Project: Sneakers Database

Created by: Juliano Nguyen JR.
Last updated: August 6, 2018

Table of Contents

---

1) Introduction

This document describes the Sneakers database. Each section will provide a description of the entities, diagrams, and tools used. The purpose of the Sneakers database is to maintain a list of upcoming shoe releases, past shoe releases, user purchases, users reselling shoes,and user reviews of the shoes.

2) Scenario

A person is waiting for a new shoe to be released. The person purchases the shoe on release day and resells the pair on a website. A user can make a review before release date to talk about if it is worth buying in order to resell it. A user can make a review after release date to talk about the quality of the shoe.

3) Constraints

Domain Constraint

Company

Attributes	**Domain Constraint**
Name	String; max 20 char; not null
City	String
State	String; not null
Year_est	Integer; after 1900; not null

Shoe

Attributes	**Domain Constraint**
Shoe_id	String; not null
Color	String; not null
Release_date	Date; YYYY-MM-DD
Retail_value	Decimal; $XXX.00

User

Attributes	**Domain Constraint**

Username	String; max 20 char; not null
First_name	String; not null
Last_name	String; not null
Shoe_size	Integer; [0-15]; not null
Date_joined	Date; YYYY-MM-DD; not null

Website

Attributes	**Domain Constraint**
Website_id	Integer; positive; not null
Website_name	String; not null
Website_email	String

Review

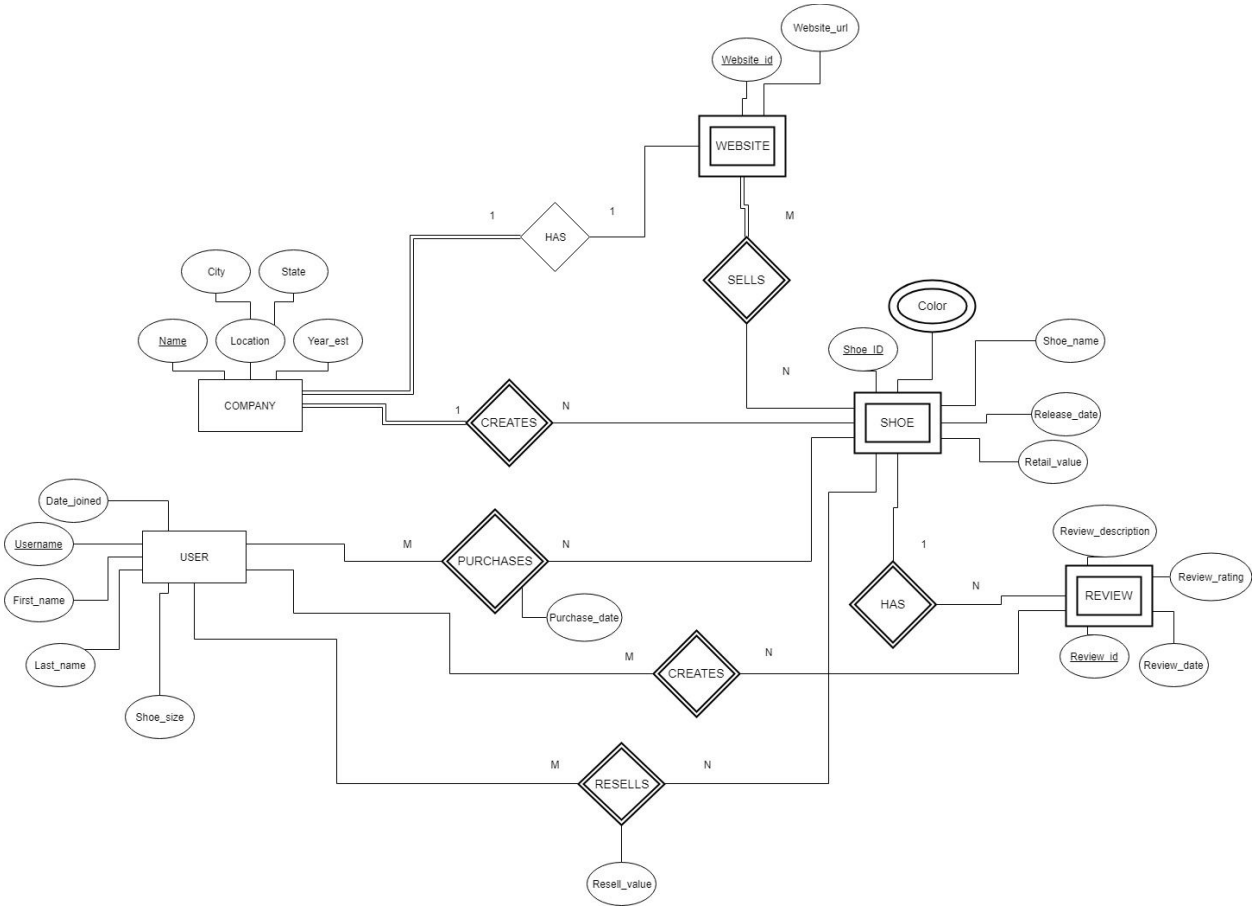
Attributes	**Domain Constraint**
Review_id	Integer; positive; not null
Review_date	Date; YYYY-MM-DD; not null
Review_rating	Integer; [0-5]; not null
Review_description	String

