**Database Management**

**Final Exam**

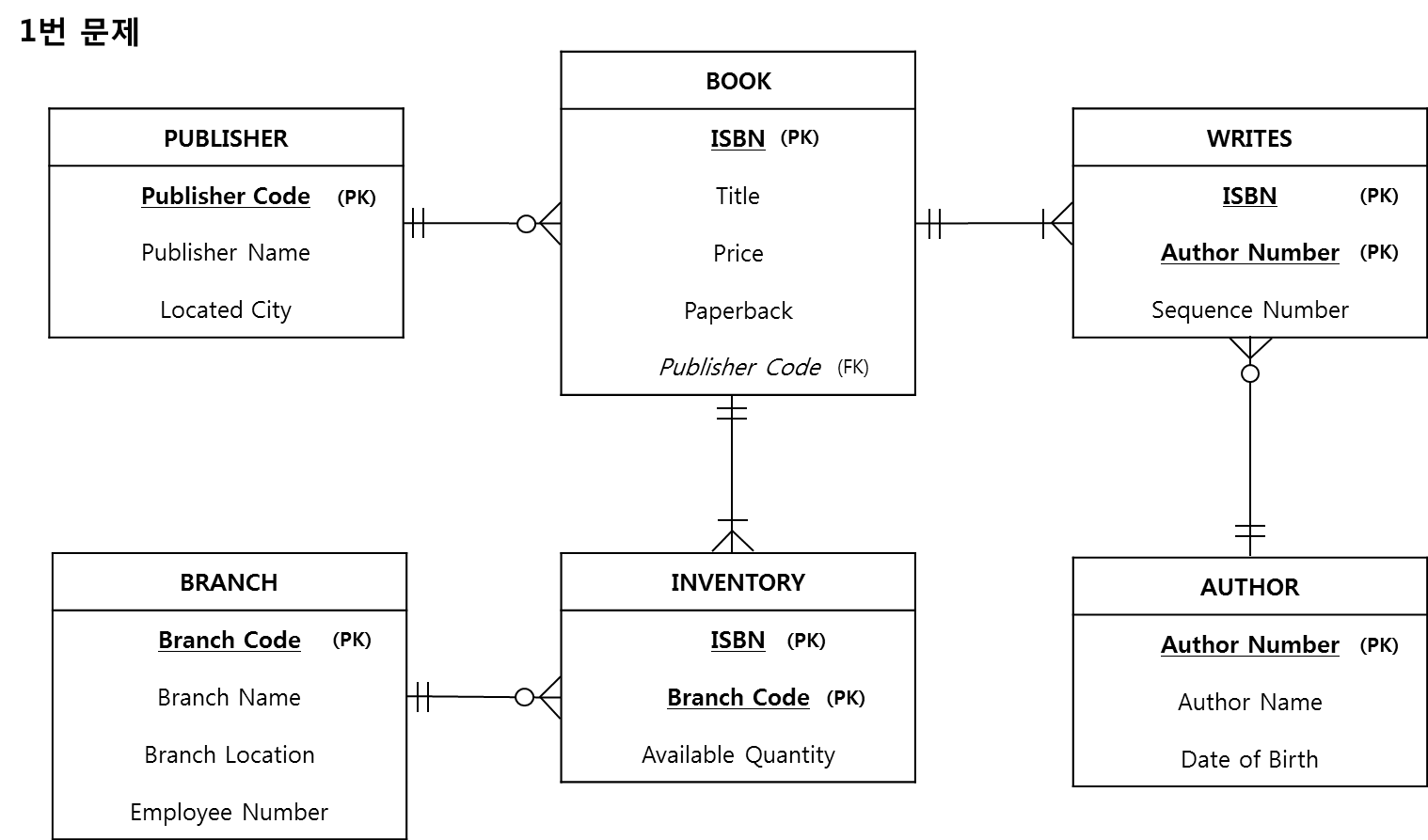
**Student ID: 20196004 Student Name: KIM MINJOON**

Q1

You will now design a database for Incredible Books, a bookstore chain. The bookstore wants to keep information on books, authors, publishers, and branches. Suppose that it has following data requirements.

* For each publisher, list the publisher code, the name, and the city in which the publisher is located.
* For each book, list its ISBN number, title, price, publisher, authors, and whether it is paperback. If there is more than one author, they must be listed in the order in which they are listed on the book.
* For each branch, list the branch code, the name, the location, and the number of employees.
* In addition, list the code and title of each book currently in the branch as well as the number of units of the book the branch currently has.

