PCI-Xpress

by

Wayne Pham & Samip Vaidh

Contributions

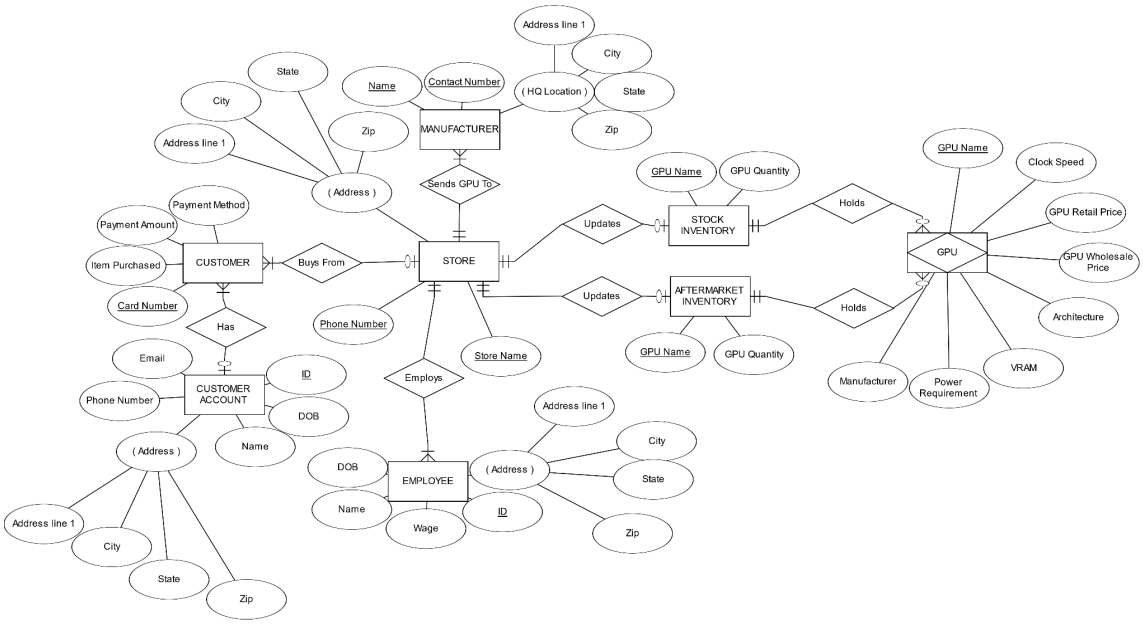
Wayne - Assisted with initial concept creation of the database including the ERD and conversion to relational schema. Converted Relational Schema into SQL code and then modified the SQL code to compile in Azure Data Studio. Wrote initial data entries and queries for the database. Setup source control to keep track of changes to the project. Also implemented the Python + SQL connection.

Samip - Assisted with initial concept creation of the database including the ERD and conversion to relational schema. Heavily researched and tried implementations of hosting the database without ADS. Help decide on hosting locally through the use of sqlite3. Setup virtual environment for writing code in Visual Studio Code. Assisted in the development of the database as well as queries. Code base was shared through Github to keep the project up to date.

Enterprise Statement

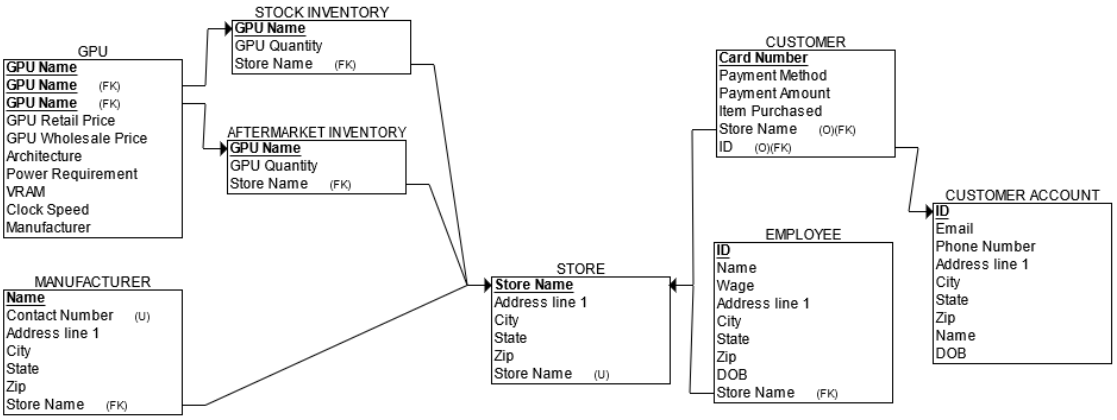
PCI-Xpress is a third party store that resells graphics cards that have connections to the GPU manufacturers. The store hires employees to manage the store. A Customer's transaction is recorded in the store. Each customer can have an account associated with the store. The store has two inventories to separate a stock product and an aftermarket product. And the inventory is based on the available graphics cards the store offers.

