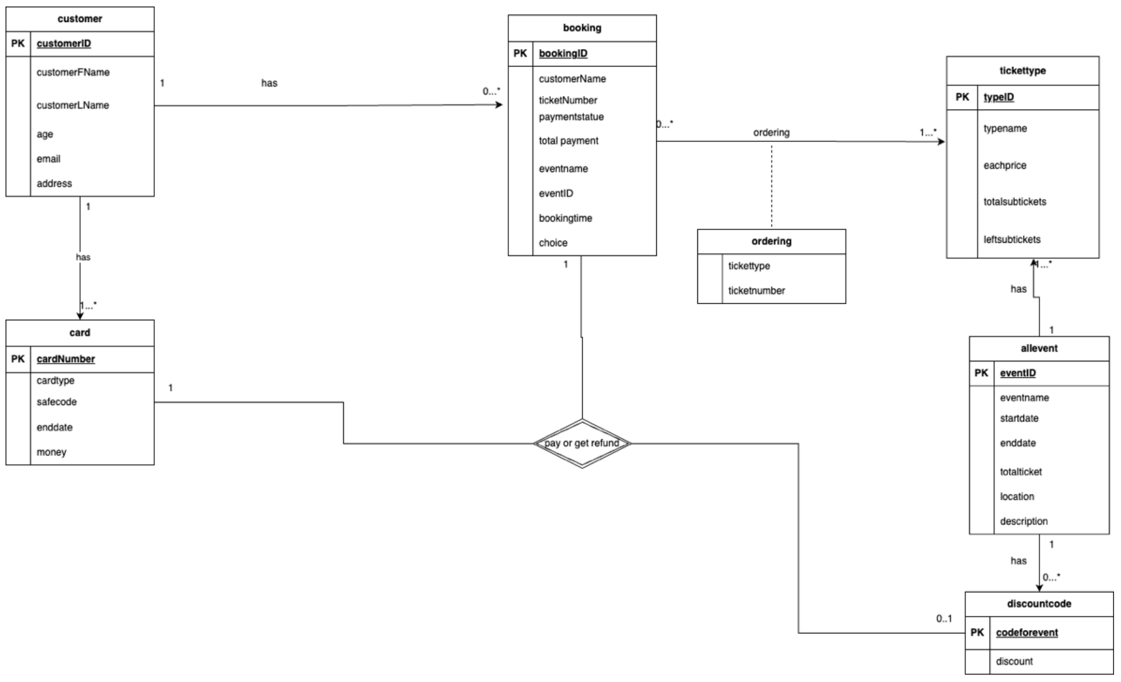
ECM2419 Database course work

Zihan Wang
Student number:720097346

Part 1 ERD and Justification

ERD:



Entities justification:

Customers:

In this entity, I assume there is a primary key customerID which is used to identify the customers who may has same name. And I define other attributes in this entity to represent the information of the customers like the age, email (may be used in book as a ticket sending choice) and address(postcode). This entity aims to satisfy the query of the information of customers.

Booking:

In this entity, I identify the bookingID as the primary key because it may have multiple bookings of one customer to reserve same event on different time. And I also use the eventID and eventname to record the events because the customer can know their booking details, which is more realistic. Then I use the payment statue to record the statue of the payment which can be paid, unpaid or refunded. I also use the total payment to document one booking's payment because the customers may use discount to pay which means we can't calculate the income of an event from the ticket number and the price. To consider one booking may have multiple ticket types so I didn't put ticket type and number in booking. Take care of the eventID and the eventname, I put them into booking which means one booking only can reserve one event. Then there are booking time and choice attributes. The booking time is supposed to be used to compare the start date of event to determine whether it can be cancelled and refunded but after considering the realistic life that people can cancelled the booking at anytime if the date is before the start date of event but the booking time can't be changed as the date going and we can't cancelled the booking when the booking time is 2023/11/2 and the start date of event is 2023/11/20 at today because today is 2023/11/29. So there is no point to compare the booking time and the start date of event, I just use it as a record. The last attribute choice is used as to represent in what way the costumers will take the tickets.

