Final Project Proposal

Project Participants: Matt Johnson

Title: Comic Tracking

Executive Summary:

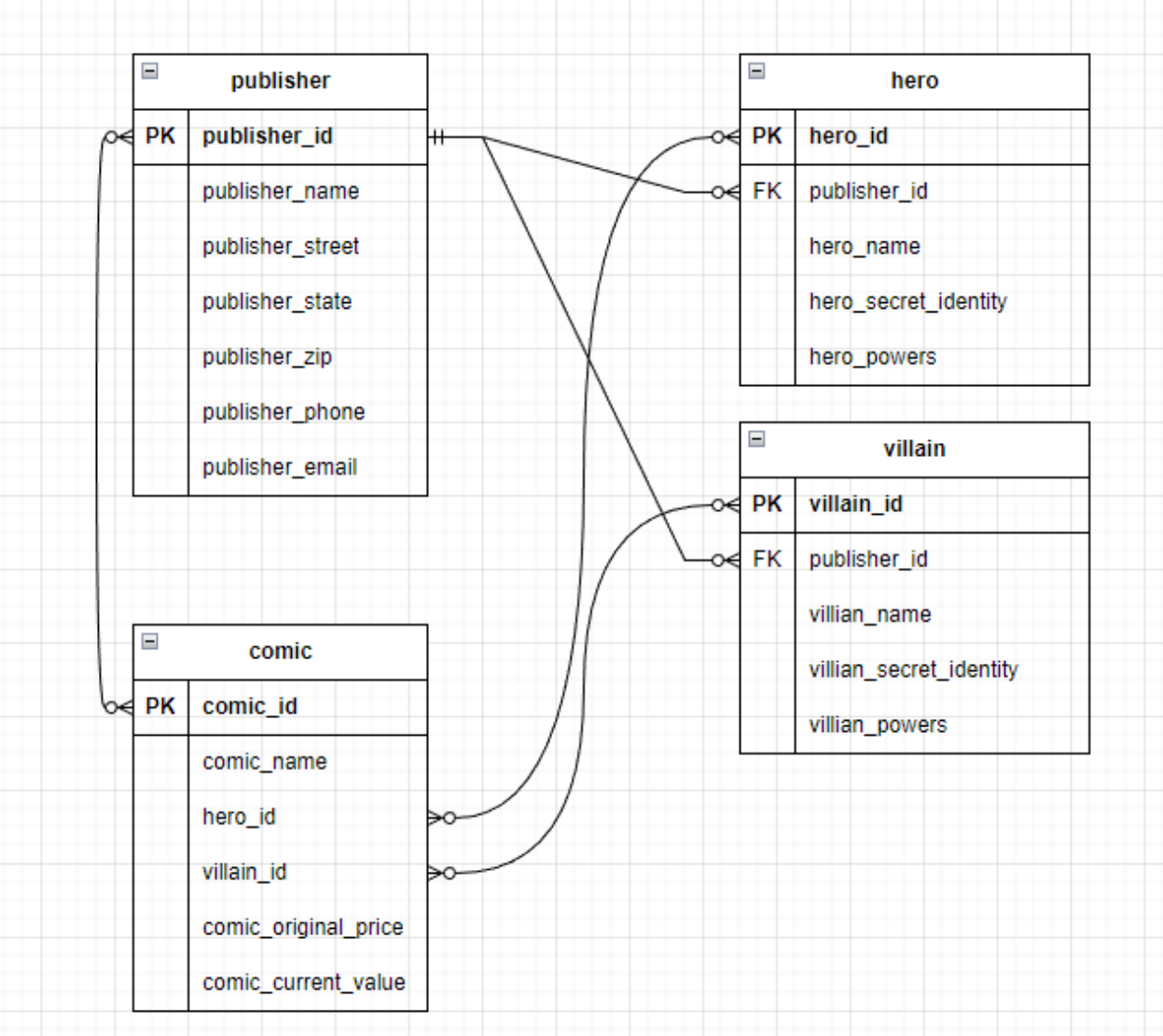
I really love comic books and movies so I will create a fictional database that will aim to create a comprehensive database system for tracking comic books, including publishers, heroes, villains, and their associated comics as described below:

- The **Publisher** table will contain information about the publisher such as name, street, state, zip, phone and email.
- The **Hero** table will contain information about the hero such as the publisher, hero name, secret identity, and powers.
- The **Villain** table will contain information about the villain such as the publisher, name, secret identity, and powers.
- The **Comic** table will contain the information about who published the comic as well as name, the hero, and villain, original price, and current value.

Information about relationships between entities:

- Publisher and Hero tables: One-to-many relationship. Publisher table's publisher_id is referenced as a foreign key in the Hero table.
- Publisher and Villain tables: One-to-many relationship. Publisher table's publisher_id is referenced as a foreign key in the Villain table.
- Hero and Comics tables: Many-to-many relationship. Hero table's hero_id is referenced as a foreign key in the Comics table. There will be a join table to establish the relationship between Hero and Comics. The join table will store the associations between multiple heroes and multiple comics.
- Villain and Comics tables: Many-to-many relationship. There will be a join table to establish the relationship between Villains and Comics. The join table will store the associations between multiple villains and multiple comics.

Entity Relationship Diagram:



Description of endpoints/features:

/publisher

*note: only one parameter would be included in any given request or the application will return a 400 response code with an additional message to the user specifying this requirement. All parameters sent to GET /program would take the form of query strings.

Verb	Parameters	Response	Request Body
GET	None	A list of all publishers and associated data.	None
	hero_id	List of all heros.	
	villain_id	List of all villains	
	comic_id	List of all comics	
POST	None	The created publisher and a status code	JSON representing the publisher entity

publisher/publisher_id

Verb	Response	Request Body
GET	The publisher entity with the id specified or a 404 status if the publisher is not found.	None
PUT	The updated publisher entity and a 200 status code.	JSON representing the publisher entity

/hero

*note: only one parameter would be included in any given request or the application will return a 400 response code with an additional message to the user specifying this requirement. All parameters sent to GET /program would take the form of query strings.

Verb	Response	Request Body
GET	A list of all heros and their data	None
PUT	The updated hero entity and a 201 status code.	JSON representing the publisher entity

/hero/hero_id

Verb	Response	Request Body
GET	A specific hero, or a 404 status code if the hero id is not found.	None
PUT	The updated hero entity and 200 status code	
POST	A 204 (no content) status code if successful. If hero already exist a 403 status is returned.	JSON representing the publisher entity

/villain

*note: only one parameter would be included in any given request or the application will return a 400 response code with an additional message to the user specifying this requirement. All parameters sent to GET /program would take the form of query strings.

Verb	Response	Request Body
GET	A list of all villains and their data	None
PUT	The updated villain entity and a 201 status code.	JSON representing the publisher entity

/villain/villain_id

Verb	Response	Request Body
GET	A specific hero, or a 404 status code if the villain id is not found.	None
PUT	The updated villain entity and 200 status code	
POST	A 204 (no content) status code if successful. If villain already exist a 403 status is returned.	JSON representing the publisher entity

/comic

*note: only one parameter would be included in any given request or the application will return a 400 response code with an additional message to the user specifying this requirement. All parameters sent to GET /program would take the form of query strings.

Verb	Response	Request Body
GET	A list of all comic and their data	None
PUT	The updated comic entity and a 201 status code.	JSON representing the publisher entity

/comic/comic_id

Verb	Response	Request Body
GET	A specific comic, or a 404 status code if the villain id is not found.	None
PUT	The updated comic entity and 200 status code	
POST	A 204 (no content) status code if successful. If comic already exist a 403 status is returned.	JSON representing the publisher entity

Stretch goals if time allows:

- Create a query that will allow you to search when which books a hero and villain were in together.
- Create a query that will allow you to see the most valuable comic for each hero.