ERD



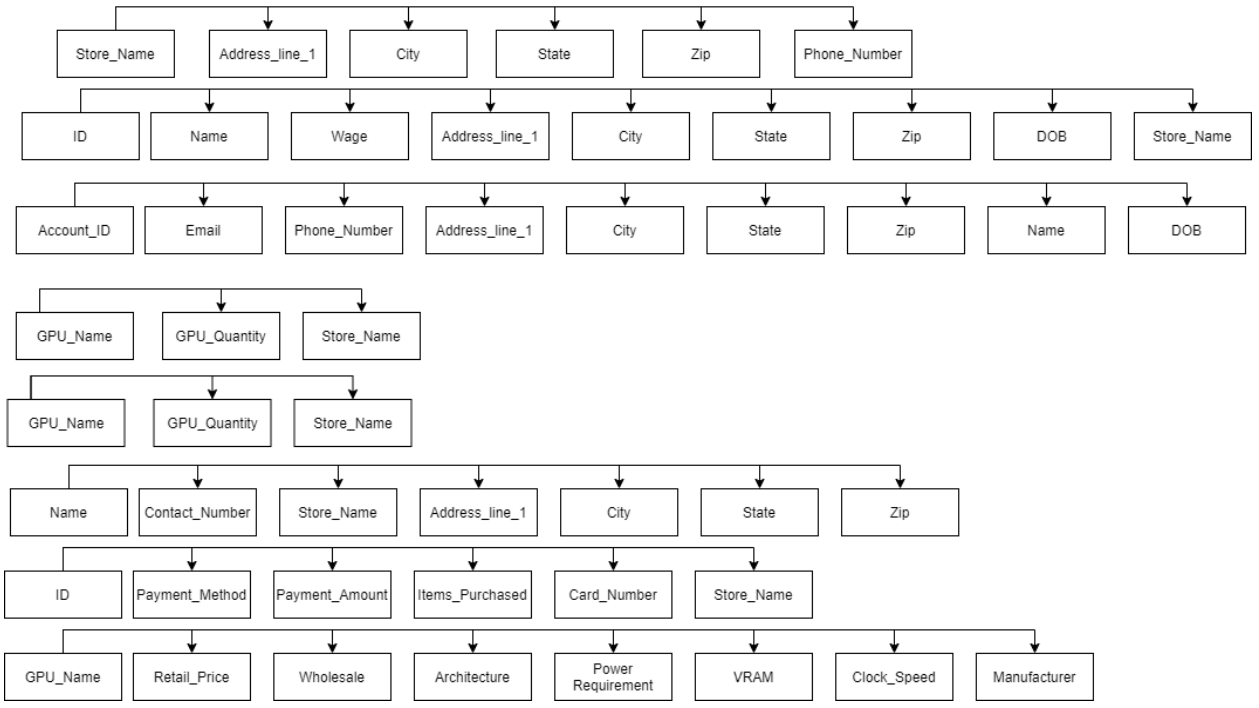
Relational Schema

Condensed version of the ERD that shows the strict relationships between each table which helped us then convert this to SQL code.



Dependency Diagram

From top to bottom: Store, Employee, Customer_Account, Aftermarket Inventory, Stock_Inventory, Manufacturer, Customer, GPU.



