<u>**Database Fundamentals Assignment Part C-**</u>

Gianni Pessotto (13199219) & Daniel Boustani (13207509) Tutorial:Wrk1 Activity:02

<u>1. Revised Business rules and assumptions (From part B)</u>

Customers

- Customers will be shortly identified via a unique ID after creating a booking via email.
- Customers and their details (name, address, phone number etc.) will be recorded and added to the system after the booking is made with an agent or officer.
- Customers need to contact an agent or an office to purchase or rent a property.

Offices

- Office information such as opening time, number, address and other important information should be accessible and easily noticeable for every customer.
- Offices will have both agent and non-agent staff.
- Offices will need to maintain customer information.

Staff

- Staff can't hide or manipulate any information in the databases.
- Staff can be agents and non-agents.
- Staff maintain offices.
- Contains the basic Agents information.

Agents

- Agents information must be accessible at all times for customers.
- Agents are staff.
- Only Agents can update and sell multiple Properties.
- General staff information will be kept in the Staff table.

Bookings

- Booking must have multiple options available to the customers.
- One customer can't make multiple bookings at one time.
- Bookings don't need to be held at the offices.
- Bookings need to be uniquely identifiable.
- Bookings also require relevant entity identifiers.
- Bookings require an agent to be present.

Property

- A property being rented or bought by someone must not be rented to another buyer or tenant.
- Must encompass all information relative to all properties.
- Is a supertype to Buy and Rent tables.
- Agents are responsible for a properties.

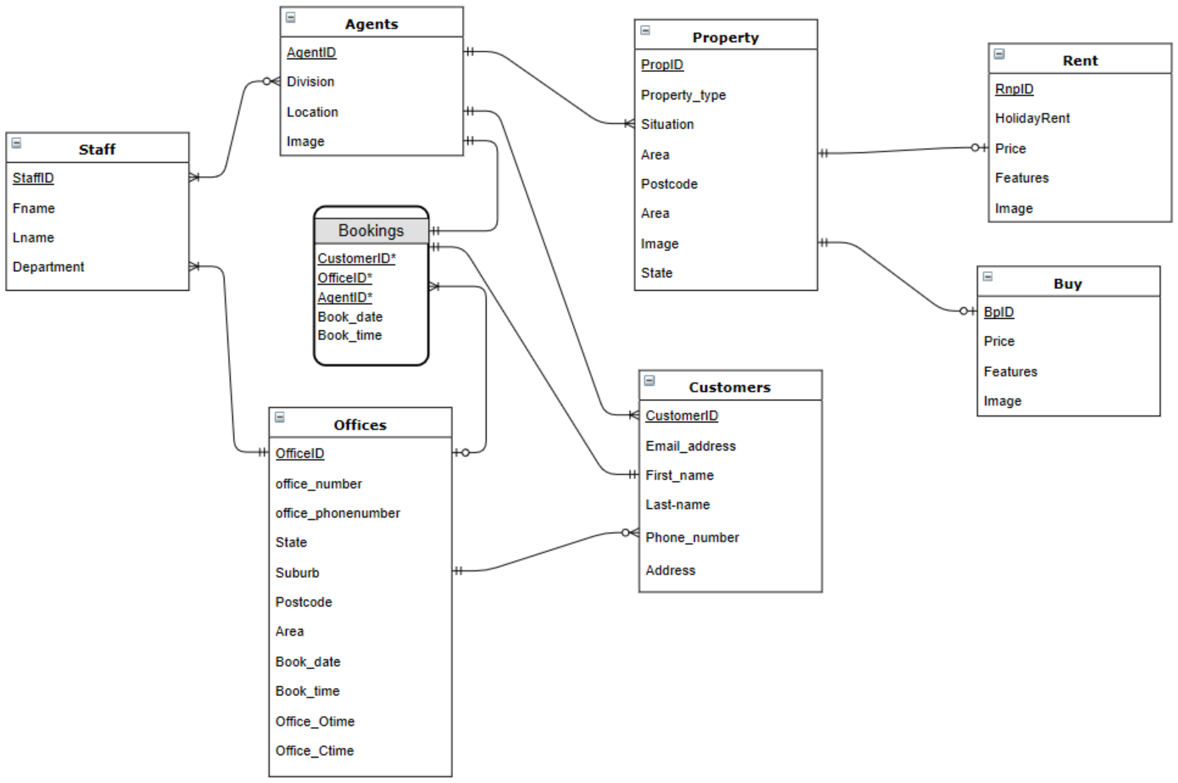
Buy

- Contains records of properties specifically for sale. Must be accurately updated.
- Cannot have entries in the Rent entity table.
- Inherits property information from Property.

Rent

- Contains records of properties specifically for rent. Must be accurately updated.
- Cannot have entries in the Buy entity table.
- Inherits property information from Property.

2. Revised ERD of Part B



---

3. Relations

Staff (StaffID, Fname, Lname, Department)

Agents(AgentID, Division, Location, Image, StaffID*)

StaffID references Staff

Offices(OfficeID, o_number, o_phone, Street, State, Suburb, Postcode, Area, o_open, o_close, StaffID*, AgentID*, CustomerID*)
StaffID references Staff
CustomerID references Customers
AgentID references Agents

Bookings(CustomerID*, OfficeID*, AgentID*, book_date, book_time)
CustomerID references Customers
OfficeID references Office
AgentID references Agent

Property(PropID, property_type, Situation, Area, Street, Postcode, State, Suburb, RnPID*, BPID*, AgentID*)
RPID references Rent
BPID references Buy
AgentID references Agents

Buy(BPID, PropID*, Price, Features, Image)
PropID references Property

Rent(RPID, PropID*, Holidayrent, Price, Features, Image)
PropID references Property

Customers(CustomerID, email_address, first_name, last_name, Phone, Area, Street, Postcode, State, Suburb)

4. List of functional dependencies related to each business rules

StaffID -> Fname, Lname, Department
(Contains all employee information, including Agents.)