Referential Integrity Constraint

- Company is related to Shoe
- User is related to Purchases, Resells_On, Review
- Shoe is related to Review, Purchases, Shoe_Color
- Website is related to Resells_On

4) Entity-Relationship Diagram

Figure 1: ER Diagram for Sneakers Database



Introduction

This is the ER diagram for the Sneakers Database. It lists the relationships, entities, weak entities, and their corresponding attributes.

Description

The Sneakers Database contains a User, Company, Website, Shoe, and Review. The User can purchase a shoe, resell a shoe, and make a review for a shoe. The Company has an official website and makes the shoe product.

Design Decisions

- I made the User entity to be able to make a shoe purchase, to resell a shoe, and to make a review of the shoe.
- The Shoe is a weak entity because it does not exist without the company that creates it.
- The Website is a weak entity because it does not exist without a company because a company will have an official website to sell the shoe.
- The Review is a weak entity because it does not exist without a shoe.

Limitations

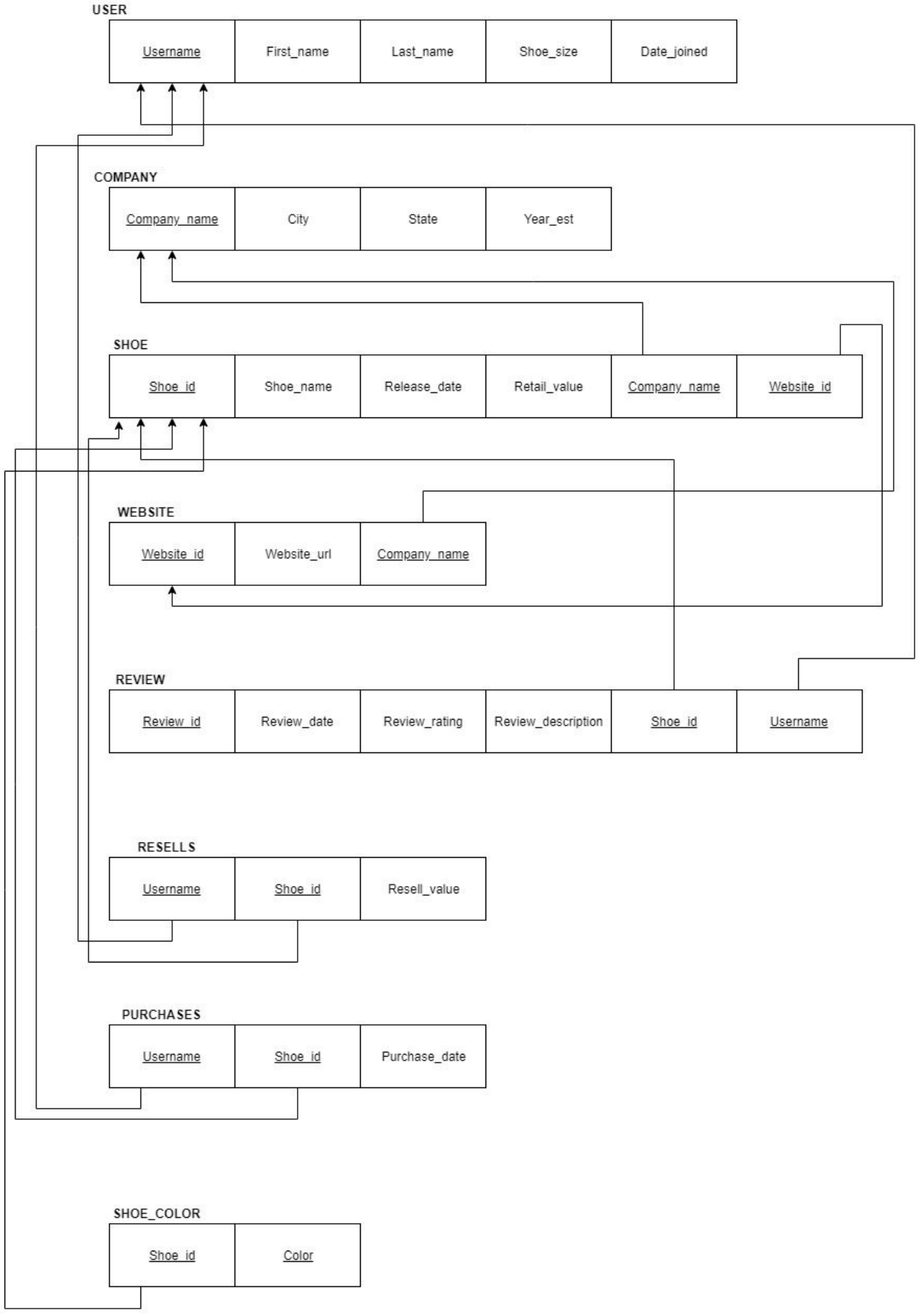
- Does not support payment transactions
- Does not support user account creation
- Does not support reselling process and transactions
- Does not support shoe product creation

