**CS4416 Project Spring 2016**

**Group Members**

* Ben Smith 14160668
* Jonathan Singer 14136988
* Jack Torpey 14167972

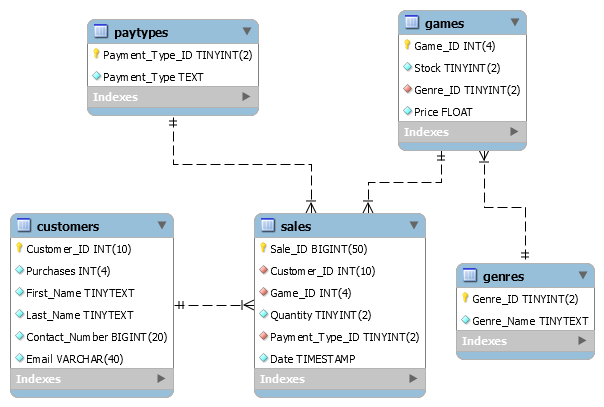
**Assigned work:**

* Database design – Jonathan, Ben
* schema.sql – Jonathan
* queries.sql – Jonathan
* trigger.sql – Jonathan
* procedures.sql – Ben
* Report – Ben

**Database Information (2):**

The database we have created is designed for a game store. The database consists of 5 3NF tables they are called game, genre, customers, sales and paytype. The database stores a variety of information including customer information the sale information and the game information.

When a sale is made the sale is given a ID the customers ID, game ID, quantity, payment Type ID and the date is attributed to the sale. If the customer is a new customer a trigger is called and the customer is given an id and all the relative information attributed to the Customer ID.

**ENTITY RELATIONSHIP DIAGRAM**

**Database Design (3):**

|  |
| --- |
| Primary Key |

games

|  |  |  |  |
| --- | --- | --- | --- |
| Game\_ID | Stock | Genre\_ID | Price |
| 1 | 20 | 1 | 29.99 |
| 2 | 6 | 3 | 24.99 |

genres

|  |  |
| --- | --- |
| Genre\_ID | Genre\_Name |
| 1 | First Person Shooter |
| 2 | Role Playing |

customers

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Customer\_ID | Purchases | First\_Name | Last\_Name | Contact\_Number | Email |
| 1 | 2 | Alfred | Zane | 353874305928 | zane.alfred@gmail.com |
| 2 | 1 | Bruce | O’Leary | 353892368475 | oleary.bruce@gmail.com |

sales

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Sale\_ID | Customer\_ID | Game\_ID | Quantity | Payment\_Type\_ID | Date |
| 1 | 1 | 1 | 1 | 1 | 2016-01-23 15:22:00 |
| 2 | 2 | 4 | 1 | 5 | 2016-01-24 13:45:00 |

paytypes

|  |  |
| --- | --- |
| Payment\_Type\_ID | Payment\_Type |
| 1 | Cash |
| 2 | Visa Credit |

**Functional Dependencies (4):**

**gameshop\_game:** Game\_ID –> Game\_ID

Game\_ID –> Stock

Game\_ID –> Genre\_ID

Game\_ID –> Price

**gameshop\_genre:** Genre\_ID –> Genre\_ID

Genre\_ID –> Genre\_Name

**gameshop\_customers:** Customer\_ID –> Customer\_ID

Customer\_ID –> Purchases

Customer\_ID –> First\_Name

Customer\_ID –> Last\_Name

Customer\_ID –> Contact\_Number

Customer\_ID –> Email

**gameshop\_sales:** Sale\_ID –> Sale\_ID

Sale\_ID –> Customer\_ID

Sale\_ID –> Game\_ID

Sale\_ID –> Quantity

Sale\_ID –>Payment\_Type\_ID

Sale\_ID –> Date

**gameshop\_paytype:** Payment\_Type\_ID –> Payment\_Type\_ID

Payment\_Type\_ID –> Payment\_Type

**Proof That Tables are 3NF (5):**

Contains only atomic values: **☑**

There are no repeating groups: **☑**

All non-key attributes are fully functional dependent on the primary key: **☑**

There is no transitive functional dependency: **☑**

**Usefulness of Queries (6):**

**Games that cost more than the average game price:**

This could be used in tandem with query number 3 to see if the price matters to your customers.

**Customers that have bought 5, 10, or 15 games:**

This could be used for loyalty rewards.

**Games that have sold more than the average game:**

This would be useful for seeing what games to keep in stock and what game to sell off in a sale.

**Analysis of the speed of the queries (7):**

Query number 1 takes on average 0.0006 seconds to complete.

Query number 2 takes on average 0.0004 seconds to complete.

Query number 3 takes on average 0.0011 seconds to complete.

Indexes proposes made little to no difference in terms of the speed.

**Justification for the necessity of the triggers and procedures (8):**

**Triggers:**

**after\_sales\_insert\_update\_purchases:**

This is necessary to update the purchases for the customer when a sale is made. This would save a lot of time so the purchases wouldn’t have to be updated manually.

**after\_sales\_insert\_update\_stocks:**

This is necessary to update the stock for a game in the game table after a purchase has been made. This would save a lot of time and prevent errors in the database.

**Procedures:**

**CountSalesByGameID:**

This procedure counts the total sales by a game ID and is useful for finding out the most popular games and make business decisions based on the data that is returned.

**DisplayGamesToBeSoldInSale:**

This procedure display the game ID, Stock, Genre Name,and Price based on the price and stock of the game. This procedure would be useful for seeing what games need to be sold off in a sale.