Normalization

The tables that our group created were already in 2nd normalized form when initially creating the database. The values in each table only contain a single value and each is unique of their own. This was decided at the beginning to avoid the issue of normalization. There is a partial dependency between stock and aftermarket inventory since they both are a form of the GPU table. This is preventing the database to be in third normalized form.

Extension of Course Material

Our group wanted to add more complexity to the project. Therefore, we decided to integrate a database into an intermediate language and also run SQL code using that language. We were exploring options of languages to use like JavaScript and Java but in order to keep the project lightweight and portable, we settled on using Python and sqlite. Sqlite is a popular option for a portable database since it writes to a file for all sql execution, and that file can be moved anywhere. This isn't ideal for a production database but for our purposes, sqlite met our requirements. Conveniency was key and Python already has sqlite built-in as a library so this made using sqlite much easier than we initially thought.

SQL Examples

After refactoring the code and cleaning up the files. We then decided to keep the file structure organized and then make a separate file to run the database queries. Therefore, we created a function in python that takes in two parameters. A title or the query name, and the SQL code itself.

```
def query(queryTitle, sqlCode):

  print(queryTitle)

  cursor.execute(sqlCode)

  result = cursor.fetchall()

  for row in result:

    print(row)

  print("\n")
```

Now call the function and give the parameters of title and SQL code.

1.

```
query("All available GPUs in your inventory",

      "SELECT * FROM GPU")
```

The output

```
All available GPUs in your inventory
('RTX 3090 Founders Edition', 1500.0, 600.0, 'Ampere', 1000, 24, 1395, 'Nvidia')
('RTX 3080 Founders Edition', 700.0, 280.0, 'Ampere', 750, 10, 1440, 'Nvidia')
('RTX 3070 Founders Edition', 500.0, 200.0, 'Ampere', 650, 8, 1500, 'Nvidia')
('RTX 3060 Founders Edition', 330.0, 132.0, 'Ampere', 550, 12, 1320, 'Nvidia')
('RTX 3060 Ti Founders Edition', 400.0, 160.0, 'Ampere', 600, 8, 1410, 'Nvidia')
('Radeon RX 6700XT', 479.0, 287.0, 'Big Navi', 600, 12, 2321, 'AMD')
('ASUS ROG STRIX NVIDIA GeForce RTX 3090 Gaming', 1800.0, 600.0, 'Ampere', 1000, 24, 1395, 'Nvidia')
('EVGA GeForce RTX 3080 XC3 ULTRA HYBRID GAMING', 900.0, 280.0, 'Ampere', 750, 10, 1440, 'Nvidia')
('MSI GeForce RTX 3070 GAMING X TRIO', 700.0, 200.0, 'Ampere', 650, 8, 1500, 'Nvidia')
('GIGABYTE AORUS GeForce RTX 3060 ELITE', 530.0, 132.0, 'Ampere', 550, 12, 1320, 'Nvidia')
('ZOTAC GAMING GeForce RTX 3060 Ti Twin Edge OC', 400.0, 160.0, 'Ampere', 600, 8, 1410, 'Nvidia')
('Sapphire Pulse AMD Radeon RX 6700XT', 579.0, 287.0, 'Big Navi', 600, 12, 2321, 'AMD')
```

2.

```
query("GPUs with Clock Speed above 1.4MHz",

      "SELECT GPU_Name FROM GPU WHERE Clock_Speed > 1400")
```

```
GPUs with Clock Speed above 1.4MHz
('RTX 3080 Founders Edition',)
('RTX 3070 Founders Edition',)
('RTX 3060 Ti Founders Edition',)
('Radeon RX 6700XT',)
('EVGA GeForce RTX 3080 XC3 ULTRA HYBRID GAMING',)
('MSI GeForce RTX 3070 GAMING X TRIO',)
('ZOTAC GAMING GeForce RTX 3060 Ti Twin Edge OC',)
('Sapphire Pulse AMD Radeon RX 6700XT',)
```

