Online Hotel Reservation System

Deliverable 2: Relational Model

Name: John Deignan
Course: CS 631-108

The second task in the creation of the online hotel reservation system is to create a relational schema based off of the ER model provided to us. The goal of this phase of the project is to accurately represent what relations must be made between each entity in order for the database to function properly. The relations must be made using the appropriate attributes, some of which are implemented during the creation of the model in order to fulfill all relationships required.

In order to convert from the ER diagram to the Relational Mapping, an algorithm was used as follows:

- For each regular (non-weak) entity type, create a relation which includes all simple (single-valued) attributes of the entity type.
    - Relations 'HOTEL', 'CUSTOMER', 'RESERVATION', 'CREDIT CARD', and 'REVIEW' were created with appropriate attributes and key(s).
- For each weak entity type, create a relation including all attributes of the weak entity type in addition to a foreign key which is the primary key of the weak entity's owner.
    - Relations 'ROOM', 'BREAKFAST', 'SERVICE', 'ROOM_RES', and 'OFFERROOM' were created.
    - Relations 'ROOM', 'BREAKFAST', and 'SERVICE' all include their corresponding simple attributes in addition to a foreign key which refers to the primary key of the 'HOTEL' relation (Hotel_ID). The primary key of each relation is a combination of the foreign key which was added, along with their partial key.
    - Relation 'OFFERROOM' includes its simple attributes in addition to a foreign key which refers to the primary key of the 'ROOM' relation (RNumber). The primary key of this relation is only the foreign key as it is a weak entity and has no partial key.
- For each 1:1 relationship type, include a foreign key in one of the relations which refers to the primary key of the other. This occurs only once throughout the ER diagram.
    - The 'CUSTOMER' relation was fit to include a foreign key which refers to the primary key of 'CREDIT CARD'. Seeing as though both relations have total participation within the relationship, no relation was preferred in the decision as to which relation would receive the foreign key.
    - This allows a customer to now be connected to their corresponding credit card number.
- For each 1:N relationship type, include a foreign key in the entity type at the N-side of the relationship.
    - The 'ROOM', 'BREAKFAST', and 'SERVICE' relations all received foreign keys which refer to the primary key of 'HOTEL' prior to this step, so no further foreign keys were necessary.

  - Similar to the situation above, the ‘ROOM_RES’ relation is of weak entity type and a foreign key was added to refer to the primary key of ‘ROOM’ prior to this step.
  - The ‘ROOM_RES’ relation was fit to include a foreign key (ResInvNumber) which refers to the primary key of ‘RESERVATION’.
  - The ‘RESERVATION’ relation was fit to include a foreign key (ResCustID) which refers to the primary key of ‘CUSTOMER’.
  - The ‘REVIEW’ relation was fit to include a foreign key (Cust_ID) which refers to the primary key of ‘CUSTOMER’.
  - The ‘RREVIEW’, ‘BREVIEW’, and ‘SREVIEW’ relations will all receive a foreign key (RevRNumber, RevBType, RevSType respectively) which refer to the primary key of their corresponding relations when they are created at a later step.
- For each M:N relationship type, create a new relation to represent the relationship between the two entities which includes the primary keys of each participating relation.
  - The ‘INCLUDES’ relation was created with the foreign keys (BTypeInc, InDateInc) which refer to the primary keys of the relations ‘BREAKFAST’ and ‘ROOM_RES’ along with its only attribute (NoOfOrders).
  - The ‘CONTAINS’ relation was created with the foreign keys (STypeCon, InDateCon) which refer to the primary keys of the relations ‘SERVICE’ and ‘ROOM_RES’.
- For each multivalued attribute, create a new relation which includes an attribute corresponding to the attribute in addition to the primary key of its corresponding entity,
  - The ‘HOTEL_PHONE’ relation was created with a foreign key of the primary key from its corresponding entity (Hotel_ID) along with another attribute (PhoneNum). These attributes together make the primary key of the relation.
- For each *n-ary* relationship type, create a new relation to represent the relationship which includes a foreign key to each entity along with any attributes it may already hold.
  - The ‘RESERVES’ relation was created with multiple foreign keys (CNumber, InvoiceNumber, CID) which refer to the primary keys of ‘CREDIT CARD’, ‘RESERVATION’, and ‘CUSTOMER’. The primary key of this relation is ‘InvoiceNumber’ as ‘RESERVATION’ holds a maximum constraint equal to 1.
- For each subclass relationship type, create a new relation which will hold the attributes to each entity type along with a foreign key which refers to the primary of its superclass (Option A).
  - The ‘RREVIEW’, ‘BREVIEW’, and ‘SREVIEW’ relations were created with a foreign key (RID) which refers to the primary key of the superclass (REVIEW) along with all attributes each entity already held. The primary key of each relation is the foreign key which refers to the superclass.

