**BestBuy Design**

CS669

SiCheng Yi

**‘BestBuy’ Project Direction Overview**

I want to develop an application called 'BestBuy', its purpose is that when someone needs to buy something (especially a lot of items), he may refer to a shopping site (such as 'Walmart', 'Amazone' and so on), may refer to many shopping websites, and the same shopping website also has a large number of similar items to refer to, maybe he will find some items he thinks well in the process, and hope to compare the details to find the best among them after looks many items. At this time, they need to keep a large number of web pages and watch them one by one for comparison, which will cause many web pages to exist above the browser, and the basic information also needs to be memorized by themselves. It's easy to feel complicated.

'BestBuy' allows users to provide a 'storage' option when they see a good choice, and can finally categorize them in a unified manner, and put the same type of items sort by select (such as 'price', 'rating') , so that the user can more easily distinguish the better ones.

Here is are some brief examples of how someone would use the application：

SiCheng has just moved to Boston, and he needs to buy furniture for his newly rented house, which includes bed frames, mattresses, chairs, and tables. He browses many items on Amazone and Walmart, and when he sees a good item, he chooses to store it on 'BestBuy'. After five hours, he was done browsing and was ready to make his final choice. He opened the 'BestBuy' interface, and sorted the 'bed frame' type products he stored by price, and then entered the detailed interface through the URL of each product to refer to the evaluation. In this way, he made a final comparison of the bed frame, mattress, chair, and table and purchased the most satisfactory product among them.

While obviously there will be a significant programming component for 'BestBuy', for this course I am focusing on the database component. The database of 'BestBuy' will be used by many people, so it needs to store each user's personal information, the database will need to associate the purchases with different accounts. 'BestBuy' will record each item's 'type', 'name', 'price', 'rating', 'sale platform', 'stored date' and so on. Some details It will be more clear in the coming weeks.

**BestBuy Use Cases and Fields**

The first important usage of the database is when a person signs up for an account and installs the application.

Account Signup/Installation Use Case

1. The person visits TrackMyBuys’ website or app store and installs the application.

2. The application asks them to create an account when its first run.

3. The user enters their information and the account is created in the database.

4.The application asks them to install browser plugins so that their purchases can be automatically tracked when they make them, and store the information when the user needs it.

From a database perspective, this use case requires storing information about accounts (from steps 2 and 3). Steps 1 and 4 apply to the user and application but not the database directly.

Significant fields for an account for this application are listed in the table below.

|  |  |  |
| --- | --- | --- |
| **Field** | **What it Stores** | **Why it’s Needed** |
| UserName | The name displayed by the user in the program will be used for some sharing, ranking and so on. | Some user maybe have many account, and some people also don’t want other people know their real name. |
| FirstName | This is the first name of the  account holder. | This is necessary for displaying the person’s name on screens and addressing them when sending them emails or other communications |
| LastName | This is the last name of the  account holder. | This is necessary for displaying the person’s name on screens and addressing them when sending them emails or other communications |
| CreatedDate | Date the account was created. | It is useful to stores the date io distinguish old and new users, it can be used for various activities. |
| Mail | Stores the e-mail of user. | Can be used to send various emails to users, this function can be turned off during registration. |
| PhoneNumber | Stores the phone number of user. | Use for contacting users. |
| Balance | Stores the balance of user. | Some features may require payment to use, and are therefore used to store the user's current balance. |

The second important usage of the database is to use browser extensions to store product information that the user wants to store.

Save Store Product Information Use Case

1. The user goes to the product detail page and chooses to save.
2. BestBuy receives the order, and stores the product's selling platform, itemtype, name, price, evaluation, and URL in the database.

Significant fields are detailed below.

|  |  |  |
| --- | --- | --- |
| **Field** | **What it Stores** | **Why it’s Needed** |
| SellingPlatform | The item’s selling platform name | Let the user know which on-line shopping platform sell the item. |
| Type | The item’s type. | Used to categorize products according to their type. |
| Name | The item’s name. | Used to let users understand the name of the product. |
| Price | The item’s price. | Let user know the price, also can use to sort. |
| evaluation | The item’s evaluation | Let user know the evaluation, also can use to sort. Most of the current websites are on a five-point scale, and buyers' detailed reviews can be viewed by using the URL. |
| URL | The purchase URL of the item. | It is convenient for users to enter the detailed purchase interface to perform more operations. |

The third important usage of the database will be used to recommend items to users.

