

A dark blue vertical bar on the left side of the slide, with a blue arrow pointing right from its center.

4/23/2019

# Silver Server

A Silver Pricing Application

Several thin, curved lines in dark blue and light gray originating from the bottom left corner and extending upwards and to the right.

Shaun Marple  
CS669

## Table of Contents

1. Introduction.....	2
2. Use Cases.....	3
3. History Table.....	10
4. Structural Rules.....	12
5. ERDs.....	14
6. Questions & Queries.....	16
7. Summary & Reflection.....	18

## **Project Direction Overview**

Over the past couple years, there has been political wars, trade wars, and many other events that have affected how people invest money. The stock market no longer holds my full confidence as being my sole investment portfolio I have grown accustomed to. I have realized that I just don't have the time to analyze market trends, watch the news, and learn everything I need to know about building a successful portfolio in only the stock market, nor am I so naïve as to think that putting all my money into a single facet of investments for successful retirement planning. Because of these things, I have decided to expand my investment portfolio in the direction of buying precious metals.

For this project, I will be constructing SilverServer, a web database for silver, with purity of rating of .999, using US currency. Users will be able to track when they purchased a specific piece of silver, what they paid for it, what it was selling for at the time of purchase (by using recent selling prices through eBay sold data, which I will be entering manually (initially)), what the price of silver per ounce was at the time of the sale, as well as ratios between price of silver Vs. average selling price of the coin, or bullion, is worth at that time, and many other comparisons. As the database grows with new users, and historical pricing will be used between other users, rather than pulling eBay information on every piece of silver that comes out. This will also help in avoiding the double-counting of purchases at each purchase price.

This application will be designed to help users project how much their silver pieces are worth as the price of silver rises and falls. This should allow them to better price their items before selling them on eBay, pawn shops, or other places that buy and sell precious metals. This application will also help users track their purchases to help protect them from over paying or under selling their assets. Additional attributes, such as average selling price – to – silver per ounce – ratio for the month of June and minimum and maximum prices expected for the month of January would become available for its users. The expectation of these prices would use the average selling prices in comparison with the average monthly price of silver per ounce ratio to approximate summary statistics to help users identify their selling profit expectations.

This application will have attributes like silver company name, silver bullion series collection it belongs to (if applicable), weight (in ounces), grade (if applicable), purchase price, coin name, coin sub-name (if applicable), Location of where it was made, purchase date, and many other fields that help in identification of which coin it is and which category it belongs to. The major coin sellers and coin names will be researched ahead of time and added to combo boxes that should allow users to choose from a list, rather than to make up several different names, for the same piece, and over-complicate queries in identifying them all. eBay, for example, allows a seller to enter in whatever title they want, making it difficult for eBay to capture all instances of each coin.

## Use Cases and Fields

One 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 the SilverServer 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.

```
199  -- Add Accounts
200  CREATE OR REPLACE FUNCTION ADD_ACCT(
201      username_arg IN VARCHAR,
202      first_name_arg IN VARCHAR,
203      last_name_arg IN VARCHAR,
204      email_arg IN VARCHAR)
205      RETURNS VOID LANGUAGE plpgsql
206  AS $$
207  BEGIN
208      INSERT INTO Account (user_name, first_name, last_name, email)
209      VALUES (username_arg, first_name_arg, last_name_arg, email_arg);
210  END;
211  $$;
212  |
213  DO
214      $$
215  BEGIN
216      EXECUTE ADD_ACCT('shaunSilver2019','Shaun','Marple','stmarple@hotmail.com');
217      EXECUTE ADD_ACCT('JSmith45','John','Smith','ILuvPocahautus@yahoo.com');
218      EXECUTE ADD_ACCT('BlackPearlNGold','Jack','Sparrow', NULL);
219      EXECUTE ADD_ACCT('BahHumbbugDec25', 'Ebenezer', 'Scrooge', NULL);
220  END;
221  $$;
```

This stored procedure can also create one new account at a time. I am just generating multiple so I will be able query it later. The user will enter a username, first and last name as well as email in case they want a report sent to them or other communications.

Shaun Marple  
**SILVER SERVER**



A coin depicting Leonidas, leader of 300 Spartans in the battle of Thermopylae.