ordering:

The ordering is a relation which contains two attributes ticket type and ticket number. What I just mentioned in the booking entity that one booking may reserve multiple ticket type in one event, but the booking table can't contain multiple ticket types. For example, one book may reserve 2 adult ticket and one children ticket for food festival, so we can't show it on booking table in one row. But we can save the ticket type and ticket number in ordering relation as another table.

tickettype:

This entity uses the typeID to make ticket type unique because different events may have the same ticket type. And this entity also contains the left ticket numbers which also reflect the changes after booking. This entity also contains the price of current ticket type which was used to calculate the total payment of booking.

allevents:

This entity aims to save the information of events which contains the primary key eventID which identify the different events. And there are other attributes like the start date which used to satisfy the query and this attribute can be used to detect whether the booking can be cancelled and refund to the costumers. The location saves the postcode of events and the description shortly describe the events.

discountcode:

This entity contains two attributes. The discountforevent is unique in system which means that the code is unique even if there are lots of events and it can only be used one time. The attribute discount represents the discount, but it is a decimal number.

card:

This entity has a primary key cardnumber which is used to receive the refund and pay the bookings. And it also has cardtype, enddate, safecode and money. I set money in card entity because I want to reflect the booking can be successfully paid and the refund can be sent to card.

Relation justification:

Customer-booking(has):

This relation is a (1) to (0…*) relationship. I think that one customer may has zero bookings or has multiple bookings, but one booking must have one customer because there is no way that the booking hasn't an orderer.

Booking-tickettype(ordering):

This relation is a (0…*) to (1…*) relationship. Because the booking must order tickets whichi means it has at least one ticket type. But for tickettype, there are a possible that no body order this ticket. And the relation has two attributes what I just mentioned in ordering entity justification. Because the booking entity can't have two ticket type in one row then I put them ordering.

allevent-tickettype(has):

This relation is (1) to (1…*) relationship. Because one event must have tickets which means it must have at least one ticket type and this ticket must have price and numbers. But for each ticket type, it belongs to one event.

allevent-discountcode(has):

This relation is (1) to (0…*) relationship. Because one event may have multiple discount code or the event doesn't have discount code for public. And for discountcode, each code must belong to an event.

payorgetrefund:

This relation is card (1), discountcode(0…1) to booking(1) relationship. This relationship should have to action pay and refund. But in my design, I set one card can pay one or more bookings and one booking can refund to its original cards, so the relationship is one to many no matter how between card and booking. So, there is just one line between card and booking. Then I considered that the card may pay the booking by using discount code, but the discount code can only be used in one booking, so I set that one card can only use one discount code to pay one booking, and the one booking can only refund to its paid card.

Part2 Logical model design:

customer (customerID, customerFname, customerLname, age, email, address)

Primary key: customerID

card (cardNumber, safecode, enddate, cardtype, money, onwerID)

Primary key: cardNumber

Foreign key: onwerID reference customer (customerID)

booking (bookingID, customerName, ticketNumber, paymentstatue, totalpayment, eventname, eventID, bookingtime, choice, reserverID)

Primary key: bookingID

Foreign key: reserverID reference customer (customerID)

allevent (eventID, eventname, startdate, enddate, totalticket, location, description)

Primary key: eventID

tickettype (typeID, typename, eachprice, totalsubtickets, leftsubtickets, onwereventID)

Primary key: typeID

Foreign key: onwereventID reference the allevent (eventID)

ordering (tickettype, ticketnumber, orderID, typeID)

Primary key: orderID, typeID

Foreign key: orderID reference booking (booking ID)

Foreign key: typeID reference tickettype (typeID)

discountcode (codeforevent, discount, ownerevent)
Primary key: codeforevent
Foreign key: ownerevent reference allevent(eventID)

payor_getrefund(paidcard, discode, paidbooking)
Foreign key (paidcard) reference card(cardNumber),
Foreign key (discode) reference discountcode(codeforevent),
Foreign key (paidbooking) reference booking(bookingID)