Recommend Items Use Case

1. The user opens the 'BestBuy' program interface.
2. Select 'Product Recommendations'.
3. ‘BestBuy’ gives the user various conditions, such as brand, selling platform, price range, etc.
4. User choose various conditions what their want.
5. The application pulls all purchases matching the criteria from the database, sorted by the number of times the item has been stored by all users using 'BestBuy', and displayed to the user.
6. The user selects the item they are interested in.
7. The application extracts the product's storage information from the database and displays it to the user.
8. The user can store it in his own account, or make other choices.

This database will use the database which is save store product information. On this basis, some additional content will be added for recommended sorting.

Significant fields are detailed below.

|  |  |  |
| --- | --- | --- |
| **Field** | **What it Stores** | **Why it’s Needed** |
| Savetime | Save how many user save this item. | To a certain extent, this can indicate how many users are interested in him, and users can be recommended according to this data. |
| Promote | Store how much promotion fee the merchant offers. | Merchants can promote the application by paying a promotion fee, which can be arranged according to the promotion fee. |
| SellingPlatform | The item’s selling platform name | Let the user know which on-line shopping platform sell the item. Also used to provide the user with a search interval limit. |
| Type | The item’s type. | Used to categorize products according to their type. Also used to provide the user with a search interval limit. |
| Name | The item’s name. | Used to let users understand the name of the product. Also used to provide the user with a search interval limit. |
| Price | The item’s price. | Let user know the price, also can use to sort. also used to provide the user with a search interval limit. |
| evaluation | The item’s evaluation | Let user know the evaluation, also can use to sort. Most of the current websites are on a five-point scale, and buyers' detailed reviews can be viewed by using the URL. Also used to provide the user with a search interval limit. |
| URL | The purchase URL of the item. | It is convenient for users to enter the detailed purchase interface to perform more operations. |

**Structural Database Rules**

From the step: ‘Account Signup/Installation Use Case’ we can find an entity-Account.

From the step: ‘Save Store Product Information Use Case’ we can find six significant data points: selling platform, type, name, price, evaluation, and URL.

We can get some structural rule:

1:Each save is associated with an account;Each account may be associated with one to many saves.

2: Each save is associated with an selling platform; Each platform associated with one to many saves.

3: Each save has one type; Each type has one to many saves.

4: Each save has one name; Each name associated with one to many saves.

5: Each save has one price; Each price has one to many saves.

6: Each save has one to many evaluations; Each evaluation be from one save.

7: Each save has one URL; Each URL associated with one save.

From the step: ‘Recommend Items Use Case’ we can add two significant data points: Savetime and Promote.

We can get some structural rule:

8: Each save associated with one savetime; Each savetime associated with one save.

9: Each save has one promote; Each promote effect one to many saves.

So from the three use cases I have thus far, I have nine structural database rules:

1:Each save is associated with an account;Each account may be associated with many saves.

2: Each save is made from one selling platform; Each platform has one to many saves.

3: Each save has one type; Each type has one to many saves.

4: Each save has one name; Each name has one to many saves.

5: Each save has one price; Each price has one to many saves.