Create a conceptual data model in ER notation to implement the requirements described above. Do not forget attributes, identifier, and cardinalities of relationships. State your assumptions, if needed.



Q2

You are asked to design a data model for the hotel reservation system. Its data requirements are shown in the following descriptions.

Hotels: hotel id, name, address, star rating, number of rooms

Rooms: room number, bed type, room size, standard price

Customers: customer id, password, customer type, last name, first name, address, credit cards

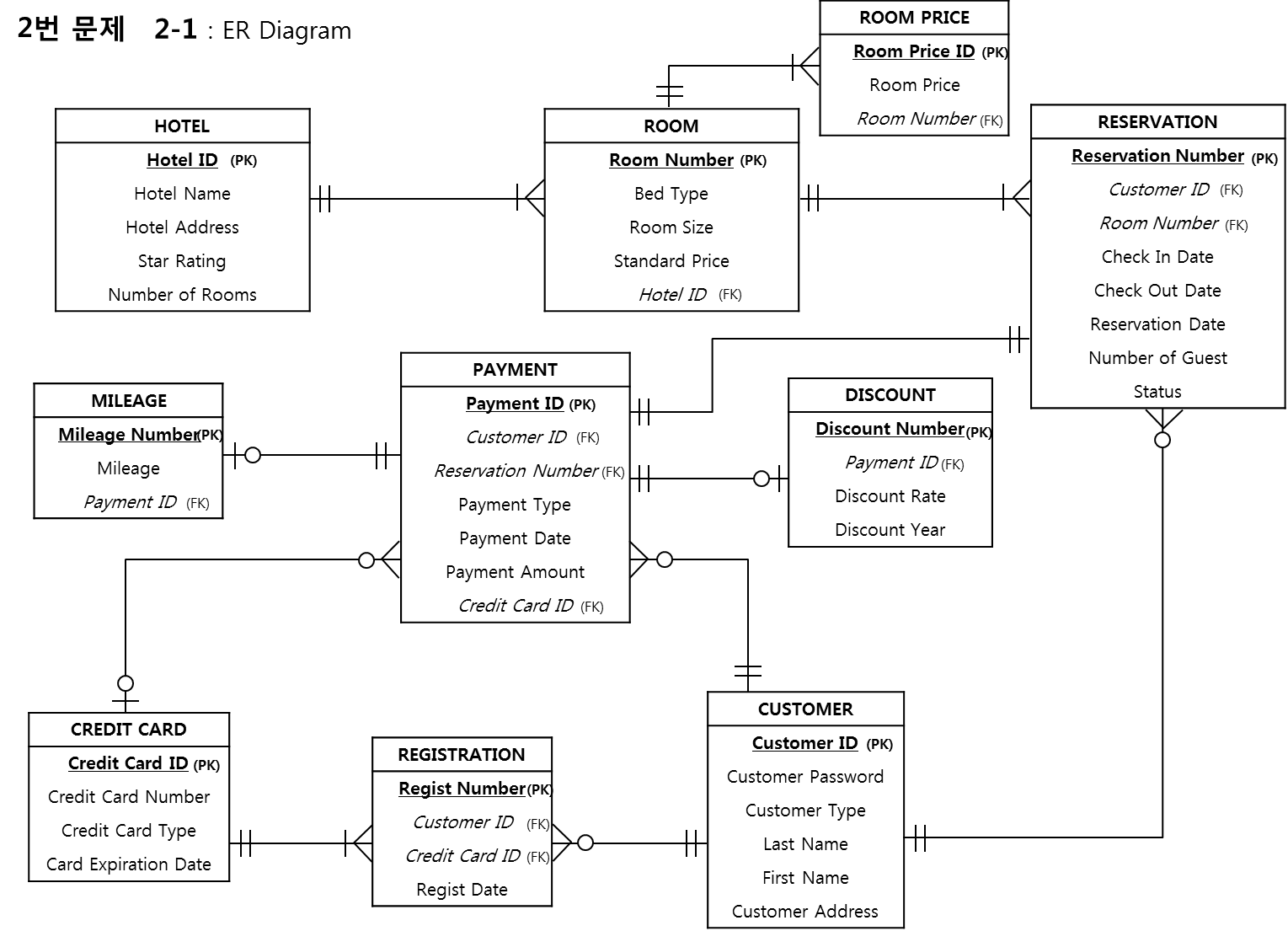
\* A hotel may offer a room price different from its standard price.

