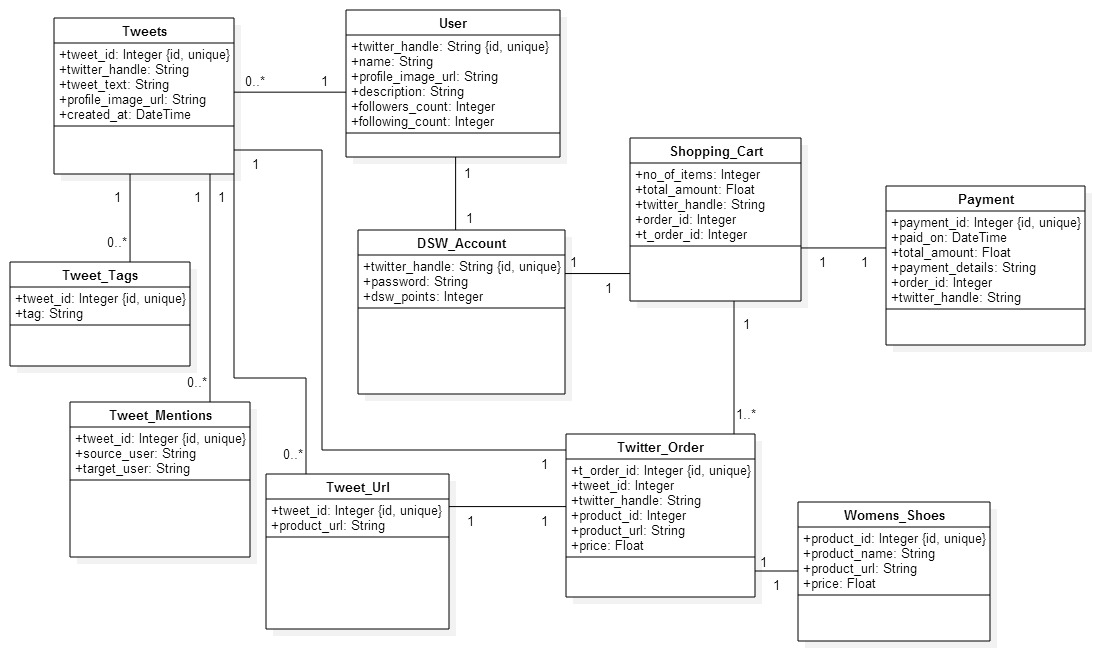
DATABASE DESIGN

WORKED EXAMPLE - ASSIGNMENT 2

**A Model on Online Shopping of Designer Shoe Warehouse using Twitter**

The online shopping domain model has been updated to be more specific for a particular shop. (Designer Shoe Warehouse). The online shopping model also incorporates Twitter database schema. In this model, the user can order shoes by tweeting about the order along with the product URL. The DSW store can also tweet about a particular shoe (women’s shoe, in our model) as a part of promotion and marketing.

Find below the UML diagram of the online Twitter shopping domain.



**Explanation on some of the design decisions:**

* The DSW account has a login and password. This login is the same as a user’s Twitter handle. The Twitter handle is unique – hence it can also be treated as the primary key of the table.
* Each user can tweet any number of tweets. The DSW-user (admin user of the DSW store who tweets about the promotional offer and ads for marketing purpose) is also one of the users and this information can be stored in the user table itself.
* A user can make an order through Twitter by tweeting about the product and mentioning the product URL. This product URL mentioned in a tweet is stored in ‘Tweet\_Url’ table. Every tweet that has a URL in it, will have an entry in ‘Tweet\_Url’ table.
* ‘Twitter Order’ has the ‘tweet\_id’ of the tweet which uniquely distinguishes each tweet, ‘product-url’ which is a foreign key referenced to the ‘product\_url’ in ‘Tweet\_Url’ table, ‘product\_id’ which is a foreign key referenced to ‘product\_id’ in ‘Womens\_Shoes’ table corresponding to the particular ‘product\_url’ mentioned by a user in the tweet.
* A user can tweet (read purchase) how many ever shoes he/she wants and add them to the shopping cart. Hence the ‘Shopping\_Cart’ has an ‘order\_id’ which is the primary key of the table (since it uniquely distinguishes each order) and multiple ‘t\_order\_id’ for an order. Note that each order in the shopping cart can have more than one Twitter order.

