeID,

ISBN → stockQuantity,
ISBN → royalty,
ISBN → lastMonthSales,
ISBN → Page_num,
ISBN → Price,
ISBN → pID,
ISBN → book_name

UID, ISBN → cartQuantity,

cartID → UID,
storeID → name,
orderID, ISBN → orderQuantity,
orderID → storeID,
orderID → cur_location,
orderID → UID,
orderID → orderBilling,

orderID → orderShipping,

pID → address,
pID → email,
pID → banking,

Phone_numbers→ pID,
Phone_numbers → pID

ISBN,Author → Page_Num,
ISBN,Author→ Price,

ISBN, Genre → page_num,
ISBN, Genre → price,
}

Canonical
Unionize

Let A = UID
Let B = userBilling
Let C = userShipping
Let D = account_type
Let E = cartID
Let F = storeID
Let G = ISBN
Let H = stockQuantity
Let I = royalty
Let J =lastMonthSales
Let K =Page_num
Let L =Price
Let M =pID
Let N = cartQuantity
Let O =name
Let P = orderID
Let Q = orderQuantity
Let R = cur_location
Let S = orderBilling
Let T = orderShipping
Let U = address
Let V = email
Let W = banking
Let X = phone_numbers
Let Y = author
Let Z = Genre
Let Δ = book_name

F_C={
UID → userBilling,userShipping,account_type,cartID,storeID
ISBN → stockQuantity, royalty, lastMonthSales, Page_num, Price, pID, book_name
UID, ISBN → cartQuantity,
cartID → UID,
storeID → name,
orderID, ISBN → orderQuantity,
orderID → storeID, cur_location, UID, orderBilling, orderShipping,
pID → address, email, banking,
Phone_numbers→ pID
ISBN,Author → Page_Num, Price
ISBN, Genre → page_num, price
}

Replacing attributes with Variables:
E,F,G,K,L,M,A,P = check if the follow values are extraneous (they show up more than once)
If **left side is paired**, check if they are extraneous

F_C={
F_1 = A → B,C,D,E,F //**NO EXTRANEOUS**

F_2 = G → H, I, J, K, L, M, Δ //**NO EXTRANEOUS,** Δ only shows up once on RHS in all dep.

F_3 = A, G → N, //**NO EXTRANEOUS**

F_4 = E → A, //**NO EXTRANEOUS**

F_5 = F → O, //**NO EXTRANEOUS**, O appears once in all dep., LHS is not paired

F_6 = P, G → Q, //**NOT EXTRANEOUS**, Q appears once in all dep., LHS are not extraneous

F_7 = P → F, R, A, S, T, // **EXT = F**

F_8 = M → U, V, W, // **NO EXTRANEOUS**

F_9 = X → M // **NO EXTRANEOUS**

F_{10} = G,Y → K, L // **EXT = K,L**

F_{11}= G, Z → K, L // **EXT = KL**
}

Remove extraneous / minimal cover proof

E,F,G,K,L,M,A,P = check if the follow values are extraneous (they show up more than once)
If **left side is paired**, check if they are extraneous

Functional Dependency	Extraneous Attribute Computation

F_1 = A → B,C,D,E,F	checking A → B,C,D,E,F
	check if A+ contains removed attribute (F)
	
	Fc'={
	A → B,C,D,E
	…
	}
	
	A+
	Result = {A}
	
	A → B,C,D,E
	Result = {A,B,C,D,E}

	E → A,
	Result = {A,B,C,D,E}
	
	G → H, I, J, K, L, M,Δ
	A, G → N,
	F → O,
	P, G → Q,
	P → F, R, A, S, T,
	M → U, V, W,
	X → M
	G,Y → K, L
	G, Z → K, L
	
	No changes to result, F is not extraneous
	
	--- 	
	
	checking A → B,C,D,E,F
	check if A+ contains removed attribute (E)
	
	Fc' =
	{
	A → B,C,D,F
	…
	}
	
	A+
	result = {A}
	
	A → B,C,D,F
	result = {A,B,C,D,F}
	
	G → H, I, J, K, L, M,Δ
	A, G → N,
	E → A,
	
	F → O,
	result = {A,B,C,D,F,O}
	
	P, G → Q,
	P → F, R, A, S, T,
	M → U, V, W,
	X → M
	G,Y → K, L
	G, Z → K, L
	
	No changes to result, E is not extraneous

F_2 = G → H, I, J, K, L, M,Δ	checking G → H,I,J,K,L,M,Δ
	check if G+ contains removed attribute (M)
	Fc' =
	{
	G → H,I,J,K,L,Δ
	…
	}
	
