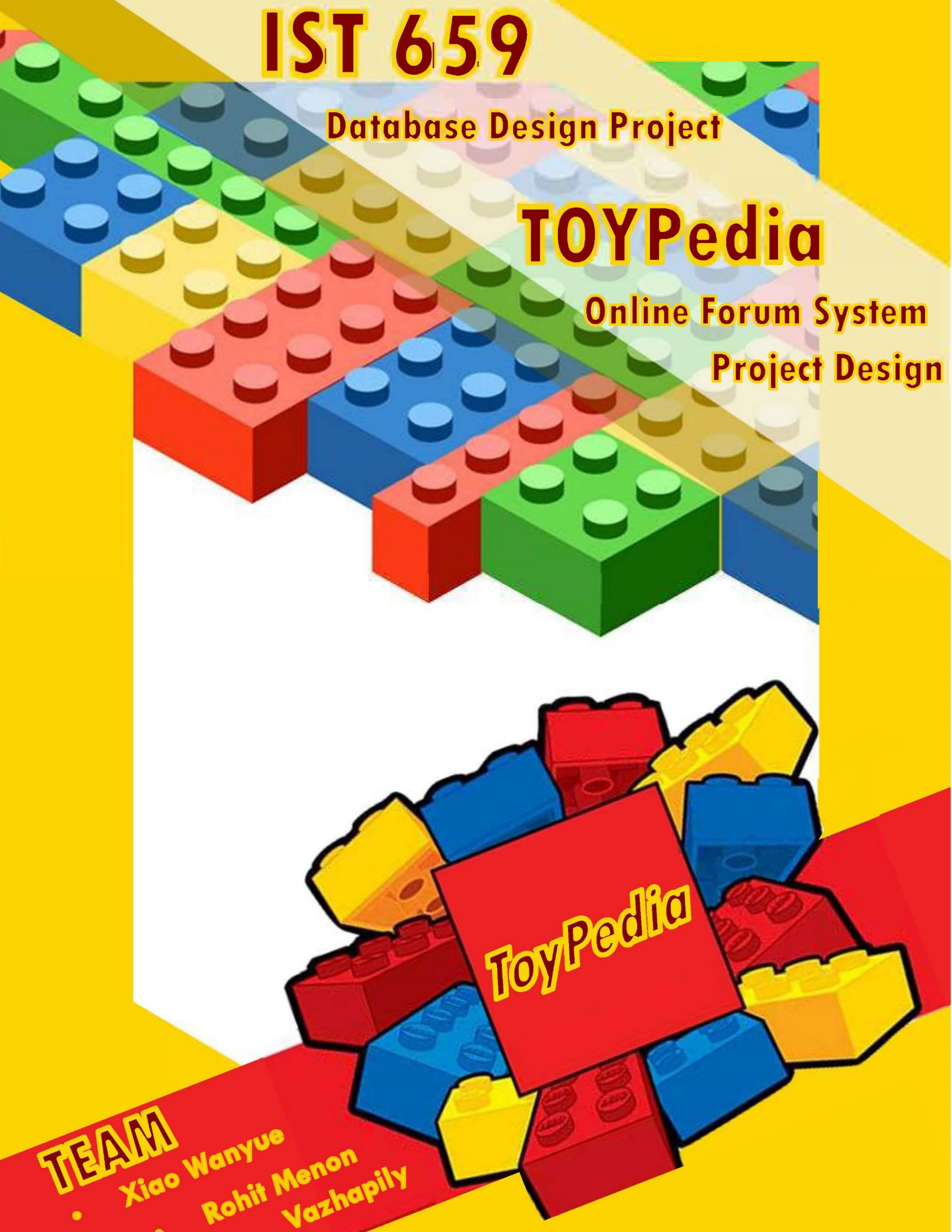
IST 659

Database Design Project

TOYPedia

Online Forum System

Project Design



TEAM

- Xiao Wanyue
- Rohit Menon Vazhapily

Table of Content

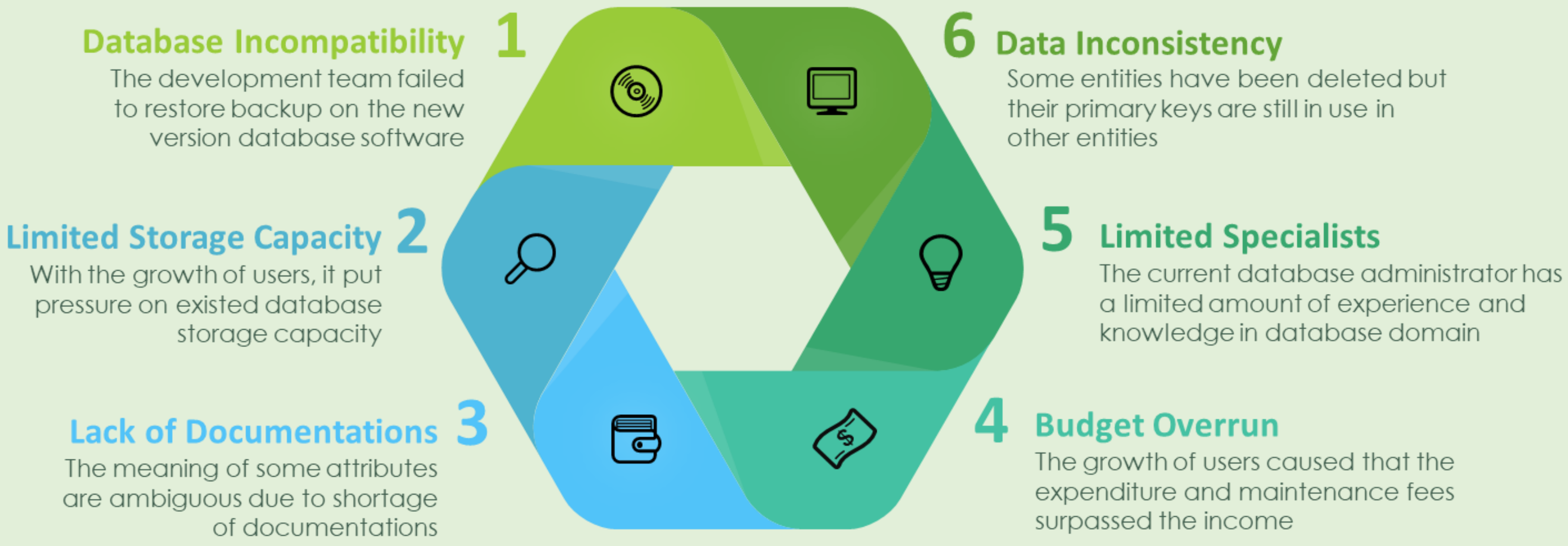
Project Summary

Background Knowledge

This project aims to design a database system for an online web-based forum called ToyPedia. As a forum developed by Yan Lee in 2012, it provides a platform for toy collectors to share information of limited collections. The original forum has several fundamental functionalities, ranging from user account management to posting and replying management. Nonetheless, the original database could not satiate the requirements of the management team due to the issues of the increasing number of users, database incompatibility, poor maintenance, and limited functions. Hence, a novel database should be developed after evaluating current business situations.

Problem Statement

Even though ToyPedia currently has a database which had already integrated with its forum, the poorly managed database caused various problems in the implementation process.


Database Incompatibility 1
The development team failed to restore backup on the new version database software
Limited Storage Capacity 2
With the growth of users, it put pressure on existed database storage capacity
Lack of Documentations 3
The meaning of some attributes are ambiguous due to shortage of documentations
4 Budget Overrun
The growth of users caused that the expenditure and maintenance fees surpassed the income
5 Limited Specialists
The current database administrator has a limited amount of experience and knowledge in database domain
6 Data Inconsistency
Some entities have been deleted but their primary keys are still in use in other entities

Proposed Solution

To tackle the problems listed above, several steps are needed to be executed.

- Detailed documentations are essential for future maintenance.
- Except keeping the original structure of the USER, EMPLOYEE, POST, and REPLY entities, the proposed database will be developed from scratch by using Microsoft SQL Server which is open-sourced and is easy to be operated.
- The rating function have been proposed. With this function, users can rate every post and every reply. Hence, new attributes that are used to store the rating score should be inserted to the original POST and REPLY entities.
- A new commercial function has been proposed. Relying on this function, users can buy products directly from the forum. To establish this function, entities related to PRODUCT, PURCHASE, SHIPPING, and PAYMENT should be created. With this commercial function, the revenue is expected to increase.
- After the development of database design, the management team could consider purchasing cloud database services, which have the benefits of cost-saving, elimination of physical infrastructure, and specialized expertise, before the data migration.

Relational Data Model

To establish a comprehensive database model which supports all the necessary basic functions an online shopping forum should equipped with, the relational data model should be divided into four parts, including

