Term Project

CS 669

SiCheng Yi

Why I want to do it.

- 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.
- '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.

Use Cases and Fields

- 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.

- Save Store Product Information Use Case

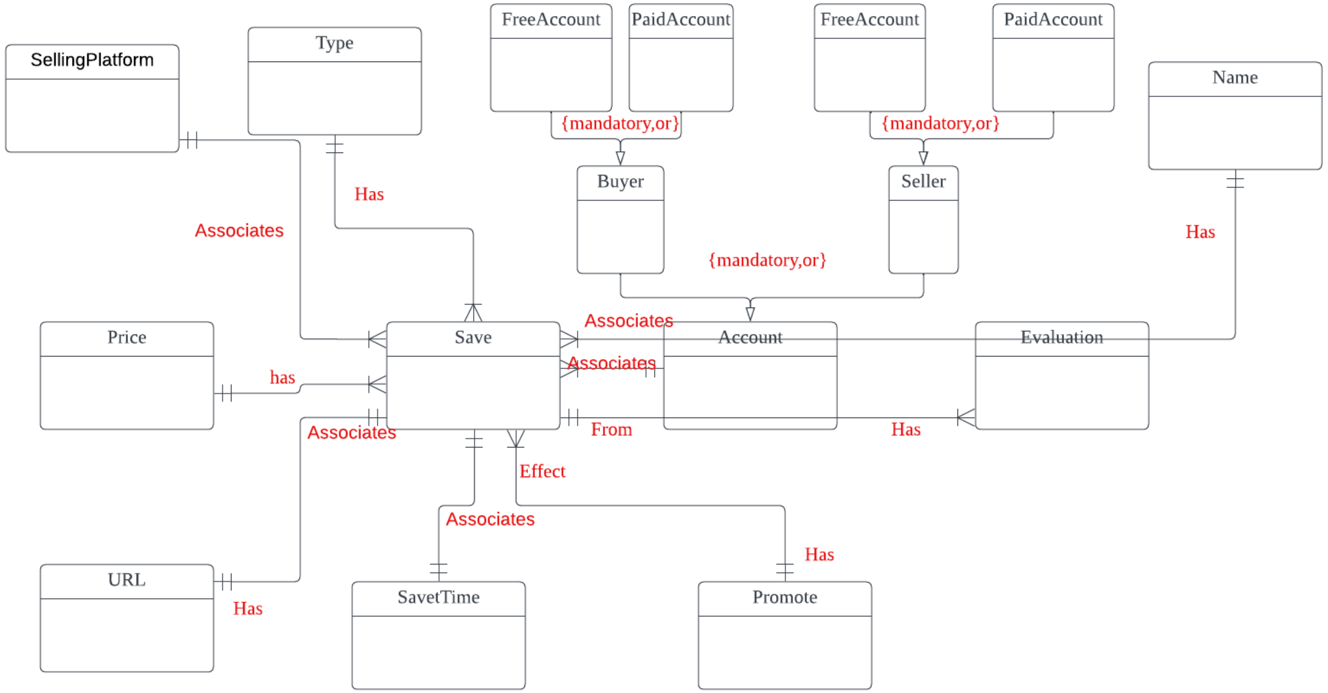
1. The user goes to the product detail page and chooses to save.
2. BestBuy receives the order, and stores the product's selling platform, itemtype, name, price, evaluation, and URL in the database.