	G+
	result = {G}
	
	G → H, I, J, K, L,Δ

result = {G,H,I,J,K,L,Δ}

A → B,C,D,E,F
A, G → N,
E → A,
F → O,
P, G → Q,
P → F, R, A, S, T,
M → U, V, W,
X → M
G,Y → K, L
G, Z → K, L

No changes to result, M is not extraneous

---

checking G → H,I,J,K,L,M,Δ
check if G+ contains removed attribute (L)
Fc' =
{
G → H,I,J,K,M,Δ
…
}

G+
result = {G}

G → H, I, J, K, M,Δ
result = {G,H,I,J,K,M,Δ}

M → U, V, W,
result = {G,H,I,J,K,M,Δ,U,V,W}

A → B,C,D,E,F
A, G → N,
E → A,
F → O,
P, G → Q,
P → F, R, A, S, T,
X → M
G,Y → K, L
G, Z → K, L

No changes to result, L is not extraneous

---

checking G → H,I,J,K,L,M,Δ
check if G+ contains removed attribute (K)
Fc' =
{
G → H,I,J,L,M,Δ
…
}

G+
result = {G}

G → H, I, J, L, M,Δ
result = {G,H,I,J,L,M,Δ}

M → U, V, W,
result = {G,H,I,J,L,M,Δ,U,V,W}

	A → B,C,D,E,F A, G → N, E → A, F → O, P, G → Q, P → F, R, A, S, T, X → M G,Y → K, L G, Z → K, L **No changes to result, K is not extraneous**
F_3 = A, G → N	CHECKING A,G->N **REMOVE A** **G->N CHECK IF G+ CONTAINS N in Fc** Result = {G} G → H, I, J, K, L, M,Δ Result = {G,H,I,J,K,L,M,Δ} M → U, V, W RESULT = G,H,I,J,K,L,M,Δ,U,V,W A → B,C,D,E,F F → O, A, G → N E → A, P, G → Q, P → F, R, A, S, T X → M G,Y → K, L G, Z → K, L **No changes to result, A is not extraneous** --- CHECKING A,G->N **REMOVE G** **A->N CHECK IF A+ contains N in Fc** Result = {A} A → B,C,D,E,F Result = {A,B,C,D,E,F} F → O, RESULT = A,B,C,D,E,F,O G → H, I, J, K, L, M,Δ A, G → N E → A, P, G → Q, P → F, R, A, S, T M → U, V, W X → M G,Y → K, L G, Z → K, L **No changes to result, G is not extraneous**

F_4 = E → A	CHECKING E → A
	check if E+ contains removed attribute (A)
	Fc' =
	{
	E →
	…
	}
	
	E+
	result = {E}
	
	A → B,C,D,E,F
	G → H, I, J, K, L, M,Δ
	A, G → N,
	E → ,
	F → O,
	P, G → Q,
	P → F, R, A, S, T,
	M → U, V, W,
	X → M
	G,Y → K, L
	G, Z → K, L
	
	No changes to result, A is not extraneous

F_5 = F → O	**No extraneous** **1. Nothing implies O except F** **2. F cannot be removed from LHS, or else nothing will imply O**
F_6 = P, G → Q	CHECKING P,G → Q **remove G, check if P+ contains Q** **P+** Result = {P} using original Fc: P → F, R, A, S, T, result = {P,F,R,A,S,T} A → B,C,D,E,F result = {P,F,R,A,S,T,B,C,D,E} E → A, result = {P,F,R,A,S,T,B,C,D,E} F → O, result = {P,F,R,A,S,T,B,C,D,E,O} G → H, I, J, K, L, M,Δ A, G → N, P, G → Q, M → U, V, W, X → M G,Y → K, L G, Z → K, L **does not contain Q, G is not extraneous**