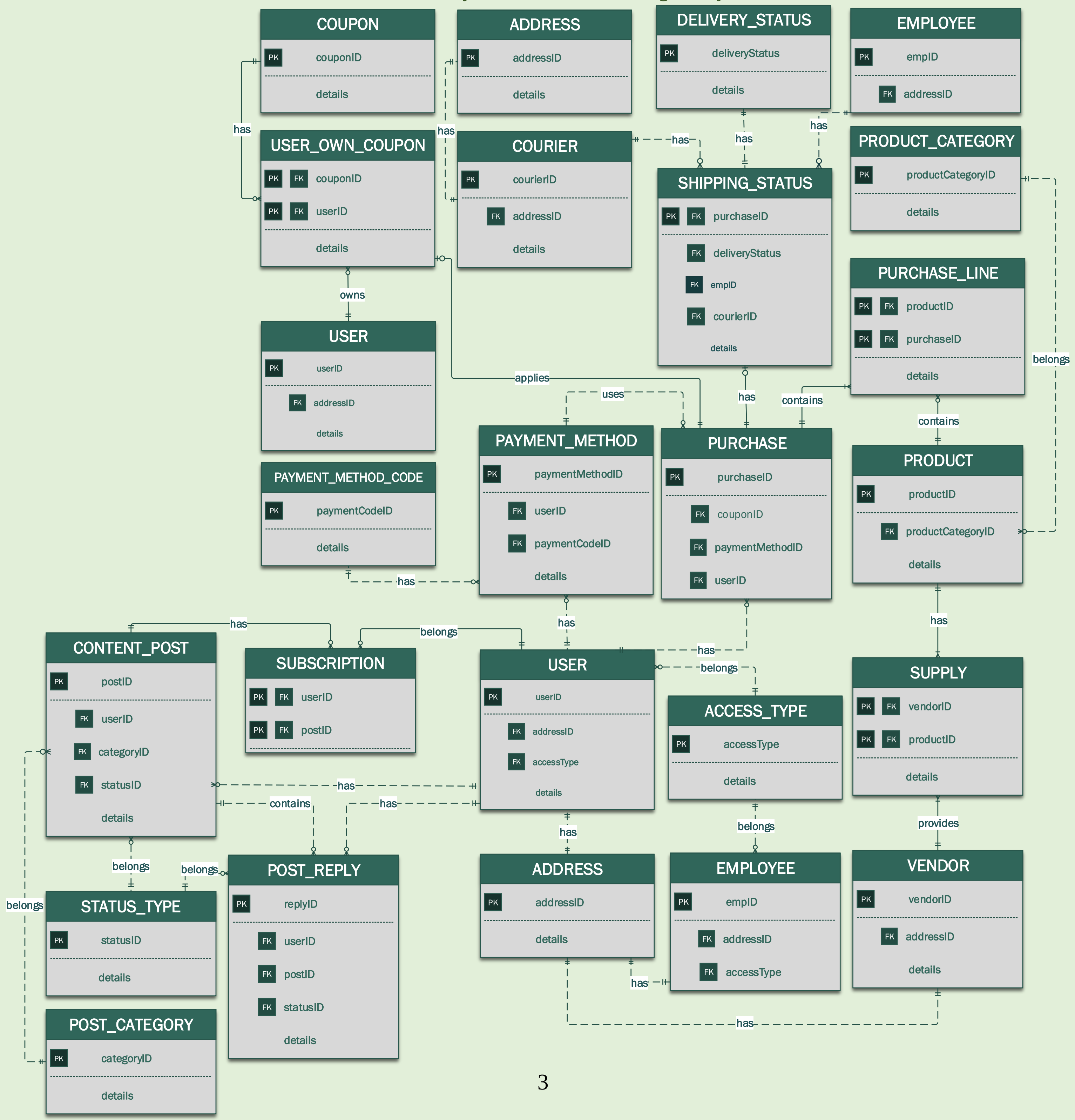
- **External User Section** which defines external users who might access data or have influence of data flow;
- **Posting and Replying section** which allow users to post, reply, rate, subscribe content;
- **Purchasing Section** which enables users to purchase products;
- **Payment and Delivery Section** which stores detailed information on payment method, transaction data, and delivery records.

General Entity Graph

Given the limited space, the entities show in general entity graph only contain primary key and foreign keys while the rest of the attributes are hidden. **The detailed entities will be illustrated in the following sections**.

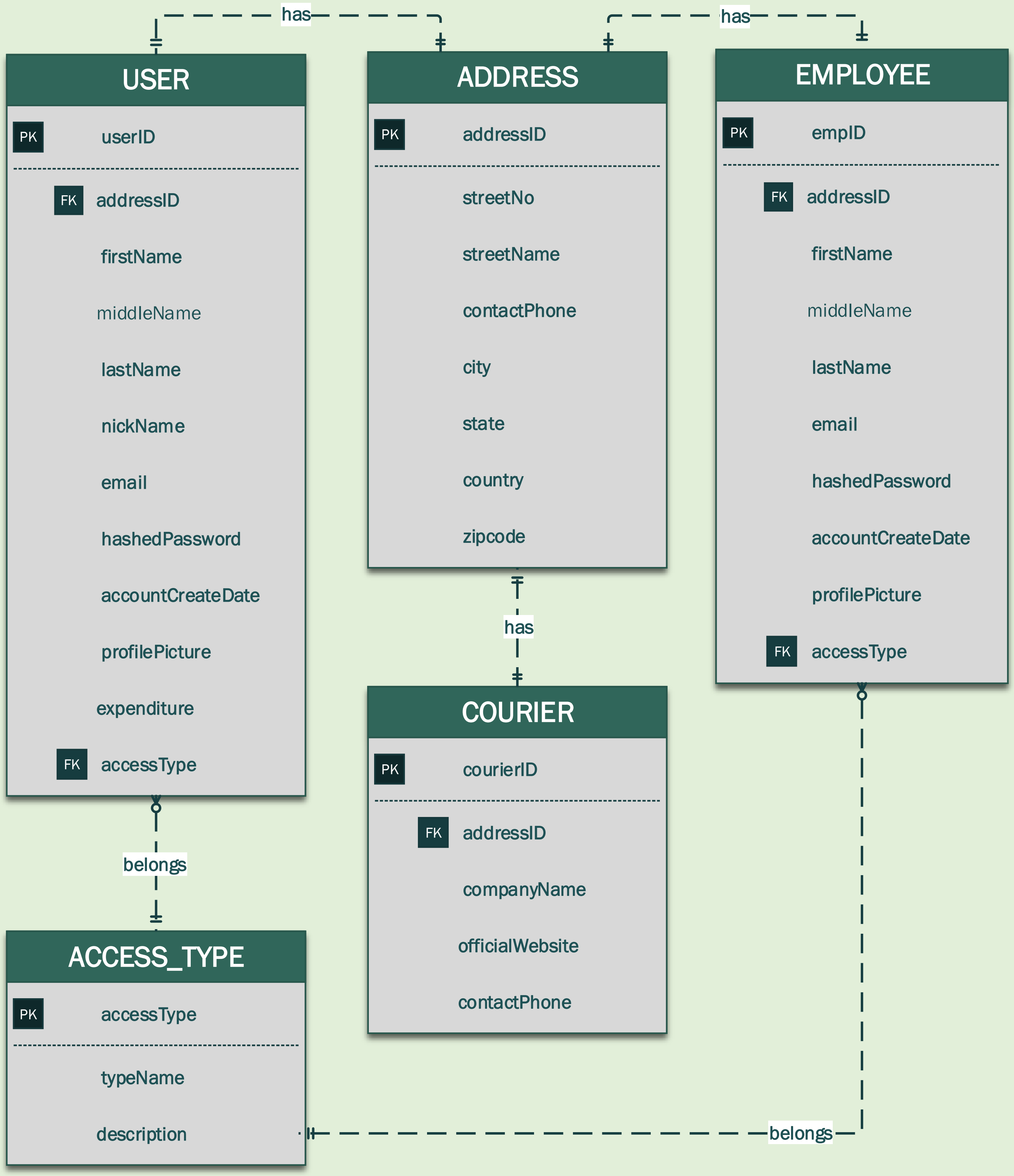
IST 659 ToyPedia Database Design Project