**Tracking Major Silver Bullion Types Use Case**

1. The user will be able to isolate their specific coins by answering a few required questions about unique types of each bullion.
2. The user will enter in which shape, whether it is round, rectangular, statue, poured, route 66, ...etc.
3. The user will enter which relief the coin is in: normal, high, or ultra-high

**Ultra High Relief (Notice the 3D look)**

**High Relief (Notice: not quite as high)**

**Normal (Not raised much at all)**



4. The user will choose a style: colorized, gold gilded, antique, shiny, ...etc.

**Gold Gilded**

**Colorized**

**Statue**



5. The user will choose whether or not the bullion belongs in a series. If it does, which series
6. The user will choose the mintage number, provided that it is minted. If not, this will be left blank
7. The user will choose location. Here are a few common ones: Serbia, Niue, Cook Islands, Solomon Islands.
8. The user will choose weight. A few of the common weights are: 1, 2, 3, 5, 10, and 100 oz.
9. The user will choose a grade, if applicable, NULL if not. The best grades are PF70 and PCGS70.
10. The user will choose manufacturing year
11. The user will choose denomination if the bullion is considered to be a coin, or currency.

In this step, many tables were created in 3NF, but they all come together in the product table, where users answer as many questions as possible to get the most accurate representation of what they hold. The following is a stored

## Shaun Marple SILVER SERVER

procedure that puts it all together into a product table:

```
CREATE OR REPLACE FUNCTION ADD_PRODUCT(  
    categ_arg VARCHAR,  
    loc_arg IN VARCHAR,  
    shape_arg IN VARCHAR,  
    relief_arg IN VARCHAR,  
    style_arg IN VARCHAR,  
  
    coin_arg IN VARCHAR,  
    coinSub_arg IN VARCHAR,  
    desc_arg IN VARCHAR,  
    yr_made_arg IN INT,  
    mintage IN INT,  
    currency_arg IN DECIMAL,  
    weight_arg IN DECIMAL,  
    grade_arg IN VARCHAR,  
    series_arg IN VARCHAR  
) RETURNS VOID  
AS $$  
DECLARE  
    var_categ_id INT;  
    var_loc_id INT;  
    var_shape_id INT;  
    var_relief_id INT;  
    var_style_id INT;  
    var_new_id INT;  
BEGIN  
    SELECT category_id INTO var_categ_id  
    FROM Category  
    WHERE category = categ_arg;  
  
    SELECT mfg_location_id INTO var_loc_id  
    FROM MFG_Location  
    WHERE mfg_location = loc_arg;  
  
    SELECT shape_id INTO var_shape_id  
    FROM SHAPE  
    WHERE shape = shape_arg;  
  
    SELECT relief_id INTO var_relief_id  
    FROM RELIEF  
    WHERE relief = relief_arg;  
  
    SELECT style_id INTO var_style_id  
    FROM STYLE  
    WHERE style = style_arg;  
  
    SELECT nextval(pg_get_serial_sequence('product', 'product_id')) INTO var_new_id;  
  
    INSERT INTO Product(product_id, category_id, mfg_location_id, shape_id, relief_id, style_id,  
        coin_name, coin_subname, coin_description, year_made, mintage_number, denomination, weight, grade)  
  
    VALUES (var_new_id, var_categ_id, var_loc_id, var_shape_id, var_relief_id, var_style_id,  
        coin_arg, coinSub_arg, desc_arg, yr_made_arg, mintage, currency_arg, weight_arg, grade_arg);  
  
    IF series_arg IS NOT NULL THEN  
        INSERT INTO Series (product_id, series_name)  
        VALUES (var_new_id, series_arg);  
    ELSE  
        INSERT INTO NOT_SERIES (product_id)  
        VALUES (var_new_id);  
    END IF;  
END;  
$$ LANGUAGE plpgsql;
```