**SQL Statements for the conceptual model:**

**User Table:**

CREATE TABLE `User` (

`Twitter\_handle` **VARCHAR**(**10**),

`name` **VARCHAR**(**20**),

`profile\_image\_url` **VARCHAR**(**200**),

`description` **VARCHAR**(**100**),

`followers\_count` **INT**,

`following\_count` **INT**,

PRIMARY KEY (`Twitter\_handle`)

);

**Tweets Table:**

CREATE TABLE `Tweets` (

`tweet\_id` **INT** NOT NULL AUTO\_INCREMENT,

`Twitter\_handle` **VARCHAR**(**10**),

`tweet\_text` **VARCHAR**(**140**),

`profile\_image\_url` **VARCHAR**(**200**),

`created\_at` **DATETIME**,

PRIMARY KEY (`tweet\_id`)

);

**Tweet\_Tags Table:**

CREATE TABLE `Tweet\_Tags` (

`tweet\_id` **INT** NOT NULL,

`tags` **VARCHAR**(**20**),

PRIMARY KEY (`tweet\_id`)

);

**Tweet\_Mentions Table:**

CREATE TABLE `Tweet\_Mentions` (

`tweet\_id` **INT** NOT NULL,

`source\_user` **VARCHAR**(**10**),

`target\_user` **VARCHAR**(**10**),

PRIMARY KEY (`tweet\_id`)

);

**Tweet\_Url Table:**

CREATE TABLE `Tweet\_Url` (

`tweet\_id` **INT** NOT NULL,

‘product\_url’ **VARCHAR**(**200**)

PRIMARY KEY (`tweet\_id`)

);

**DSW\_Account Table:**

CREATE TABLE `DSW\_Account` (

`Twitter\_handle` **VARCHAR**(**10**) NOT NULL,

`password` **VARCHAR**(**10**),

‘dsw\_points’ **INT**

PRIMARY KEY (`Twitter\_handle`)

);

**Womens\_Shoes Table:**

CREATE TABLE `Womens\_Shoes` (

`product\_id` **INT** NOT NULL AUTO\_INCREMENT,

`product\_name` **VARCHAR**(**20**),

`product\_url` **VARCHAR**(**200**),

‘price’ **FLOAT**

PRIMARY KEY (`product\_id`)

);

**Twitter\_Order Table:**

CREATE TABLE `Twitter\_Order` (

`t\_order\_id` **INT** NOT NULL AUTO\_INCREMENT,

`tweet\_id` **INT**,

`Twitter\_handle` **VARCHAR**,

`product\_id` **INT**,

‘product\_url’ **VARCHAR**(**200**)

‘price’ **FLOAT**

PRIMARY KEY (`t\_order\_id`)

);

**Shopping\_Cart Table:**

CREATE TABLE `Shopping\_Cart` (

`order\_id` **INT** NOT NULL AUTO\_INCREMENT,

`total\_amount` **FLOAT**,

`Twitter\_handle` **VARCHAR**(**10**),

`no\_of\_items` **INT**,

PRIMARY KEY (`order\_id`)

);

**Payment Table:**

CREATE TABLE `Payment` (

`payment\_id` **INT** NOT NULL AUTO\_INCREMENT,

`order\_id` **INT**,

`Twitter\_handle` **VARCHAR**(**10**),

`paid\_on` **DATETIME**,

`total\_amount` **FLOAT**,

`payment\_details` **VARCHAR**,

PRIMARY KEY (`payment\_id`)

);

**Adding Foreign Key Constraint:**

**Constraint for Tweet table:**

ALTER TABLE `Tweets`

ADD CONSTRAINT `Tweets\_fk1` FOREIGN KEY (`Twitter\_handle`)

REFERENCES User(`Twitter\_handle`);

ALTER TABLE `Tweets`

ADD CONSTRAINT `Tweets\_fk2` FOREIGN KEY (`profile\_image\_url`)

REFERENCES User(`profile\_image\_url`);

**Constraint for Tweet\_Tags table:**

ALTER TABLE `Tweet\_Tags`

ADD CONSTRAINT `Tweet\_Tags\_fk1` FOREIGN KEY (`tweet\_id`)

