Group Project Paper

Forever Resort Database

Francisco Mendoza, Francisco Moo, Youngmin Kim, and Zhiqin Zhang

IT509: Data Management

Spring Term 2021  
Professor: Dr. Virginia ILIE  
April 21, 2021

**Table of Contents**

[Introduction](#_heading=h.3dy6vkm) 2

[Assumptions](#_heading=h.1t3h5sf) 2

[Entity-Relationship Diagram 1](#_heading=h.4d34og8) 3

[Entity-Relationship Diagram 2](#_heading=h.2s8eyo1) 5

[Relational Design](#_heading=h.17dp8vu) 7

[Research on SQL Server Datatypes](#_heading=h.4b0vquu986) 8

[Data Dictionary](#_heading=h.cwh5jq8boh18) 11

[Normalization](#_heading=h.hqit56obm20m) 15

# Introduction

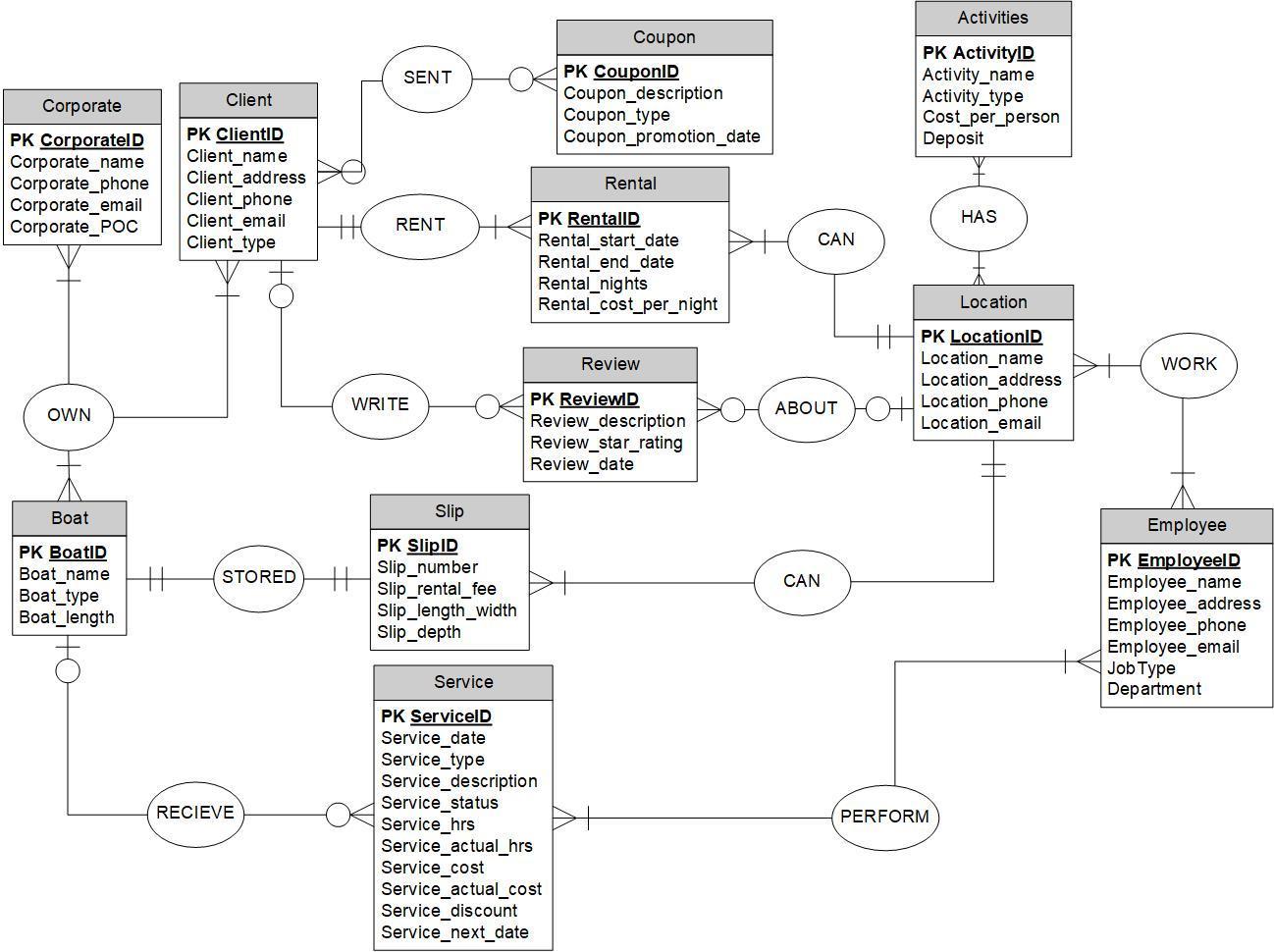
Thank you for choosing Echelon Up Inc. for your technology needs. We understand how technology can drive your business forward. The hospitality industry is dynamic and demanding constant innovation. We are delighted to bring you the vehicle that will deliver your business into the 21st century. Through our intuitive design, you will gain insight that will give you a competitive edge. Through the power of effective data management, you can make actionable decisions to drive your business forward with maximized profits.