	CHECKING P,G → Q **remove P, check if G+ contains Q** **G+** result = {G} using original Fc: G → H, I, J, K, L, M,Δ result = {G,H,I,J,K,L,M,Δ} M → U, V, W, result = {G,H,I,J,K,L,M,Δ,U,V,W} A → B,C,D,E,F A, G → N, E → A, F → O, P, G → Q, P → F, R, A, S, T, X → M G,Y → K, L G, Z → K, L **Does not contain Q, P is not extraneous**
F_7 = P → F, R, A, S, T	**check if P+ contains removed attribute {F}** Fc' = {P → ..., R, A, S, T,} P+ result = {P} A → B,C,D,E,F **result2 = {PRASTBCDEF}** G → H, I, J, K, L, M,Δ A, G → N, E → A, F → O, P, G → Q, P → ..., R, A, S, T, result = {PRAST} M → U, V, W, X → M G,Y → K, L G, Z → K, L result2 = result (Start at top) **F = EXTRANEOUS as F is contained in result2** --- **check if P+ contains removed attribute {A}** Fc' = {P → F, R, ..., S, T,} P+ result = {P} A → B,C,D,E,F G → H, I, J, K, L, M,Δ

	A, G → N, E → A, F → O, **result2 = {PFRSTO}** P, G → Q, P → F, R, ..., S, T, result = {PFRST} M → U, V, W, X → M G,Y → K, L G, Z → K, L result2 = result (Start at top) **A= NOT as A is not contained in result2**
F_8 = M → U, V, W	**No extraneous** **1. Nothing implies UVW except M** **2. M cannot be removed from LHS, or else nothing will imply UVW**
F_9 = X → M	**check if X+ contains removed attribute {M}** Fc' = {X → M} X+ result = {X} A → B,C,D,E,F G → H, I, J, K, L, M,Δ A, G → N, E → A, F → O, P, G → Q, P → F, R, A, S, T, M → U, V, W, X → ... G, Y → K, L G, Z → K, ... **result = {X}** **M = NOT as M is not contained in result**
F_{10} = G,Y → K, L	**check if GY+ contains removed attribute {K}** Fc' = {G,Y → ..., L} GY+ result = {GY} A → B,C,D,E,F G → H, I, J, K, L, M,Δ result = {GYHIJKLMΔ} A, G → N, E → A, F → O, P, G → Q, P → F, R, A, S, T,

	M → U, V, W, result = {GYHIJKLMΔUVW} X → M G,Y → ..., L **result = {GYHIJKLMΔUVW}** G, Z → K, L **K = EXT as K is contained in result** --- **check if GY+ contains removed attribute {L}** Fc' = {G,Y → K, ...} GY+ result = {GY} A → B,C,D,E,F G → H, I, J, K, L, M,Δ result = {GYHIJKLMΔ} A, G → N, E → A, F → O, P, G → Q, P → F, R, A, S, T, M → U, V, W, result = {GYHIJKLMΔUVW} X → M G,Y → K, … **result = {GYHIJKLMΔUVW}** G, Z → K, L **L = EXT as L is contained in result**
F_{11}= G, Z → K, L	**check if GZ+ contains removed attribute {K}** Fc' = {G, Z → ..., L} GZ+ result = {GZ} A → B,C,D,E,F G → H, I, J, K, L, M,Δ result = {GZHIJKLMΔ} A, G → N, E → A, F → O, P, G → Q, P → F, R, A, S, T, M → U, V, W, result = {GZHIJKLMΔUVW}

	X → M G, Y → K, L G, Z → ..., L result = {GZHIJKLMΔUVW} **K = EXT as K is contained in result** --- check if GZ+ contains removed attribute {L} Fc' = {G, Z → K, ...} GZ+ result = {GZ} A → B,C,D,E,F G → H, I, J, K, L, M,Δ result = {GZHIJKLMΔ} A, G → N, E → A, F → O, P, G → Q, P → F, R, A, S, T, M → U, V, W, result = {GZHIJKLMΔUVW} X → M G, Y → K, L G, Z → K, … **result = {GZHIJKLMΔUVW}** **L = EXT as L is contained in result**

Convert to 3NF Functional Dependencies

Updated canonical cover after removing all extraneous attributes

F_C={
F_1 = A → B,C,D,E,F
F_2 = G → H, I, J, K, L, M,Δ
F_3 = A, G → N
F_4 = E → A
F_5 = F → O
F_6 = P, G → Q
F_7 = P → R, A, S, T
F_8 = M → U, V, W
F_9 = X → M
}