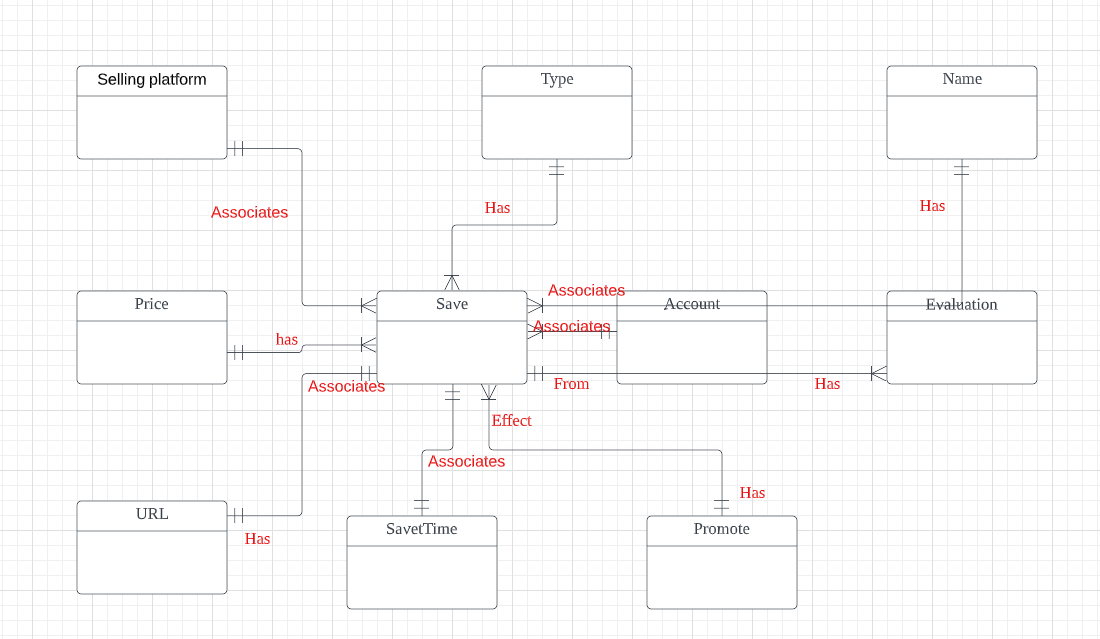
6: Each save has one to many evaluations; Each evaluation be from one save.

7: Each save has one URL; Each URL appear one save.

8: Each save associated with one savetime; Each savetime associated with one save.

9: Each save has one promote; Each promote effect one to many saves.

**Entity‐relationship diagram**



**Adding Specialization-Generalization**

By checking all my cases, after thinking, I judged that "Account Signup/Installation Use Case" can be modified.

Account Signup/Installation Use Case

1. The person visits TrackMyBuys’ website or app store and installs the application.

2. The application asks them to create an account when its first run.

3. The user enters their information and the account is created in the database.

4.The application asks them to install browser plugins so that their purchases can be automatically tracked when they make them, and store the information when the user needs it.

As expected, I will separate accounts into buyers and sellers, and I will offer a free account that has some limitations on the functionality, and a paid account which offers all the features. I modify the use case as follows：

1. The person visits TrackMyBuys’ website or app store and installs the application.

2. The application asks them to create a buyer or seller account and choose whether to pay when its first run.

3. The user enters their information and the account is created in the database.

4.The application asks them to install browser plugins so that their purchases can be automatically tracked when they make them, and store the information when the user needs it.