## **Assumptions**

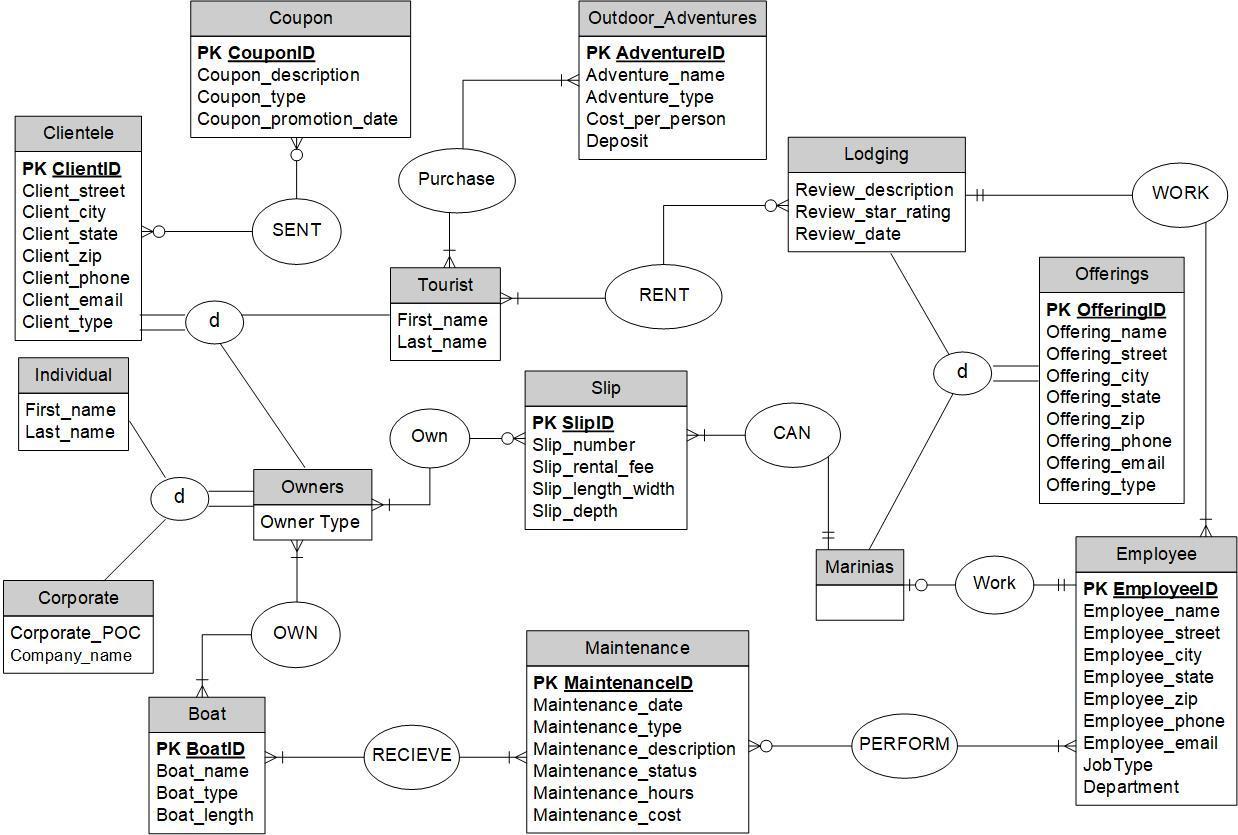
1. Client may or may not receive a coupon.
2. Client may or may/not write a review about a location.
3. Boat may or may/not receive service.
4. Employee May or may/not perform a service.

