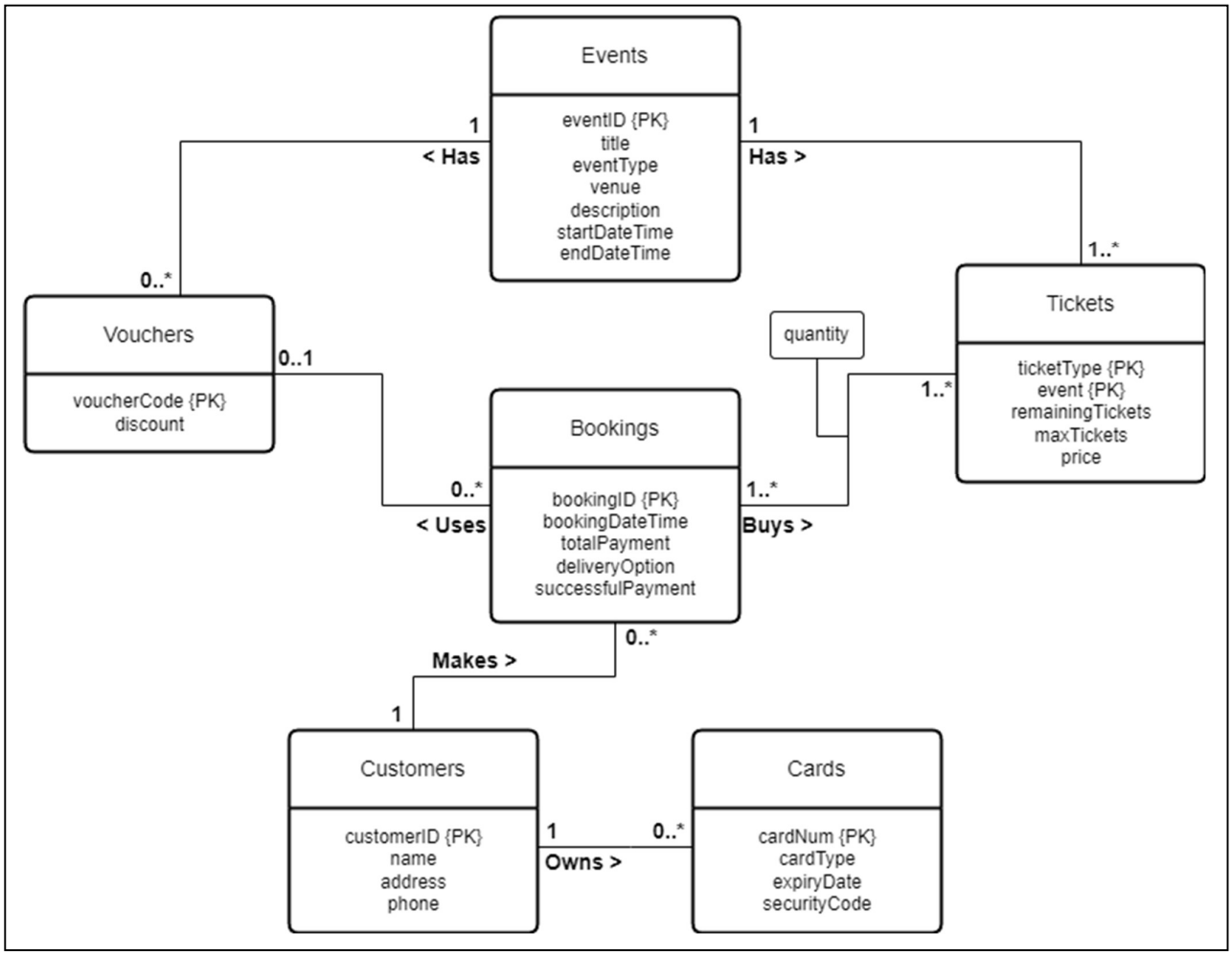
ECM2419 – Database Theory and Design – Jacob Braidley

ERD Diagram

An Entity Relationship Diagram (ERD) using UML notation.



Entities

Events

The Events entity is used to define each event within the database. Its primary key is eventID, which is a unique auto incremented field of INT datatype. This primary key is required since all other fields are not necessarily unique, so eventID provides a way to access each distinct record in Events.

The title, eventType, venue and description attributes in this entity are all VARCHAR(255) datatypes used to provide descriptive information about each event. The startDateTime and endDateTime attributes, of the DATETIME datatype, provide the date/time for the start and end of an event, respectively. Using the DATETIME datatype removes the need to have separate attributes for date and time.

Vouchers

Vouchers is an entity that defines the vouchers used on bookings to provide customers with discounts. Its primary key is voucherCode, which is a unique code of VARCHAR(255) datatype. This is the code that is referred to by the customer when they apply the voucher. Since each voucher should have unique codes, this can be used as a primary key without the need for an auto incremented id field of INT datatype.

The discount attribute, of INT datatype, is used to provide a discount when the voucher is applied. It is stored as an INT rather than a FLOAT to provide easier readability of the database when querying. However, this means it must be divided by 10 when calculating the discount (e.g., 0.2 would be used when discounting 20%).

Tickets

The Tickets entity defines the different tickets available for each event.

Its primary key is a composite key consisting of ticketType and event, the latter being a foreign key that references Events(eventID). ticketType is of VARCHAR(255) datatype and event is of INT datatype. Since both ticketType and event may not be unique separately, they are combined into a composite key. This allows for a unique reference to each type of ticket for each event.