General Entity Graph



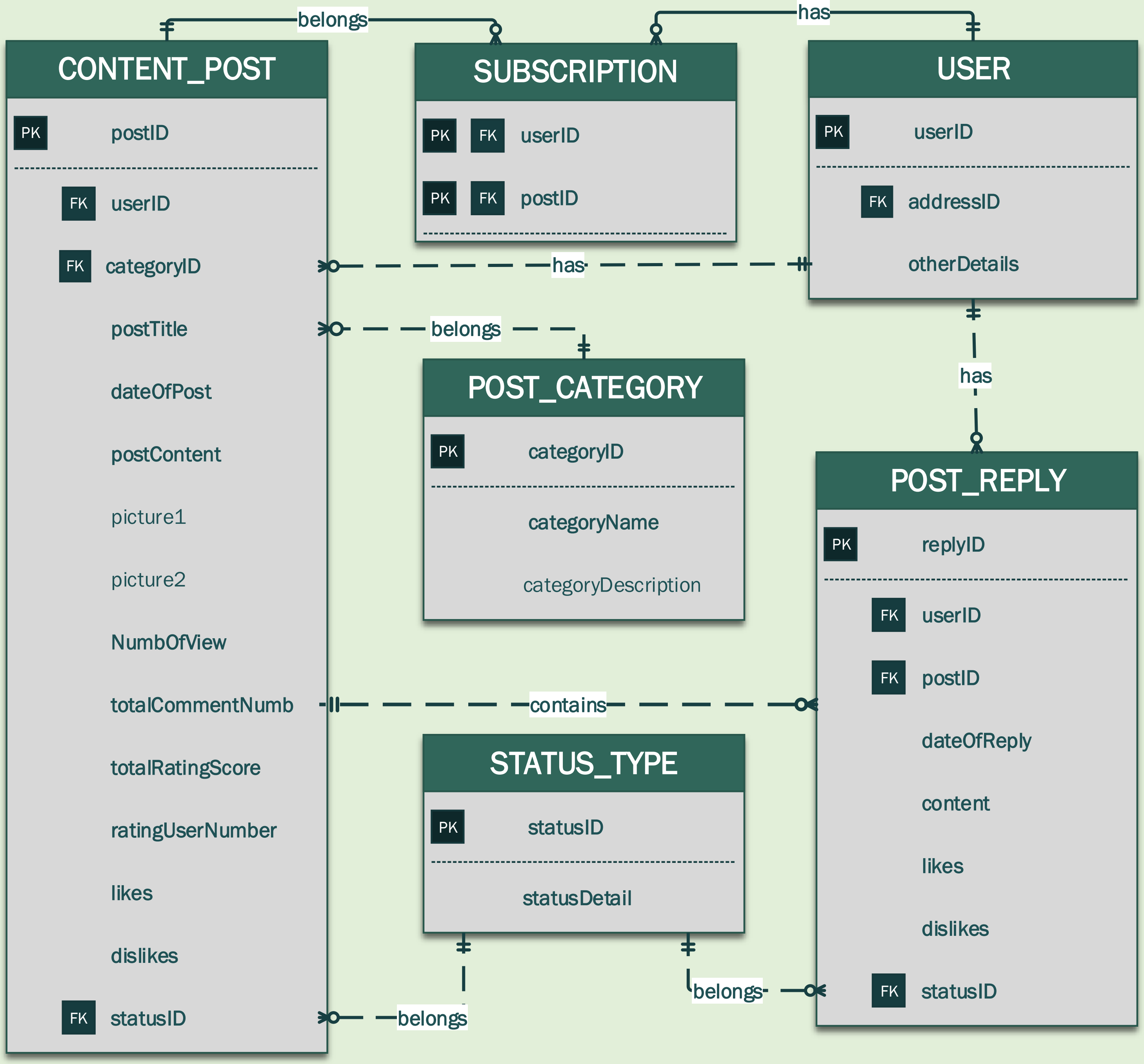
EXTERNAL USERS Entity Section

In this section, all the external users, except the VENDOR entity, who might have the right to access the database or might have a potential influence on the database will be presented with well-defined attributes.