Assumptions

- User purchases a shoe in the same shoe size listed on their account
- User reselling a shoe will be in the shoe size listed on their account
- Company designs, creates, and manufactures a shoe product
- User can only resell a shoe by listing its resell value (no actual resell)
- User can only purchase a shoe at retail value

5) Relational Model

Figure 2: RM Schema for Sneakers Database


USER
Username
First_name
Last_name
Shoe_size
Date_joined
COMPANY
Company_name
City
State
Year_est
SHOE
Shoe_id
Shoe_name
Release_date
Retail_value
Company_name
Website_id
WEBSITE
Website_id
Website_url
Company_name
REVIEW
Review_id
Review_date
Review_rating
Review_description
Shoe_id
Username
RESELLS
Username
Shoe_id
Resell_value
PURCHASES
Username
Shoe_id
Purchase_date
SHOE_COLOR
Shoe_id
Color

Introduction

This is the RM diagram for the Sneakers Database. It includes the table name and their corresponding attributes. It also lists some of the relationship tables, such as purchases. The primary key is underlined in each table. There may be multiple primary keys, such as the Shoe table, because it is a weak entity and it inherits the primary key from the normal entity connected to it.

Description

Table	Description
User [Entity]	Each user includes a username, their name, their shoe size, and date joined.
Company [Entity]	Each company includes a name, their location, and year that the company was established.
Shoe [Entity]	Each shoe includes a unique ID, name, release date, and retail value.
Website [Entity]	Each website includes a unique ID, a URL, and the company name.
Review [Entity]	Each review includes a unique ID, a date, a rating, a description, a shoe, and the user.
Resells [Relationship]	Each resell includes a user, the shoe ID, and the resell value.
Purchases [Relationship]	Each purchase includes a user, the shoe ID, and a date.
Shoe_Color [Relationship]	It includes the shoe ID and its corresponding color.

Design Decisions

- Shoe_Color was created separately because Color can contain multiple colors
- Website includes a web URL to its respective company website
- User has a shoe size so that they can purchase/resell a shoe in the same size

Limitations

- Same from Entity-Relationship Diagram section

Assumptions

- Same from Entity-Relationship Diagram section
- The State attribute in the Company can also be a Country if it is outside the U.S.
- The Color attribute in the Shoe_Color can include multiple colors

6) Queries

This section contains a sample of possible queries in English for the Sneakers database.

- List all shoes purchased by a specific user.
- List all shoes with the color “Black”.
- List all users that are reselling a shoe.
- List all “Nike” shoes.
- List all shoes that cost more than $150
- List all shoes with a resell value more than $200

7) SQL Statements

This section contains a list of sample SQL statements that will retrieve relevant information from a table or multiple tables.