Use Cases and Fields

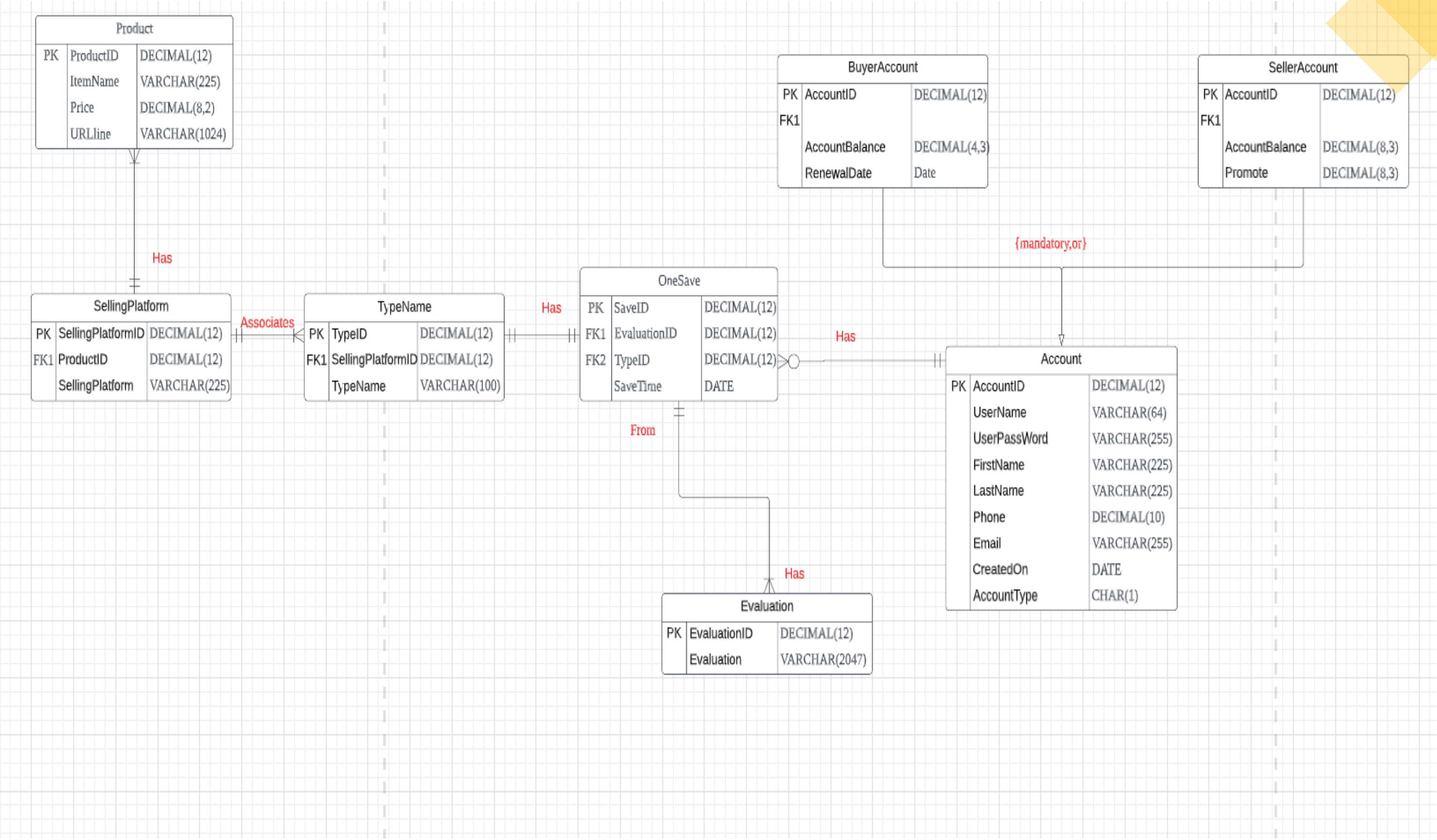
- Recommend Items Use Case

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.