## Shaun Marple SILVER SERVER

```
498 DO
499 $$
500 BEGIN
501         /* categ, loc, shape, relief, style, coin,
502            coinsub, desc, yr, mintage,denom, weight, grade, series_arg*/
503 EXECUTE ADD_PRODUCT('Star Wars','Niue','Round','Ultra High','Shiny','Millenium Falcon',
504                     NULL, NULL, 2019, 5000, 50, 1, NULL, NULL);
505
506 EXECUTE ADD_PRODUCT('Greek', 'Anonomous', 'Round', 'High', 'Shiny', 'Spartans',
507                     'Leader of Men (type 2)', NULL, 2019, NULL, 0, 1, NULL, 'Molon Labe');
508
509 EXECUTE ADD_PRODUCT('Greek', 'Anonomous', 'Round', 'High', 'Gold Gilded', 'Spartans',
510                     'With Spear', NULL, 2019, NULL, 0, 1, NULL, 'Molon Labe');
511
512 EXECUTE ADD_PRODUCT('Historical', 'Serbia', 'Round', 'Normal', 'Shiny', 'Nikola Tesla',
513                     'Alternating Current', 'Throughout the world, Nikola Tesla is recognized as one of history's
514                     greatest minds. He has over 280 patents to his name, and developed a way to harness the power
515                     that still runs the world today. In 2018, the Serbian Mint began a bullion series to honor their
516                     national hero. Like the bullion coin, the Proofs are struck from one ounce of .999 fine silver.
517                     This piece has a total mintage of 3,327 pieces, a reference to room 3327 at the New Yorker Hotel
518                     in which Tesla kept his scientific papers locked. This piece remains in GEM Proof
519                     condition.', 2018, 3327, 100, 1, NULL, NULL);
520
521 EXECUTE ADD_PRODUCT('Mythology', 'Tanzania', 'Round', 'Normal', 'Shiny', 'Griffin',
522                     NULL, 'The griffin, griffon, or gryphon is a legendary creature with the body, tail,
523                     and back legs of a lion, the head and wings of an eagle, and an eagle's talons as its
524                     front feet. Because the lion was traditionally considered the king of the beasts and the
525                     eagle the king of birds, the griffin was thought to be an especially powerful and majestic
526                     creature. The griffin was also thought of as king of all creatures. Griffins are known
527                     for guarding treasure and priceless possessions.', 2018, 499, 1500, 1, NULL, 'Mythology Series');
528
529 EXECUTE ADD_PRODUCT('Roman', 'Tuvalu', 'Round', 'High', 'Antique', 'Roman Legion',
530                     'Tuvalu Warfare', NULL, 2018, 2000, 0, 1, 'PCGS69', 'Warfare Series');
531
532 EXECUTE ADD_PRODUCT('Greek', 'Cook Islands', 'Round', 'Ultra High', 'Antique', 'Shield of Athena',
533                     'Aegis', 'Shield of Athena - aegis The aegis, as stated in the Iliad,
534                     is carried by Athena and Zeus, but its nature is uncertain. It had been
535                     interpreted as an animal skin or a shield, sometimes bearing the head
536                     of a Gorgon. There may be a connection with a deity named Aex or Aix,
537                     a daughter of Helios and a nurse of Zeus or alternatively a mistress of
538                     Zeus. The aegis of Athena is referred to in several places in the Iliad.
539                     "It produced a sound as from a myriad roaring dragons (Iliad, 4.17) and
540                     was borne by Athena in battle ... and among them went bright-eyed Athene,
541                     holding the precious aegis which is ageless and immortal: a hundred tassels
542                     of pure gold hang fluttering from it, tight-woven each of them, and each
543                     the worth of a hundred oxen".', 2018, 999, 0, 1, 'PCGS70', 'Mythology Series');
544
545 EXECUTE ADD_PRODUCT('Greek', 'Niue', 'Round', 'Ultra High', 'Hematite', 'Achilles',
546                     'Achilles Vs Hector', NULL, 2017, 650, 0, 1, NULL, 'Demigods Series');
547
548 END;
549 $$;
```

### Purchase Tracking Use Case

1. The person signs into SilverServer.
2. The person enters information, or selects from drop-down menus, in their recent purchase(s) or sale(s).
3. Once all information has been submitted, the user can view reports on all of their entered data.
4. The user will get ratio information between prices of their coins against the average ask price of silver for that month.
5. SilverServer displays all recorded information about the purchase (Which can help in doing their taxes; since dates and prices will be recorded and kept. eBay only tracks your purchases for up to a year or so).

6. The user can also view all data, pulled from other users in the database as well as initial data pulled from eBay and other Websites, such as APMEX  
or JM Bullion, and pull summary statistics on average current selling prices.
7. The user will also be able query information on which months trend high and low in pricing.

For these steps to work, the user will have to:

1. Enter whether the transaction is a Buy or a Sell (This database currently only deals with purchases)
2. Enter a purchase date
3. Choose a product through a series of drop-down menus (This database currently asks for product\_id for simplicity)
4. Enter a price
5. Enter a quantity
6. Enter whether or not the item is minted. If is minted, the quantity will be set to 1, since it is one of a kind. This minted argument will be used to ask for the number of the coin, usually inscribed in the side of the coin, or in a certificate of authenticity. (However, this thought came in retrospect, after the project was thought to be completed.) If the mint argument is left NULL, then the quantity, entered by the user, will go into effect.