SQL Statement	**Purpose**
SELECT Shoe_id, Shoe_name, Retail_value, ResUsername, Resell_value, Shoe_size FROM RESELLS, SHOE, USER WHERE Shoe_id = ResShoe_id AND Username = ResUsername;	Retrieves all users that are reselling a shoe.
SELECT Shoe_id, Shoe_name, Release_date, Retail_value, SC_color FROM SHOE, SHOE_COLOR WHERE Shoe_id = SCShoe_id ORDER BY Release_date DESC;	Retrieves all shoes with their color name. The latest shoe will be listed at the top.
SELECT Shoe_id, Shoe_name, Release_date, Retail_value FROM SHOE ORDER BY Release_date DESC;	Retrieves all shoes, where the latest shoe will be listed at the top.
SELECT Shoe_id, Shoe_name, Release_date, Retail_value, SC_color	Retrieves all shoes with no release date.

FROM SHOE, SHOE_COLOR WHERE Shoe_id = SCShoe_id AND Release_date IS NULL ORDER BY Shoe_name;	
SELECT Shoe_id, Shoe_name, Release_date, Retail_value, SC_color FROM SHOE, SHOE_COLOR WHERE Shoe_id = SCShoe_id AND Shoe_name LIKE '%Yeezy%' ORDER BY Release_date DESC;	Retrieves all shoes with the name “Yeezy”.
SELECT PUsername, SUM(Retail_value) AS Total_spent FROM PURCHASES, SHOE WHERE Shoe_id = PShoe_id GROUP BY PUsername;	Retrieves all users and the total amount each user spent on shoes.
SELECT Shoe_id, Shoe_name, SC_color, FORMAT(AVG(Resell_value), 2) AS Average_resell FROM SHOE, SHOE_COLOR, RESELLS WHERE Shoe_id = SCShoe_id AND Shoe_id = ResShoe_id GROUP BY ResShoe_id;	Retrieves the average resell value for all shoes that are being resold.
SELECT Shoe_id, Shoe_name, SC_color, Release_date, Retail_value FROM SHOE, SHOE_COLOR WHERE Shoe_id = SCShoe_id AND SC_color LIKE '%Black%' ORDER BY Release_date DESC;	Retrieves all shoes with the color name “Black”.

8) Test Data

The data will be manually inputted by using StockX for resell values for shoes and various Sneaker websites to maintain upcoming sneaker releases. New shoes will have new SQL insert statements and will have to be updated if there is no further information about the shoe, such as an unconfirmed release date or an unconfirmed retail value.

9) Tooling Assessment

Tool	**Description**
MySQL	This is used to maintain all the SQL statements and view the tables.
Notepad++	This is used to create the SQL create and insert statements for the database.

10) Normalization

USER

- In 3NF
    - Satisfies 1NF because all attributes are single atomic values
    - Satisfies 2NF because all non-primary keys are functionally dependent on Username
    - Satisfies 3NF because there is no transitive dependency; X is the super key in the FD: {Username} -> {First_name, Last_name, Shoe_size, Date_joined}

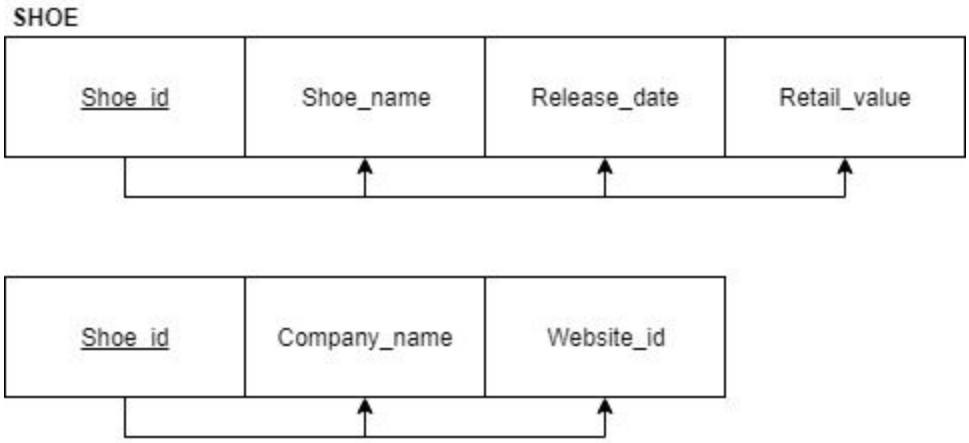
COMPANY

- In 3NF
    - Satisfies 1NF because all attributes are single atomic values
    - Satisfies 2NF because all non-primary keys are functionally dependent on Company_name
    - Satisfies 3NF because there is no transitive dependency; X is the super key in the FD: {Company_name} -> {City, State, Year_est}

