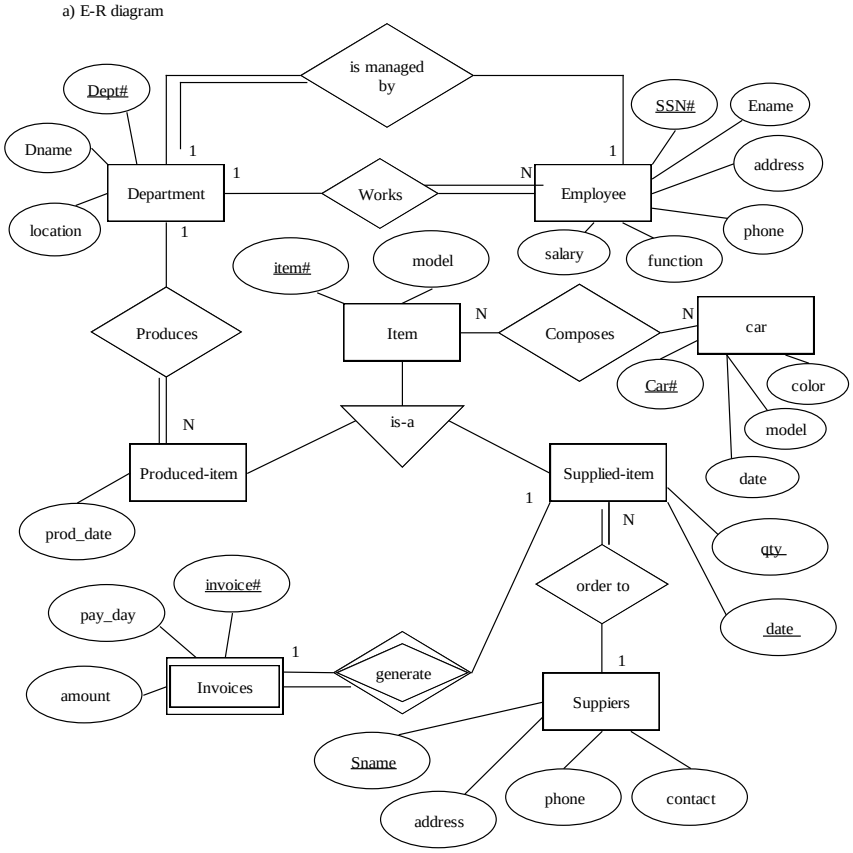
INM370 – Advanced Databases

Tutorial 1 – ER-Diagram, Relational DB Modelling and SQL

Model Answers

1. ER Diagram and Relational Scheme

a) E-R diagram



b) Department (Dept#, Dname, location, *SS#*)
Employee (SS#, Name, address, phone, function, salary, *Dept#*)
Item (Item#, model)
Produced-item (Item#, prod_date, *Dept#)*
Supplied-item (Item#, QTY, Date, *Sname, Invoice#*)
Car (Car#, color, model, date)
Invoices (invoice#, pay_day, amount)
Suppliers (Sname, address, phone, contact)
Composes (Item#, car#)

