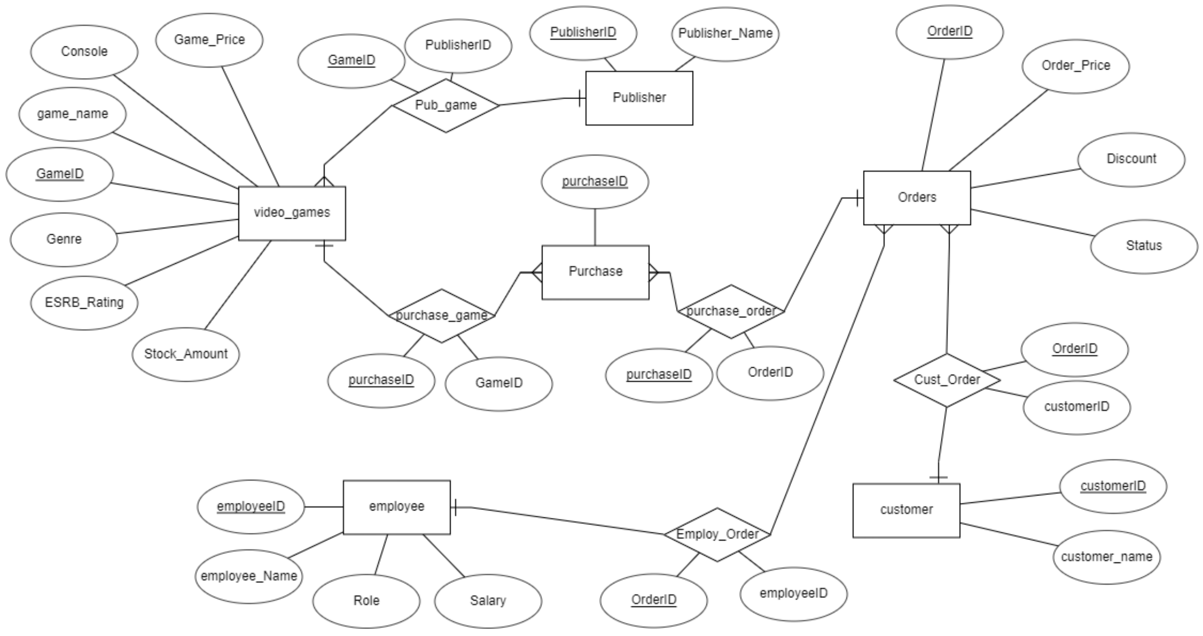
Group Members: Davonte Littleton, Devionne Littleton, Hunter Jones, Christabel Akhigbe

Prof. Saifuddin Mahmud

CS 33007

April 25th, 2022

1. Done!
2. Provide a Conceptual Design of your database
   - Need to edit create join table between games and orders



   -

3. Explain the notations you have used in the ER diagram (You are open to use any notations, but you must describe them)
   - To begin with, it is imperative to discuss why all of the entity sets within our ER Diagram are strong entity sets with simple attributes. This is because the attributes are not multivalued and they are not composite as well, making our representation of our tables simple, as there were nothing but simple attributes (as we could not

break any of our attributes down into something smaller to have multiple values within them and none of them warrant their own entity set/table). Another factor to mention is there are no weak entity sets. This is because it simply does not relate to the ER Diagram and tables that we plan to have as our tables have enough as to where they are able to competently form their own primary key, so none of them can be weak. This is our conclusion we found when concerning entity sets, but what of our relationship sets?

- There are also a multitude of relationship sets and they are between publisher and video_games, purchase and game, order and game, employee and order, and customer and order. These relationship sets are here to connect these entity sets and a majority of them are connected with many-to-one/one-to-many relationships. The reason for this is one, for logical purposes, many-to-one makes sense for all of the relationships within our ER Diagram. Furthermore, the relationships are many-to-one or one-to-many so that when they are converted into a relational schema and schema diagram, it is simple to do so as we are easily able to transfer the primary keys of the "one" side to the "many" side and then remove the relationship sets once they become redundant.