SHOE

- In 1NF
    - Satisfies 1NF because all attributes are single atomic values
    - To normalize this to 3NF, the tables have to be split into the following FD:
        - {Shoe_id} -> {Shoe_name, Release_date, Retail_value}
        - {Shoe_id} -> {Company_name, Website_id}

  I splitted the tables this way because Company_name and Website_id do not determine the other non-primary key attributes in this table. I think that deleting a company would accidentally delete other shoe information.



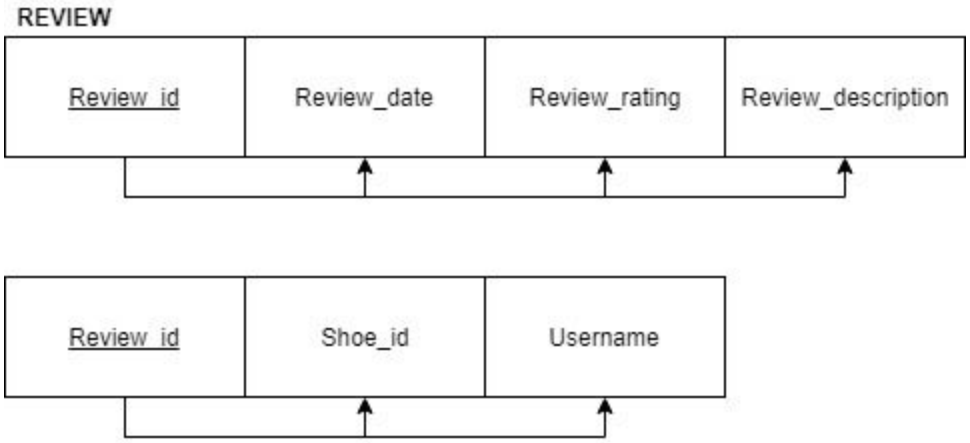
WEBSITE

- In 3NF
    - Satisfies 1NF because all attributes are single atomic values
    - Satisfies 2NF because all non-primary keys are functionally dependent on Website_id and Company_name
    - Satisfies 3NF because there is no transitive dependency; X is the super key in the FD: {Website_id, Company_name} -> {Website_url}

REVIEW

- In 1NF
    - Satisfies 1NF because all attributes are single atomic values
    - To normalize this to 3NF, the tables have to be split into the following FD:
        - {Review_id} -> {Review_date, Review_rating, Review_description}
        - {Review_id} -> {Shoe_id, Username}

  I splitted the tables this way because Shoe_id and Username do not determine the other non-primary key attributes in this table. I think that deleting a username would accidentally delete other review information.



RESELLS

- In 3NF
    - Satisfies 1NF because all attributes are single atomic values
    - Satisfies 2NF because all non-primary keys are functionally dependent on Username and Shoe_id
    - Satisfies 3NF because there is no transitive dependency; X is the super key in the FD: {Username, Shoe_id} -> {Resell_value}

PURCHASES

- In 3NF
    - Satisfies 1NF because all attributes are single atomic values
    - Satisfies 2NF because all non-primary keys are functionally dependent on Username and Shoe_id
    - Satisfies 3NF because there is no transitive dependency; X is the super key in the FD: {Username, Shoe_id} -> {Purchase_date}

SHOE_COLOR

- In 1NF
    - Satisfies 1NF because all attributes are single atomic values
    - To normalize this to 3NF, the tables have to be split into the following FD:
        - {Shoe_id} -> {Color}

    I made the table this way because it would not make sense to have a table with both attributes as primary keys.

11) Schedule

Description	**Due Date**
Determine requirements for database	7/6/2018
Design ER and RM diagrams V1	7/13/2018
Refine ER and RM diagrams V2	7/20/2018
Populate database with sample data	7/20/2018
Develop SQL statements for database	7/27/2018