# Entity-Relationship Diagram 1

1. Original

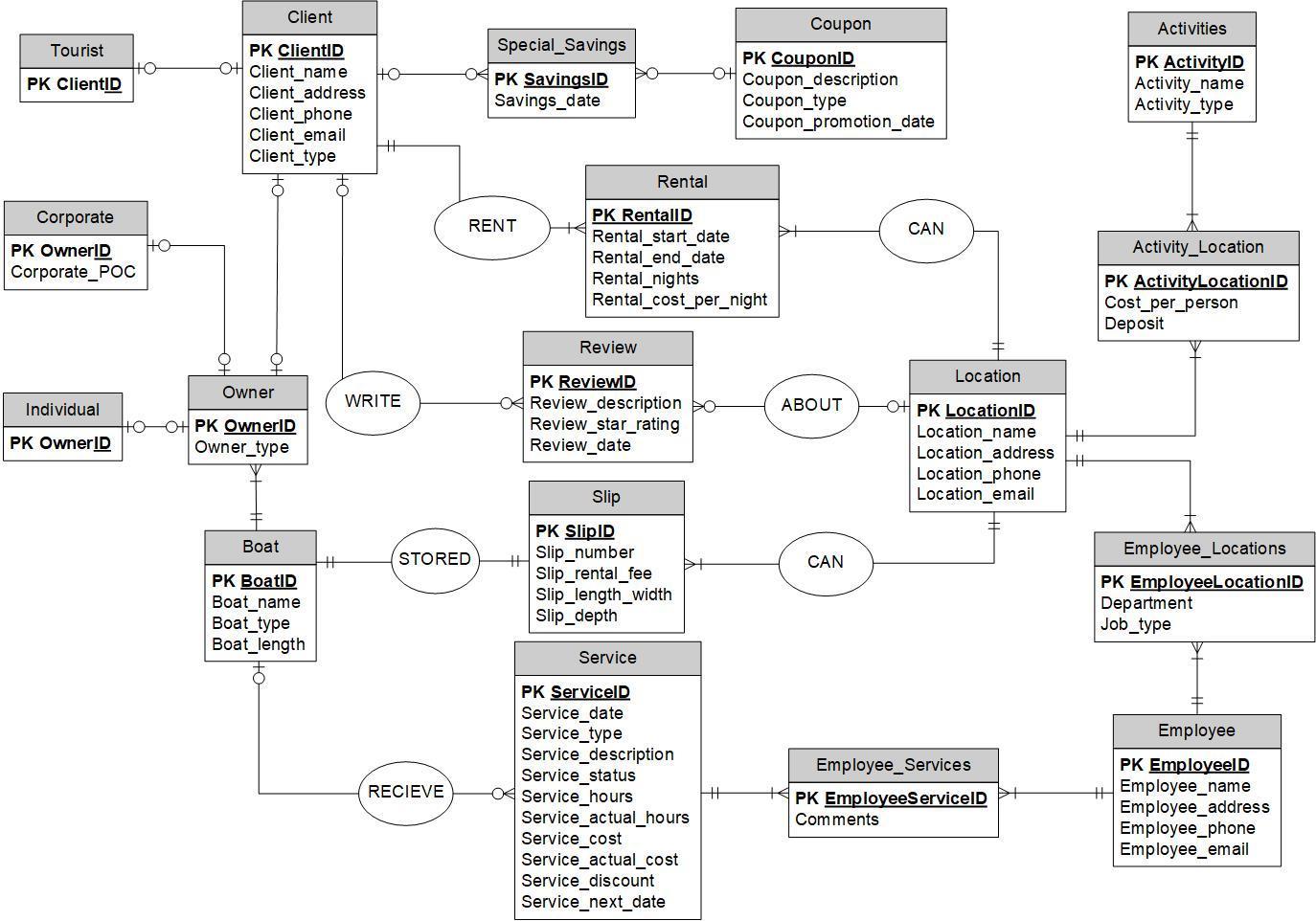


2. Revised

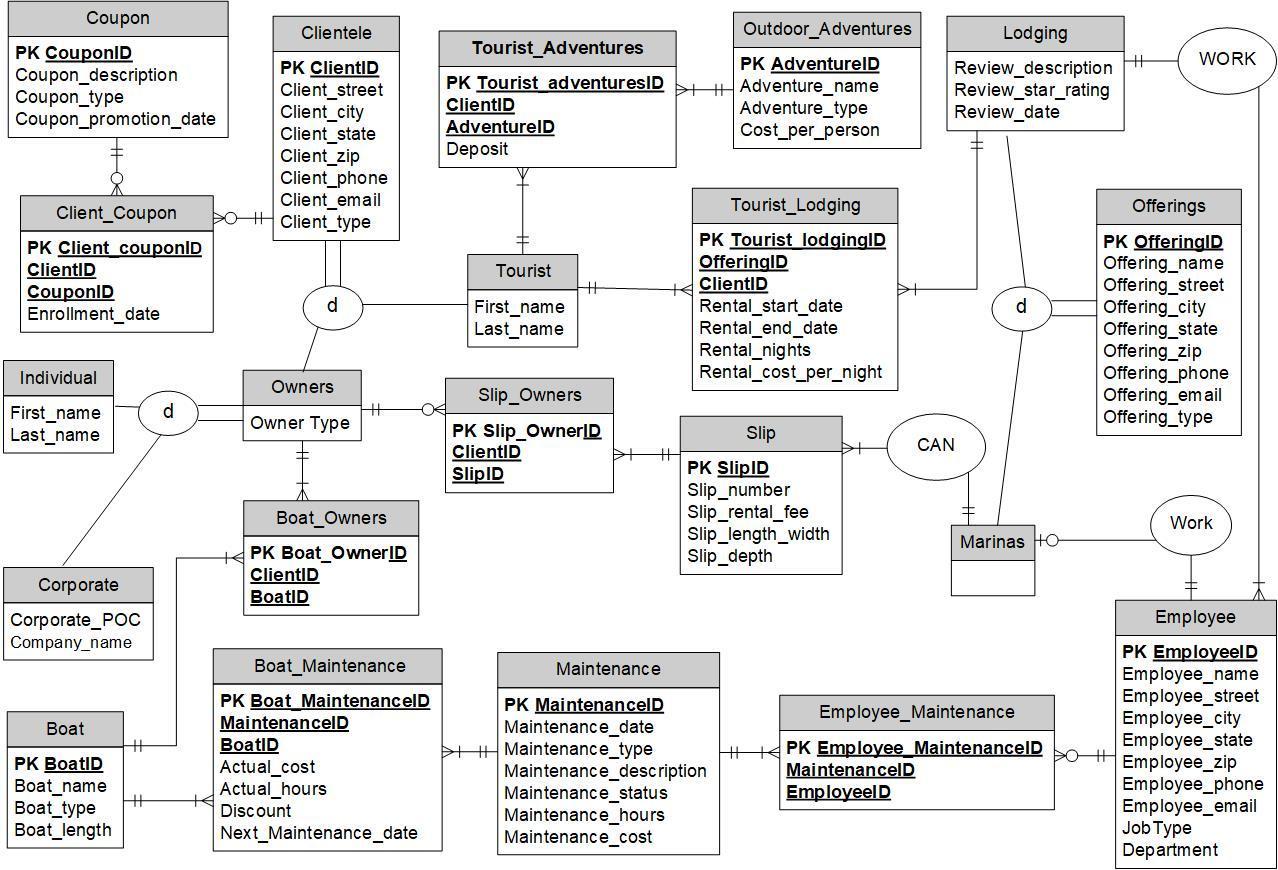


# Entity-Relationship Diagram 2

1. Original



1. Revised



# Relational Design



# Research on SQL Server Datatypes

| **EXACT NUMERICS** |  |  |
| --- | --- | --- |
| **Ms. SQL Server DATATYPE** | **DESCRIPTION** | **WHEN TO USE** |
| bigint(*size*) | A larger integer that ranges from -9223372036854775808 and 9223372036854775807. The size parameters specifies maximum display. | Used when trying to capture large integer values that exceed an integer. |
| bit | An integer data type that can take a value of 1,0, or NULL. | Used to capture boolean values. |
| decimal(*size,d*) | Extract numeric type defined by the total number of digits(size) and number of decimal points(d). | Used when storing numeric types that require decimal values. Dollars, percentages, etc. |
| int(*size*) | Integers ranging from -2,147,483,648 and 2,147,483,647. The size parameters specifies maximum display. | Used when capturing integers within a specific range. Primary integer type. |
| money | Money or currency values between -922,337,203,685,477.5808 and 922,337,203,685,477.5807. | Used to capture currency values. |
| numeric(*p,s*) | Fixed precision and scale numbers defined by the total number of decimal digits(p) and number of decimal points to the right(s). | Used to capture literal representation of a numbers value. |
| smallint(*size*) | Integers ranging between -32,768 and 32,767. The size parameters specifies maximum display. | Used to capture smaller integers values. |
| smallmoney | Money or currency values between - 214,748.3648 and 214,748.3647. | Used to capture smaller currency values. |
| tinyint(*size*) | Integers ranging from 0 to 255. | Used to capture integers no more than 255. |