Here is the ERD I do first time in the right:



- 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:



Product

PK	ProductID	DECIMAL(12)
	ItemName	VARCHAR(225)
	Price	DECIMAL(8,2)
	URLline	VARCHAR(1024)

SellingPlatform

PK	SellingPlatformID	DECIMAL(12)
FK1	ProductID	DECIMAL(12)
	SellingPlatform	VARCHAR(225)

TypeName

PK	TypeID	DECIMAL(12)
FK1	SellingPlatformID	DECIMAL(12)
	TypeName	VARCHAR(100)

OneSave

PK	SaveID	DECIMAL(12)
FK1	EvaluationID	DECIMAL(12)
FK2	TypeID	DECIMAL(12)
	SaveTime	DATE

BuyerAccount

PK	AccountID	DECIMAL(12)
FK1		
	AccountBalance	DECIMAL(4,3)
	RenewalDate	Date

SellerAccount

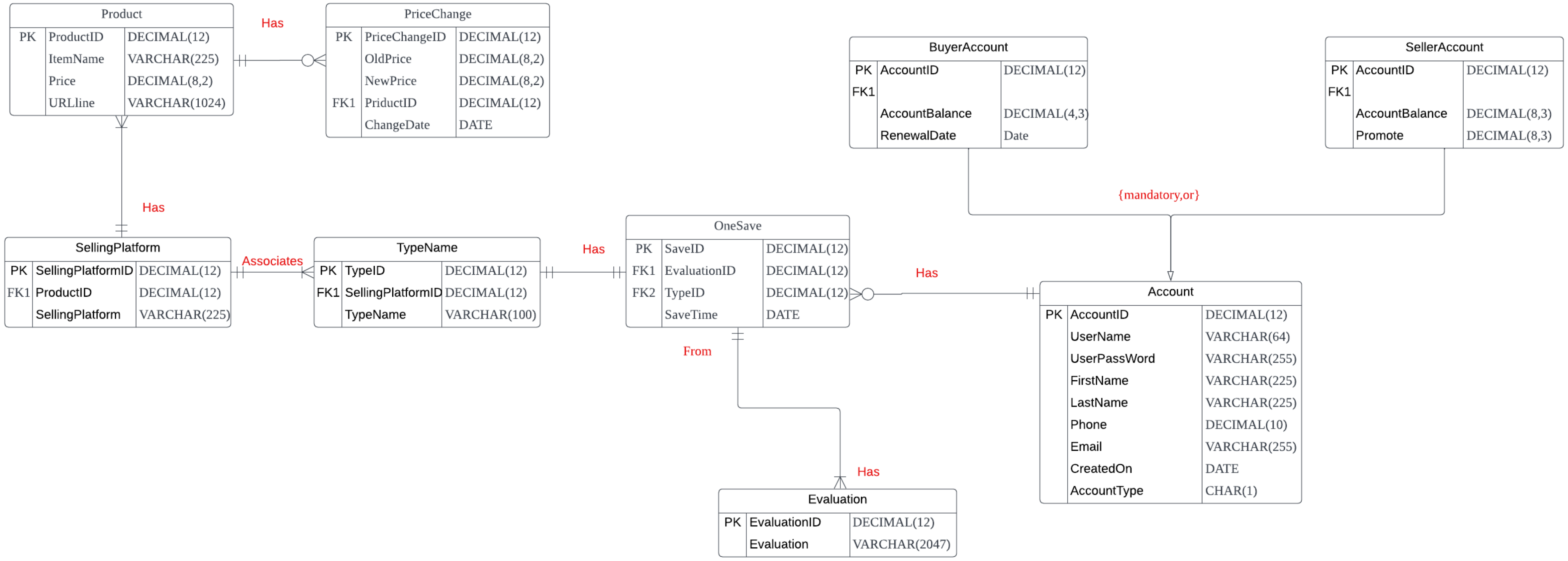
PK	AccountID	DECIMAL(12)
FK1		
	AccountBalance	DECIMAL(8,3)
	Promote	DECIMAL(8,3)

Account

PK	AccountID	DECIMAL(12)
	UserName	VARCHAR(64)
	UserPassWord	VARCHAR(255)
	FirstName	VARCHAR(225)
	LastName	VARCHAR(225)
	Phone	DECIMAL(10)
	Email	VARCHAR(255)
	CreatedOn	DATE
	AccountType	CHAR(1)

