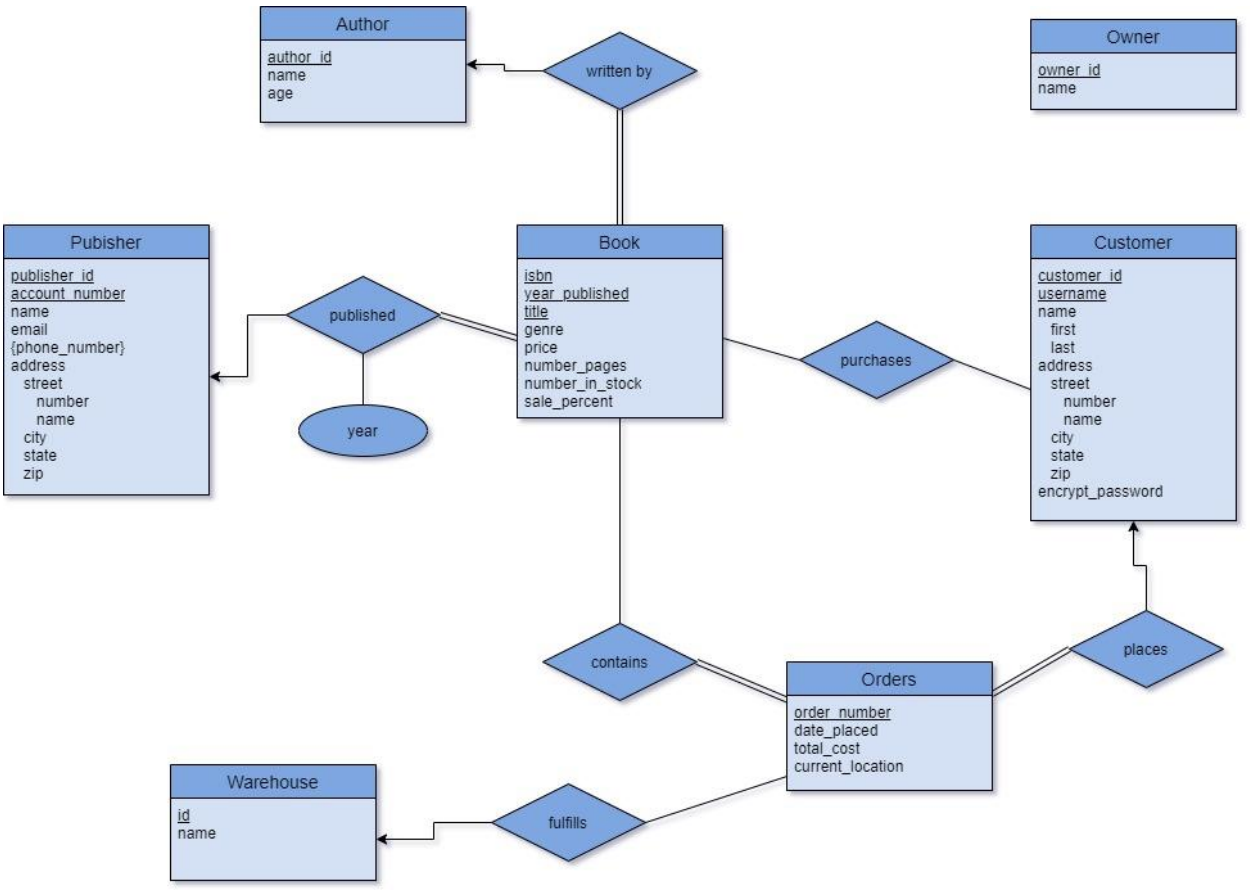
COMP 3005 Project Report

Jack and Daniels Books

Authored by: Riley Franolla 101071323

Conceptual Design



The conceptual design of the database is meant to include all the relevant information needed in order to “run” the bookstore. There are a few assumptions being made, one of them being that for a book to be included in the database, it is assumed that it must have one author and one publisher. This should not be against the statement and though not entirely accurate in real life (one book could have many authors) but in this world they do. Another assumption made was when reading the problem statement, the owner didn’t seem to have any real relations from a database perspective. The owner entity is strictly used for login purpose and all owner functionality is down on the application side. For the orders entity, an order must be placed by one customer, must contain books (can’t order nothing) and it must be fulfilled by one warehouse. Other than those assumptions the overall design was focused on the problem statement to meet all requirements