|  |  |  |
| **APPROXIMATE NUMERICS** | |  |
| **Ms. SQL Server DATATYPE** | **DESCRIPTION** | **WHEN TO USE** |
| float*(n)* | Approximate number data type, n is the number of bits used to store the mantissa of the float number in scientific notation. Value must be between 1 and 53. | Used for scientific Measurements |
| real | Approximation of a real number. Can range between 1.79E+308 to -2.23E-308, 0 and 2.23E-308 to 1.79E+308. | Used when capturing approximation of a real number. |
|  |  |  |
| **DATE AND TIME** |  |  |
| **Ms. SQL Server DATATYPE** | **DESCRIPTION** | **WHEN TO USE** |
| date | Stores a date only from January 1, 0001 to December 31, 9999. **Ex. YYYY-MM-DD** | Used when trying to capture date by year, month, and day. |
| datetime2 | Stores a date with 100 nanoseconds from January 1, 0001 to December 31, 9999. **Ex. YYYY-MM-DD hh:mm:ss[.nnnnnnn]** | Used when trying to capture date, hours, minutes, seconds, and nanoseconds. |
| datetime | Stores a date with 3.33 milliseconds from January 1, 0001 to December 31, 9999. **Ex. YYYYMMDD[ hh:mm:ss[.mmm]]** | Used when trying to capture date, hours, minutes, seconds, and milliseconds. |
| datetimeoffset | Similar to datetime2 in addition with a time zone offset. **YYYY-MM-DDThh:mm:ss[.nnnnnnn]Z (UTC)** | Used when trying to capture date, hours, minutes, seconds, nanoseconds and time zone. |
| smalldatetime | Stores a date in combination with a time of day. **Ex. YYYYMMDD[ hh:mm:ss]** | Used when trying to capture date, hours, minutes, and seconds. |
| time | Defines a time of day without a time zone. **Ex. hh:mm:ss** | Used when trying to capture hours, minutes, and seconds. |
|  |  |  |
| **CHARACTER STRINGS** | |  |
| **Ms. SQL Server DATATYPE** | **DESCRIPTION** | **WHEN TO USE** |