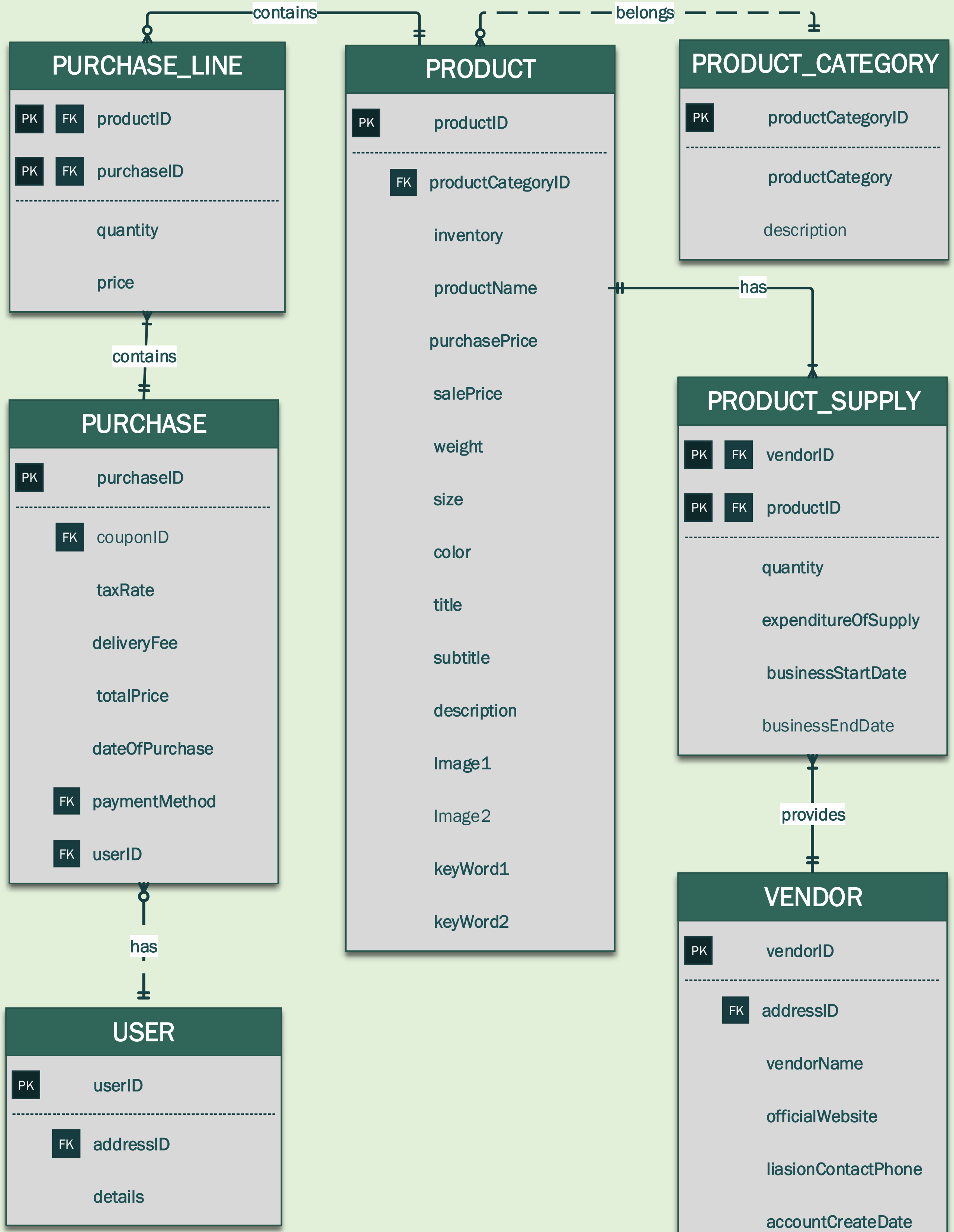
POSTING and REPLYING Section

For the section which takes in charge of posting and replying functionalities, all the entities that associate with and, more importantly, support with these functionalities will be introduced.



