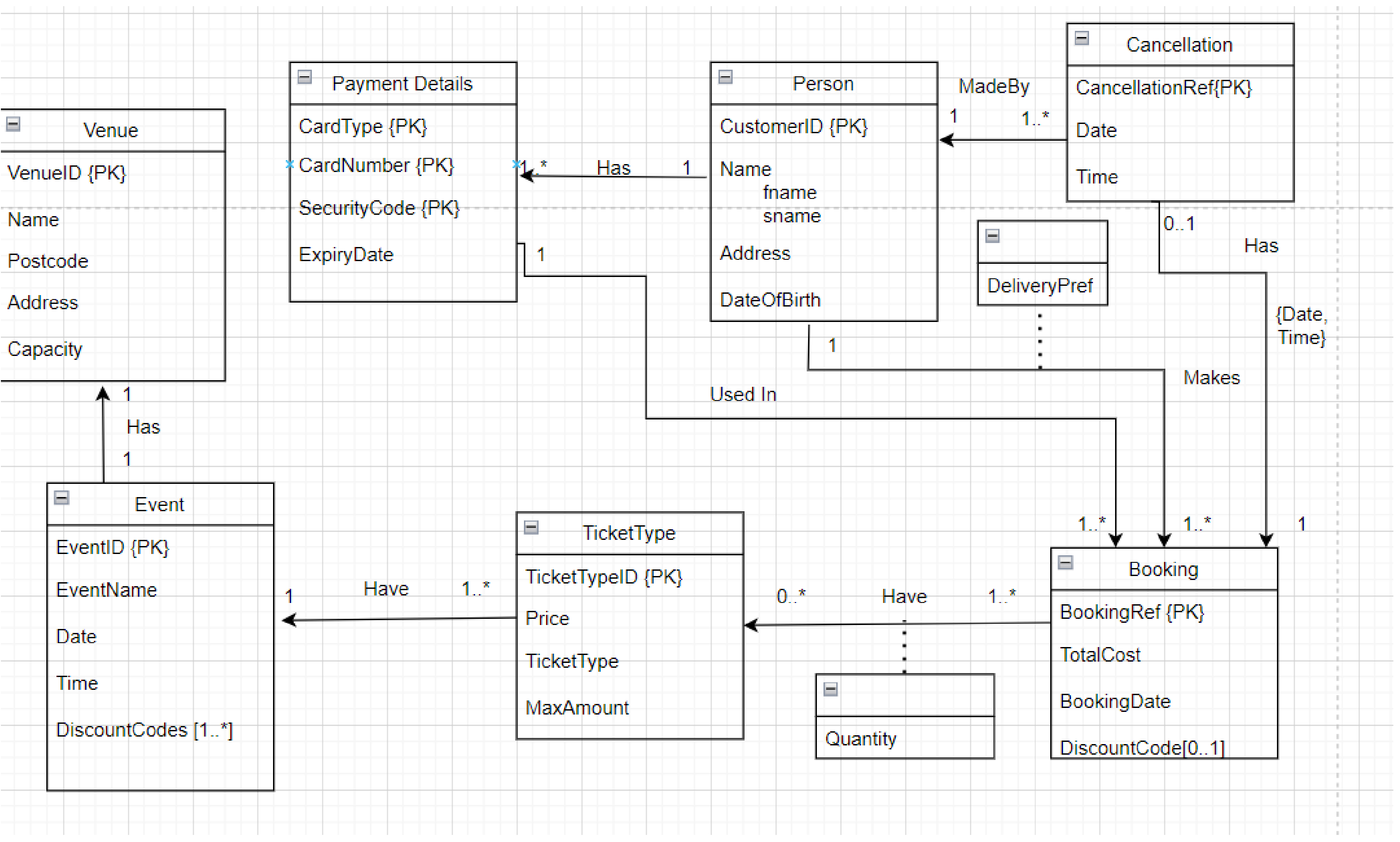
ECM2419 CA

Thomas Shannon

Candidate No: **099864**

The conceptual model Design:



Description of the design:

The Event entity has a primary key EventID identifier that uniquely identifies each occurrence, it also contains information about the event and a multivalued attribute DiscountCodes which contains all discount codes related to that event, this will have to be turned into a new entity when converted to the relational model.