Reduction to Relation Schemas

owner(owner_id, name)

author(author_id, name)

book(isbn, title, year_published, genre, price, number_pages, number_in_stock, sale_percent, author_id, publisher_id)

publisher(publisher_id, account_number, name, email, street_name, street_number, city, state, zip)

publisher_phone(publisher_id, phone_number, type)

customer(customer_id, username, first_name, last_name, street_number, street_name, city, state, zip,encrypt_password)

orders(order_number, date_placed, total_cost, current_location, customer_id)

order_containes(isbn, order_number)

warehouse(id, name)

fulfillments(id, order_number)

Normalization of Relation Schemas

owner(owner_id, name):

F= {
owner_id, name → name
name → name
owner_id → name
}
This relation is in good normal form under BCNF as the first two functional dependencies are both trivial as name is a subset of both [owner_id, name] and [name]. The third dependency alpha, or in this case owner_id, is a superkey since there are no tuples with the same owner_id

author(author_id, name):

F= {
author_id, name → name
name → name
author_id → name
}
This relation is in good normal form under BCNF as the first two functional dependencies are both trivial as name is a subset of both [author_id, name] and [name]. The third dependency alpha, or in this case author_id, is a superkey since there are no tuples with the same author_id

book(isbn, title, year_published, genre, price, number_pages, number_in_stock, sale_percent, author_id, publisher_id):

F= {
isbn, title → year_published, genre, number_pages, number_in_stock , author_id, publisher_id
isbn, title, year_published → genre, number_pages, number_in_stock, author_id, publisher_id
}
This relation is in good normal form under BCNF as both the function dependencies, the left-hand side is a superkey. The isbn, by definition, is a unique number that identifies the different editions of books, even the same one. So, no two books will every have the same isbn, thus the isbn determines every important attribute of a book.

publisher(publisher_id, account_number, name, email, street_name, street_number, city, state, zip):

F= {
publisher_id → account_number, name, email, street, city, zip
city → zip, street_name
state → city
}
This relation is in good normal for under 3NF. This first functional dependency has alpha as its suprekey. The publisher_id is unique, and no two tuples will have the same id. The last two dependencies follow the rule of "Each attribute A in (beta – alpha) is contained in a candidate key for R (each attribute can be in a different candidate key)". (beta – alpha = zip, street_name) for the first dependency and (beta – alpha = city) for the third are both contained under the candidate key (street_name, street_number, city, state, zip) since the address can uniquely determine a publisher as no two publishers will have the same address.

publisher_phone(publisher_id, phone_number, type):

F= {
publisher_id, phone_number → phone_number
}
This relation is in good normal form under BCNF as it only has one functional dependency that is trivial as phone_number is a subset of [publisher_is, phone_number].

customer(customer_id, username, first_name, last_name, street_number, street_name, city, state, zip, encrypt_password):

F= {
customer_id → username, name, street_number, street_name, city, state, zip, encrypt_password
name, city → zip, street_name
name, state → city
}
This relation is in good normal for under 3NF. This first functional dependency has alpha as its suprekey. The customer_id is unique, and no two tuples will have the same id. The last two dependencies follow the rule of "Each attribute A in (beta – alpha) is contained in a candidate key for R (each attribute can be

in a different candidate key)". (beta – alpha = zip, street_name) for the first dependency and (beta – alpha = city) for the third are both contained under the candidate key (street_name, street_number, city, state, zip) since the address can uniquely determine a customer as no two customers with the same name (first and last) will have the same address.

orders(order_number, date_placed, total_cost, current_location, customer_id)

F= {
order_number → date_placed, total_cost, customer_id
}
This relation is in good normal form under BCNF as for the one function dependency, the left-hand side is a superkey. The order_number a unique number that identifies all the different orders. So a order_number can uniquely identify the tuples of the table.

book_orders(isbn, order_number):

F= {
isbn, order_number → isbn
}
This relation is in good normal form under BCNF as it only has one functional dependency that is trivial as isbn is a subset of [isbn, oder_number].