Shaun Marple  
**SILVER SERVER**

These steps will likely go in succession, with steps 5 and 6 reversed, in a web design application to reduce redundant questions.

```
-- ADD Purchases,
CREATE OR REPLACE FUNCTION ADD_PURCHASE(
  --price_arg IN VARCHAR,
  product_id_arg IN INTEGER,
  acct_arg IN VARCHAR,
  trans_arg IN VARCHAR,

  date_arg IN DATE,
  price_arg IN DECIMAL,
  qty_arg IN INT,
  mint_arg IN VARCHAR
) RETURNS VOID

AS $$
DECLARE
  var_price_id INT;
  var_acct_id INT;
  var_trans_id INT;
  var_new_id INT;
  var_inv_id INT;

BEGIN
  SELECT Raw_Pricing.monthly_avg_id INTO var_price_id
  FROM Raw_Pricing
  WHERE to_char(date_arg, 'YYYYMM') = to_char(Raw_Pricing.purchase_date, 'YYYYMM');

  SELECT acct_id INTO var_acct_id
  FROM Account
  WHERE user_name = acct_arg;

  SELECT transaction_id INTO var_trans_id
  FROM Transaction
  WHERE transaction = trans_arg;

  SELECT nextval(pg_get_serial_sequence('purchase', 'purchase_id'))
  INTO var_new_id;

  INSERT INTO Purchase(purchase_id, monthly_avg_id, acct_id, transaction_id,
    product_id,
    purchase_date, price)
  VALUES (var_new_id, var_price_id, var_acct_id, var_trans_id,
    product_id_arg,
    date_arg, price_arg);

  IF mint_arg IS NOT NULL THEN
    INSERT INTO Minted (purchase_id, quantity)
    VALUES (var_new_id, 1);
  ELSE
    INSERT INTO NOT_minted (purchase_id, quantity)
    VALUES (var_new_id, qty_arg);
  END IF;

  SELECT ADD_INVENTORY(var_acct_id, product_id_arg, qty_arg) INTO var_inv_id;

  INSERT INTO Pur_Inv_Lk(purchase_id, inventory_id) VALUES (
    var_new_id, var_inv_id);

END;
$$ LANGUAGE plpgsql;
```

Shaun Marple  
**SILVER SERVER**

```
DO
$$
BEGIN
    EXECUTE ADD_PURCHASE(
        1, --product id
        'shaunSilver2019', -- username
        'Buy', --trans_arg
        to_date('10/12/2017', 'MM/DD/YYYY'), --date_arg
        40, --price_arg
        1, -- qty_arg
        NULL
    );
    EXECUTE ADD_PURCHASE(
        1, --product id
        'JSmith45', -- username
        'Buy', --trans_arg
        to_date('10/12/2017', 'MM/DD/YYYY'), --date_arg
        40, --price_arg
        1, -- qty_arg
        NULL
    );
    EXECUTE ADD_PURCHASE(
        2, --product id
        'shaunSilver2019', -- username
        'Buy', --trans_arg
        to_date('10/25/2017', 'MM/DD/YYYY'), --date_arg
        37, --price_arg
        4, -- qty_arg
        NULL
    );
END;
$;
```

The above code shows how purchases can be made. The user selects a product id. The username will be automatic, base on whomever is currently signed in. The user will then choose buy/sell, followed by the date, the price, the quantity, and finally, the mintage. If the mintage is NULL, it is expected that it doesn't carry a mintage.

## SilverServer History

Although it would make perfect sense to provide historical reference to just about all the tables provided, I thought I'd add a historical table that many companies always seem to leave out: The last name change. It seems near impossible to change your name in the system of so many websites and stores once you are in the system. One may hypothesize that this could be why men and women who get married, choose not to keep their spouse's last name. It is just too painful to go through the name changing process, if not, impossible in some places.

So:

### The last name change historical table with trigger

Attribute	Description
history_id	SERIAL Primary key (auto-increment, synthetic)
acct_id	the user's account information id
old_last_name	last name before the change
new_last_name	new last name
date_of_change	date of change DATE


```
CREATE OR REPLACE FUNCTION Name_Change_History_func()
RETURNS TRIGGER LANGUAGE plpgsql
AS $$
BEGIN
    IF OLD.last_name <> NEW.last_name THEN
        INSERT INTO Name_Change_History (acct_id, old_last_name, new_last_name, date_of_change)
        VALUES(NEW.acct_id, OLD.last_name, NEW.last_name, CURRENT_DATE);
    END IF;
    RETURN NEW;
END;
$$;

CREATE TRIGGER Name_Change_History_trg
BEFORE UPDATE ON Account
FOR EACH ROW
EXECUTE PROCEDURE Name_Change_History_func();
```

## Testing the Trigger


Before:

```
754 --Test trigger
755 -- Test
756 SELECT * FROM ACCOUNT;
```

Data Output	Explain	Messages	Notifications	
 acct_id integer	user_name character varying (16)	first_name character varying (32)	last_name character varying (32)	email character varying (64)
1	1 shaunSilver2019	Shaun	Marple	stmarple@hotmail.com
2	2 JSmith45	John	Smith	ILuvPocahautus@yahoo.c...
3	3 BlackPearlNGold	Jack	Sparrow	[null]
4	4 BahHumbugDec25	Ebenezer	Scrooge	[null]