Evaluation

PK	EvaluationID	DECIMAL(12)
	Evaluation	VARCHAR(2047)



Product

PK	ProductID	DECIMAL(12)
	ItemName	VARCHAR(225)
	Price	DECIMAL(8,2)
	URLline	VARCHAR(1024)

PriceChange

PK	PriceChangeID	DECIMAL(12)
	OldPrice	DECIMAL(8,2)
	NewPrice	DECIMAL(8,2)
FK1	PriductID	DECIMAL(12)
	ChangeDate	DATE

BuyerAccount

PK	AccountID	DECIMAL(12)
FK1		
	AccountBalance	DECIMAL(4,3)
	RenewalDate	Date

SellerAccount

PK	AccountID	DECIMAL(12)
FK1		
	AccountBalance	DECIMAL(8,3)
	Promote	DECIMAL(8,3)

SellingPlatform

PK	SellingPlatformID	DECIMAL(12)
FK1	ProductID	DECIMAL(12)
	SellingPlatform	VARCHAR(225)

TypeName

PK	TypeID	DECIMAL(12)
FK1	SellingPlatformID	DECIMAL(12)
	TypeName	VARCHAR(100)

OneSave

PK	SaveID	DECIMAL(12)
FK1	EvaluationID	DECIMAL(12)
FK2	TypeID	DECIMAL(12)
	SaveTime	DATE

Account

PK	AccountID	DECIMAL(12)
	UserName	VARCHAR(64)
	UserPassWord	VARCHAR(255)
	FirstName	VARCHAR(225)
	LastName	VARCHAR(225)
	Phone	DECIMAL(10)
	Email	VARCHAR(255)
	CreatedOn	DATE
	AccountType	CHAR(1)

Evaluation

PK	EvaluationID	DECIMAL(12)
	Evaluation	VARCHAR(2047)

Add index and history

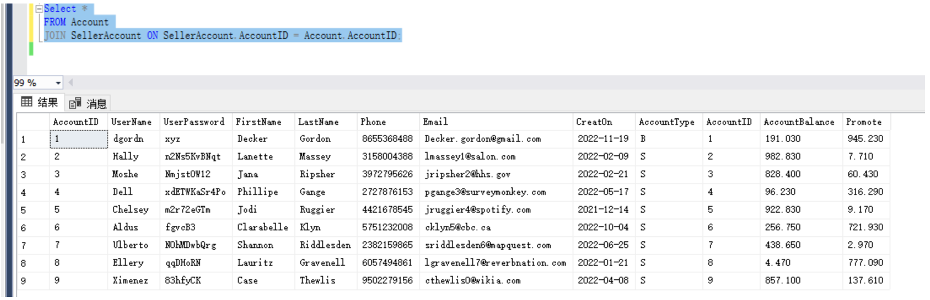
- I find that the price will change in many times for one product, like the activity price and normal price, so it can benefit from historical data updates.
- The new structural database rule is: Each product can have many price changes; each price change is for one Product.
-
- Here is the new update ERD:

Question and Query

- **List the number of users who have invested more than 100$ from account create after 2022 and display the corresponding number according to the three items of 100-300, 300-500, and 500$+.**