ref.: primary keys and *foreign keys*

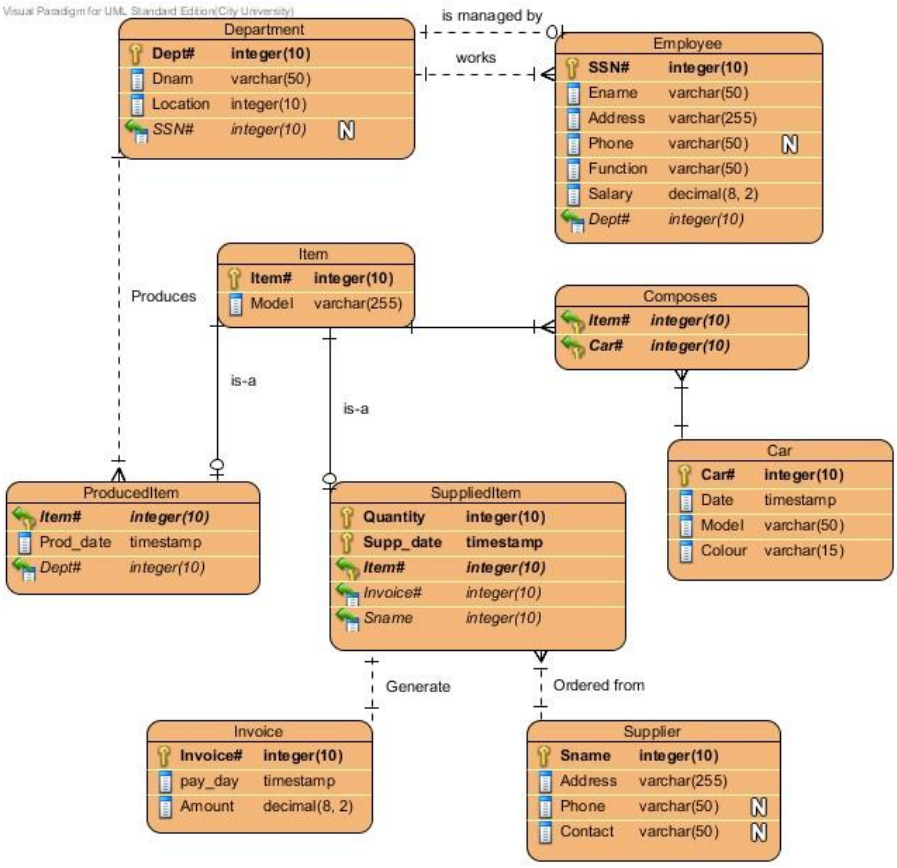
Commented [SV1]: Since Invoice appears to be a Weak entity type, should not Invoice have an FK to point to Supplied-Item, and not vice-versa?

The question stated: "An invoice is generated for each request of an item.". Thus one Invoice for one Supplied-Item. Supplied-Item is in Total participation with Invoice: all Supplied-Items have a corresponding Invoice. Therefore, FK can be in Supplied-Item, too

On the other hand, maybe the "weakness" of Invoice entity type should be removed?! But Invoice cannot exist without Supplied-Item!

IMPORTANT: There is no standard notation for ER diagrams; many notations exist, such as original Peter Chen's[1], Crow's Foot, UML-based, etc. Mostly, the UML-based notation has been used in Lecture 1 slides (some of you might know of, or have used, UML in other contexts, and some might have dealt with it in other modules, e.g. INM330, or UG Stage 4 students in IN2013), but different notations will be used throughout the module – you ought to know them! Additional explanation about different notations are given in the Preface of the module's textbook (p.36), and the Appendix C. Similarly, there is a plethora of tools for drawing ER diagrams.
NOTE: This is a model answer, and like in other (database) modelling tasks, in principle, other solutions (at least in part) are possible.

An alternative solution for E-R diagram, using "Crow's Foot" notation



[1] Chen, P.P.-S., *The Entity-Relationship Model - toward a Unified View of Data*. ACM Trans. Database Syst., 1(1): p. 9-36. 1976.
Chen, P., *Entity-Relationship Modeling: Historical Events, Future Trends, and Lessons Learned*, in *Software Pioneers*, B. Manfred and D. Ernst, Editors, Springer-Verlag New York, Inc. p. 296-310, 2002.

NB: ER diagrams are not meant to resolve many-to-many relationships. Some software tools do that, however. The tool used for the above diagram is Visual Paradigm (a version of which is available in the labs), which automatically resolves many-to-many relationships, e.g. "Composes".
Many ER diagramming tools exist: ERwin, Microsoft Visio, Rational Rose, System Architect, Visual Paradigm etc. Some free software diagramming tools just draw the ER notational shapes, e.g., Dia (part of the GNOME project office suite), or www.draw.io. These kinds of tools usually do not enforce any semantic rules (let alone generate SQL).