After:

```
758 UPDATE ACCOUNT
759 SET last_name = 'Black'
760 WHERE first_name = 'Jack';
761
762 SELECT * FROM ACCOUNT;
```

Data Output	Explain	Messages	Notifications	
 acct_id integer	user_name character varying (16)	first_name character varying (32)	last_name character varying (32)	email character varying (64)
1	1 shaunSilver2019	Shaun	Marple	stmarple@hotmail.com
2	2 JSmith45	John	Smith	ILuvPocahautus@yahoo.c...
3	4 BahHumbugDec25	Ebenezer	Scrooge	[null]
4	3 BlackPearlNGold	Jack	Black	[null]

Historical changes table (after a few changes):

```
758 UPDATE ACCOUNT
759 SET last_name = 'Sparrow'
760 WHERE first_name = 'Jack';
761
762 SELECT * FROM Name_Change_History;
```

Data Output

Explain

Messages

Notifications

	history_id integer	acct_id integer	old_last_name character varying (32)	new_last_name character varying (32)	date_of_change date
1	1	3	Black	Keroac	2019-04-23
2	2	3	Keroac	Black	2019-04-23
3	3	3	Black	Sparrow	2019-04-23

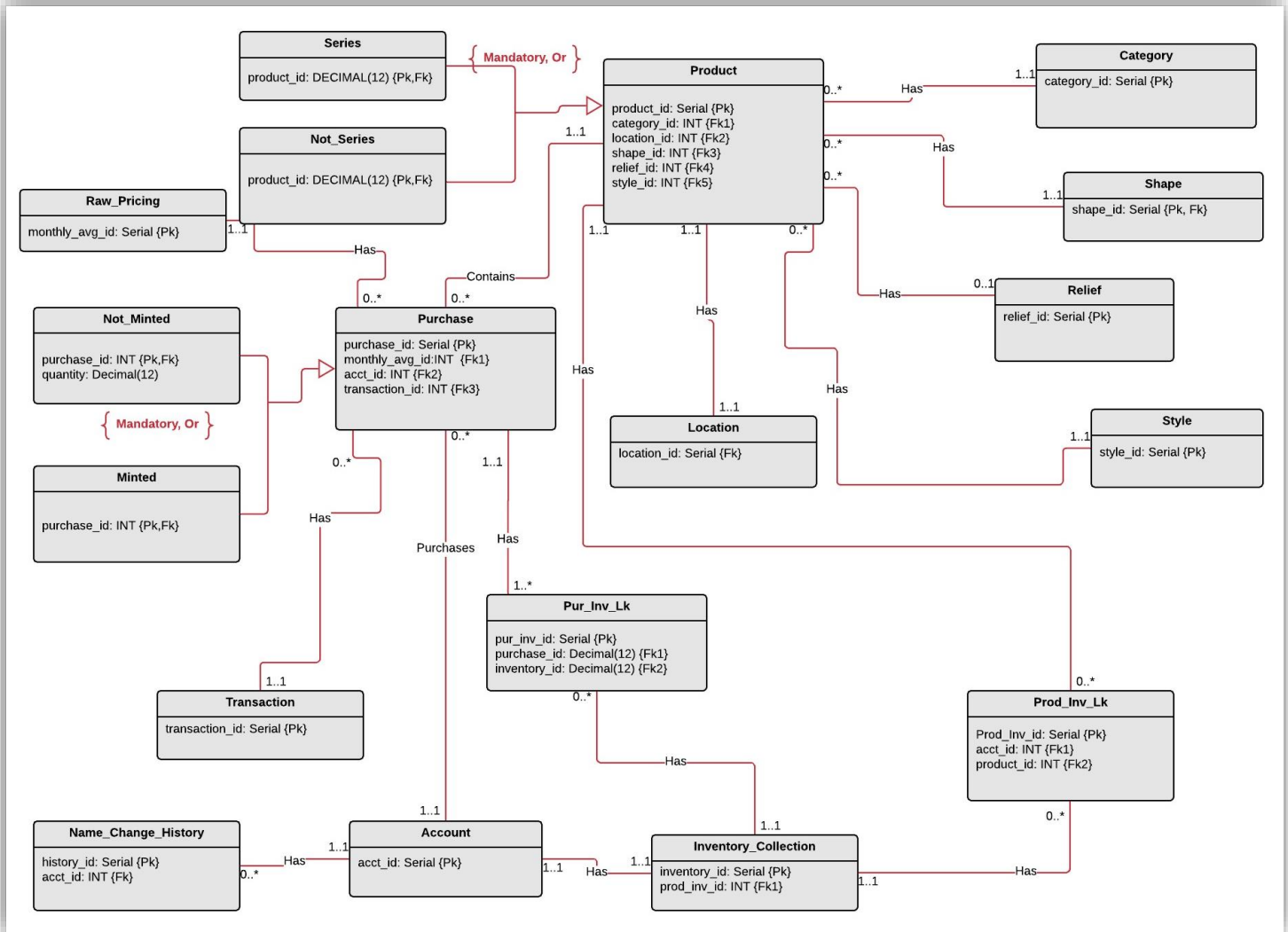
## Structural Rules

Business Rule	Participation	Plurality
Each product may be contained in multiple users' inventory_collections	Optional	Plural
Each user inventory_collection may contain multiple products	Optional	Plural
Each user may have many purchases of a "non-minted" products or only a single purchase of a "minted" product	Optional	Plural
Each purchase may be associated with a single user	Optional	Singular
Each product is created at a company	Mandatory	Singular
Each company creates many products	Mandatory	Plural
Each user has an account	Mandatory	Singular
Each account is held by a single user	Mandatory	Singular
Each user inventory_collection may have many purchases	Optional	Plural