\* A customer should be able to reserve rooms for specific check-in and check-out dates. They can reserve more than one room at a time.

\* Customers are of two types: corporate and individual. Corporate customers are companies that reserve and pay for the rooms for their employees. This type of customers has special promotion discount rates. Discount rates are updated every year for each corporate customer. Individual customers can accumulate their mileage (about 5% of their payments) and use it for future payments.

\* A customer pays with their credit cards or mileage. They can save more than one credit card information.

1. Create a conceptual data model in ER notation. Make sure to indicate the various attributes of each entity and relationship set. Also specify identifiers and cardinality constraints. State your assumptions, if needed.

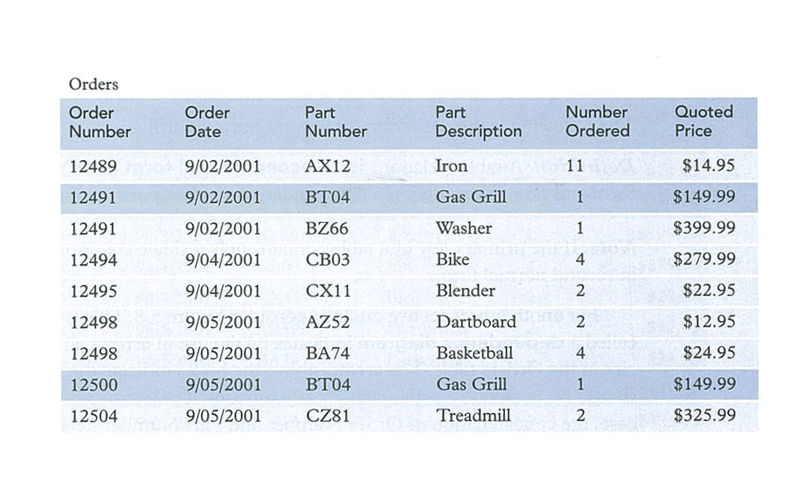


1. Create a relational schema based on the ER model you created in (1)



Q3

We consider the following relation:



\* Let us assume that the quoted price is fixed for each part.

1. Is the above table in the first normal form? What are the primary key fields?

- Yes, the relation is provided in the First Normal Form (1NF)

: no multivalued attributes (every attribute value is atomic)

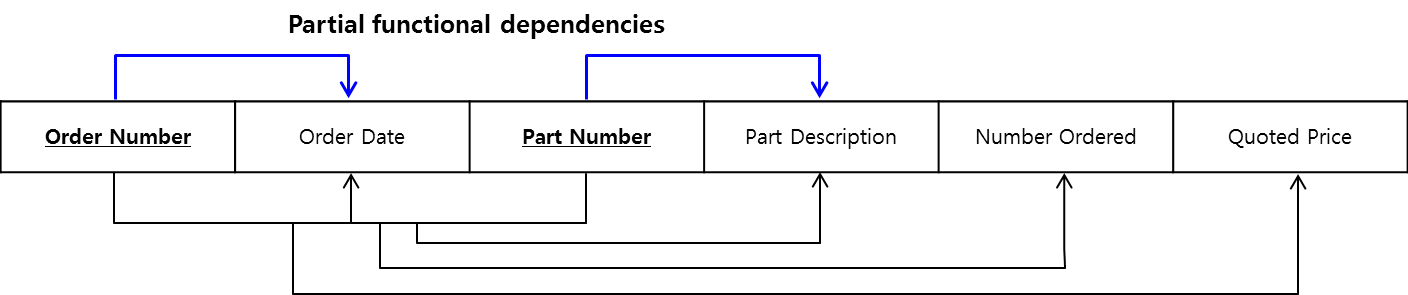
- Primary Keys are Order Number, Part Number

1. Explain potential problems with insertion, update, and deletion anomalies. Show partial functional dependencies, if any, among attributes.

Insertion : if new part is ordered, some information must be re-entered, causing duplication.

Deletion : if we delete the Iron from Oder Number 12496, we lose some information.