REFERENCES Tweets(`tweet\_id`);

**Constraint for Tweet\_Mentions table:**

ALTER TABLE `Tweet\_Mentions`

ADD CONSTRAINT `Tweet\_Mentions\_fk1` FOREIGN KEY (`tweet\_id`)

REFERENCES Tweets(`tweet\_id`);

ALTER TABLE `Tweet\_Mentions`

ADD CONSTRAINT `Tweet\_Mentions\_fk2` FOREIGN KEY (`source\_user`)

REFERENCES User(`Twitter\_handle`);

ALTER TABLE `Tweet\_Mentions`

ADD CONSTRAINT `Tweet\_Mentions\_fk3` FOREIGN KEY (`target\_user`)

REFERENCES User(`Twitter\_handle`);

**Constraint for Tweet\_Url table:**

ALTER TABLE `Tweet\_Url`

ADD CONSTRAINT `Tweet\_Url\_fk1` FOREIGN KEY (`tweet\_id`)

REFERENCES Tweets(`tweet\_id`);

**Constraint for DSW\_Account table:**

ALTER TABLE `DSW\_Account`

ADD CONSTRAINT `DSW\_Account\_fk1` FOREIGN KEY (`Twitter\_handle`)

REFERENCES User(`Twitter\_handle`);

**Constraint for Twitter\_Order table:**

ALTER TABLE `Twitter\_Order`

ADD CONSTRAINT `Twitter\_Order\_fk1` FOREIGN KEY (`tweet\_id`)

REFERENCES Tweet\_Url(`tweet\_id`);

ALTER TABLE `Twitter\_Order`

ADD CONSTRAINT `Twitter\_Order\_fk2` FOREIGN KEY (`Twitter\_handle`)

REFERENCES Tweets(`Twitter\_handle`);

ALTER TABLE `Twitter\_Order`

ADD CONSTRAINT `Twitter\_Order\_fk3` FOREIGN KEY (`product\_id`)

REFERENCES Womens\_Shoes(`product\_id`);

ALTER TABLE `Twitter\_Order`

ADD CONSTRAINT `Twitter\_Order\_fk4` FOREIGN KEY (`product\_url`)

REFERENCES Tweet\_url(`product\_url`);

ALTER TABLE `Twitter\_Order`

ADD CONSTRAINT `Twitter\_Order\_fk5` FOREIGN KEY (`price`)

REFERENCES Womens\_Shoes(`price`);

**Constraint for Shopping\_Cart table:**

ALTER TABLE `Shopping\_Cart`

ADD CONSTRAINT `Shopping\_Cart\_fk1` FOREIGN KEY (`Twitter\_handle`)

REFERENCES Twitter\_Order(`Twitter\_handle`);

ALTER TABLE ` Shopping\_Cart `

ADD CONSTRAINT ` Shopping\_Cart \_fk2` FOREIGN KEY (`t\_order\_id`)

REFERENCES Twitter\_Order(`t\_order\_id`);

ALTER TABLE ` Shopping\_Cart `

ADD CONSTRAINT ` Shopping\_Cart \_fk3` FOREIGN KEY (`Twitter\_handle`)

REFERENCES Tweet\_Order(`Twitter\_handle`);

**Constraint for Payment table:**

ALTER TABLE `Payment`

ADD CONSTRAINT `Payment\_fk1` FOREIGN KEY (`order\_id`)

REFERENCES Shopping\_Cart(`order\_id`);

ALTER TABLE `Payment`

ADD CONSTRAINT `Payment\_fk2` FOREIGN KEY (`Twitter\_handle`)

REFERENCES Shopping\_Cart(`Twitter\_handle`);