3.

```
query("GPU quantity greater than 7 from the stock inventory",

      "SELECT GPU_Name FROM STOCK_INVENTORY WHERE GPU_Quantity > 7")
```

```
GPU quantity greater than 7 from the stock inventory
('RTX 3090 Founders Edition',)
('RTX 3080 Founders Edition',)
('RTX 3070 Founders Edition',)
```

4.

```
query("Customers from the city of Mondstadt or the state of Snezhnaya",

      "SELECT * FROM CUSTOMER_ACCOUNT WHERE City = 'Mondstadt' OR State =
'SN'")
```

```
Customers from the city of Mondstadt or the state of Snezhnaya
(2, 'Klee@pcixpress.com', 7654321, '6456 Jumpty dumpty Rd', 'Mondstadt', 'MD', 97897, 'Klee', '12/11/2010')
(3, 'Nora@pcixpress.com', 6762655, '4564 Imaginary ln', 'Mondstadt', 'MD', 56486, 'Nora', '12/11/2010')
(4, 'Fatooey@pcixpress.com', 1666666, '4898 Tsarista Ct', 'Zapolyarny Palace', 'SN', 79786, 'Fatooey', '12/11/1990')
```

5.

```
query("All aftermarket cards",

      "SELECT GPU_Name FROM AFTERMARKET_INVENTORY WHERE GPU_Quantity > 7
ORDER BY GPU_Name ASC")
```

```
All aftermarket cards
('ASUS ROG STRIX NVIDIA GeForce RTX 3090 Gaming',)
('EVGA GeForce RTX 3080 XC3 ULTRA HYBRID GAMING',)
('MSI GeForce RTX 3070 GAMING X TRIO',)
```

6.

```
query("GPUs with VRAM higher than 10GB",
```

```
      "SELECT GPU_Name FROM GPU WHERE VRAM > 10 ORDER BY GPU_Name DESC")
```

```
GPUs with VRAM higher than 10GB
('Sapphire Pulse AMD Radeon RX 6700XT',)
('Radeon RX 6700XT',)
('RTX 3090 Founders Edition',)
('RTX 3060 Founders Edition',)
('GIGABYTE AORUS GeForce RTX 3060 ELITE',)
('ASUS ROG STRIX NVIDIA GeForce RTX 3090 Gaming',)
```

7.

```
query("What customers used a credit card to make a purchase at PCI-Xpress?
For security reasons list the email only and type of payment.",

      "SELECT CUSTOMER_ACCOUNT.Email, CUSTOMER.Payment_Method FROM
CUSTOMER_ACCOUNT INNER JOIN CUSTOMER ON CUSTOMER_ACCOUNT.Account_ID =
CUSTOMER.ID WHERE CUSTOMER.Payment_Method LIKE 'CREDIT'")
```

```
What customers used a credit card to make a purchase at PCI-Xpress? For secruity reasons list the email only and type of payment.
('Blub@pcixpress.com', 'CREDIT')
('Nora@pcixpress.com', 'CREDIT')
('Fatooey@pcixpress.com', 'CREDIT')
```

8.

```
query("What Employees make more than $100.00?",

      "SELECT Name, Wage FROM EMPLOYEE WHERE Wage > 100")
```

```
What Employees make more than $100.00?
('Wayne Pham', 200.0)
('Samip Vaidh', 200.0)
```

9.

```
query("Which customers purchased more than 2 items from PCI-Xpress?",
```

```
      "SELECT CUSTOMER_ACCOUNT.Email, CUSTOMER.Items_Purchased FROM
CUSTOMER INNER JOIN CUSTOMER_ACCOUNT ON CUSTOMER_ACCOUNT.Account_ID =
CUSTOMER.ID WHERE CUSTOMER.Items_Purchased > 2")
```

```
Which customers pruchased more than 2 items from PCI-Xpress?
('Nora@pcixpress.com', 3)
('Fatooey@pcixpress.com', 4)
```

10.

```
query("What GPU Manufacturers are in Santa Clara?",

      "SELECT Name, Contact_Number FROM MANUFACTURER WHERE City LIKE
'Santa Clara' ORDER BY Name ASC")
```

```
What GPU Manufacturers are in Santa Clara?
('AMD', 18772841566)
('Nvidia', 18007976530)
```