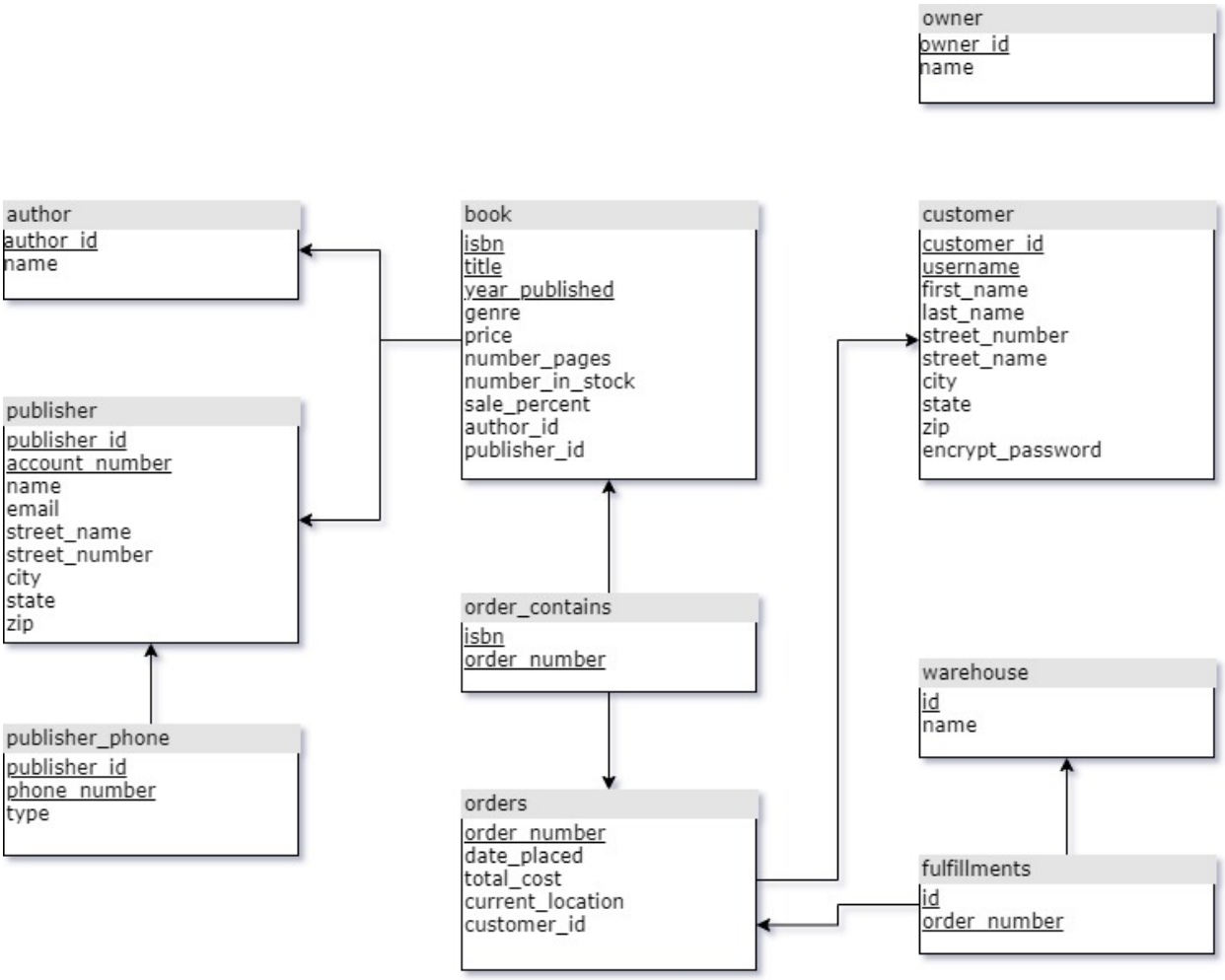
warehouse(id, name):

F= {
id, name → name
name → name
id → name
}
This relation is in good normal form under BCNF as the first two functional dependencies are both trivial as name is a subset of both [id, name] and [name]. The third dependency alpha, or in this case id, is a superkey since there are no tuples with the same id so id can uniquely identify each row in the table

fulfillments(id, order_number):

F= {
id, order_number → order_number
}
This relation is in good normal form under BCNF as it only has one functional dependency that is trivial as order_number is a subset of [id, oder_number].

Database Schema Diagram



Implementation

I was not able to fully complete the application, but I will briefly go over what I had envisioned the design would be. The application would follow be a simple command line interface and follow a Model View Control design pattern. The view as the interaction between the user and application and the control using the input. The user would be prompted to either continue as an owner or user. Users could login, register, view books, or view orders. While the owner could add new books, remove books, or get sales reports. In order to get sales reports the owner would specify either a type of report, what they want to view, and/or the time range to view them from. Given that information queries would be made and look at all the orders that fit that criteria.

GitHub Repository

As stated in the previous section I was not able to get the application finished, but in the hopes of little part marks here is the repository link:

https://github.com/rfranolla/COMP3005-Book-Store-Project