| char(*n)* | Fixed size string data, n defines the string size in bytes and be a value between 1 - 8,000. | Used when the size of the column entry is consistent. |
| text | Variable length non-Unicode data with a max string length of 2,147,483,647. | Used for larger string data. |
| varchar(*n|max*) | Variable size string data. n defines the string size in bytes and must be a value between 1 - 8,000 or use max to apply the maximums storage size (2GB). | Used when the size of the column entry varies, or the string length exceeds 8,000 bytes(max). |
|  |  |  |
| **UNICODE CHARACTER STRINGS** | |  |
| **Ms. SQL Server DATATYPE** | **DESCRIPTION** | **WHEN TO USE** |
| nchar*(n)* | Unicode string data with a fixed size string data where *n* defines the string size a value 1 - 4,000. | Used when the size of the column data entry are consistent. |
| ntext | Unicode string data with max string length of 1,073,741,823. | Used for larger string data. |
| nvarchar*(n|max)* | Unicode string data where *n* defines the string size value 1 - 4,000. Max indicates the max storage which is 2GB. | Used when the size of the column entry varies, or the string length exceeds 4,000 bytes(max). |
|  |  |  |
| **BINARY STRINGS** | |  |
| **Ms. SQL Server DATATYPE** | **DESCRIPTION** | **WHEN TO USE** |
| binary(*n*) | Binary string data with a length of *n* bytes. Max size is 8,000 bytes. | Used to capture images, word files, text files, etc. |
| image | Binary string data that is used to store image files (BMP, TIFF, GIF, or JPEG). Max size is 2GB. | Used to capture image files. (BMP, TIFF, GIF, or JPEG). |
| varbinary(*n|max*) | Binary string data that is used to store images, pdfs, and word documents. Max size is 2GB. | Used to capture images, pdfs, and word documents. |

