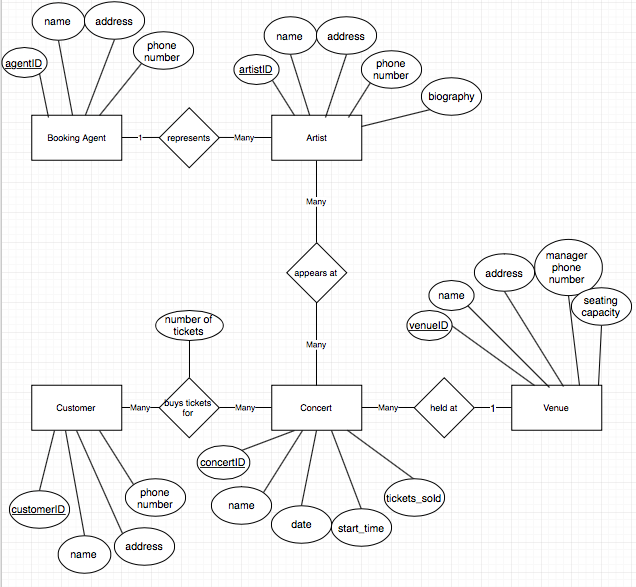
**CSC 8008 Database Assignment – Jeremy Williams**

**Question 1.**

****

**Relationship Assumptions**

In the above design I have made the following assumptions regarding the relationships present:

* A single booking agent can represent many artists.
* A single artist can be represented by at most one booking agent.
* A single artist can appear at many concerts.
* A single concert can have many artists appearing at it.
* A single customer can book many tickets to a concert.
* A single concert can have many customers booked to see it.
* A single concert can be held at one venue.
* A single venue can hold many concerts.

**Further Assumptions**

* Each entity will be given its own ID which will form its primary key
* No participation constraints (i.e. total participation) have been imposed to ensure data flexibility. Some could be imposed in a further design if needed. An example might be that every concert must be held at a venue.
* No weak entities have been used. I considered making Booking Agent a weak entity dependent on Artist but decided not to use this approach due to the fact that a Booking Agent can exist in his/her own right, has lots of attributes and that there would be lots of data duplication.

**Question 2.**

SQL Commands to create the database:

CREATE TABLE BookingAgent

(bookingAgentId INTEGER NOT NULL,

name VARCHAR(20) NOT NULL,

address VARCHAR(30) NOT NULL,

phone VARCHAR(15) NOT NULL,

PRIMARY KEY (bookingAgentId))

ENGINE=InnoDB;

CREATE TABLE Artist

(artistId INTEGER NOT NULL,

name VARCHAR(20) NOT NULL,

address VARCHAR(30) NOT NULL,

phone VARCHAR(15) NOT NULL,

biography VARCHAR(200) NOT NULL,

bookingAgentId INTEGER,

PRIMARY KEY (artistId),

FOREIGN KEY (bookingAgentId) REFERENCES BookingAgent(bookingAgentId))

ENGINE=InnoDB;

CREATE TABLE Venue

(venueId INTEGER NOT NULL,

name VARCHAR(20) NOT NULL,

address VARCHAR(30) NOT NULL,

manager\_phone\_number VARCHAR(15) NOT NULL,

seating\_capacity INTEGER NOT NULL,

PRIMARY KEY (venueId))

ENGINE=InnoDB;

CREATE TABLE Concert

(concertId INTEGER NOT NULL,

name VARCHAR(20) NOT NULL,

date VARCHAR(20) NOT NULL,

start\_time VARCHAR(15) NOT NULL,

tickets\_sold INTEGER NOT NULL,

venueId INTEGER,

PRIMARY KEY (concertId),

FOREIGN KEY(venueId) REFERENCES Venue(venueId))

ENGINE=InnoDB;

CREATE TABLE ConcertArtist

(concertId INTEGER NOT NULL,

artistId INTEGER NOT NULL,

PRIMARY KEY (concertId, artistId),

FOREIGN KEY (concertId) REFERENCES Concert(concertId),

FOREIGN KEY (artistId) REFERENCES Artist(artistId))

