Michael Zwartz

November 15th, 2022

GitHub Link:

https://github.com/michaelzwartz/Database-Systems.git

Timmz Atlantic

Database Systems Project

# Part 1: Design

1. The product I sell are underwater-themed action figures for the wildly popular (and made up) television show Sebastian and the Mer-Puppies. The show depicts Sebastian, the crab from Little Mermaid, and his adventures running an underwater doggie daycare. We designs and manufactures action figures based on the characters from the show. The toys are marketed for kids aged 4-10 and are designed to be played with in water at the pool/lake or during bath time at home. Our organization is called Timmz Atlantic. Timmz Atlantic is headquartered in Hilton Head, South Carolina. The company purchases raw materials from around the world. The toys are manufactured here in the USA, then shipped and sold in the USA, Canada, and many countries in Western Europe.
2. Timmz Atlantic is a company that’s comparable to Jakks Pacific. Jakks Pacific is an American toy company founded in 1995 and is now a publicly traded. Jakks produces toys for brands such as Super Mario, Sonic the Hedgehog, and APEX Legends. The 2022 third quarter report shows net sales were $323.0 million, a year-over-year increase of 36%. JAKKS’ products are manufactured in factories all over the world, including China, India, Mexico, and the United States. Jakks toys has over 750 employees worldwide but is headquartered in Santa Monica, California. Financial information found [here](https://www.jakks.com/investors/). Employee information found [here](https://www.jakks.com/corporate/careers.php).

Text

Description automatically generated with medium confidence

1. Timmz Atlantic has about 500 employees worldwide. At Timmz, we will use a structured four-tier client-server.
   1. Tier 1 manages user interface. This includes access for all our employee devices on the system.
   2. Tier 2 is focused on security and checks authorization for all attempted access.
   3. Tier 3 performs the runs the applications by performing business logic and data processing.
   4. Tier 4 in the database server is used for data access, validation, and storage. The four-tier system can be visualized on the right.

server will mainly be used to check authorization and provide recovery control.

1. This business will need to store and access many different data types. The main aspects of Timms Pacific is buying from third-party manufactures, shipping information, and customer data. Primary data could be the name of the toy’s character, the location it’s being shipped to, or the retailer its being sold at. Integers can be used for part numbers and floats will be used for costs and quantities of raw materials. This database will also need to store data about the dates/timestamps of shipped items at each location along the route from manufacture to customer.
2. For our old company files that are not compatible with this new Cloud database set up we will middleware to transfer into our new system. Some examples of this tool being used can be found here: [Spiceworks: Top Middleware Software Platforms](https://www.spiceworks.com/tech/cloud/articles/top-middleware-software-platforms/). This article highlights Flow Middleware platform, IBM WebSphere Application Server, JBoss EAP, Oracle Fusion Middleware, and Microsoft BizTalk Server as some of the best middleware platforms. Common formats that systems use to transfer product data include comma separated values, JSON, and XML files.

# Part 2: Data Entities, Attributes, and Files

1. Five entities Timmz Atlantic maintains records for include: Employee data, Third party manufacturers, products, sales, and shipping information:

* Employee data contains the employee ID numbers, first and last names, location, and salary. Our employees work all over the world. The location will be the city in which there is an office for Timmz Atlantic or remote for our remote workers. The ID number starts with the last two digits of the year they were hired and random numbers after that.
* All of our products are made by third part manufacturers. We choose our partners based on quality, reliability and price. This table records the manufacturer’s name, location (country), max production capacity per day, a cost rating, and a reliability rating.
* The products are tracked by their name, manufacturer, and cost per unit.
* Sales are tracked using an increasing order number, the store name its being sold to, the product name, and quantity sold.
* The shipping information includes the order number, destination city, shipping company, and unique tracking number.

1. Create 5 tables (1 for each entity)

|  |  |
| --- | --- |
| **Employee Data** | |
| Employee\_ID | varchar[10] |
| First name | varchar[15] |
| Last name | varchar[20] |
| Location | String |
| Salary | Float |

|  |  |
| --- | --- |
| **Shipping Information** | |
| Order\_num | varchar[20] |
| Destination City | varchar[10] |
| Carrier | varchar[10] |
| Tracking number | int |

|  |  |
| --- | --- |
| **Third Party Manufacturers** | |
| Name | varchar[20] |
| Location | varchar[10] |
| Capacity per day | Float |
| Cost rating (1-5) | Int |
| Reliability (1-5) | int |

|  |  |
| --- | --- |
| **Sales** | |
| order\_num | int |
| Store\_id | varchar[10] |
| product | varchar[10] |
| quantity | int |

|  |  |
| --- | --- |
| **Product** | |
| Product name | varchar[20] |
| Manufacturer | varchar[10] |
| Cost per unit | Float |

# Part 3: Entity Relationship Diagrams

UML Physical diagram:

A picture containing timeline

Description automatically generated

Crows foot notation:

A picture containing graphical user interface

Description automatically generated

Description of relationships:

Third party manufacturer has a one mandatory to many optional relationship with the products produced at that manufacturer. The manufacturer could produce one or more of our products at one time.

The sales data has a many option to many optional relationship with the product names. A sale might contain one product or many products. The products could be part of just one sale or they could be part of many different sales.