Each purchase must be made by a user	Mandatory	Singular
Each silver product has to be purchased from a silver company	Mandatory	Singular
Each silver company creates one or more silver products	Mandatory	Plural
Each product is associated with only one category	Mandatory	Singular
Each category may be associated with many products	Optional	Plural
Each account has one inventory_collection	Mandatory	Singular
Inventory_collection must be held by an account (user)	Mandatory	Singular
Inventory_collection may have many products in it	Optional	Plural
Each Purchase may be in each inventory_collection	Optional	Singular
Each purchase has a daily raw silver price associated with it	Mandatory	Singular
Each raw silver price may be associated with many purchases	Optional	Plural
Each purchase contains one or more products	Mandatory	Plural

Shaun Marple  
**SILVER SERVER**

<b>Each product may be associated by 0 or more purchases</b>	Optional	Plural
Each product may or may not be part of a series	Optional	Singular
Each series contains 1 or more coins	mandatory	plural
Each purchased product may or may not be minted	Optional	Singular
Each purchase is associated with a transcation [type]. Each purchase may be a sale or a purchase	Optional	Singular
Each category can have 0 or more sub-categories	Optional	Plural
Each sub-category can belong to 1 or more categories	Mandatory	Plural
Each product is associated with a shape	Mandatory	Singular
Each product is associated with a relief	Mandatory	Singular
Each shape can be associated with many many products	Optional	Plural
Each relief may be associated with many products	Optional	Plural