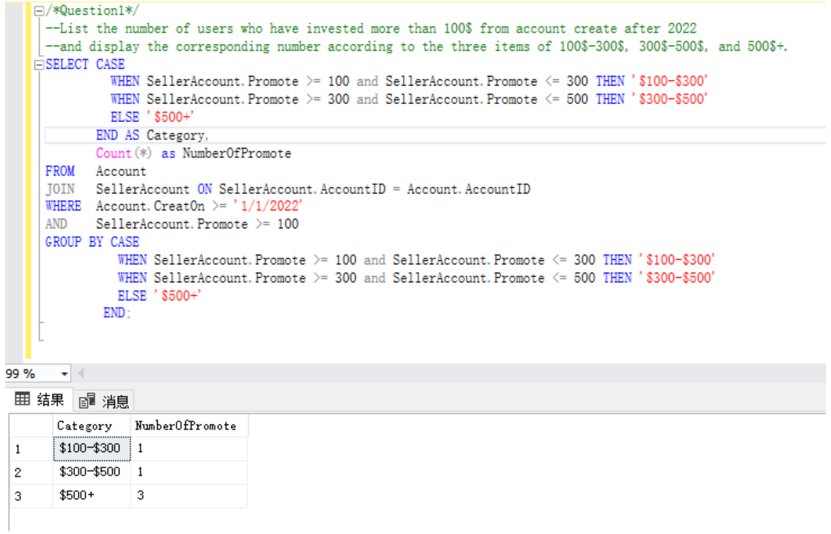
This question can help us confirm the investment of new users this year, and solve many problems, such as confirming the economic situation, the degree of attention of the app, and so on.

Here is a screenshot of the query I use:



```sql
Select *
FROM Account
JOIN SellerAccount ON SellerAccount.AccountID = Account.AccountID;
```

	AccountID	UserName	UserPassword	FirstName	LastName	Phone	Email	CreatOn	AccountType	AccountID	AccountBalance	Promote
1	1	dgordn	xyz	Decker	Gordon	8655368488	Decker.gordon@gmail.com	2022-11-19	B	1	191.030	945.230
2	2	Hally	n2Ns5KvBNqt	Lanette	Massey	3158004388	lmassey1@salon.com	2022-02-09	S	2	982.830	7.710
3	3	Moshe	NmjstOW12	Jana	Ripsher	3972795626	jripsher2@hhs.gov	2022-02-21	S	3	828.400	60.430
4	4	Dell	xdETWKaSr4Po	Phillipe	Gange	2727876153	pgange3@surveymonkey.com	2022-05-17	S	4	96.230	316.290
5	5	Chelsey	m2r72eGTm	Jodi	Ruggier	4421678545	jruggier4@spotify.com	2021-12-14	S	5	922.830	9.170
6	6	Aldus	fgvcB3	Clarabelle	Klyn	5751232008	cklyn5@cbc.ca	2022-10-04	S	6	256.750	721.930
7	7	Ulberto	NOhMDwbQrg	Shannon	Riddlesden	2382159865	sriddlesden6@mapquest.com	2022-06-25	S	7	438.650	2.970
8	8	Ellery	qqDHoRN	Lauritz	Gravenell	6057494861	lgravenell7@reverbnation.com	2022-01-21	S	8	4.470	777.090
9	9	Ximenez	83hfyCK	Case	Thewlis	9502279156	cthewlis0@wikia.com	2022-04-08	S	9	857.100	137.610



```sql
/*Question1*/
--List the number of users who have invested more than 100$ from account create after 2022
--and display the corresponding number according to the three items of 100$-300$, 300$-500$, and 500$+.
SELECT CASE
         WHEN SellerAccount.Promote >= 100 and SellerAccount.Promote <= 300 THEN '$100-$300'
         WHEN SellerAccount.Promote >= 300 and SellerAccount.Promote <= 500 THEN '$300-$500'
         ELSE '$500+'
       END AS Category,
       Count(*) as NumberOfPromote
FROM   Account
JOIN   SellerAccount ON SellerAccount.AccountID = Account.AccountID
WHERE  Account.CreatOn >= '1/1/2022'
AND    SellerAccount.Promote >= 100
GROUP BY CASE
          WHEN SellerAccount.Promote >= 100 and SellerAccount.Promote <= 300 THEN '$100-$300'
          WHEN SellerAccount.Promote >= 300 and SellerAccount.Promote <= 500 THEN '$300-$500'
          ELSE '$500+'
        END;
```

	Category	NumberOfPromote
1	$100-$300	1
2	$300-$500	1
3	$500+	3

Thank you!