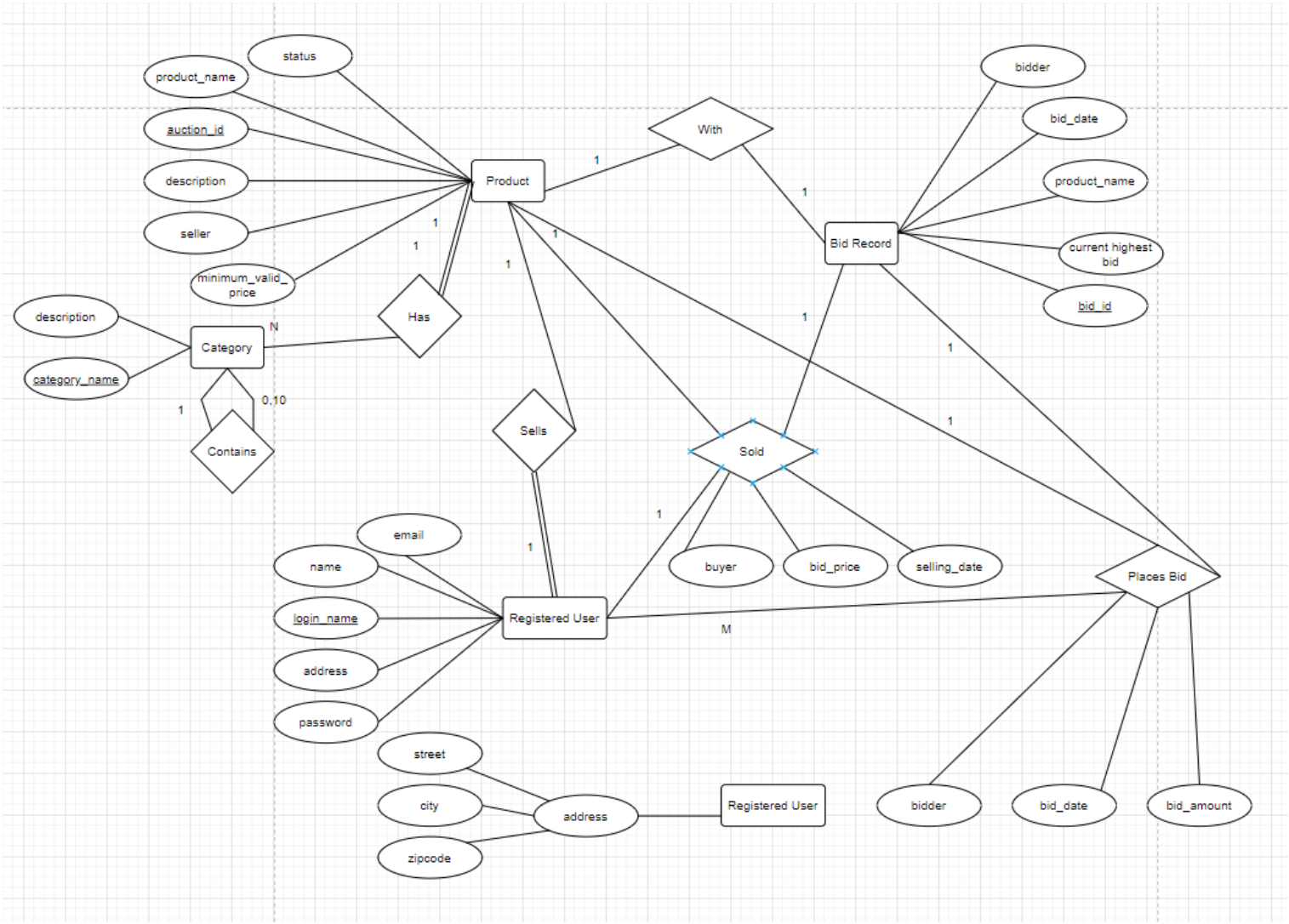
CS 1555: Homework 7 - Lu, Gordon

Task 1:



Assumptions:

Registered User:

For a registered user, I make the following assumptions:
1) The password cannot be unique. Some users may have the same password. Consider the password, "12345678", multiple users may have the same password, so under this assumption, we should not use password as the Primary Key.

2) The name cannot be unique but can be NULL. This would be similar to something that could be optionally filled out in an "Additional Information" tab on an Auction website.

3) The address cannot be unique and can be NULL. Unless something needs to shipped to a certain address in the case that a Registered User is a buyer, then this would be an instance where the address would not be NULL. This would occur only in the instance where a user buys a product. Additionally, the address can be shared among two users, as two people from the same household can hold an account.

4) The email can be unique and cannot be NULL. Typically, we would expect people registering using emails from different websites such as Yahoo and Gmail. So, we can expect people to have the same prefixes to the @[WEBSITE]. For example, we could expect "John@gmail.com" and "John@yahoo.com", but these would never match as they come from different email domains. This would make Email a candidate key.

Product:

For a product put for auction, I make the following assumptions:

1) The status cannot be NULL. This status would be updated through some trigger on the update of a tuple.

2) I assume that each product put for auction must have a corresponding product name, however, this product name does not need to be unique. Consider multiple bikes put on auction.

3) I assume that the description can be empty.

4) I derive the seller from the Registered Users that have put the product on for auction. This does not need to be unique, as sellers can put multiple items for auction.

5) For categories, I assume that a single product can take on multiple categories.

Bid Record:

In interpreting what Bid Record is, for a given bidder, I assume that the Bid Record holds the information for a given buyer, along with the current highest bid amount, and a corresponding bid_id which can be used to match the auctions that have a bid. With the product on auction, I assume that the Bid Record is the history of all bids for a given product.