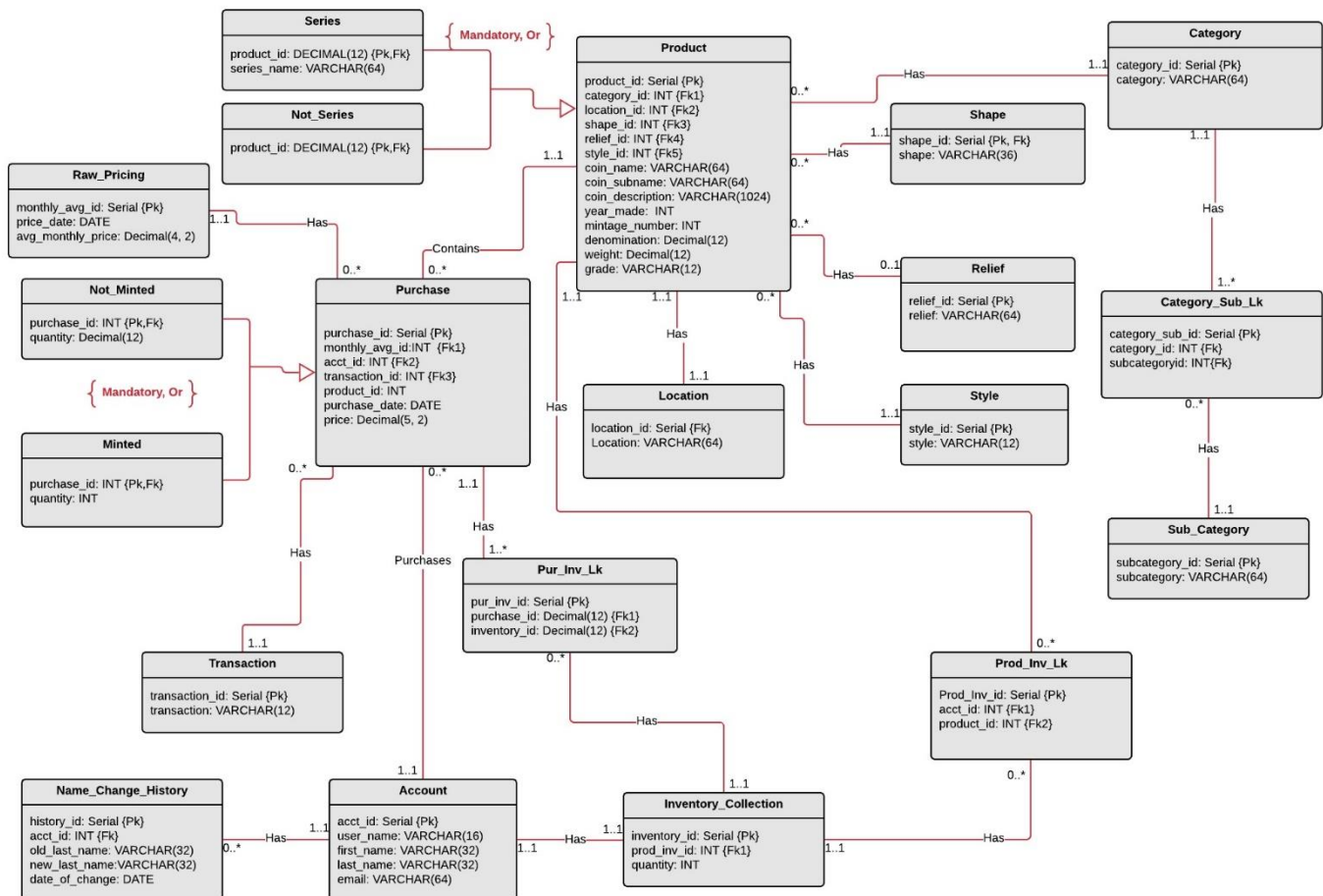
### Conceptual ERD (With keys)



This ERD has changed a bit since the last iteration. For one, the shape subtypes were removed since these were redundant. Originally thought to be able to distinguish a round bullion versus a round coin. However, this would lead to redundancies in the supertypes, for all the possible shapes that other countries may use for currency. Either that, or I would need to add a child of a subtype to break out the possible shapes that are both bullion and currency. I.e. Coins are round and many bullions are round too.

**Note:** Bullions are basically silver coins without denominations inscribed. They can be highly detailed and very artistic in nature. Coins generally have some form of denomination on them and often one of the sides will include a head one of their leaders, like a queen or president.

### Physical ERD (Normalized)



Other changes that have been made from previous iterations is the locations of a few different attributes like grade and weight moved to product since they deal more with the product than they do to the purchase. All of the Boolean attributes that tied in the supertypes with their corresponding subtypes, were removed as well. Rather than keeping additional attributes of character inputs that seem to be of little use, I decided that it would be better to include these inputs as part of the stored procedures when adding them to the purchase and product tables. If and when I decide to add meaning to these subtypes and bring them in as additional attributes, it'll be a simple change to the tables and stored procedure. For example, rather than asking if the product has mintage, I can ask for the coins minted number. If NULL, it is expected to not have a mintage associated with it.



## Shaun Marple SILVER SERVER

### SilverServer Questions and Queries

1. What are all the series names for all of your products and how many products do you have for each series?

```
726 SELECT year_made, series_name, COUNT(product.coin_name) AS Product_Count
727 FROM Product
728 JOIN Series ON product.product_id = Series.product_id
729 JOIN Not_Series ON product.product_id = Series.product_id
730 GROUP BY year_made, Series_name
731 ORDER BY Product_Count DESC;
732
```

	year_made numeric (4)	series_name character varying (64)	product_count bigint	
1	2018	Mythology Series	4	
2	2019	Molon Labe	4	
3	2017	Demigods Series	2	
4	2018	Warfare Series	2	

It looks like there are 4 products in from the 2018 mythology series, 4 from the 2019 Molon Labe series, 2 from the 2017 Demigods Series, and 2 from the warfare series.