# Data Dictionary

| **TABLE: CLIENTELE** |  |  |
| --- | --- | --- |
| **Column Name** | **DataType** | **Constraints** |
| **ClientID** | INT | Primary Key for CLIENTELE table |
| Client\_street | VARCHAR(50) |  |
| Client\_city | VARCHAR(50) |  |
| Client\_state | CHAR(2) |  |
| Client\_zip | VARCHAR(10) |  |
| Client\_phone | CHAR(10) |  |
| Client\_email | VARCHAR(50) |  |
| Client\_type | VARCHAR(20) |  |
|  |  |  |
| **TABLE: COUPON** |  |  |
| **Column Name** | **DataType** | **Constraints** |
| **CouponID** | INT | Primary Key for OUTDOOR\_ADVENTURES table |
| Coupon\_description | VARCHAR(100) |  |
| Coupon\_type | VARCHAR(50) |  |
| Coupon\_promotion\_date | DATETIME |  |
|  |  |  |
| **TABLE: BOAT** |  |  |
| **Column Name** | **DataType** | **Constraints** |
| **BoatID** | INT | Primary Key for BOAT table |
| Boat\_name | VARCHAR(50) |  |
| Boat\_type | VARCHAR(50) |  |
| Boat\_length\_ft | INT |  |
|  |  |  |
| **TABLE: SLIP** |  |  |
| **Column Name** | **DataType** | **Constraints** |
| **SlipID** | INT | Primary Key for SLIP table |
| Slip\_number | INT |  |
| Slip\_rental\_fee | DECIMAL(5,2) |  |
| Slip\_length\_width | DECIMAL(5,2) |  |
| Slip\_depth | DECIMAL(5,2) |  |
|  |  |  |
| **TABLE: MAINTENANCE** |  |  |
| **Column Name** | **DataType** | **Constraints** |
| **MaintenanceID** | INT | Primary Key for MAINTENANCE table |
| Maintenance\_type | VARCHAR(50) |  |
| Maintenance\_description | VARCHAR(100) |  |
| Maintenance\_status | VARCHAR(50) |  |
| Maintenance\_hours | INT |  |
| Maintenance\_cost | DECIMAL(7,2) |  |
|  |  |  |
| **TABLE: EMPLOYEE** |  |  |
| **Column Name** | **DataType** | **Constraints** |
| **EmployeeID** | INT | Primary Key for EMPLOYEE table |
| Employee\_name | VARCHAR(50) |  |
| Employee\_street | VARCHAR(50) |  |
| Employee\_city | VARCHAR(50) |  |
| Employee\_state | CHAR(2) |  |
| Employee\_zip | VARCHAR(10) |  |
| Employee\_phone | CHAR(10) |  |
| Employee\_email | VARCHAR(50) |  |
| Job\_type | VARCHAR(50) |  |
| Department | VARCHAR(20) |  |
|  |  |  |
| **TABLE: OFFERINGS** |  |  |
| **Column Name** | **DataType** | **Constraints** |
| **OfferingsID** | INT | Primary Key for OFFERINGS table |
| Offerings\_name | VARCHAR(50) |  |
| Offerings\_street | VARCHAR(50) |  |
| Offerings\_city | VARCHAR(50) |  |
| Offerings\_state | CHAR(2) |  |
| Offerings\_zip | VARCHAR(10) |  |
| Offerings\_phone | CHAR(10) |  |
| Offerings\_email | VARCHAR(50) |  |
| Offerings\_type | VARCHAR(20) |  |
|  |  |  |
| **TABLE: OUTDOOR\_ADVENTURES** | |  |
| **Column Name** | **DataType** | **Constraints** |
| **AdventureID** | INT | Primary Key for OUTDOOR\_ADVENTURES table |
| Adventure\_name | VARCHAR(50) |  |
| Adventure\_type | VARCHAR(50) |  |
| Cost\_per\_person | DECIMAL(5,2) |  |
|  |  |  |
| **TABLE: CLIENT\_COUPON** |  |  |
| **Column Name** | **DataType** | **Constraints** |
| **Client\_CouponID** | INT | Primary Key for CLIENT\_COUPON table |
| ***ClientID*** | INT | Primary Key for CLIENT\_COUPON table; Foreign Key to CLIENTELE table |
| ***CouponID*** | INT | Primary Key for CLIENT\_COUPON table; Foreign Key to COUPON table |
| Enrollment\_date | DATETIME |  |
|  |  |  |
| **TABLE: TOURIST** |  |  |
| **Column Name** | **DataType** | **Constraints** |
| ***ClientID*** | INT | Primary Key for TOURIST table; Foreign key for CLIENTELE table |
| First\_name | VARCHAR(50) |  |
| Last\_name | VARCHAR(50) |  |
|  |  |  |
| **TABLE: TOURIST\_ADVENTURES** | |  |
| **Column Name** | **DataType** | **Constraints** |
| **Tourlist\_AdventuresID** | INT | Primary Key for TOURIST\_ADVENTURES table |
| ***ClientID*** | INT | Primary Key for TOURIST\_ADVENTURES table; Foreign Key to CLIENTELE table |