4. Describe at least two relationships among entity sets from your ER diagram.
   - Cust_Order: Both Customer and Orders are Strong Entity Sets with Simple Attributes. The reason the relationship is set up is to show the connection between the customer and whatever game or games they ordered. This allows for the customer to be immediately linked to the order, due to the customer and order IDs. Due to its many-to-one relationship, the relationship is set with Order_ID

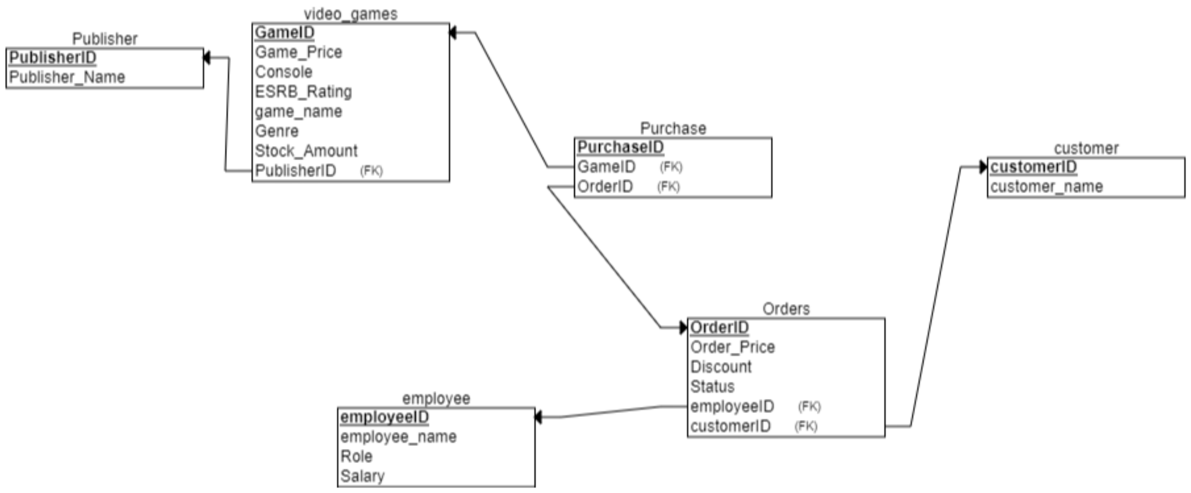
being the primary key, so when it is turned into a relational schema, Orders would take on Customer_ID as a foreign key. The many-to-one relationship is based on the fact that one customer can have multiple orders. This is important because the Orders need to have the customer_id so that people who work at the video game store are aware of the exact customer who had the order at all (so it allows for better verification). That way, if a customer has an issue with their order, the staff can know if that order is truly theirs based on the customerID that is connected to the specific order.

- Employ_Order: Both Employee and Orders are Strong Entity Sets with Simple Attributes. The relationship is set in order to connect the employees to the orders that they send out to the customers. This allows us to know what employee sent out each order. The many-to-one relationship makes it so the Order_ID is the primary key within the relationship. When turned into a relational schema, Orders would take EmployeeID on as a foreign key. The many-to-one relationship is based on the fact that one employee can send out multiple orders. The reason as to why Orders need the EmployeeID is so upper management can keep track of which employees took which orders just in case something goes wrong (such as there being an order incorrectly issued or labeled). So, with Orders having EmployeeID, we can better track the employee who completed the order/transaction and if they were the reason for any issue.

5. Convert ER diagram to Relational Schemas
    - Publisher(<u>PublisherID</u>, Publisher_Name)

- video_games(GameID, Game_Price, Console, ESRB_Rating, game_name, Genre, In_Stock, Stock_Amount, PublisherID (FK))
- Purchase(PurchaseID, GameID(FK), OrderID(FK))
- customer(customerID, customer_name)
- Orders(OrderID, Order_Price, Discount, Status, employeeID(FK), customerID(FK))
- employee(employeeID, employee_name, Role, Salary)

6. From Relational Schemas/directly from the ER diagram, draw the Schema Diagram of your database.



-

7. SQL queries and features such as insert (open bank account), delete (delete account), update (update balance after money withdraw), joins, view, trigger etc
   - Finished, check the source code for details