ENGINE=InnoDB;

CREATE TABLE Customer

(customerId INTEGER NOT NULL,

name VARCHAR(20) NOT NULL,

address VARCHAR(30) NOT NULL,

phone VARCHAR(15) NOT NULL,

PRIMARY KEY (customerId))

ENGINE=InnoDB;

CREATE TABLE Ticket

(ticketTransactionNumber INTEGER NOT NULL,

numberOfTicketsPurchased INTEGER NOT NULL,

concertId INTEGER NOT NULL,

customerId INTEGER NOT NULL,

PRIMARY KEY (ticketTransactionNumber),

FOREIGN KEY (concertId) REFERENCES Concert(concertId),

FOREIGN KEY (customerId) REFERENCES Customer(customerId))

ENGINE=InnoDB;

**Notes on implementation changes from original diagram design**

When I came to implement my design in MySQL, I made one small change. In my diagram, I show a many to many relationship between the entity ‘Customer’ and the entity ‘Concert’. This is because a single customer can buy many tickets to a concert and a concert can have many customers buying tickets for it. I originally planned to make this relationship into a join table with one extra attribute for the number of tickets purchased. It would have had a primary key composed of the customerId and the concertId.

When I came to implement this, however, I realised that if a customer purchased a ticket for a Concert and then at a later date came back and purchased any more tickets for the same Concert, this would cause a problem within the database. A duplicate entry for the customerId-concertId primary key would not be allowed. To avoid this problem, I decided to model the relationship using another entity. I have created the entity ‘Ticket’ to represent this. It has its own primary key ‘ticketTransactionNumber’, which will avoid the problem mentioned above.

In order to insert data into the above tables, I created .SQL files and uploaded them:

INSERT INTO BookingAgent VALUES

(1, 'Virgin Records', '32 Paradise st, New York', '07867445334'),

(2, 'Simon Cowell', '21 Star Avenue, California', '07889667556'),

(3, 'Warner Brothers', '45 Super St, Washington', '07554332554'),

(4, 'Sony Music', '1 Tokyo rd, Tokyo', '07667564234'),

(5, 'Universal', '5 Kissimee Dr, Orlando', '07897765879')

;

INSERT INTO Artist VALUES

(1, 'Beatles', '24 London st, London', '07765299100','Cool old tunes',1),

(2, 'Rolling Stones', '56 Rock Avenue, California', '07006956386','Rockin and rollin',4),

(3, 'Arctic Monkeys', '1 Freeze Lane, Birmingham', '07883662112','Super cool',3),

(4, 'Nirvana', '4 Dream Lane, New York', '07907876234','Grunge rock tunes',1),

(5, 'ACDC', '5 Metal rd, Seattle', '07846132089','Super classic rock',2),

(6, 'Metallica', '7 Machine rd, Chicago', '07667889977','Metal in your face',5),

(7, 'Iron Maiden', '8 Cannock rd, London', '07123554422','Can you feel the rock',3),

(8, 'Nickelback', '36 Treehouse rd, Manchester', '07098667788','Canadian rock legends',1),

(9, 'Little Mix', '3 Westgate rd, Newcastle', '07123067456','Pop girls from Newcastle',3),

(10, 'Royksopp', '88 Norway Lane, Norway', '07098446876','Norwegian chilled music',4)

;

INSERT INTO Venue VALUES

(1, 'Civic Centre', '1 Newcastle St, Newcastle', '07634123321',10000),

(2, 'Sage', '44 Westgate rd, Newcastle', '07765677098',2000),

(3, 'Radio Arena', '34 High st, Newcastle', '07345877777',3500)

;

INSERT INTO Concert VALUES

(1, 'Rockathon', '1st December', '8pm', 100, 1),

(2, 'Chillzone', '3rd January', '9pm', 400, 2),

(3, 'AnthemWorld', '5th June', '7pm', 500, 3),

(4, 'Lollapalooza', '10th September', '8pm', 350, 1),

(5, 'RockoutMetalNight', '30th August', '7pm', 70, 2)