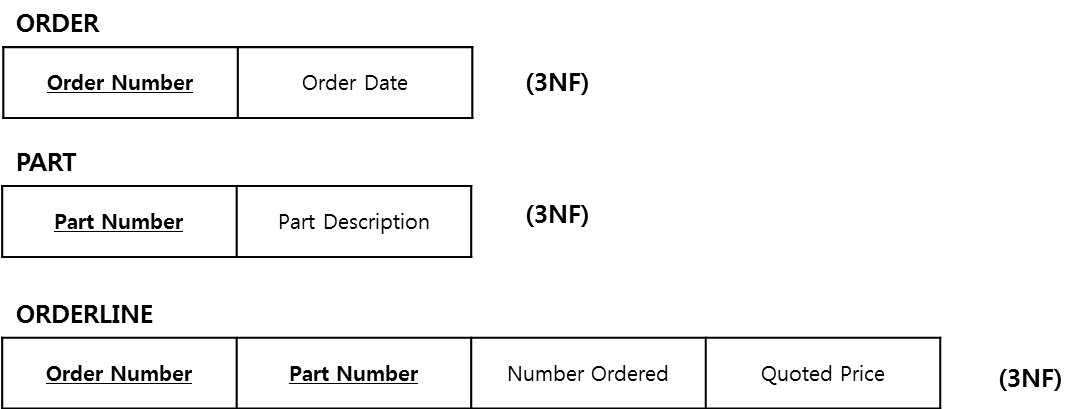
Update : if change certain value, it requires update in multiple records.



1. Create relations in the second normal form.

After Removing Partial functional dependencies,

it was converted the First Normal Form to Third Normal Form



1. Show transitive functional dependencies, if any.

There is no Transitive Functional Dependencies

.(no functional dependencies on non-primary-key attributes)

1. Create relations in the third normal form.

It is aleady the Third Normal Form

Q4

1. Cretae a new table, Customer. It has only two fields: CustomerID (Integer) and CustomerName (Varchar(30).

CREATE TABLE Customer (

CustomerID INTEGER NOT NULL,

CustomerName VARCHAR(30),

CONSTRAINT Customer\_PK PRIMARY KEY (CustomerID));

1. Create a new table, Movie. It has three fields: MovieID (Integer), Title (Varchar(30)), and ReleaseYear (Integer of four digits).

CREATE TABLE Movie (

MovieID INTEGER NOT NULL,

Title VARCHAR(30),

ReleaseYear INT(4),

CONSTRAINT Movie\_PK PRIMARY KEY (MovieID));

1. Create a new table, Review. It has four fields: ReviewID (Integer), ReviewText (Varchar(30)), CustomerID (Integer), and MovieID (Integer). CustomerID and MovieID are foreign key fields that reference primary key fields of Customer and Movie tables, respectively.

CREATE TABLE Review (

ReviewID INTEGER NOT NULL,

ReviewText VARCHAR(30),

CustomerID INTEGER,

MovieID INTEGER,

CONSTRAINT Review\_PK PRIMARY KEY (ReviewID),

CONSTRAINT Review\_FK1 FOREIGN KEY (CustomerID) REFERENCES Customer(customerID),

CONSTRAINT Review\_FK2 FOREIGN KEY (MovieID) REFERENCES Movie(MovieID));

1. Add a new column, ReviewDate to Review table. The data type of ReviewDate is Date.

ALTER TABLE Review ADD ReviewDate DATE;

1. Change the size of ReviewText in Review table from up to 30 characters to 300 characters.

ALTER TABLE Review MODIFY ReviewText VARCHAR(300);

Q5

Add new records to Customer, Movie and Review table.   
  
ID: 101 Name: John Doe  
ID: 102 Name: Jane Doe   
  
ID: 1001 Title: Avengers ReleaseYear: 2010

ID: 1002 Title: Last Jedi ReleaseYear: 2017

ID: 90001 Review Text: It s\*\*\*s! CustomerID: 101 MovieID: 1002

ID: 90002 Review Text: Mind blowing. CustomerID: 102 MovieID: 1001

INSERT INTO Customer (CustomerID, CustomerName) VALUES (101, "John Doe");

INSERT INTO Customer (CustomerID, CustomerName) VALUES (102, "John Doe");

INSERT INTO Movie (MovieID, Title, ReleaseYear) VALUES (1001, "Avengers", "2010");

INSERT INTO Movie (MovieID, Title, ReleaseYear) VALUES (1002, "Last Jedi", "2017");

INSERT INTO Review (ReviewID, ReviewText, CustomerID, MovieID) VALUES (90001, "It s\*\*\*s!", 101, 1002);

INSERT INTO Review (ReviewID, ReviewText, CustomerID, MovieID) VALUES (90002, "Mind blowing.", 102, 1001);

(Before answering questions) If you have not installed PVFC database yet, please do so by downloading and executing script files from Blackboard. There are three files: PVFC\_Create.sql, PVFC\_Load1.sql, and PVFC\_Load2.sql. The data model for PVFC database is in PVFC\_ERD.pdf file.