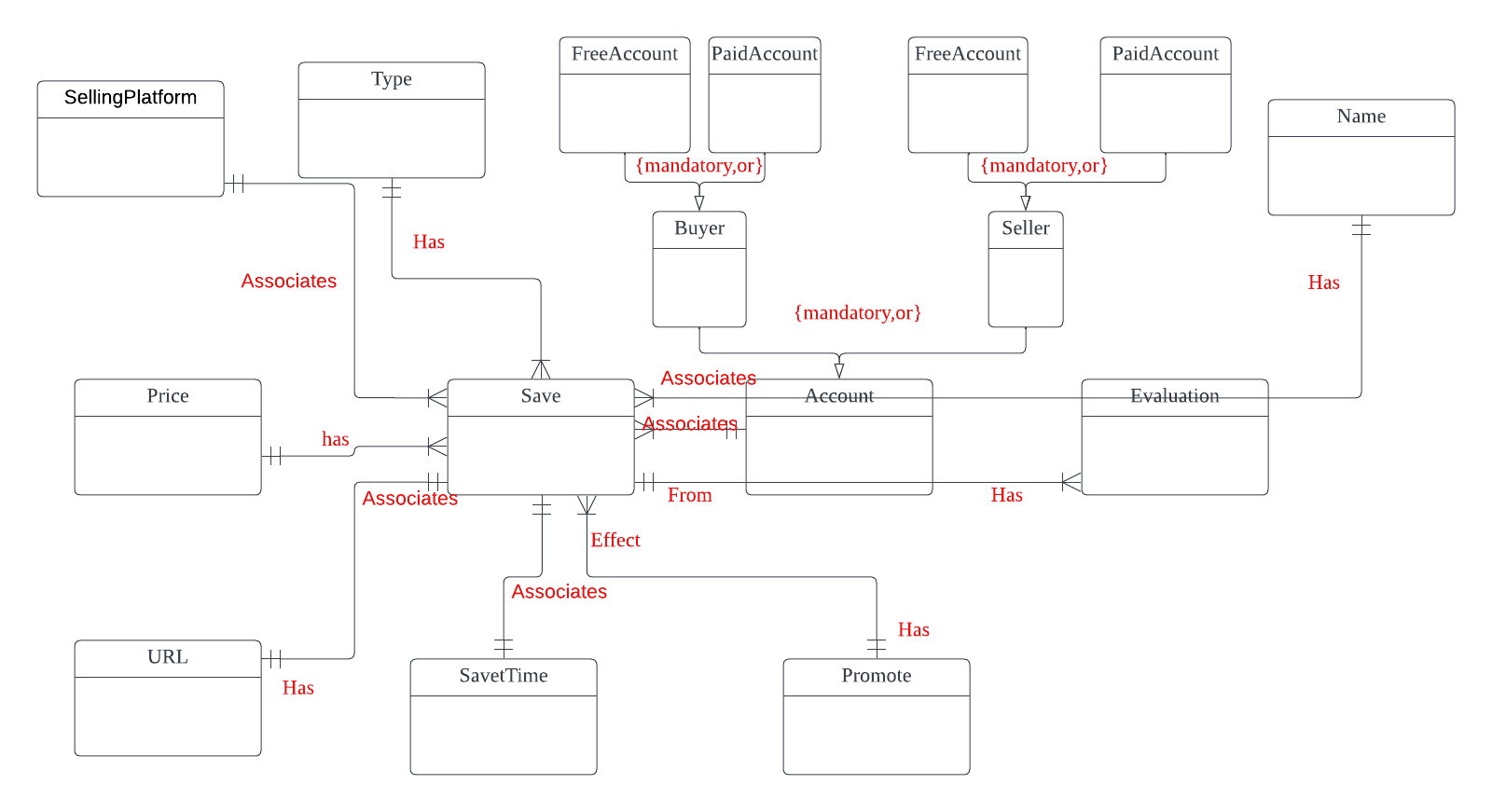
According to the #2 change, I deduce two rules:

8. An account is a buyer account or a seller account.

9. An account is a free account or a paid account.

My database only has two kinds of accounts: buyer and seller, and that is the complete list. And each account has two relationships: free and paid. It's also a complate list, the relationship is totally complete. The relationship is disjoint.