AgentID -> Division, Location, Image, StaffID*

(Agents are staff also, why StaffID is present.
All general agent information such as name is kept within staff table to minimise duplication.)

OfficeID -> o_number, o_phone, Street, State, Suburb, Postcode, Area, o_open, o_close, StaffID*, AgentID*, CustomerID*
(CustomerID and StaffID necessary here as staff operate/maintain the offices and customers can potentially make bookings with the office.
AgentID needed also as customers must book with an agent.
Office information is easily accessible for customers with all necessary information.)

Bookings -> CustomerID*, OfficeID*, AgentID*, book_date, book_time
(OfficeID is optional as not necessary for bookings to be completed.
Bookings is uniquely identifiable via composite key and has an agent and a customer present always which are the entity identifiers (CustomerID, AgentID and OfficeID)
Customers can only make one booking at a time also.)

PropertyID -> property_type, Situation, Area, Street, Postcode, State, Suburb, RnPID*, BPID*, AgentID*
(All information regarding the property makes up PropertyID and as such is easily updatable and storable.
Having the attribute situation accounts for what is occuring in regards to the property to avoid for example, being bought twice.
Agents can update and change properties.)

BPID, PropID* -> Price, Features, Image
(Only contains buy specific information so does not appear in the rent table.
Having different identifiers for buying and renting properties ensures no duplicate entries.)

RPID, PropID* -> Holidayrent, Price, Features, Image
(Only contains rent specific information so does not appear in the buy table.
Having different identifiers for buying and renting properties ensures no duplicate entries.)

CustomerID -> email_address, first_name, last_name, Phone, Area, Street, Postcode, State, Suburb
(Addresses that customer information is stored after contact and a unique ID is made for each customer.
Bookings must be made to purchase or rent a property.)

5. Normalisation

a) Functional dependencies:

All relations and their primary keys uniquely identify each row of an entity and have no partial or transitive functional dependencies as all entity attributes rely on the entire primary key only and nothing else. It is already in 3NF.
There are no atomic attributes remaining in the relations and all entities have been separated into their own tables to store their specific information.

b) Justifications:

Staff -
Staff is already in 3NF as both name fields and department rely on StaffID.
This is due to **StaffID -> Fname, Lname, Department,** where no derived or atomic attributes exist, it is a single key and finally there are no interdependent attributes.

Customer-
Customers(CustomerID, email_address, first_name, last_name, Phone, Address)
Customers(CustomerID, email_address, first_name, last_name, Phone, Street, Suburb, State, Postcode)
In 3NF after splitting Address into state, suburb, street etc. as it was an atomic attribute.
Single key ensures no transitive attributes and no interdependencies.

Office-
Offices(OfficeID, o_add, o_number, o_phone, State, Suburb, Postcode, Area, o_open, o_close, StaffID*, AgentID*, CustomerID*)
Offices(OfficeID, o_number, o_phone, Street, State, Suburb, Postcode, Area, o_open, o_close, StaffID*, AgentID*, CustomerID*)
o_add removed as atomic attribute can be split into street, suburb etc.
Everything else is dependent upon the individual office ID such as open/closing times, thus there are no interdependencies or transitive attributes. Only dependent on the single key.

Bookings-
The book date and time are dependent completely on the entire composite primary key, which means there are no partial or transitive functional dependencies. The remaining attributes also depend on the entire key, not one part of the key meaning it is in 3NF.

Agents-
Agent entity contains agent only data, gets specific employee data from Staff table and as such is already in 3NF with only full dependencies.

Property-
Property(<u>PropID,</u> property_type, Situation, Area, Postcode, State, Suburb, Address, Image, RnPID*, BPID*)
Property(<u>PropID,</u> property_type, Situation, Area, Street, Postcode, State, Suburb, Image, RnPID*, BPID*)
Address removed as atomic attribute can be split into street, suburb etc. Entire relation is dependent upon the single key that defines the entity. This creates unique instances only and as a result it is in 3NF.

Buy-
Buy(<u>BPID, PropID*</u>, Price, Features, Image)
Removed all duplicate data such as address and general information that was also inside the property table. The last attributes rely on the single key representing each unique property for sale.

Rent-
Rent(<u>RPID, PropID*,</u> Holidayrent, Price, Features, Image)
Removed all duplicate data such as address and general information that was also inside the property table. The last attributes rely on the single key representing each unique property for rent.

<u>6. Reflection</u>

Prior to this assignment, the Normal Forms and constructing ERDs were known of, but not known how to do. Initially, they were confusing to understand and replicate.
With this assignment, actually utilising the 3 forms of Normal Form has aided in understanding these concepts and their use. We were able to test our understanding

and process of applying these concepts. As a result, we were required to go back and ensure we had learnt the concept well as we must use it for the assignment.
This taught us how to piece ERDs together and make connections between entities whether they be in class, at home or at work as connections exist everywhere, especially within the other applications we use daily.
As for other subjects, it allows better structure in responses in technical subjects but also provides how these ideas also affect non-technical subjects and how to approach them.
For further improvements, better reinforcement in class of concept knowledge and better summaries of information can aid students in consolidating knowledge and making it easier for them to apply it.