

The King

William Hotel

*By:*

Mathew Migliore, Tyler Segovia, and Kevin Romero

*Presented to:*

Basil J. Cleese and The Development Team

*of:*

The King William Hotel

*Date:*

2020-11-26

Table of Contents

Introduction2

Mission Statement3

Objectives4

User View Preface5

User View6

System Boundary Preface7

System Boundary8

3NF Preface9

3NF Diagram10

3NF Graph11

Functional Dependencies Preface12

Functional Dependencies13-14

Data Dictionary Preface15

Data Dictionary16-17

Final Word18

Dictionary19

# **Introduction**

The King William Hotel, a historic much-loved landmark built in 1923. It serves the community of Ontario and offers guests hospitality and services that makes the guest want to come back. Due to the hotel’s older age and transition to the modern world, the hotel has renovated it’s rooms to look part of an early 20th era, Although the rooms have changed, the hospitality and service remain the same. This transition to the modern world and its perpetual top of the line service has come at cost. Although the staff maintain a great system to book the hotel’s information, it is now time for their system to adopt a new modern take, not only to facilitate but to also improve their service. With this Report, our team will be doing our best to show case the many different approaches we have taken to meet the client’s requests with as much accuracy as possible and to present the application we have created. Without further delay, here is our solution to The King William case.

# **MISSION STATEMENT**

The purpose of the King Williams Database Application is to store and manage information to improve efficiency and control of hotel processes.

# **OBJECTIVES**

- To maintain (enter, update, delete) data on rooms

- To maintain (enter, update, delete) data on room reservations

- To maintain (enter, update, delete) data on customers

- To maintain (enter, update, delete) data on customer billing

- To maintain (enter, update, delete) data on chargeable items

- To maintain (enter, update, delete) data on transactions

- To maintain (enter, update, delete) data on employees

- To maintain (enter, update, delete) data on employment positions

- To perform searches on rooms

- To perform searches on room reservations

- To perform searches on customers

- To perform searches on customer billing

- To perform searches on chargeable items

- To perform searches on transactions

- To perform searches on employees

- To perform searches on employment positions

- To track the status of rooms

- To track the status of customer bills

- To track the status of customer requests

- To report on room reservations

- To report on customers

- To report on customer billing

- To report on chargeable items

- To report on transactions

- To report on employees

# **User View Preface**

For the user views we took into consideration the employees that would access the system and what each employee should or should not be able to see and/or do while logged in. We managed to break the system down into 6 categories ranging from low to full access of the system.

## **Low access:**

This is a split between Kitchen Staff, Room Cleaning, Accounting & Stock, and Front Desk. They are each able to view, update, and edit certain parts of the Users information, but are not able to do everything for the users. They are also unable to edit the information for themselves or any other higher-level user.

## **High access:**

Management has access over all the previous users, can edit, update, and the records, but do not have full access. They cannot delete records.

## **Full access:**

Admin has full access over everything. They can edit, update, and delete all records and all users.

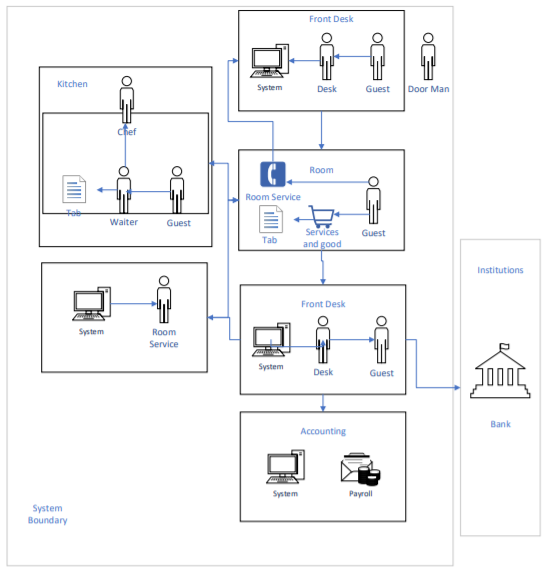
# **User View**



# **System Boundary Preface**

For the System Boundary we’ve took a lot of inspiration from the user view and mostly wanted to focus on the six major groups and their basic operations within the system that they would go through when creating, viewing, and modifying any information from the system. Once we had that we thought about other external groups that interacted with those functions.