Missing candidate keys and attributes Z,Y, create new relation

TESTING ATTRIBUTES THAT ONLY APPEAR ON THE LEFT SIDE, since we will need those attributes on the RHS to form new relation

G+ =
result={G}

F2 = G → H, I, J, K, L, M,Δ
result = {G,H,I,J,K,L,M,Δ}

F8 = M → U, V, W
result = {G,H,I,J,K,L,M,Δ,U,V,W}

F1 = A → B,C,D,E,F
F3 = A, G → N
F4 = E → A
F5 = F → O
F6 = P, G → Q
F7 = P → R, A, S, T
F9 = X → M

NO CHANGES IN RESULT
result = {G,H,I,J,K,L,M,Δ,U,V,W}
G+ = {G,H,I,J,K,L,M,Δ,U,V,W}

P+ =
result={P}

F7 = P → R, A, S, T
result={P,R,A,S,T}

F1 = A → B,C,D,E,F
result={P,R,A,S,T,B,C,D,E,F}

F5 = F → O
result={P,R,A,S,T,B,C,D,E,F,O}

F2 = G → H, I, J, K, L, M,Δ
F3 = A, G → N
F4 = E → A
F6 = P, G → Q
F8 = M → U, V, W
F9 = X → M

NO CHANGES IN RESULT
result = P,R,A,S,T,B,C,D,E,F,O
P+ = {P,R,A,S,T,B,C,D,E,F,O}

X+ =
result={X}

F9 = X → M
result={X,M}

F8 = M → U, V, W
result={X,M,U,V,W}

F1 = A → B,C,D,E,F
F2 = G → H, I, J, K, L, M,Δ
F3 = A, G → N

F4 = E → A
F5 = F → O
F6 = P, G → Q
F7 = P → R, A, S, T

NO CHANGES IN RESULT
result = X,M,U,V,W
X+ = {X,M,U,V,W}

Augment G,P,X TOGETHER
GPX -> {G,H,I,J,K,L,M,Δ,U,V,W,P,R,A,S,T,B,C,D,E,F,O,X}

AUGMENT
Y AND Z
GPXYZ -> {G,H,I,J,K,L,M,Δ,U,V,W,P,R,A,S,T,B,C,D,E,F,O,X,Y,Z}
Thus GPXYZ is the candidate key and superkey

Create new relations by unioning RHS and LFS, plus the relation with the candidate keys

R_1= {A, B, C, D, E, F}
R_2= {G, H, I, J, K, L, M,Δ}
R_3={A, G, N}
R_4= {E, A}
R_5= {F, O}
R_6= {P, G, Q}
R_7= {P, R, A ,S, T}
R_8= {M, U, V, W}
R_9= {X, M}
R_{10}={G,P,X,Y,Z}

If any schema Ri is contained another schema, delete it

R4 ⊂ R1: Delete R4

R_1= {A, B, C, D, E, F}
R_2= {G, H, I, J, K, L, M,Δ}
R_3={A, G, N}
R_4= {F, O}
R_5= {P, G, Q}
R_6= {P, R, A ,S, T}
R_7= {M, U, V, W}
R_8= {X, M}
R_9={G,P,X,Y,Z}

Lossless and dependency preservation

3NF is by definition lossless and dependency preserving.

Converting relation schemas and functional dependencies back to original attribute names

FC={

F1 = UID → userBilling,userShipping,account_type,cartID,storeID

F2 = ISBN → stockQuantity, royalty, lastMonthSales, Page_num, Price, pID, book_name

F3 = UID, ISBN → cartQuantity

F4 = cartID → UID

F5 = storeID → name

F6 = orderID, ISBN → stockQuantity

F7 = orderID → cur_location, UID, orderBilling, orderShipping

F8 = pID → address, email, banking

F9 = phone_numbers → pID

}

User = {UID, userBilling, userShipping, account_type, cartID, storeID}

Book= {ISBN, stockQuantity, royalty, lastMonthSales, Page_num, Price, pID, book_name}