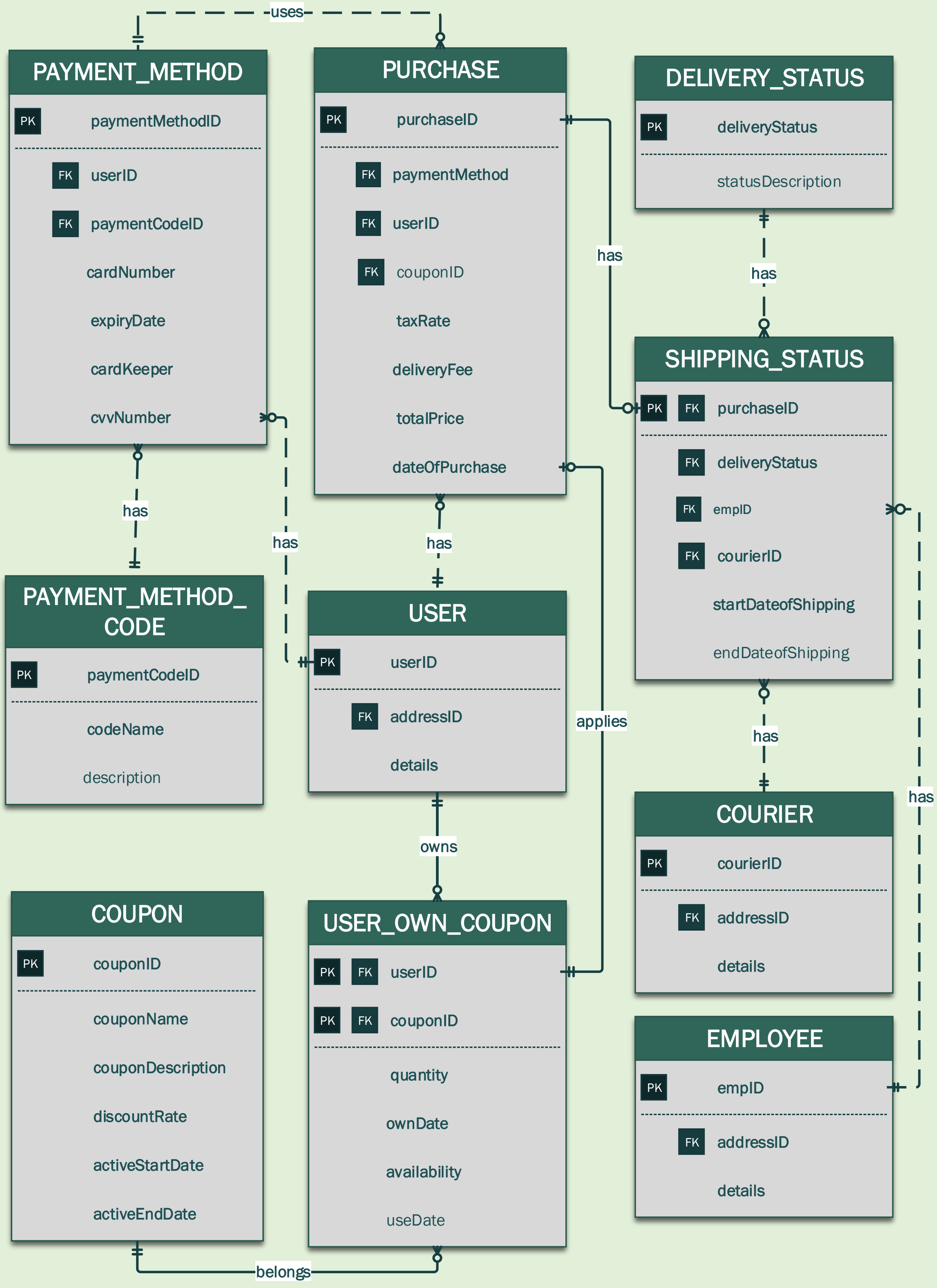
PURCHASING Section

As a section responsible for purchasing or shopping function, it illustrates entities that show relevance with order making, product selection, replenishment, and vendor information presenting.



PAYMENT & DELIVERY Section

The last section, which manages functions of bill payment and package delivery, stores not only information of payment methods but also tracks records of transactions and delivery status once transactions are being marked as completed.



Entity and Attribute Table

To help readers comprehend the diagrams and entities listed above, the table below illustrates what attributes or contents each entity should contain, explains what type of data will stored in each attribute, and indicates which entity a foreign key should be associated with and what relationship between among entities.

Entities and Attributes	Description
USER	
userID	**Primary Key**
addressID	**Foreign Key** (*ADDRESS Entity)* ♦ Each user can only have one address
accessType	**Foreign Key** *(ACCESS_TYPE Entity)* which helps to identify different types of user ♦ Each user can only belong to one access type recorded in the ACCESS_Type entity
firstName	
middleName (optional)	
lastName	
nickName	
emailAddress	
hashedPassword	Encrypted password that is used to verify the authenticity of user
accountCreateDate	Date on which the user creates his/her account
profilePicture	The working directory of picture that user chooses to present in his/her account
expenditure	Numeric value which indicates the total expenditure of each user
EMPLOYEE	
empID	**Primary Key**
addressID	**Foreign Key** (*ADDRESS Entity)* ♦ Each employee can only have one address
accessType	**Foreign Key** *(ACCESS_TYPE Entity)* ♦ Each user can only belong to one access type recorded in the ACCESS_TYPE entity
firstName	
middleName (optional)	
lastName	
emailAddress	
hashedPassword	Encrypted password that is used to ensure account safety
accountCreateDate	Date on which the employee creates his/her account
profilePicture	The working directory of the picture that employee chooses to present in their account
ACCESS_TYPE	**Stores different access right each user or employee belongs to**
accessType	**Primary Key** which records numeric number that represents different access rights (Example: 1 for Visitor; 2 for Normal User; 3 for VIP User; 4 for Normal Employee; 5 for Sales Manager; 6 for Administrator)
typeName	Name of Role (Example: Visitor, Normal User, VIP User, Normal Employee, Sales Manager; Administrator)