| ***AdventureID*** | INT | Primary Key for TOURIST\_ADVENTURES table; Foreign Key to OUTDOOR\_ADEVNTURES table |
| Deposit | DECIMAL(5,2) |  |
|  |  |  |
| **TABLE: TOURIST\_LODGING** | |  |
| **Column Name** | **DataType** | **Constraints** |
| **Tourist\_lodgingID** | INT | Primary Key for TOURIST\_LODGING table |
| ***OfferingsID*** | INT | Primary Key for TOURIST\_LODGING table; Foreign Key to OFFERINGS table |
| ***ClientID*** | INT | Primary Key for TOURIST\_LODGING table; Foreign Key to CLIENTELE table |
| Rental\_start\_night | DATETIME |  |
| Rental\_end\_night | DATETIME |  |
| Rental\_cost\_per\_night | DECIMAL(5,2) |  |
| Rental\_nights | INT |  |
|  |  |  |
| **TABLE: LODGING** |  |  |
| **Column Name** | **DataType** | **Constraints** |
| ***OfferingsID*** | INT | Primary Key for LODGING table; Foreign Key to OFFERINGS table |
| ***EmployeeID*** | INT | Primary Key for LODGING table; Foreign Key to EMPLOYEE table |
| Review\_star\_rating | CHAR(10) |  |
| Review\_description | VARCHAR(255) |  |
| Review\_date | DATETIME |  |
|  |  |  |
| **TABLE: OWNERS** |  |  |
| **Column Name** | **DataType** | **Constraints** |
| ***ClientID*** | INT | Primary Key for OWNERS table; Foreign Key for CLIENTELE table |
| Owner\_type | VARCHAR(50) |  |
|  |  |  |
| **TABLE: INDIVIDUAL** |  |  |
| **Column Name** | **DataType** | **Constraints** |
| ***ClientID*** | INT | Primary Key for INDIVIDUAL table; Foreign Key for CLIENTELE table |
| First\_name | VARCHAR(50) |  |
| Last\_name | VARCHAR(50) |  |
|  |  |  |
| **TABLE: CORPORATE** |  |  |
| **Column Name** | **DataType** | **Constraints** |
| ***ClientID*** | INT | Primary Key for CORPORATE table; Foreign Key for CLIENTELE table |
| Corporate\_POC | VARCHAR(50) |  |
| Corporate\_name | VARCHAR(50) |  |
|  |  |  |
| **TABLE: BOAT\_OWNERS** |  |  |
| **Column Name** | **DataType** | **Constraints** |
| **Boat\_OwnerID** | INT | Primary Key for BOAT\_OWNERS table |
| ***ClientID*** | INT | Primary Key for BOAT\_OWNERS table; Foreign Key for CLIENTELE table |
| ***BoatID*** | INT | Primary Key for BOAT\_OWNERS table; Foreign Key for BOAT table |
|  |  |  |
| **TABLE: BOAT\_MAINTENANCE** | |  |
| **Column Name** | **DataType** | **Constraints** |
| ***B*oat\_MaintenanceID** | INT | Primary Key for BOAT\_MAINTENANCE table |
| ***MaintenanceID*** | INT | Primary Key for BOAT\_MAINTENANCE table; Foreign Key to MAINTENANCE table |
| ***BoatID*** | INT | Primary Key for BOAT\_MAINTENANCE table; Foreign Key to BOAT table |
| Maintenace\_date | DATETIME |  |
| Actual\_cost | DECIMAL(7,2) |  |
| Actual\_hours | INT |  |
| Discount | DECIMAL(3,2) |  |
| Next\_Maintenance\_date | DATETIME |  |
|  |  |  |
| **TABLE: EMPLOYEE\_MAINTENANCE** | |  |
| **Column Name** | **DataType** | **Constraints** |
| **Employee\_MaintenanceID** | INT | Primary Key for EMPLOYEE\_MAINTENANCE table |
| ***EmployeeID*** | INT | Primary Key for EMPLOYEE\_MAINTENANCE table; Foreign Key to EMPLOYEE table |
| ***MaintenanceID*** | INT | Primary Key for EMPLOYEE\_MAINTENANCE table; Foreign Key to MAINTENANCE table |
|  |  |  |
| **TABLE: SLIP\_OWNERS** |  |  |
| **Column Name** | **DataType** | **Constraints** |
| **Owner\_SlipID** | INT | Primary Key for SLIP\_OWNERS table |
| ***SlipID*** | INT | Primary Key for SLIP\_OWNERS table; Foreign Key for SLIP table |
| ***ClientID*** | INT | Primary Key for SLIP\_OWNERS table; Foreign Key for CLIENTELE table |
|  |  |  |
| **TABLE: MARINAS** |  |  |
| **Column Name** | **DataType** | **Constraints** |
| ***OfferingsID*** | INT | Primary Key for MARINA table; Foreign Key for OFFERINGS table |
| ***SlipID*** | INT | Primary Key for MARINA table; Foreign Key to SLIP table |
| ***EmployeeID*** | INT | Primary Key for MARINA table; Foreign Key to EMPLOYEE table |