Cart={UID, ISBN, cartQuantity}

Bookstore= {storeID, name}

Order_contents= {orderID, ISBN, orderQuantity}

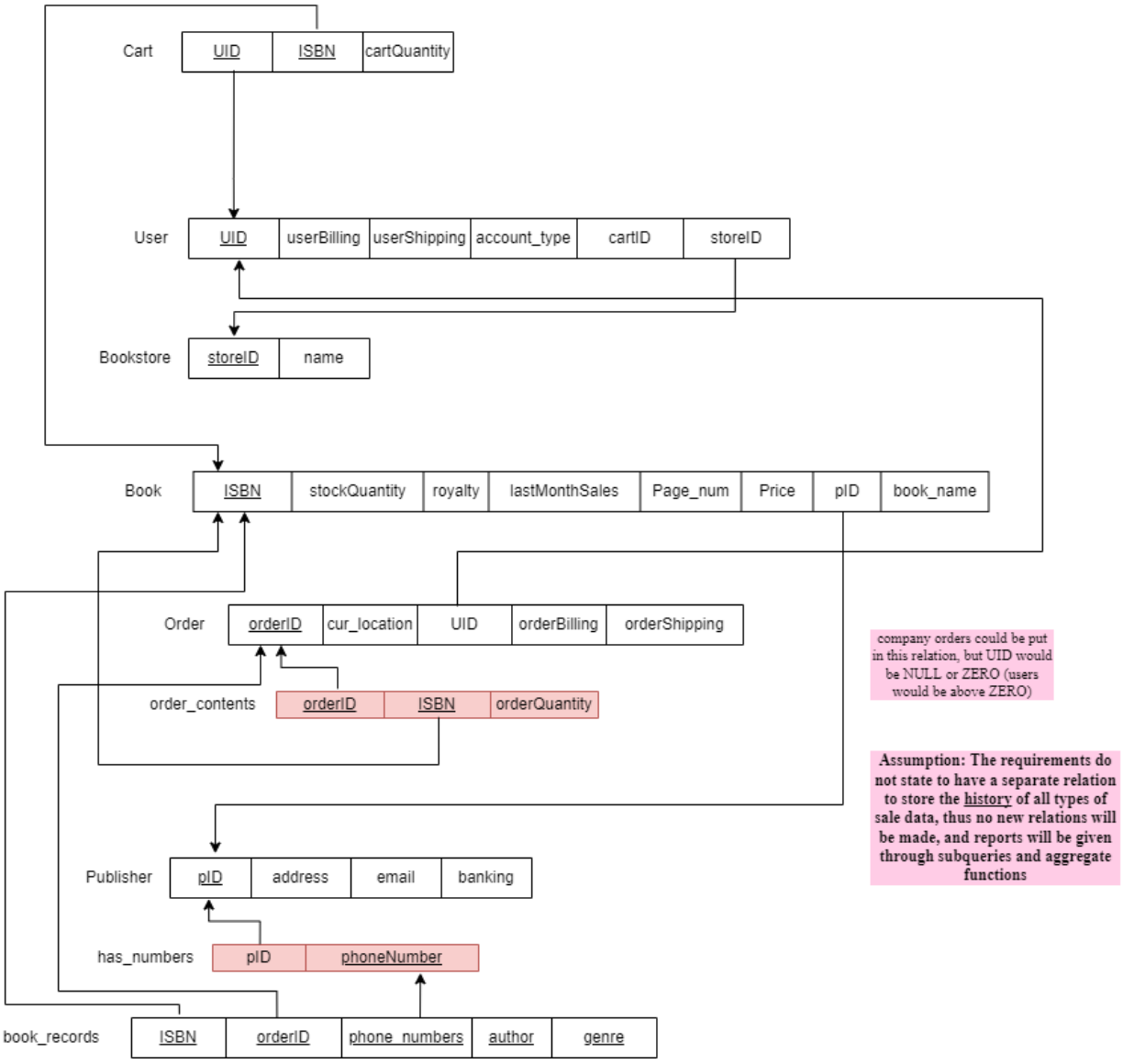
Order= {orderID, cur_location, UID ,orderBilling, orderShipping}