The Event entity has a one-one relationship with the Venue entity as an event would be held at one location on one specific date and time hence it makes sense to use a one-one relationship here. The venue is uniquely identified by its primary key Venue ID which will be assigned to each venue when an occurrence is created. The venue also contains further details about the location and capacity of the venue. In the relational model these two tables will be combined.

The TicketType entity is uniquely identified by the primary key TicketTypeID that will be assigned to each occurrence as they are made. The TicketType entity also contains information about the price, ticket type and max amount of available tickets for this type. This Ticket entity has a many-one relationship with Event as many ticket types will have one specific event related to them.

The Booking entity is uniquely identified by the primary key BookingRef which is created when a person makes a booking. The booking entity also contains the total cost of the booking, date of the booking and the discount code (this has an optional participation value as a customer may not use a discount code). The Booking entity has a many-to-many relationship with the TicketType entity, but the TicketType entity has optional participation as tickets that haven't been sold don't have a booking associated with them. The relationship between booking and TicketType contains a relationship attribute Quantity to define the number of Tickets of each type within the booking, this relationship attribute will become an intermediate entity in the logical model.

The Payment Details entity is uniquely identified by a composite key containing the (CardType, CardNumber, SecurityCode) as this combination will uniquely identify each occurrence. The Payment Details entity also stores the expiry date of the card. This entity has a one-to-many relationship with the Booking entity as one card can be used for many bookings.

The Person entity contains all details about the customer and is identified by the primary key CustomerID which uniquely identifies each occurrence. The entity has a one-many relationship with Payment Details as one person can have multiple credit/debit cards that can be used for payment, while one card must have only one card holder. The Person Entity also has a one-many relationship with Booking as one person can make many bookings for different events whereas one booking must have a single person it was made by. The relationship between Person and Booking has a relationship attribute DeliveryPref which is used to identify whether the customer wants their tickets to be delivered or collected.

The Cancellation Entity Is uniquely identifiable by the CancellationRef which will be identified alongside the Date and Time attributes when an occurrence is made. The Date and Time are used as a constraint and will be checked alongside the Event Date and Time attributes to see if the cancellation request is valid. The Cancellation Entity has a one-one relationship with the Booking entity as one cancellation must be associated with one booking but it also has optional participation as a booking is not always cancelled. The Cancellation Entity also has a many-one relationship with person as many cancellations can be made by one person. This cancellation table will not need to be used in the relational model as when implemented in SQL the entities will just be removed from the booking table making there no need to store the data in the cancellation table as this will create issues with foreign key constraints.

The overall structure of the model avoids both chasm and fan traps which makes the process of converting it into the logical model design much easier and avoids complications with accessing data in the final database.

The logical model

Event(EventID, EventName , Date, StartTime,EndTime, VenueName ,Postcode, Address, VenueCapacity,Details)

PrimaryKey: EventID

DiscountCodes(Code, DiscountAmount, ExpirationDate, EventID**)**

Primary Key: Code

Foreign Key: EventID references Event(EventID)

TicketType(TicketTypeID, Price, TicketType, MaxAmount, Event ID**)**

Primary Key: TicketTypeID

Foreign Key: EventID references Event(EventID)

BookingTicketType(BookingRef, TicketTypeID,Quantity**)**

Primary Key: BookingRef, TicketTypeID

Foreign Key: BookingRef references Booking(BookingRef)

Foreign Key: TicketTypeID refences TicketType(TicketTypeID)

Booking(BookingRef, TotalCost, BookingDate, Code, CardType, CardNumber, SecurityCode, CustomerID, DeliveryPref**)**

Primary Key: BookingRef

Foreign Key : Code references DiscountCode(Code)

Foreign Key : CardType references PaymentDetails (CardType)

Foreign Key : CardNumber references PaymentDetails (CardNumber)

Foreign Key : SecurityCode references PaymentDetails (SecurityCode)

Foreign Key: CustomerID references Person(CustomerID)

PaymentDetails(CardType, CardNumber, SecurityCode, ExpiryDate, CustomerID **)**

Primary Key: CardType, CardNumber, SecurityCode

Foreign Key: CustomerID references Person(CustomerID)

Person(CustomerID, Fname, Sname, Address, DateOfBirth**)**

Primary Key: CustomerID