# **SYSTEM BOUNDARY**



**3NF Preface**

**Introduction:** For the data normalization our group took a different approach than the standard, instead of starting with the list, we wanted to create the graph so we could envision how the data would really work within our heads, and how we could organize it in a way that it was programmatically easy to insert, create and update tables. Once the graph was completed, we translated that to the regular 3NF list, which helped us find more logical error we missed in the graph.

**How did we come up with the design?**

We began the design with the most important table, which was reservations. We tried to envision how the entire system came together from the time a guest enters the hotel, to the moment they leave. We then visualized in which moments that guest interacted with the system and how the workers did so as well. Finally, we thought about how each individual worker interacted with the system, which also includes when they’re not attending the guests but doing work at the back.

**What were the biggest challenges?**

The biggest challenges were thinking how we could fulfil every need each worker might have when interacting with the system, without going out of scope.

**Why did our group choose those tables?**

It is our belief that these tables store all the necessary data to perform the functions required and also allow the development team to expand their system for future endeavors.

**Legend**

**Table Name (Primary Key (PK), Foreign Key (FK),** Attribute Name…**)**

# **3NF Diagram**

**reservations** (**reservationID (PK)**, **roomNumber (FK)**, **customerID (FK)**, **billID (FK)**, numberOfGuests, startDate, endDate, notes)