1) Every time a user bids on a product, this will be updated in the Bid Records.

2) The current highest bid would be used at the end of the bid, and the corresponding user with the highest bid would be awarded the product.

3) I assume that a Bid Record can exist without a Product, meaning an empty record, and a Product can exist without a Bid Record, as this would be a Product with no Bids.

Category:

In the Category entity, I interpreted it to be defined recursively. Meaning for a single category, there can be 0 to 10 subcategories, and for each of those subcategories there can be 0 to 10 more subcategories, until there are no more subcategories.

1) I assume that description for a given category can be empty.

Sold Relationship:

For all auctions where the product was successfully sold, I assume a ternary relationship, requiring the Product, the Bid Record and the Registered User, which will be the buyer.

Task 2:

Entities:

1) Product(status, product_name, auction_id, description, $\overline{seller}$, minimum_valid_price, categories)

2) Registered User(email, name(fname, lastname), login_name, address(street, city, zipcode), password)

3) Bid Record($\overline{bidder}$, bid_date, product_name, description, current_highest_bid, bid_id)

4) Category(description, category_name)

Relationships:

1) Sells <Registered User, Product> 1:1, TOTAL/PARTIAL

2) Contains <Category, Category> 1:0,10, PARTIAL/PARTIAL

3) Has <Product, Category> 1:N, TOTAL/PARTIAL

4) Sold <Registered User, Product, Bid Record> 1:1:1, PARTIAL/PARTIAL/PARTIAL, buyer, bid_price, selling_date, seller

5) With <Product, Bid Record> 1:1, PARTIAL/PARTIAL

6) Places Bid <Registered User, Product, Bid Record> M:1:1, PARTIAL/PARTIAL/PARTIAL, bidder, bid_date, bid_amount

Schemas after Mapping (without Constraints and PKs):

REGISTERED_USER(email, name, login_name, address, password)

PRODUCT(status, product_name, auction_id, description, $\overline{seller}$, minimum_valid_price, categories)

FK(seller) → REGISTERED_USER(login_name)

FK(categories) → CATEGORY(category_name)

BID_RECORD((bidder, bid_id), bid_date, product_name, description, current_highest_bid,)

FK(bidder) → REGISTERED_USER(login_name)

FK(bid_id) → PRODUCT(auction_id)

BIDS_PLACED((bid_id, bidder) bid_date, valid_bid_amount)

FK(bidder) → REGISTERED_USER(login_name)

FK(bid_id) → BID_RECORD(bid_id)

SOLD_SUCCESSFULLY((buyer, bid_id), sell_date, bid_price)

FK(buyer) → BID_RECORD (bidder)

FK(bid_id) → BID_RECORD(bid_id)

CATEGORY(description, (category_name, auction_id))

FK(auction_id) → PRODUCT(auction_id)

FK(category_name) → CATEGORY(category_name)

Schemas after Mapping (with Constraints and PKs):

REGISTERED_USER(email, name, login_name, address, password)

PK(login_name)

CHECK(email IS NOT NULL)

CHECK(password IS NOT NULL)

PRODUCT(status, product_name, auction_id, description, seller, minimum_valid_price, categories)

PK(auction_id)

FK(seller) → REGISTERED_USER(login_name)

FK(categories) → CATEGORY(category_name)

CHECK(UNIQUE seller)

CHECK(product_name IS NOT NULL)

CHECK(status in ('under auction', 'sold', 'withdrawn')

CHECK(minimum_valid_price IS NOT NULL)

BID_RECORD(bidder, bid_id, bid_date, product_name, current_highest_bid,)

PK(bidder, bid_id)

FK(bidder) → REGISTERED_USER(login_name)

FK(bid_id) → PRODUCT(auction_id)

CHECK(product_name IS NOT NULL)

CHECK(bid_date IS NOT NULL)

CHECK(current_highest_bid IS NOT NULL)

BIDS_PLACED(bidder, bid_id, bid_date, valid_bid_amount)

PK(bidder, bid_id)

FK(bidder) → REGISTERED_USER(login_name)

FK(bid_id) → BID_RECORD(bid_id)

CHECK(valid_bid_amount IS NOT NULL)

SOLD_SUCCESSFULLY(buyer, bid_id, sell_date, bid_price)

PK(buyer, bid_id)

FK(buyer) → BID_RECORD (bidder)

FK(bid_id) → BID_RECORD(bid_id)

CHECK(sell_date IS NOT NULL)

CHECK(bid_price IS NOT NULL)

CATEGORY(description, category_name, auction_id)

PK(category_name, auction_id)

FK(auction_id) → PRODUCT(auction_id)

FK(category_name) → CATEGORY(category_name)

Schema Assumptions:

BIDS_PLACED:

In the bids_placed schema, I assume the primary key to be **(bidder, bid_id)**, what I assume is that since Bid Record is related to Product through the auction_id, in the sense that bid_id represents an auction_id that has a bid on it, it is not necessary to include the auction_id. As retrieving the correct auction_id would involve a join of sorts between he auction_id and the bid_id. Which makes auction_id seem extraneous of sorts.

SOLD_SUCCESSFULLY:

In the sold_successfully schema, I assume the primary key to be **(bidder, bid_id)**, for a similar reason to the bids_placed schema. What I assume is that since Bid Record is related to Product through the auction_id, in the sense that bid_id represents an auction_id that has a bid on it, it is not necessary to include the auction_id. As retrieving the correct auction_id would involve a join of sorts between he auction_id and the bid_id. Which makes auction_id seem extraneous of sorts.