The remainingTickets and maxTickets attributes, both being the INT datatype, are used to keep track of how many tickets are yet to be sold and how many tickets in total can be sold, respectively for each ticket type for an event. These attributes are accessed when customers make orders, in order to ensure the tickets can be bought (i.e., not sold out). The attribute price is a FLOAT (limited to two digits after the decimal) that provides the price for each specific ticket.

Bookings

Bookings is an entity defining the transactions by customers when buying tickets.

bookingID is its primary key, being an auto incremented INT value that provides a unique reference to each record in Bookings. Since multiple bookings of the same tickets and events can be made, this primary key is required to allow for a unique field in every record.

bookingDateTime allows the date and time, of when the booking is made, to be stored in the table. It is of the DATETIME datatype in order to group both date and time into one field. The FLOAT attribute totalPayment provides the total cost of the booking (the sum of the price of all tickets bought). If a voucher is used in the booking, the amount of the discount is taken into consideration when calculating the value of totalPayment. deliveryOption is a VARCHAR(255) attribute that specifies how the customer wants to receive their tickets, through ‘email’ or ‘pick-up’. successfulPayment is an attribute of BOOL (where 0 represents false, and positive integers, typically 1, represent true) that declares whether the payment of the booking was successful or not.

Customers

The Customers entity defines each customer that is able to buy tickets for events in the database.

customerID is used as the primary key. It is an auto incremented INT that provides a unique reference to each record in Customers. Since multiple customers can have the same details (e.g., name or address), an id type primary key is required.

The three attributes in the Customers entity are all of VARCHAR(255) datatype. The attributes are used to provide information about each customer in the database (i.e., their name, address, and phone number).

Cards

Cards is an entity defining a customer’s card that can be used to pay bookings.

Its primary key is cardNum, which defines the card number on the card. Since each card number is required and must be different for every card, the cardNum attribute will be unique and not null, making it suitable to be the entity’s primary key.

cardType, expiryDate, and securityCode are all VARCHAR(255) attributes that provide information about each card in the database.

Quantity

For each booking of tickets, the amount bought must be defined. This creates the quantity associative entity, avoiding the many-to-many relationship that would be experienced by Tickets and Bookings otherwise. Since it is an associative entity, it does not have a primary key. The associative entity is made into a table TicketsInBookings in the relational schema below.

Relationships

Events HAS Vouchers

The Events to Vouchers relationship is that of one-to-many, having multiplicity constraints of 1 and 0..*. The Events entity has mandatory participation, whilst Vouchers has optional; an event can have either none, one or multiple vouchers associated with it, however a voucher will only be associated to one event.

Events HAS Tickets

The Events to Tickets relationship is that of one-to-many, having multiplicity constraints of 1 and 1..*. Both the Events and Tickets entities have mandatory participation; an event will have at least one ticket for it and each ticket will be for exactly one event.

Bookings USES Vouchers

The Bookings to Vouchers relationship is that of one-to-many, having multiplicity constraints of 0..* and 0..1. Both the Bookings and Vouchers entities have optional participation; a booking can use either none or one voucher and a voucher can be used on either none, one or many bookings.

Bookings BUYS Tickets

The Bookings to Tickets relationship is that of many-to-many, having multiplicity constraints of 1..* and 1..*. Both the Bookings and Tickets entities have mandatory participation; a booking must have at least one ticket and a ticket will be used in at least one booking.

Due to the many-to-many relationship, an associative entity is formed named quantity. This provides the amount of tickets bought within a booking, resolving the many-to-many relationship. In the database, this will be its own table.

Customers MAKES Bookings

The Customers to Bookings relationship is that of one-to-many, having multiplicity constraints of 1 and 0..*. The Customers entity has mandatory participation, whilst Bookings has optional; a customer can make either none, one or multiple bookings, however a booking must be made by exactly one customer.

Customers OWNS Cards

The Customers to Cards relationship is that of one-to-many, having multiplicity constraints of 1 and 0..*. The Customers entity has mandatory participation, whilst Cards has optional; a customer can own either none, one or multiple cards, however a card must be owned by exactly one customer.

Relational Schema

A relational schema designed from the ERD diagram above.

Events (<u>eventID</u>, title, eventType, venue, description, startDateTime, endDateTime)
- PK: eventID

Vouchers (<u>voucherCode</u>, discount, event)
- PK: voucherCode
- FK(s):
  - event – references **Events** (eventID)

Tickets (<u>ticketType</u>, <u>event</u>, remainingTickets, maxTickets, price)
- PK: ticketType, event
- FK(s):
  - event – references **Events** (eventID)

Customers (<u>customerID</u>, name, address, phone)
- PK: customerID

Cards (<u>cardNum</u>, cardType, expiryDate, securityCode, customer)
- PK: cardNum
- FK(s):
  - customer – references **Customers** (customerID)

Bookings (<u>bookingID</u>, bookingDateTime, totalPayment, deliveryOption, successfulPayment, customer, voucher)
- PK: bookingID
- FK(s):
  - customer – references **Customers** (customerID)
  - voucher – references **Vouchers** (voucherCode)

TicketsInBookings (<u>booking</u>, <u>event</u>, <u>ticketType</u>, quantity)
- PK: booking, event, ticketType
- FK(s):
  - booking – references **Bookings** (bookingID)
  - event, ticketType – references **Tickets** (event, ticketType)