Q6

List name, city, state, and supervisor names of all the employees who were hired in 1999.

SELECT a.EmployeeName, a.EmployeeCity, a.EmployeeState, b.EmployeeName

FROM employee\_t a INNER JOIN employee\_t b

ON (a.employeesupervisor = b.employeeID)

WHERE a.employeedatehired BETWEEN "1999-01-01" and "1999-12-31";

Q7

List order id, order date, product description, unit price, ordered quantity, and subtotal for the orders placed by Eastern Furniture. Sort the results by order date, from latest to old ones.

SELECT a.OrderID, a.OrderDate, b.ProductDescription, b.ProductStandardPrice, sum(c.OrderedQuantity) AS Subtotal\_Orders

FROM order\_t a, product\_t b, orderline\_t c, customer\_t d

WHERE a.OrderID=c.OrderID AND b.ProductID=c.ProductID AND a.CustomerID=d.CustomerID

AND d.CustomerName = "Eastern Furniture"

GROUP BY a.OrderID, a.OrderDate, b.ProductDescription, b.ProductStandardPrice

ORDER BY a.OrderDate DESC;

Q8

Display each customer’s name and total order amount (in dollars, not the ordered quantities.) If a customer has never placed any order, its total amount should be shown as zero. Sort the results by total amount, from big to small.

SELECT a.CustomerName, COALESCE(SUM(c.OrderedQuantity \* d.ProductStandardPrice), 0) AS TotalOrderAmount\_dollars

FROM customer\_t a LEFT OUTER JOIN order\_t b

ON (a.CustomerID=b.CustomerID) LEFT OUTER JOIN orderline\_t c

ON (b.OrderID=c.OrderID) LEFT OUTER JOIN product\_t d

ON (c.ProductID=d.ProductID)

GROUP BY a.CustomerName

ORDER BY TotalOrderAmount\_dollars DESC;

Q9

Show the name and address of the customer who has the largest sales amount (calculated in Q8). The total amount should be displayed on the result view as well.

SELECT d.CustomerName, d.CustomerAddress, sum(c.OrderedQuantity \* b.ProductStandardPrice) AS TotalOrderAmount\_dollars

FROM order\_t a, product\_t b, orderline\_t c, customer\_t d

WHERE a.OrderID=c.OrderID AND b.ProductID=c.ProductID AND a.CustomerID=d.CustomerID

GROUP BY d.CustomerName, d.CustomerAddress

HAVING d.CustomerName = "Eastern Furniture";

Q10

This time, show the sales amount by product line. Display product line id, line name, and total sales amount. Sort the results by total sales amount, from big to small.

SELECT a.ProductLineID, a.ProductLineName, sum(c.OrderedQuantity \* b.ProductStandardPrice) AS TotalSalesAmount\_dollars

FROM productline\_t a, product\_t b, orderline\_t c

WHERE a.ProductLineID=b.ProductLineID AND b.ProductID=c.ProductID

GROUP BY a.ProductLineID, a.ProductLineName

ORDER BY TotalSalesAmount\_dollars DESC;

Q11

Show customer name, territory name of their business, and salesperson who covers the same territory.

SELECT DISTINCT a.CustomerName, b.TerritoryName, c.SalespersonName

FROM customer\_t a, territory\_t b, salesperson\_t c, doesbusinessin\_t d

WHERE a.CustomerID=d.CustomerID AND b.TerritoryID=d.TerritoryID AND b.TerritoryID=c.SalesTerritoryID;

Q12.

Base on the result in Q11, list all the customer name, territory name, and number of salespersons in that territory.

SELECT a.CustomerName, b.TerritoryName, COUNT(c.SalespersonName) AS NumOfSalespersons

FROM customer\_t a, territory\_t b, salesperson\_t c, doesbusinessin\_t d

WHERE a.CustomerID=d.CustomerID AND b.TerritoryID=d.TerritoryID AND b.TerritoryID=c.SalesTerritoryID

GROUP BY a.CustomerName, b.TerritoryName;

**SQL Captures : Q.4 ~ Q.12**