**rooms**(**roomNumber (PK)** , **floorNumber (FK)**, **roomTypeID (FK)**, **statusID (FK)**, notes)

**roomStatus**(**statusID (PK)**, statusDescription)

**roomTypes**(**roomTypeID (PK)**, roomType, pricePerNight)

**customers**(**customerID (PK)**, firstName, lastName, phoneNumber, **addressID**)

**address**(**addressID (PK)**, addressLine1, addressLine2, city, **provinceID (FK)**, country, postalCode)

**provinces**(**provinceID (PK)**, provinceCode, provinceName)

**customerBilling**(**billID (PK)**, billAmount, **reservationID (FK)**, paymentType, amountOwing)

**chargeableItems**(**itemID (PK)**, itemName, itemDescription, itemPrice)

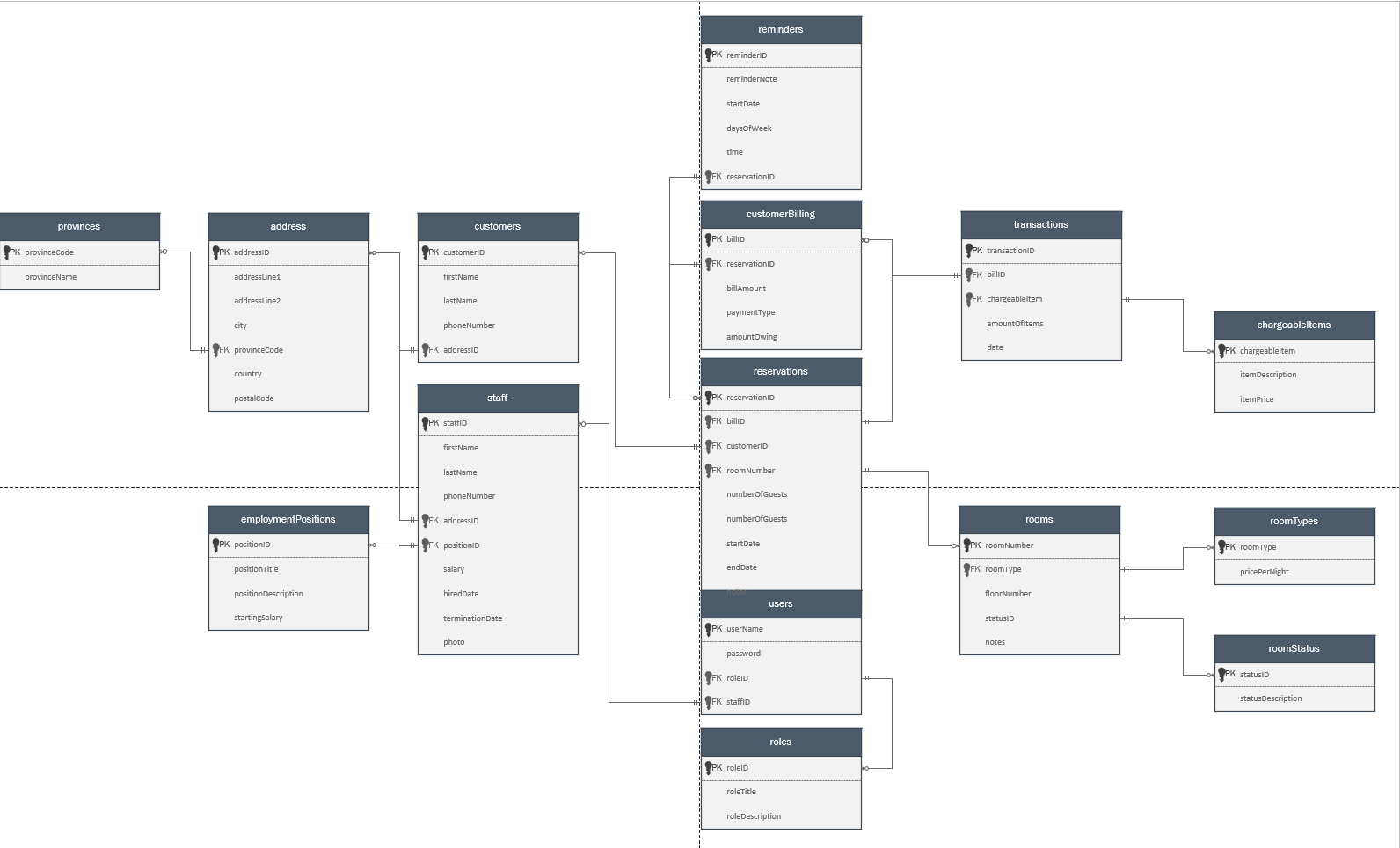
**transactions**(**transactionID (PK)**, **billID (FK)**, **itemID (FK)**, amountOfItems, date)

**staff**(**staffID (PK)**, firstName, lastName, phoneNumber, **addressID (FK)**, **positionID (FK)**, salary, hiredDate, terminationDate, photo)

**employmentPositions**(**positionID (PK)**, positionTitle, positionDescription, startingSalary)

**users**(**userID (PK)**, username, password, **roleID (FK)**, **staffID (FK)**)

**roles**(**roleID (PK)**, roleTitle, roleDescription)



**Functional Dependencies Preface**

**Introduction:** For the functional dependencies we thought about which attributes would logically determine the others within the database, this is where the graph we made for the 3NF design came in handy, we were able to visualize within a table each attribute’s role within the database.

**How did we come up with the design?**

The standard Functional Dependency format was followed.

**What were the biggest challenges?**

The biggest challenges laid in determining which attributes in the table would allow us to search for other attributes and which ones wouldn’t.

**Legend**

**Table Name:**

{Determining Attribute/s (X)} -> {Functionally Dependent Attribute/s (Y)}

**Functional Dependencies**

**reservation:**

{reservationID} -> {roomNumber, customerID, billID, numberOfGuests, startDate, endDate, notes}

{numberOfGuests, startDate, endDate, notes} -> {reservationID, roomNumber, customerID, billID}

**rooms:**

{roomNumber} -> {floorNumber, roomType, statusID, notes}

{statusID} -> {roomNumber, floorNumber}

**roomStatus:**

{statusID} -> {statusDescription}

**roomTypes:**

{roomType} -> {pricePerNight}

**Customers:**

{customerID} -> {firstName, lastName, phoneNumber, addressID}

{firstName, lastName, phoneNumber} -> {customerID, addressID}

**address:**

{addressID} -> {addressLine1, addressLine2, city, provinceCode, country, postalCode}