2. Name all the products whose prices are greater than the average.

```
751 --Name all the products whose prices are greater than the average.
752 SELECT Category.category, Series.series_name, Product.coin_name, Cast(Purchase.price as money) AS Cash
753 FROM Category
754 JOIN Product ON Category.category_id = Product.category_id
755 JOIN Purchase ON Product.product_id = Purchase.product_id
756 JOIN Series ON Series.product_id = Product.product_id
757 WHERE (
758     (SELECT AVG(Purchase.price)
759     FROM Purchase) > Purchase.price);
760
```

	category character varying (64)	series_name character varying (64)	coin_name character varying (64)	cash money	
1	Greek	Molon Labe	Spartans	\$37.00	
2	Greek	Molon Labe	Spartans	\$37.00	
3	Mythology	Mythology Series	Griffin	\$37.00	

Shaun Marple  
SILVER SERVER

What are the purchase price-to-raw silver price ratios of the cost of 1 oz?

```
762 SELECT DISTINCT Category.category, Product.coin_name, Cast(Purchase.price as money) as Cash_Money,
763                    TRUNC(Purchase.price / Raw_Pricing.average_monthly_price, 3) AS Purchase_Ratio
764 FROM Category
765 RIGHT JOIN Product ON Category.category_id = Product.category_id
766 RIGHT JOIN Purchase ON Product.product_id = Purchase.product_id
767 RIGHT JOIN Raw_Pricing ON Purchase.monthly_avg_id = Raw_Pricing.monthly_avg_id
768 WHERE (to_char(Purchase.purchase_date, 'YYYYMM') = to_char(Raw_Pricing.price_date, 'YYYYMM'))
769        AND (Product.weight = 1))
770 ORDER BY purchase_ratio DESC;
```

Data Output		Explain	Messages	Notifications	
	category character varying (64)	coin_name character varying (64)	cash_money money	purchase_ratio numeric	
1	Mythology	Griffin	\$90.00	5.528	
2	Star Wars	Millenium Falcon	\$40.00	2.340	
3	Greek	Spartans	\$37.00	2.193	
4	Mythology	Griffin	\$37.00	2.193	
5	Greek	Spartans	\$37.00	2.165	

### Summary and reflection

This database will be a web application called SilverServer, which tracks silver purchases, then compares price information against average silver ask price per ounce for the time of purchase, then compares that ratio against other coins that are from the same mintage, year, country, relief type, grade and/or company. SilverServer looks to provide interfacing for entering purchases as well as other additional information about the coins in order to help users sell the coins in the future at the highest profit possible with up-to-date information and basic analytics. SilverServer will become the silver-surfing web application for the average person looking to invest in precious metals, without the need of hiring a broker, and getting the most from their silver sales.

The structural database rules and conceptual ERD for the database design contain such entities as Raw\_Pricing, Purchase, Account, Inventory\_Collection, Product, Category, and Company. The design contains hierarchies of: Account, premium versus free, and Purchase, minted versus not minted. The DBMS physical ERD contains the required entities and relationships of the database design along with synthetic primary keys used in best practice. I expect I'll be using auto-incrementing number for these primary keys.

The script creates all the tables, inserts practice data, and follows the requirements as outlined by the DBMS physical ERD. Indexes were added for extra efficiency in running queries and normalization was held in a strict manner to reduce the databases redundant data.

A historical data table for last name changes has also been incorporated to reflect competencies in both stored procedures and triggers. Furthermore, although not part of the database implementation process, I used a python script to extract the raw pricing averages for my database table (Raw\_pricing), from a non-user-friendly Excel workbook.

In conclusion of this term project, I'd like to reflect on the overall adventure it has been. Though painfully stressed in completing this seemingly never-ending project, in finding the time to squeeze in, or sleep time I can skip out on, I wanted to finally get my vision out into a design I can be proud of. This database was a long time coming and I am glad I finally have something that not only meets the requirement of the class, but also gives me a working database that I can use in my silver bullion collecting hobby. I have been looking for an application to allow me to track my purchases, but have been unsuccessful in the attempts and Excel isn't my favorite tool for something this is ever changing as well. Upon completion of the class, I intend on populating all my purchases within the database, and likely will continue to develop it as I continue in the CIS program.

As for the skills I have learned in this rigorous 5 week boot camp session: I've enhanced my weak SQL, I've increased the depth of database implementation beyond Microsoft Access, looking onward to web applications that are full-SQL based, I've learned how to build a well designed database, and most importantly, how crucial it is to have a well developed design when implementing the code. I'm sure the code would have been a nightmare had I have had a poor design! I coded just about 20 tables in less than 2 days; and what is more interesting: it works! 😊

Soon enough, I'll be delivering the successful web application of Silver Server to lead others to Silver Surf their way through tons of silver purchase and sales data that will not only help in maximizing profits, but will also be able to further educate possible new buyers. A few queries and a ton of data can aid in the education of what to look for and what to run away from (Never buy any silver from China. It's known for being the land of the fake reviews and selling silver "plated" metal).

This database design and implementation process has sure been a ride, but it is nice to know that I made it. I also understand that I still have much more to learn. However, I now feel like I have a solid understanding of how the process works as well as a solid foundation to continue to build on.