description	Description that details the content of each access type
COURIER	**Company that helps the management team to ship packages to user**
<u>courierID</u>	**Primary Key**
AddressID	**Foreign Key** *(ADDRESS Entity)* ♦ Each courier can only have one address recorded in the database.
companyName	
officialWebsite	Stores the URL of official website of each courier
contactPhone	
ADDRESS	
<u>addressID</u>	**Primary Key**
streetNo	Records apartment or room number
streetName	Records the street name
contactPhone	
city	
state	
country	
zipcode	
STATUS_TYPE	**Records different status that applied to user's posting and replying**
<u>statusID</u>	**Primary Key**
statuseDetail	Status that assigns to each post and reply (Example: Waiting for Approval; Approved, Rejected, Reported as Spam)
POST_CATEGORY	**Records different category or section posting belongs to**
<u>categoryID</u>	**Primary Key**
categoryName	
categoryDescription (optional)	Detailed explanation of each post category
CONTENT_POST	
<u>postID</u>	**Primary Key**
userID	**Foreign key** *(USER Entity)* ♦ Each user can have many postings while each posting only belongs to one specific user
categoryID	**Foreign key** *(POST_CATEGORY Entity)* which implies the category each posting belongs to ♦ Each posting belongs to exactly one posting category
statusID	**Foreign key** *(STATUS_TYPE Entity)* which implies the status of each posting ♦ Each posting only can be marked with one status.
postTitle	Title of post that creates by user
dateOfPost	Date on which user posts the content
postContent	
picture1 (optional)	The working directory of first of picture that user decides to insert into the content
picture2 (optional)	The working directory of second of picture that user decides to insert into the content
numberOfView	The number of views each posting has

totalCommentNumb	The number of comments that replied by users
totalRatingScore	A score that accumulates by score of all users who rate the post
ratingUserNumber	A numeric value that indicates what rating score each posting could gain, setting the default value is 0
likes	The number of users who like the posting, setting default value is 0
dislikes	The number of users who dislike the posting, setting default value is 0
POST_REPLY	**Records all the replies that post by users**
replyID	**Primary Key**
userID	**Foreign Key** *(USER Entity)* ♦ Each user can have many replies while each reply only belongs to one specific user
postID	**Foreign Key** *(CONTENT_POST Entity)* ♦ Each reply only belongs to one specific posting while one posting can have many replies
statusID	**Foreign Key** *(STATUS_TYPE Entity)* ♦ Each reply only can be marked with one status
dateOfRply	Date on which user posts the reply
content	
likes	The number of users who like the posting, setting default value is 0
dislikes	The number of users who dislike the posting, setting default value is 0
SUBSCRIPTION	**Records the subscription detail between postings and users**
userID	**Primary Key + Foreign Key** *(USER Entity)* ♦ Each tuple (row) in subscription entity can only contain one specific user
postID	**Primary Key + Foreign Key** *(CONTENT_POST Entity)* ♦ Each tuple (row) in subscription entity can only contain one specific post
PRODUCT	**Stores every detailed information of each product that available for user to purchase**
productID	**Primary Key**
productCategoryID	**Foreign Key** *(PRODUCT_CATEGORY Entity)* which implies the category of each product ♦ Each product can only belong to one specific category that recorded in PRODUCT_CATEGORY Entity
inventory	Available inventory number of each product
productName	
purcahsePrice	Price that management team paid for each product
salePrice	Price that management team sold to customer
weight	
size	
color	
title	A short introduction of each product
subtitle	A detailed introduction of each product
description	Detailed information of each product in every aspect
image1	The working direction of the first picture that shows the appearance of product

image2 (optional)	The working direction of the second picture that shows the appearance of product
keyWord1	The keyword which represents the feature of product
keyWord2	The keyword which represents the feature of product
PRODUCT_CATEGORY	**Stores different types of category that each product belongs to**
productCategoryID	**Primary Key**
productCategoryName	Records different type of product that is available in the purchasing section (Example: Model; Puzzle; Comic Book; Game)
description	Description or explanation of product category
VENDOR	
vendorID	**Primary Key**
addressID	**Foreign Key** *(ADDRESS Entity)* ♦ Each vendor can only have one address recorded in database
vendorName	
officialWebsite	Stores the URL of the official website of each vendor
liaisonContactPhone	Stores the phone number of each vendor's liaison
accountCreateDate	Date on which the vendor account has been created
PRODUCT_SUPPLY	**Records every transaction between the management team and each vendor**
vendorID	**Primary Key + Foreign Key** *(VENDOR Entity)* ♦ Given that one product can be supplied by various vendors while one vendor can supply plenty of products. Each tuple (row) of PRODUCT_SUPPLY Entity should only contain one specific vendor ♦ One specific product must be supplied by at least one specific vendor while one specific vendor must supply one specific product (Example: A bookstore must supply one product, such as the Lord of The Ring, to ToyPedia. If not, the information of this vendor will be deleted. Similarly, a product must be supplied by one specific vendor, otherwise this product will be deleted from the database.)
productID	**Primary Key + Foreign Key** *(PRODUCT Entity)* ♦ Each tuple (row) of PRODUCT_SUPPLY Entity should only contain one specific product.
quantity	The quantity of product ordered by the management team for each transaction
businessStartDate	Date on which the management team makes the order
businessCategory	Is this the first transaction that occurs between the management team and vendor? (Yes/No)
businessEndDate (optional)	Date on which management team received the products shipped by vendor
PURCHASE	**Records the information of exactly one purchase, including the total expenditure and tax rate of products contained in the purchase, coupon that might be for the purchase, and date that purchase made**
purchaseID	**Primary Key**
couponID (optional)	**Foreign key** *(USER_OWN_COUPON Entity)* that implies which coupon user decides to use in the purchase ♦ Each coupon can only be applied to one specific purchase if user decide to use that coupon