# Normalization

By looking at our logic design and resolving associative entities, our design appears to be normalized. As discussed in class, there are different forms of normalization, each critical to our logical design and will be discussed in detail.

First Normal Form, identifies that our relationships do not contain multivalued attributes, and each attribute value is atomic or indivisible. In our design, all composite keys that do not produce unique instances have been resolved. Also, we have also resolved the many to many relationships into associative entities to prevent multivalued attributes. Therefore, our logic design satisfies the first normal form.

Second Normal Form consists of the First Normal Form rules and removing partial dependencies. Removing partial dependencies is done by separating Boats, Boat Maintenance, and Maintenance into separate entities. Maintenance date, actual cost, discount, and next maintenance date is partially dependent on Boat and Maintenance. By creating a Boat Maintenance entity, we create full functional dependencies for the attributes that pertain to the actual maintenance done on the boat. Therefore, this design satisfies the second normal form.

Third Normal Form consists of Second Normal Form rules and no transitive dependencies. Our logical design does not consist of transitive dependencies. All attributes refer back to their primary keys and are not dependent on other primary keys with different attributes. By doing this, we eliminate orphan data which is caused by removing data being shared in the same entity and can be referred back to if needed. In conclusion, our logic design appears to be normalized and allows us to streamline our database. We eliminate the idea of redundant data and normalize all our relations with anomalies by creating fully dependent entities.