;

INSERT INTO ConcertArtist VALUES

(1, 1),

(1, 2),

(1, 3),

(2, 4),

(3, 5),

(3, 6),

(4, 1),

(4, 2),

(5, 1),

(5, 7),

(5, 8),

(5, 9),

(5, 10)

;

INSERT INTO Customer VALUES

(1, 'Bob Jenkins', '34 Newquay rd, Newcastle', '07457698034'),

(2, 'Dave Watts', '3 Great rd, Sunderland', '07845227586'),

(3, 'Will Davies', '5 Low Lane, Durham', '07234995566'),

(4, 'Clare Roberts', '78 High rise, Manchester', '07879108597'),

(5, 'Kate Smith', '45 Ridge way, Newcastle', '07987108456')

;

INSERT INTO Ticket VALUES

(1, 1, 2, 3),

(2, 2, 2, 1),

(3, 1, 3, 4),

(4, 4, 1, 2),

(5, 1, 4, 2),

(6, 2, 4, 2)

;

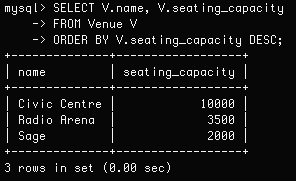
**Question 3.**

A)

SELECT V.name, V.seating\_capacity

FROM Venue V

ORDER BY V.seating\_capacity DESC;

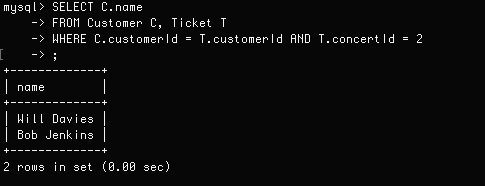


B)

SELECT C.name

FROM Customer C, Ticket T

WHERE C.customerId = T.customerId AND T.concertId = 2;



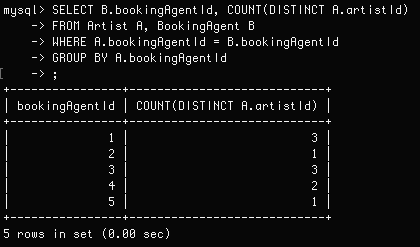
C)

SELECT B.bookingAgentId, COUNT(DISTINCT A.artistId)

FROM Artist A, BookingAgent B

WHERE A.bookingAgentId = B.bookingAgentId

GROUP BY A.bookingAgentId;

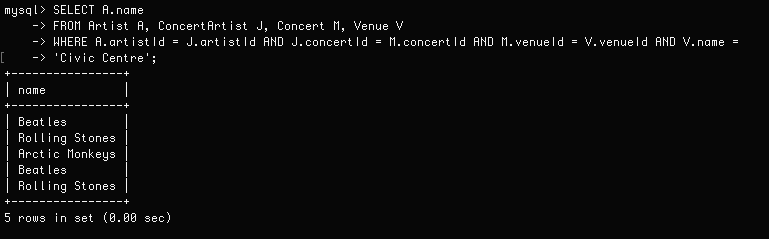


D)

SELECT A.name

FROM Artist A, ConcertArtist J, Concert M, Venue V

WHERE A.artistId = J.artistId AND J.concertId = M.concertId AND M.venueId = V.venueId AND V.name = ‘Civic Centre’;



**NB** 2 artists (Beatles and Rolling Stones) are displayed twice as they both play at 2 separate concerts at the Civic Centre venue.

**Question 4.**

You could find out the seating capacity at the venue that concert 1 is being held at.

You could then find out the total number of tickets sold to customers for concert 1.

Then subtract the amount of tickets sold for concert 1 from the seating capacity of its venue.

SELECT V.seating\_capacity - C.tickets\_sold AS TicketsRemaining

FROM Venue V, Concert C

WHERE C.venueId = V.venueId AND C.concertId = 1;