**Provinces:**

{provinceCode} -> {provinceName}

**customerBilling:**

{billID} -> {billAmount, reservationID, paymentType, amountOwing}

**chargableItems:**

{chargableItem} -> {itemDescription, itemPrice}

{itemDescription, itemPrice} -> {chargableItem}

**transactions:**

{trasactionID} -> {billID, chargeableItem, amountOfItems, date}

{chargableItem, billID, amountofItems} -> {transactionID, date}

{chargeableItem, amountOfItems, date} -> {transactionID, billID}

**staff:**

{staffID} -> {firstName, lastName, phoneNumber, addressID, positionID, salary, hiredDate, terminationDate, photo}

{positionID} -> {staffID, firstName, lastName, phoneNumber, salary, hiredDate, terminationDate}

**employmentPositions:**

{positionID} -> {positionTitle, positionDescription, startingSalary)

**users:**

{userName} -> {password, roleID, staffID}

{staffID} -> {password, roleID, userName}

**roles:**

{roleID} -> {roleTitle, roleDescription}

**Data Dictionary Preface**

**Introduction:** For the data dictionary, we spend quite a bit of time looking at the things it was being asked of us again in the system request. We really wanted to make sure that the sizes of data were consistent with the report that were presented and the other possible type of data that could be stored. We tried to have a balance with the storage and the user friendliness when entering data. One of the reasons we took the graphical approach for the 3NF design was because we knew the immense help it would provide use for developing the Data Dictionary. With the combination of the 3NF design, functional dependencies, and this data dictionary, we were able to catch errors it the original designs and correct them before it was handed over to the development team.

# **Data Dictionary**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Table | Column | Data Type | References | Default | Not Null |
| <reservations> | reservationID | INT |  | Identity | Y |
| <reservations> | roomNumber | INT | <rooms>.roomNumber |  | Y |
| <reservations> | customerID | INT | <customers>.customerID |  | Y |
| <reservations> | billID | INT | <customerBilling>.billID |  | Y |
| <reservations> | numberOfGuests | INT |  |  | Y |
| <reservations> | startDate | DATETIME |  |  | Y |
| <reservations> | endDate | DATETIME |  |  | Y |
| <reservations> | notes | VARCHAR |  |  | N |
| <rooms> | roomNumber | INT |  | Identity | Y |
| <rooms> | floorNumber | INT |  |  | Y |
| <rooms> | roomTypeID | INT | <roomTypes>.roomTypeID |  | Y |
| <rooms> | statusID | INT | <roomStatus>.statusID |  | Y |
| <rooms> | notes | VARCHAR |  |  | N |
| <roomStatus> | statusID | INT |  | Identity | Y |
| <roomStatus> | statusDescription | VARCHAR |  |  | Y |
| <roomTypes> | roomTypeID | INT |  | Identity | Y |
| <roomTypes> | roomType | VARCHAR |  |  | Y |
| <roomTypes> | pricePerNight | MONEY |  |  | Y |
| <customers> | customerID | INT |  | Identity | Y |
| <customers> | firstName | VARCHAR |  |  | Y |
| <customers> | lastName | VARCHAR |  |  | Y |
| <customers> | phoneNumber | VARCHAR |  |  | Y |
| <customers> | addressID | INT | <addresses>.addressID |  | Y |
| <addresses> | addressID | INT |  | Identity | Y |
| <addresses> | addressLine1 | VARCHAR |  |  | Y |
| <addresses> | addressLine2 | VARCHAR |  |  | N |
| <addresses> | city | VARCHAR |  |  | Y |
| <addresses> | provinceID | INT | <provinces>.provinceID |  | Y |
| <addresses> | country | VARCHAR |  | ‘Canada’ | Y |
| <addresses> | postalCode | CHAR |  |  | Y |
| <provinces> | provinceID | INT |  | Identity |  |
| <provinces> | provinceCode | CHAR |  |  | Y |
| <provinces> | provinceName | VARCHAR |  |  | Y |
| <customerBilling> | billID | INT |  | Identity | Y |
| <customerBilling> | billAmount | MONEY |  |  | Y |
| <customerBilling> | reservationID | INT | <reservations>.reservationID |  | Y |
| <customerBilling> | paymentType | VARCHAR |  |  | Y |
| <customerBilling> | amountOwing | MONEY |  |  | Y |
| <chargeableItems> | itemID | INT |  | Identity | Y |
| <chargeableItems> | itemName | VARCHAR |  |  | Y |
| <chargeableItems> | itemDescription | VARCHAR |  |  | Y |
| <chargeableItems> | itemPrice | MONEY |  |  | Y |
| <transactions> | transactionID | INT |  | Identity | Y |
| <transactions> | billID | INT | <customerBilling>.billID |  | Y |
| <transactions> | itemID | INT | <chargeableItems>.itemID |  | Y |
| <transactions> | amountOfItems | INT |  |  | Y |
| <transactions> | date | DATETIME |  |  | Y |
| <staff> | staffID | INT |  | Identity | Y |
| <staff> | firstName | VARCHAR |  |  | Y |
| <staff> | lastName | VARCHAR |  |  | Y |
| <staff> | phoneNumber | VARCHAR |  |  | Y |
| <staff> | addressID | INT | <addresses>.addressID |  | Y |
| <staff> | positionID | INT | <employmentPositions>.positionID |  | Y |
| <staff> | salary | MONEY |  |  | Y |
| <staff> | hiredDate | DATE |  |  | Y |
| <staff> | terminationDate | DATE |  |  | N |
| <staff> | photo | VARCHAR |  |  | N |
| <employmentPositions> | positionID | INT |  | Identity | Y |
| <employmentPositions> | positionTitle | VARCHAR |  |  | Y |
| <employmentPositions> | positionDescription | VARCHAR |  |  | Y |
| <employmentPositions> | startingSalary | MONEY |  |  | Y |
| <users> | userID | INT |  | Identity | Y |
| <users> | username | VARCHAR |  |  | Y |
| <users> | password | VARCHAR |  |  | Y |
| <users> | roleID | INT | <roles>.roleID |  | Y |
| <users> | staffID | INT | <staff>.staffID |  | N |
| <roles> | roleID | INT |  | Identity | Y |
| <roles> | roleTitle | VARCHAR |  |  | Y |
| <roles> | roleDescription | VARCHAR |  |  | Y |