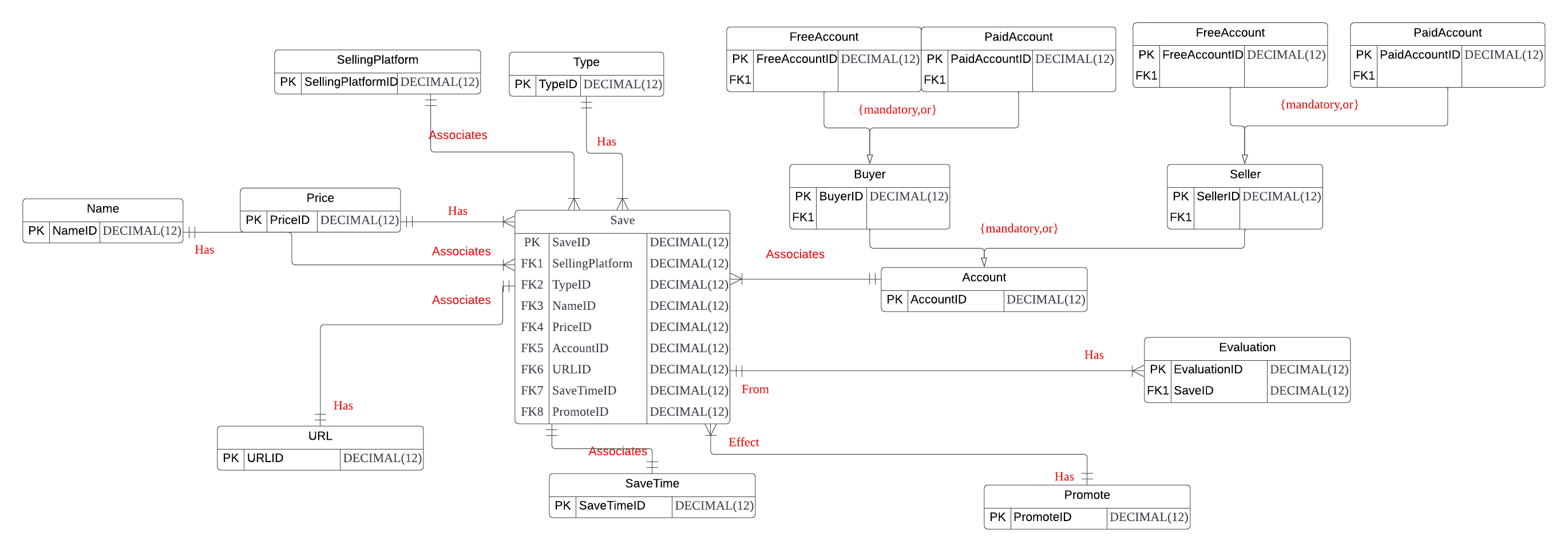
Here is the ERD:



**Initial DBMS Physical ERD**

I opted to use the conceptual ERD to identify the relationships. After observation, I found that my ERD is basically composed of 1:1 and 1:M, so it is relatively simple to convert it into a physical ERD, just add the corresponding PK and RK.

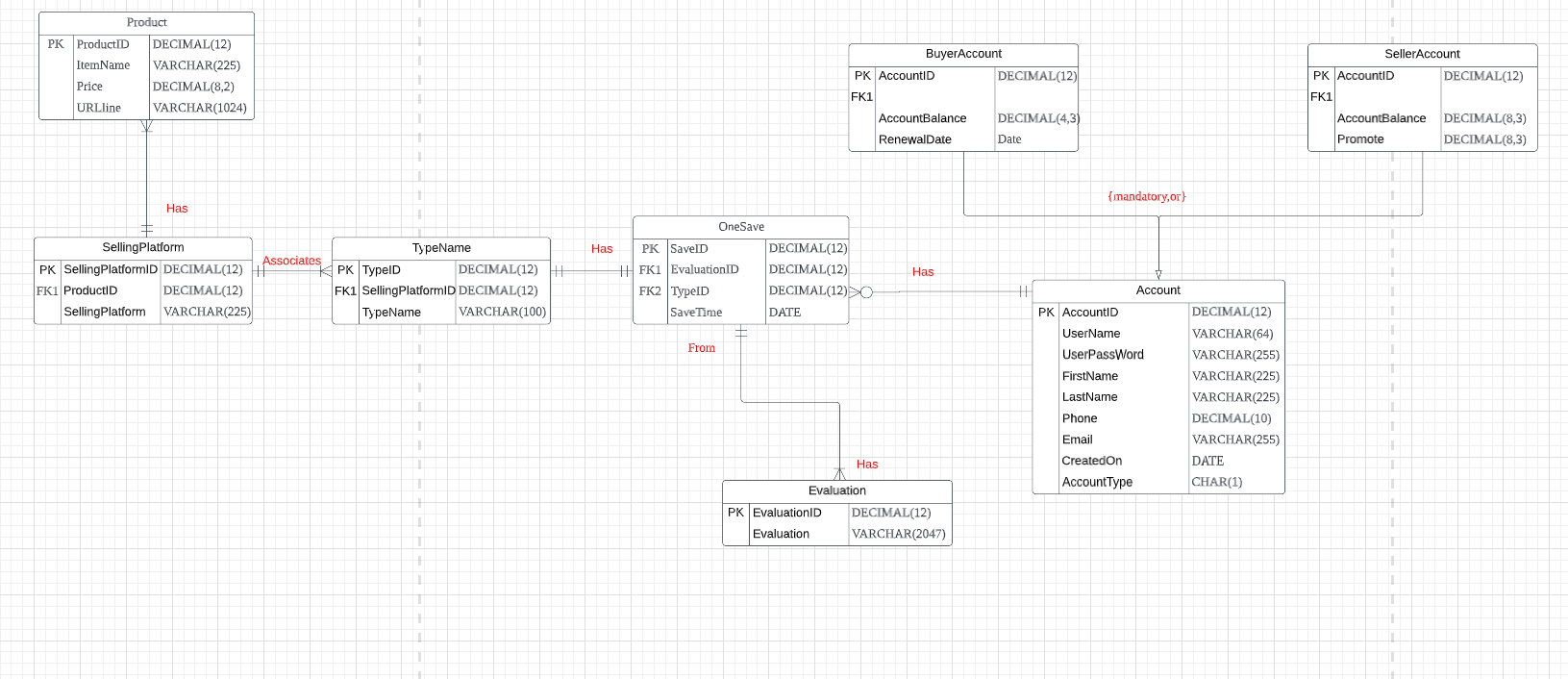
Here is the Physical ERD:



**BestBuy Attributes**

In order to make the structure clearer and reduce redundant consumption, I made major changes. I unified various product information into a new table: Product, including ItemName, Price, URL, and then Associate Product to each platform, and then to the type of product, Associate the Type with a table containing the data savetime as a storage. I no longer divide the account into paid and free, but simply separate it as Buyer and Seller, where Buyer can open a limited-time paid function by paying a small fee, and Seller can provide recommendations for their stored products by paying an amount. Here are the details:

|  |  |  |  |
| --- | --- | --- | --- |
| **Table** | **Attribute** | **Datatype** | **Reasoning** |
| Product | ItemName | VARCHAR(225) | This is just the name of the item, so I don’t think it need pretty long of the name, 225 is enough. |
| Product | Price | DECIMAL(8,2) | This is the price of the item, I think tens of millions of dollars is the limit for online shopping transactions, and many merchants like to add .99 to their prices, so I provide 2 decimal places. |
| SellingPlatform | SellingPlatform | VARCHAR(225) | This is the SellingPlatform name, just like the item name. |
| TypeName | TypeName | VARCHAR(100) | I think the type should not be pretty long, so the varchar(100) is enough. |
| Product | URLline | VARCHAR(1024) | I noticed that some sites have very long URLs, so I designed a larger space. |
| Evaluation | Evaluation | VARCHAR(2047) | Many people will write a paragraph or even several paragraphs in some comments, so I give more space for users to write more content. |
| OneSave | SaveTime | DATE | This is the time of the date save on the database, so it is a DATE data. |
| Account | UserName | VARCHAR(64) | This is just the username of the account, not pretty long and short. |
| Account | UserPassword | VARCHAR(255) | I just set enough space to let the user can make a long password for safety. |
| Account | FirstName | VARCHAR(225) | I don’t think the name of a parson will be pretty long, this is just for guarantee have enough space. |
| Account | LastName | VARHCAR(225) | I don’t think the name of a parson will be pretty long, this is just for guarantee have enough space. |
| Account | Phone | DECIMAL(10) | I think 10 digits is enough to provide area codes + phone numbers for most countries. |
| Account | Email | VARCHAR(255) | Provide enough space for various email accounts. |
| Account | CreatedOn | DATE | A date variable representing the date the account was created. |
| Account | AccountType | CHAR(1) | A char used to distinguish buyer account from seller account, expected to be 0 and 1. |
| BuyerAccount | AccountBalance | DECIMAL(4,3) | Buyer users only need to pay a small fee to use paid features for a period of time, so I don't think anyone will save more than $10,000. |
| BuyerAccount | RenewalDate | DATE | Expiration date for account paid features |
| SellerAccount | AccountBalance | DECIMAL(8,3) | Seller users are likely to pay more for their product to get a referral, so I offer a larger amount. |
| SellerAccount | Promote | DECIMAL(8,3) | This variable is meant to represent the amount that has been used |



I noticed that after the adjustment, it seems to be in the canonical state, and I don't need to make any additional adjustments.