**USE-CASE**

1. **Use Case:** Register for an account in DSW

**Description:** User registers for an account in DSW

**Actor:** User

**Precondition:** When a customer wants to buy something from shop, firstly he will be registered

**Steps:**

**Actor action:** User request for registration

**System Responses:** If customer information is correct then customer is registered and use case ends.

**Post Condition:** Customer successfully registered

**Alternate Path:** The customer request is not correct and system throws an error

**Error:** User information is incorrect

1. **Use Case:** Make an order in DSW

**Description:** User makes an order of a product in DSW store

**Actors:** User

**Precondition:** User must have a unique Twitter handle to tweet

**Steps:**

**Actor action –** User tweets about a product to order along with the product URL

**System Responses –** An order is made for the product that matches the product URL

**Post Condition:** An order is added to Twitter\_Order table for the product the user tweeted.

**Alternate Path:** The product not currently available in the store

**Error:** Product Not Available

1. **Use Case:** View a product already ordered through Twitter by a user

**Description:** User views a product already ordered

**Actors:** User

**Precondition:** User must have made an order

**Steps:**

**Actor action –** User views a product from its URL

**System Responses –** product URL would be displayed

**Post Condition:** system displays product URL

1. **Use Case:** View the products above a particular price (say $100)

**Description:** Use views the products above a particular price

**Actor:** User

**Precondition:**

**Steps:**

**Actor action:** User views the products above a particular price

**System Responses:** the list of products above a price are displayed

**Post Condition:** system displays the list of products for the condition

1. **Use Case:** View the orders made by a user

**Description:** User views the orders made by him/her

**Actor:** User

**Precondition:** User must have made at least one order to view an order

**Steps:**

**Actor action:** User views the history of orders

**System Responses:** Displays all the orders made by a user

**Alternate Path:** There are no orders made by a user

**Error:** No history of orders available.

**RELATIONAL-ALGEBRA EXPRESSIONS FOR THE USE CASES**

1. Use Case: View a product already ordered through Twitter

Π{w.product\_url}(σ{w.product\_id = t.product\_id ∧ t.Twitter\_handle = '@alice' }(ρ{w}(Womens\_Shoes) × ρ{t}(Twitter\_Order)))

2. Use Case: View the products above a particular price (say $100)

Π{w.product\_url, w.product\_url}(σ{w.price > 100 }(Womens\_Shoes))

3. Use Case: View the orders made by a user

Π{s.Twitter\_handle, s.order\_id}(σ{s.Twitter\_handle = '@bob' }(Shopping\_Cart))

**SQL STATEMENTS**

1. Use Case: Register for an account in DSW

INSERT INTO DSW\_Account

(Twitter\_handle, password, dsw\_points)

VALUES (@john, john123, **0**)

INSERT INTO DSW\_Account

(Twitter\_handle, password, dsw\_points)

VALUES (@alice, alice123, **0**);

INSERT INTO DSW\_Account

(Twitter\_handle, password, dsw\_points)

VALUES (@bob, bob123, **0**);

1. Use Case: Make an order in DSW

INSERT INTO Tweet

(tweet\_id, Twitter\_handle, tweet\_text, profile\_image\_url, created\_at )

VALUES (**12321**, @john, ‘I would like to purchase www.dsw.com/shoe/product\_id=**2341**’ , ‘www.facebook.com/john.smith/ photo.php?fbid=**10205**’ , **02**-**02**-**2015** );

INSERT INTO Tweet\_url

(tweet\_id, tweet\_url )

VALUES (**12321**,‘www.dsw.com/shoe/product\_id=**2341**’);

INSERT INTO Twitter\_Order

(t\_order\_id, tweet\_id, Twitter\_handle, product\_id, product\_url, price)

VALUES (**4532**, **12321**, @john, **2341**, ,‘www.dsw.com/shoe/product\_id=**2341**’, **26.4** )

INSERT INTO Shopping\_Cart

(order\_id, t\_order\_id, no\_of\_items, total\_amount, Twitter\_handle)

VALUES ( **9876**, **4532**, **1**, **26.4**, @john )

3. Use Case: View a product already ordered through Twitter

SELECT w.product\_url

FROM Womens\_Shoes w, Twitter\_Order t

WHERE

t.product\_id = w.product\_id AND

t.Twitter\_handle = '@alice'

4. Use Case: View the products above a particular price (say $100)

SELECT w.product\_name, w.product\_url

FROM Womens\_Shoes w

WHERE

w.price > **100**;

5. Use Case: View the orders made by a user

SELECT s.Twitter\_handle, s.order\_id

FROM Shopping\_Cart s

WHERE

s.Twitter\_handle = '@bob';