paymentMethodID	**Foreign key** *(PAYMENT_METHOD Entity)* that implies which available payment method user chooses to pay the bill ♦ Each purchase can only be paid by one specific payment method
userID	**Foreign Key** *(USER Entity)* ♦ Each purchase should only be made by one user
taxRate	Stores specific tax rate according to different region user belongs to
deliveryFee	Stores the delivery fee which might vary according to user's shipping address and courier chosen by user
totalPrice	Stores the total expense of each purchase, which equals to the sum of prices stored in ORDER Entity with same purchaseID plus the tax of that sum
dateOfPurchase	Date on which user ordered the purchase
PURCHASE_LINE	**Stores the product ID, quantity of that product, and the total price after multiplying the net sale price with the quantity of the product**
<u>productID</u>	**Foreign Key + Primary Key** *(PRODUCT Entity)* ♦ Each order can only contain one specific product
<u>purchaseID</u>	**Foreign Key + Primary Key** *(PURCHASE Entity)* ♦ One purchase can associate with many purchase lines while one purchase line can only belong to one specific purchase ♦ One purchase must associate with at least one specific purchase line
quantity	The number of product that stored in each order
price	Equals to the value of multiplying the quantity of that product with its corresponding net sale price
COUPON	Stores information about different type of coupon
<u>couponID</u>	**Primary Key**
couponName	The specific name of each coupon
couponDescription	Detailed description of each coupon
discountRate	Specific discount rate or amount of money that will be used to deduct the total expense when user using this coupon
activeStartDate	Date from which user can use the coupon
activeEndDate	Date on which coupon will expire
USER_OWN_COUPON	**Stores coupons that owns by each user**
<u>couponID</u>	**Primary Key + Foreign Key** *(COUPONS Entity)* ♦ Given that each type of coupon can be available for diverse users while one user can possess various type of coupons, each tuple (row) in the USER_OWN_COUPON Entity can only contain one specific coupon (Example: user A can have a coupon named "SAVEMORE" and a coupon named "30DISCOUNT". It will take two rows to store the ID of user A and these couponID.)
<u>userID</u>	**Primary Key + Foreign Key** *(USER Entity)* ♦ Each tuple (row) in the USER_OWN_COUPON Entity can only contain one user
quantity	The number of one specific coupon that owns by user (Example: user A have six "SAVEMORE" coupons, implying that user A can apply this coupon in six different purchase.)

ownDate	Data on which user owns the coupon
availability	Does this type of coupon still available to use for user? (Yes/No)
useDate (optional)	Date on which user used the coupon
PAYMENT_METHOD	**Stores information of various payment method that user might use to pay their bill**
paymentMethodID	**Primary Key**
userID	**Foreign Key** *(USER Entity)* ♦ Each payment method only belongs to one specific user
paymentCodeID	**Foreign Key** *(PAYMENT_METHOD_CODE Entity)* ♦ Each payment method should only have one payment code
cardNumber	The card number of the payment method
expiryDate	Date on which this card will expire
cardKeeper	Stores the name of card owner
cvvNumber	
PAYMENT_METHOD_CODE	**Stores different types of payment method that user can choose when paying the bill**
paymentCodeID	**Primary Key**
codeName	Name of different payment method (Example: CC stands for Credit Card, DC stands for Debit Card)
description (optional)	
SHIPPING_STATUS	**A statement user receives after paying the bill, recording information of purchase ID, courier who takes in charge of package delivery, the status of delivery**
purchasID	**Primary Key + Foreign Key** *(PURCHASE Entity)* ♦ Each shipping_status belongs to one purchase ♦ Correspondingly, one purchase can either have only one specific shipping_status or no shipping_status if the payment of that purchase does not finish
deliveryStatus	**Foreign Key** *(DELIVERY_STATUS Entity)* ♦ Each shipping_status can only have one delivery_status at one time, even though the delivery_status will change with the according to the location of package ♦ One type of delivery status can be applied to multiple invoices
courierID	**Foreign Key** *(COURIER Entity)* ♦ Each invoice can only have courier for package shipping, even though the courier might change in case accidence happened
empID	**Foreign Key** *(EMPLOYEE Entity)* ♦ Each shipping_status can only be authorized by one specific employee while one employee can either authorize none or many shipping_status
startDateofShipping	The date on which courier ship the package(s)
endDateofShipping (optional)	The date on which the package is received by user
DELIVERY_STATUS	**Stores information that represent different type of delivery status**
deliveryStatus	**Primary Key** which stores possible delivery status (Example: Delivery Completed; Out of warehouse; Damaged; Lost; Preparing for shipping)
description (optional)	Detailed explanation about each delivery status

Business Rules

1) Each employee has a unique discount rate when they purchase an item.
2) The delivery fee of an order should be free if the order is above $20.
3) The final rating of a post should be the average of all the ratings given by many users.
4) A product can be supplied by many vendors and a vendor can supply multiple products.
5) A user must provide his or her credit/debit card details in order to purchase a product.

Major Data Questions

1) Which product is sold the most by the forum?
2) Which user buys the greatest number of products from the forum?
3) Which product has the highest revenue?
4) What is the average number of orders per user?
5) What is the annual revenue of each product supplied by the forum?
6) Which city has the greatest number of orders?
7) Which user's post gets the highest rating?
8) Which product is sold the least by the forum so that the forum can discontinue the product?