# **Final word**

We hope that the solutions that we have created above are exactly to your specifications and likings. We have taken into consideration every piece of information you have given us and concluded that our research and findings will provide you with an excellent application and database. If you do find anything you would like to change about our report or the application, please do not hesitate to contact us. Do not hesitate to provide any feedback on any of the information we have provided here today.

Thank you for choosing us, and we hope to hear back from you soon.

Best Regards,

Mathew Migliore, Tyle Segovia, & Kevin Romero.

**Dictionary**

* **DBMS:** Short for Database Management System. The system that manages the storage and querying of a database.
* **Query:** A logical request made to the database to return specified results.
* **Mission Statement:** Defines an understanding of what the mission of the project is and what the company wants to accomplish from this project.
* **Mission Objectives:** Defines in a simple format what the objectives of our team are for development in order to deem the project successfully

completed.

* **System Boundary:** Defines the boundaries that are internally in the facility and externally. It also defines the actors, items and other systems that interact with the company’s system.
* **Use Case:** Defines the objects and functions that the actors within the system interact with in each systematic cycle.
* **3NF:** Third Normal form, is the most common used normalization form to design a database which uses principles to reduce the duplication of data, avoid data anomalies, ensure referential integrity, and simplify data management.
* **Functional Dependencies:** defines a relationship between two attributes, typically between the PK and other non-key attributes within a table.
* **Data Dictionary:** a set of information describing the contents, format, and structure of a database and the relationship between its elements, used to control access to and manipulation of the database.
* **Attribute:** In terms of databases it refers to as a field. Used interchangeably with the term “column”
* **Column:** **:** In terms of databases it refers to as a field. Used interchangeably with the term “Attribute”
* **Primary Key:** The field used to Identify a record.
* **Foreign Key:** The field that reference the primary key of another table.