Project 1 : Data Models and Query Languages (CSE 560)

Shubham Sharma, Person No.: 50290293, UBIT: ss628

April 6, 2019

Entity Relationship Model

The entity relationship schema for the web portal *shoes.com* is shown in figure 1. Following are the details:

Entity Types:

- CUSTOMER: This is a strong entity having attributes USERNAME, PASSWORD, C_FNAME (Customer First Name), C_LNAME (Customer Last Name), STREET, CITY, STATE(Full name of the state), ZIPCODE, COUNTRY(Full Name of the country). Two Subclasses namely, BUS_CUST (Business Customers) with attribute COMP_NAME (Company Name) and IND_USERS (Individual Users) are created for this entity type with a disjointness constraint. The primary key for this entity type is USERNAME.
- PRODUCT: This is a strong entity with attributes CATEGORY, DESCRIPTION, PRICE, SHOE_NAME, PROD_ID (Unique product ID for each product) which acts as the primary key here and P_DISCOUNT (Discount offers on products)
- AVAILABILITY: This is a weak entity which borrows its key from PRODUCT. This has attributes SIZE and QUANTITY where SIZE(integer) is the key here and PROD_ID is borrowed from PRODUCT(partial key). This entity was created since we might have different sized products for a particular PROD_ID and thus to avoid repetition this step was taken.
- ORDERS: This is a strong entity with attributes ORDER_NUM (Order Number which is the primary key), O_DISCOUNT (Order Discount) and TOTAL_AMT (Total Amount for the order).

Relationship Types:

- PLACED_RETURN: This is a 1:N relationship between CUSTOMER and ORDER which handles the order placed by customers as well as the returns requested by the customer. This implies that one customer can place between 0 to N orders and one order can only be placed by exactly one customer.
- CONTAINS: This is an M:N relationship between ORDER and PRODUCT which tells the products contained in an order. This has the attributes O_QUANTITY (tells the quantity of each product ordered in a particular order number), R_QUANTITY (tells the quantity of product requested to be returned) and STATUS (tells the status of the order 0 - order placed, 1 - order processed, 2 - order delivered, 3 - return requested, 4 - return processed). This implies that 1 to M products can be present in one order and 1 product can be in 0 to N orders.
- IS_AVAILABLE: This is a 1:N relationship between PRODUCT and AVAILABILITY which gives the description of the quantities of different sizes available for a particular product. One product can have 1 to N availability while one availability tuple can be specified for only 1 product.

The Relational Database Schema is as follows:

Tables:

The two subclasses: BUS_CUST and IND_USERS are created as tables.