Publisher= {pID, address, email, banking}

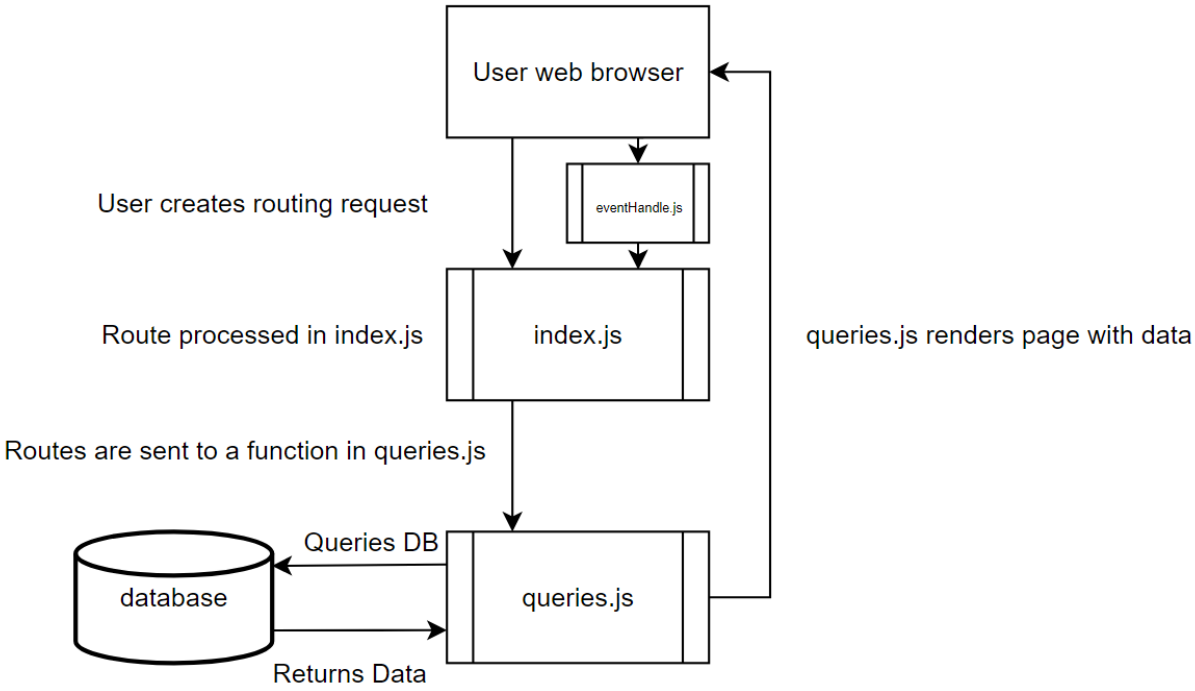
has_numbers= {phone_numbers, pID}

book_records={ISBN, orderID, phone_numbers, author,Genre}

2.4 Database Schema Diagram



2.5 Implementation



The web app was designed using NodeJS, Javascript, Express and node-Postgres (npm package that connects our server to the database) to handle the backend, pug to help render and display/format webpages and PostgreSQL 15.1 to act as our database server. Upon initialization, SQL files are run through a query that helps automatically setup all the relations in the database and populates them with mock data needed in order to simulate a real bookstore application. It follows a straightforward approach from receiving a request to serving pages. All requests are made through the browser and sent to `index.js` with specific html event handling sent to `eventHandle.js`. This file handles all routing, it catches the URL and calls an appropriate function within `queries.js`. The query file contains all functions related to querying the database and processing the returned data. Within the query file, there is pug file rendering whose purpose is to use the returned data + any modifications/additional parameters and serve a rendered pug file. This file is then sent as a response to the user, and their browser will update depending on what is served to them. (I.E: If a user hits checkout, a new cart + a re-rendered cart pug file is sent back as a response allowing them to make more orders)

2.6 Bonus features

Some of our bonus features are(but **not limited to)**:

- HTML redirection
- CSS customization
- Privilege based access
    - Only owners/admins can access control panel
- Fully interactable interface
- A nice Favicon.ico
- Uses Pug for clean and adjusts on-demand to changes while rendering of pages
- Dropdown menus
- Dynamic interface
    - Updates depending on the user you are
    - UserID at the top of site will signify which account you are
    - Only owners can see redirect buttons to admin control panel

2.7 Github

https://github.com/spencerlow/bookstore-comp3005final
Follow README.md for instructions on how to run application

File structure:
Bookstore-comp3005final
- ↪**diagrams**
- ↪**node-api-postgres** (SERVER + code + SQL files)
  - ↪**code**
  - ↪**sql**
- ↪**screenshots** (Screen grabs of customer vs owner screens)
- ↪**.gitignore**
- ↪**README.md**
- ↪**ProjectReport.pdf**