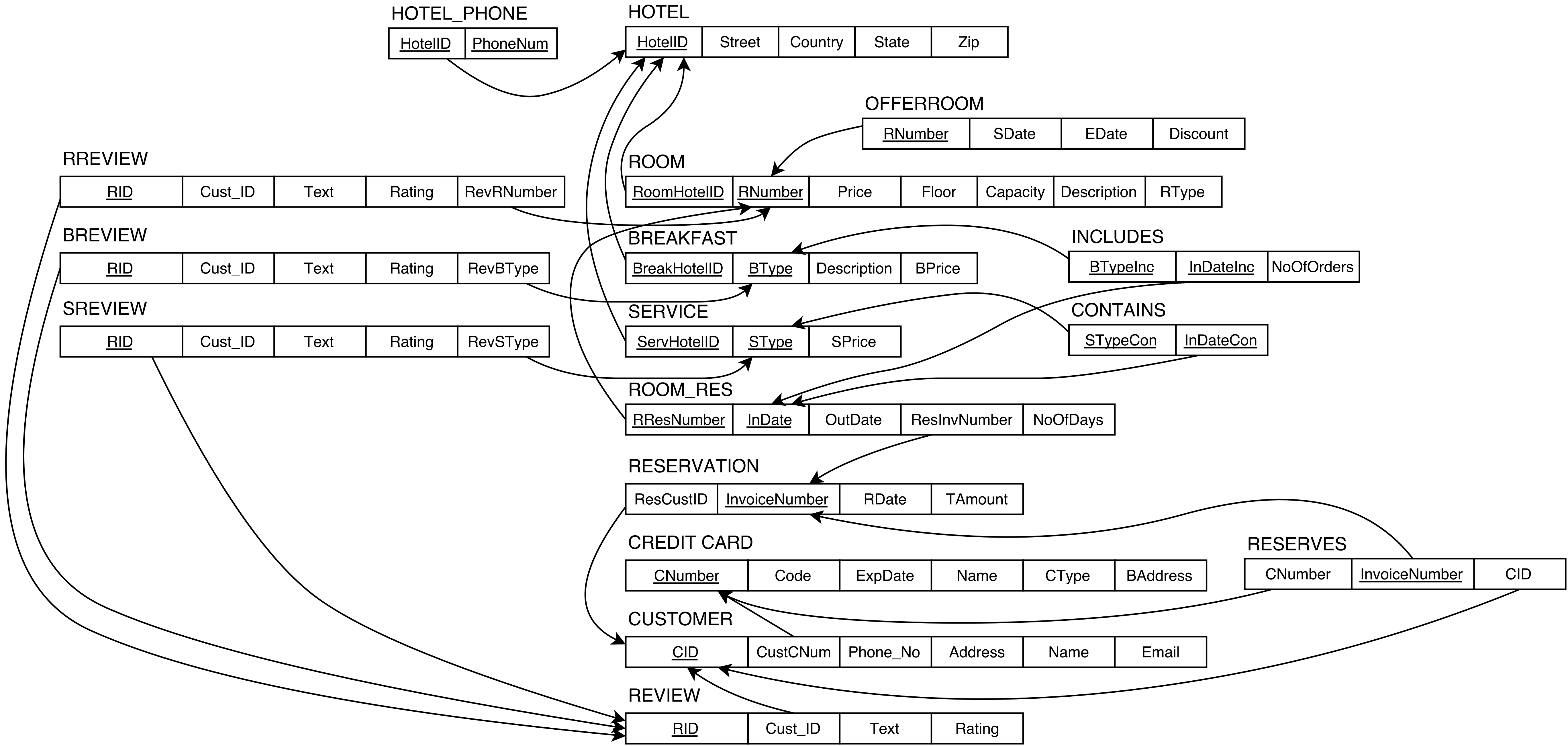
- The ‘OFFERROOM’ relation was created with a foreign key (RNumber) which refers to the primary key of its superclass along with all attributes the entity already held. The primary key of this relation is the foreign key.

Every relation must have a key in order to accurately identify a specific tuple within it. The minimal alternate key(s) of any applicable relation are not shown in the relational mapping and are as follows:

- HOTEL: {Street, Country, Zip}
- CREDIT CARD: Although highly unlikely, any combination of attributes not including the primary key may return more than one credit card. There is a very minute chance that a repeating customer may make a reservation with the same card type that holds the same security code and expiration date. If this possibility is dismissed, a candidate key would have to include all remaining attributes to limit the possibilities of repetition ({Name, BAddress, CType, ExpDate, Code}).
- CUSTOMER: {Email}, {Phone_No}

In the implementation of this database, some foreign key constraints may cause problems due to overlapping. Relations which experience this difficulty are ‘INCLUDES’, ‘CONTAINS’, ‘RESERVES’, ‘ROOM_RES’, and all review subclasses. For example, if a breakfast type was removed, ‘BREVIEW’ would experience a key constraint violation when attempting to cascade all tuples of that breakfast type review.

As always, some difficulties were faced in the completion of this assignment. One of the main issues I experienced was attempting to decide where foreign keys were completely necessary, especially when dealing with conflicting situations such as a weak entity that is on the N side of a 1:N relationship. A weak entity had already received a foreign key to refer toits owner entity, but a 1:N relationship calls for a foreign key to be generated for the N side relation which refers to the primary key of the 1 side relation.


HOTEL_PHONE
HotelID | PhoneNum
HOTEL
HotelID | Street | Country | State | Zip
OFFERROOM
RNumber | SDate | EDate | Discount
RREVIEW
RID | Cust_ID | Text | Rating | RevRNumber
ROOM
RoomHotelID | RNumber | Price | Floor | Capacity | Description | RType
BREVIEW
RID | Cust_ID | Text | Rating | RevBType
BREAKFAST
BreakHotelID | BType | Description | BPrice
INCLUDES
BTypeInc | InDateInc | NoOfOrders
SREVIEW
RID | Cust_ID | Text | Rating | RevSType
SERVICE
ServHotelID | SType | SPrice
CONTAINS
STypeCon | InDateCon
ROOM_RES
RResNumber | InDate | OutDate | ResInvNumber | NoOfDays
RESERVATION
ResCustID | InvoiceNumber | RDate | TAmount
CREDIT CARD
CNumber | Code | ExpDate | Name | CType | BAddress
RESERVES
CNumber | InvoiceNumber | CID
CUSTOMER
CID | CustCNum | Phone_No | Address | Name | Email
REVIEW
RID | Cust_ID | Text | Rating