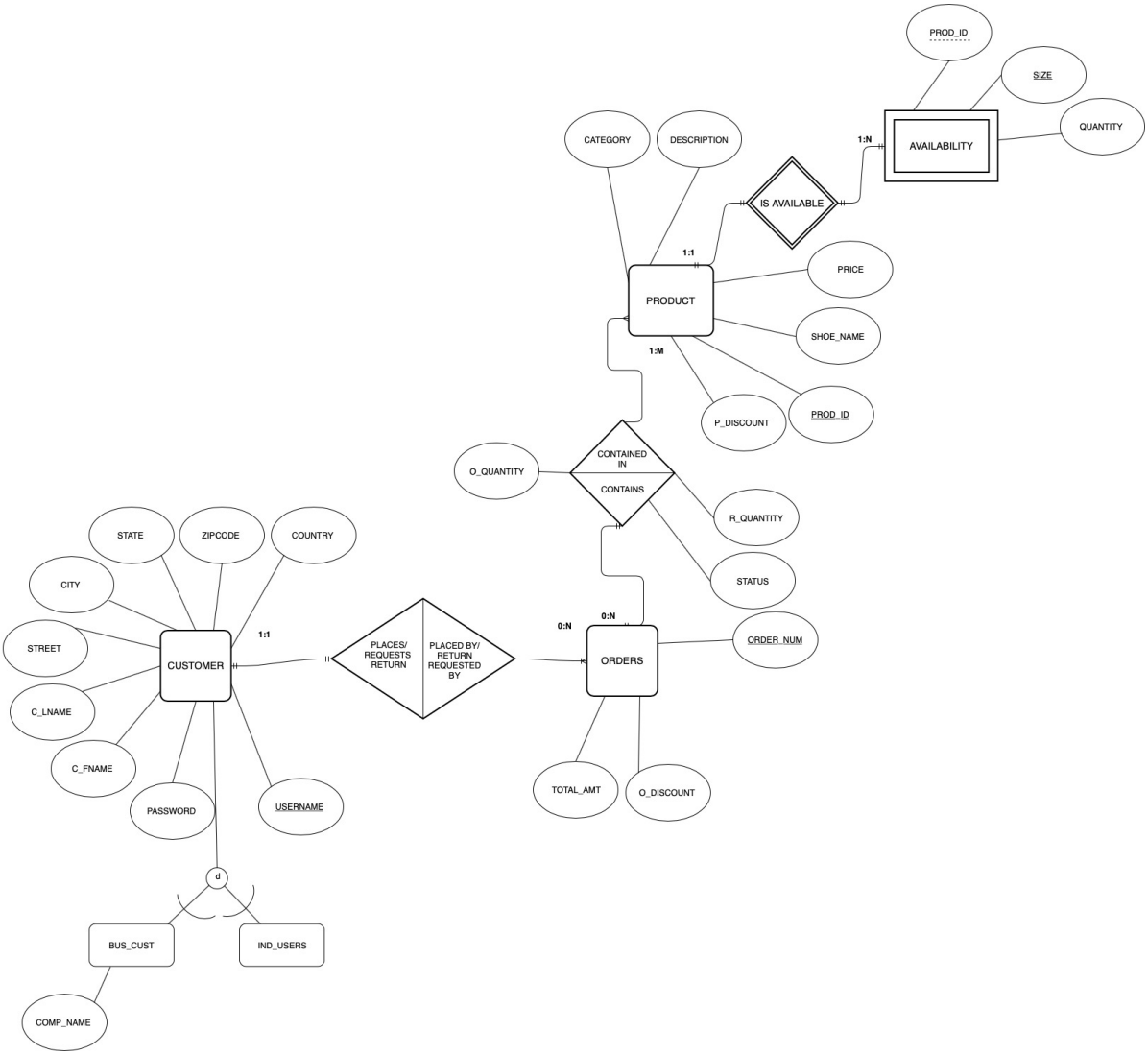

Figure 1: Entity relationship schema for the web portal *shoes.com*

- Bus_Cust: This has the following attributes: username(primary key), passwd, first_name, last_name, street, city, state, zip(between 0 and 99999), country, company_name.
- Ind_Users: This has the following attributes: username(primary key), passwd, first_name, last_name, street, city, state, zip(between 0 and 99999), country.
- Product: This has the following attributes prod_id (Unique product ID for each product which acts as the primary key), p_discount(between 0 to 100), price(¿=0), shoe_name, category, descript.
- Availability: This has the following attributes size(between 1 - 20; primary key), prod_id foreign key referencing Product and quantity(¿0)
- OrderProduct: This has the following attributes o_quantity (¿=0), r_quantity (¿=0 and ¡= o_quantity), order_num (which is foreign key referencing Orders), p_status(status of order as specified earlier), prod_id (foreign key referencing Product). This table tells information about the quantity of each product placed or requested to be returned in an order.
- Orders: This has the following attributes order_num (unique order number for each order; primary key), total_amount (total amount for the order), o_discount (order discount; 0 - 100 %), username (foreign key referencing Ind_Users and Bus_Cust since it has a 1:N relationship between them)

The attribute prod_id is introduced to identify each product uniquely. The attribute p_discount identifies which product is on sale (by noting which products have non-zero discount values). We introduce the attribute p_status on the relationship/table OrderProduct to support the returns of orders, using which we record the order number from the ORDERS entity type, returned products from the PRODUCT entity type and the return status is specified by the attribute p_status itself. The attribute r_quantity specifies the quantity of the product which is requested to be returned. This will be 0 for the orders in which no return is requested which will act as an additional flag to check if return is requested or not. Only atomic attributes are considered and thus Customer name and Address are broken into atomic attributes.

Advantages and Disadvantages:

The advantages are as follows:

- The design effectively specializes the customers as Business cutomers and Individual Users, thus reducing redundancy in the design.
- The support to handle return for the orders is efficiently handles which can handle returns of specific products in a given order and teacks its status
- The design handles all integrity constraints and ensures that no garbage value is present in the database.
- The creation of weak entity AVAILABILITY reduces redundancy in the design by avoinding the repetiition of information.
- Nested queries could be executed very fast in this design as the queries would be on key attributes only.

The disadvantages are as follows:

- A better design would have been to create composite attribute for Customer Name and Address which could be broken down into subparts
- The table *Orderproduct* has redundant data in r_quantity for products where return is not requested.