**BestBuy CREATE TABLE**

**图形用户界面, 应用程序, Teams

描述已自动生成**

**BestBuy Indexing**

As far as primary keys which are already indexed, here is the list:

Account.AccountID

BuyerAccount.AccountID

SellerAccount.AccountID

OneSave.SaveID

Evaluation.EvaluationID

TypeName.TypeID

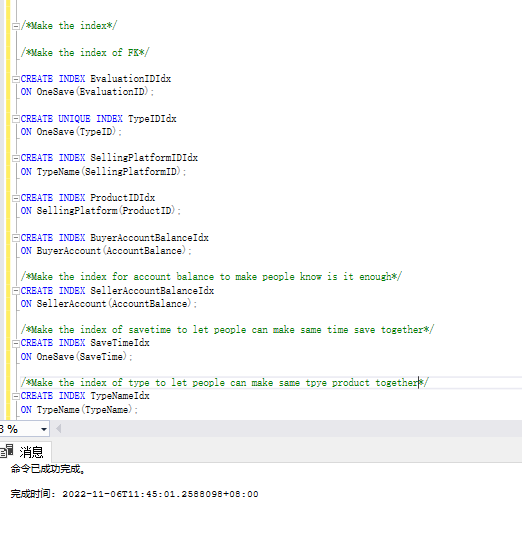
SellingPlatform.SellingPlatformID

Product.ProductID

As far as foreign keys, all of it need to have an index:

|  |  |  |
| --- | --- | --- |
| **Column** | **Unique** | **Description** |
| OneSave.EvaluationID | Not unique | Not unique because one save can have many evaluation |
| OneSave.TypeID | Unique | Unique because one save can have only one type |
| TypeName.SellingPlatformID | Not unique | Not unique because one type can from many platform |
| SellingPlatform.ProductID | Not unique | Not unique because one platform can have many product |

As far as the three query driven indexes, First I set is account balance for many people want to know their balance. Second is savetime because we often need to put together like the save on one day. Third I set the TypeName to make many people need the same type together.



**BestBuy Summary and Reflection**

My database is an application called 'BestBuy', usually, each shopping platform can only sort all the products on the platform, and 'BestBuy' will store all the products that the user is interested in, and in the way the user wants The sorting of categories is convenient for users to perform more refined filtering. The program is dedicated to creating a list of favorite products for users, and can even rate and recommend products based on other people's storage. I also divided the account into buyer account and seller account, and each one is divided into free account and paid account.

When I was designing the database, I realized that 'BestBuy' might be considered a platform, so it might gradually expand and store more and more data, including that it might be necessary to provide separate services for buyers and sellers. specialized program. Keep track of bills, payment records, credit cards, and more. And based on the third case, some basic storage records are needed. My current idea is to wait until there are certain users and enough information to be stored before turning on the recommended function. I hope there are some suggestions on these.

The structural database rules and ERD for my database design contain the important entities of Account, selling platform, type, name, price, evaluation, URL, Savetime and Promote as well as relationships between them.

I saved a lot of entities in the use case, which greatly increased my work volume. Even a small use case may use a lot of entities. It can be seen from this attempt: 'The number of entities is not determined by the size of the use case, but by the system, the people using the system, and the database.'

My database is taking shape with a DBMS Physical ERD. This means I can start building the database now.

Mapping the conceptual ERD to the physical ERD is just a mechanical process for me, hope it and specialization‐generalization rules and EERD will give me enough help when I build the database.

In the process, I optimized and adjusted it, I merged some instances, which can save running time and space consumption, and then I established Table and established various corresponding indexes, which is very happy , the database is finally really created.

Although there may be various issues, but I'm excited to develop this project to turn the 'BestBuy' application into a usable, convenient program. I will be happy to receive any suggestions to make it better.