2. SQL

a) List all double or family rooms with a price below £40.00 per night, in ascending order of price.

```
SELECT *
FROM Room
WHERE price < 40 AND type IN ('D', 'F')
ORDER BY price;
```

NOTE: An assumption is that the column `type (CHAR(1))` stores data in the given format/datatype; other options are possible – this is a model answer. State what data type you assumed in your answer.

b) List the bookings for which no dateTo has been specified.

```
SELECT *
FROM Booking
WHERE dateTo IS NULL;
```

c) List the number of rooms in each hotel.

```
SELECT hotelNo, COUNT(roomNo) AS roomCount
FROM Room
GROUP BY hotelNo;
```

d) Create a view containing the guests who are from United Kingdom

```
CREATE VIEW GuestFromUK
AS
      SELECT *
      FROM Guest
      WHERE guestAddress like "%United Kingdom%" OR
            guestAddress like "%UK%"
```

3.
a)[2]

```
CREATE TABLE Play (
  PlayNum VARCHAR(5) NOT NULL,
  Title VARCHAR(30),
  Period VARCHAR(15),
  Type VARCHAR(15),
  Writer VARCHAR(25),
  PRIMARY KEY (PlayNum)
);

CREATE TABLE Actor (
  ActNum VARCHAR(5) NOT NULL,
  Name VARCHAR(20),
  Address VARCHAR(45),
  Type VARCHAR(10),
  PRIMARY KEY (ActNum)
);

CREATE TABLE Participate (
  PlayNum VARCHAR(5) NOT NULL,
  ActNum VARCHAR(5) NOT NULL,
  PlayDate DATE NOT NULL,
  StartTime VARCHAR(5),
  PRIMARY KEY (PlayNum, ActNum, PlayDate),
  FOREIGN KEY (PlayNum) REFERENCES Play,
  FOREIGN KEY (ActNum) REFERENCES Actor
);

CREATE TABLE Ticket (
  SeatNum VARCHAR(4) NOT NULL,
  PlayDate DATE NOT NULL,
  PlayCode VARCHAR(5) NOT NULL,
  StartTime VARCHAR(5),
  Price NUMBER(4),
  PRIMARY KEY (SeatNum, PlayDate, PlayCode),
  FOREIGN KEY (PlayCode) REFERENCES Play
);
```

b) Some examples of INSERT operations:

```
INSERT INTO Play VALUES
('P11', 'Madamme Butterfly', '19:30', 'Spring 2019', 'Opera', 'G.
Puccini');

INSERT INTO Actor VALUES
('A1', 'Mary White', '15 High Street London SW1', 'Opera');

INSERT INTO Participate VALUES
```

[2] Use of other, but suitable, data types is possible.

```
('P11', 'A1', '10-Jun-2019', '19:30');

INSERT INTO Ticket VALUES
('A12', '10-Jun-2019', 'P11', '19:30', 50,);
```

Etc.

c)

i)

```
SELECT *
FROM   Play
WHERE  Play.Title LIKE '%Family%';
```

ii)

```
SELECT     SUM(Ticket.Price)
FROM       Ticket
WHERE      Ticket.PlayCode = 'P12'
AND        Ticket.PlayDate = '20-Jun-2019';
```

iii)

```
SELECT Actor.Name
FROM   Play, Actor, Participate
WHERE  Play.PlayNum = Participate.PlayNum AND
       Actor.ActNum = Participate.ActNum AND
       Participate.PlayDate = '20-Jun-2019' AND
       Play.Writer = 'Shakespeare';
```

iv)

```
CREATE VIEW GoodSeats AS
(
  SELECT SeatNum[3]
  FROM Ticket
  WHERE Ticket.Price >= 25
  AND Ticket.PlayCode IN
     (SELECT PlayNum
      FROM   Play
      WHERE  Play.title = 'Madame Butterfly')
);
```

4. See the model answers for Q 3) and come up with analogous solutions.

[3] An alternative is possible that uses JOIN operator of the SQL language. More generally, remember that SQL allows for alternative solutions: SQL operations can be re-phrased into logically equivalent ones, i.e. the same results can be obtained using different language constructs.