The employee data has a one-to-many optional relationship. The sales employees would be connected to many different sales orders. Some employees who aren’t in sales might not be connected to any sales.

The sales data and shipping information have a one-to-one relationship. Each order number will have a unique tracking number with it.

# Part 4a: Creating the Database

For this part of the project I will be using XAMPP on Windows and running the code in the command shell. The script used to create the tables can be found in my GitHub under the create\_tables.sql file.

With the tables created, here is the screen shot of the empty tables this script created.

Text

Description automatically generated

Once the tables were created, I uploaded the data using the XML files. Here is a screenshot of the code used to upload the data from the .csv files. The script can be found in GitHub under the fill\_tables.sql fie.

Text

Description automatically generated

Finally, here is a screenshot of the full tables printed out in the command prompt using the SELECT \* FROM command.

A screenshot of a computer

Description automatically generated with medium confidence

# Part 4b: SQL Commands

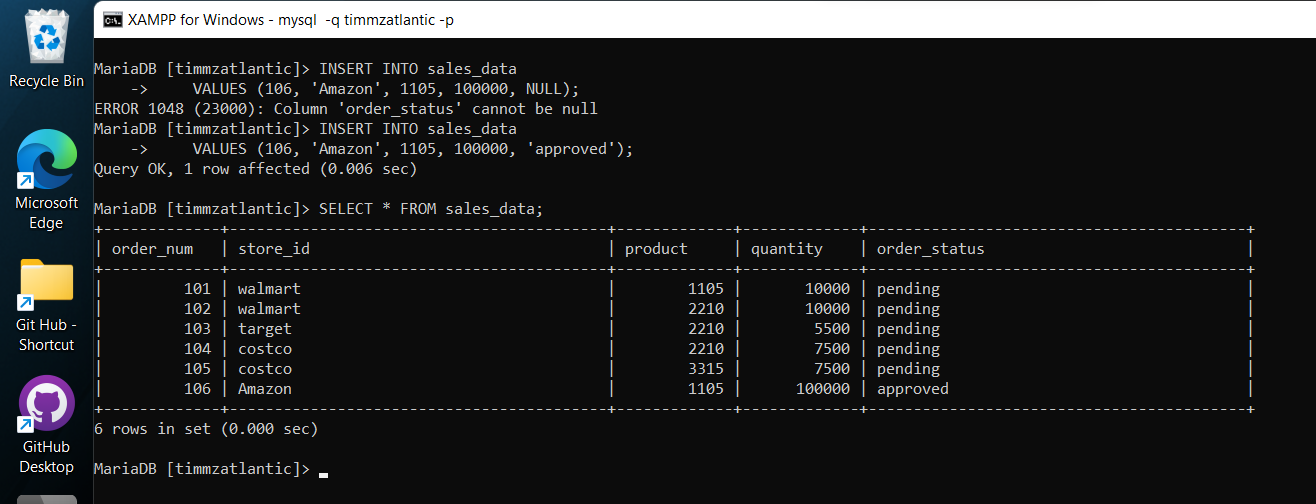
1. The command in the screenshot below adds a column to the sales data table. The new column shows the orders status. Any order can be ‘pending’, ‘approved’, ‘preparing’, ‘in-transit’, or ‘delivered’. This list of order status is the list of the ENUM variable. At the end of the command is the NOT NULL constraint that limits the data from containing a null value.

Here is a screenshot of the table before the command and after the command:

A screenshot of a computer

Description automatically generated

1. This next command will insert a new row into the sales data table.
   1. The first command has a null value for ‘order\_status’ and returns and error.
   2. The second command corrects this error and fills in ‘approved’ for this order status



1. The rest of the rows in this data table need to be updated from the sales ‘pending’ to different statuses. After some time, all of the sales should be approved, packed, shipped and delivered. The order status should reflect the current status of each sale.

A screenshot of a computer

Description automatically generated with medium confidence

1. This next command deletes the new record we just created as seen here.

Graphical user interface

Description automatically generated

1. In the sales data table the order number will increment by 1. We can set the script to create that unique order number for us instead of having the user input it. The AUTO\_INCREMENT feature can be added to the order\_num column because it is the primary key for this table. Another record is added to the table and shows the function or auto\_increment.

**Text

Description automatically generated**

1. The code here sets the default order status to ‘pending’ when a new order is created. That can be seen in the screenshot here.

Text

Description automatically generated

# Python Project:

For this part of the project I will be using Anaconda Spyder to run python code that connects to the XAMPP MYSQL server. The screen has a text editor on the left, a variable list on the top left, and the terminal in the bottom right.

## Part I – Display an entire table

Below you’ll see a screenshot of the code and output to display the entire sales data table from the Timmz Atlantic database.

A screenshot of a computer

Description automatically generated with medium confidence

## Part II – Insert Rows

Below is a screenshot of the same sales data table but with an added row. One interesting note, the SQL table is set up to automatically set the order number to the next number in order and also set the order status as ‘pending’. In the code you’ll see that neither of those two are defined in the new row. After adding and deleting rows a few times, order 125 is the next order number available.

A screenshot of a computer

Description automatically generated with medium confidence

## Part III – Deleting Rows

A screenshot of a computer

Description automatically generated with medium confidence