CSE 3241 Introduction to Database Systems
Autumn 2019 – Final Exam
Take-home exam

Student´s name: __yufei zhou___________________________

Score: ______________ / 100

Instructions:

- Please read the Honor statement at the bottom and **sign it.**
- You should show all your work to support your answers
- **Considering that is a take-home exam, all your answers must be clear to read and understand; in a professional manner**
- YOU MUST SUBMIT YOUR EXAM TO CARMEN DROP BOX. No printed exams. Submit a single file either on word or pdf format
- You may type and draw diagrams using software tools or you may write down/draw by hand and then scan. Be neat and organized.
- Exams must be submitted to Carmen before Saturday December 7th at 12:00 AM

Confidentiality and Academic conduct:

This test should reflect strictly your work. The information on this exam is NOT to be shared in any way. This exam is 100% your own work.

Honor statement

In accordance with The Ohio State University Code of Student Conduct, I certify that:

- I have received no aid on this exam from any other person
- I have not given anyone aid on this exam
- I shall NOT discuss the contents of this exam with anyone who has not already taken this exam or is taking this exam
- I shall NOT post, publish, email, share the exam on internet

By submitting the exam, I pledge on my honor that I have not received or given any unauthorized assistance in this exam

____

Signature

Good Neighbor Store (GNS) is a growing and successful retail institution oriented to small businesses and neighborhoods. The company is concerned about its ability to manage their business information, for that reason they started a comprehensive project to incorporate a secure database system as one its pillars of their strategic technology plan, so they can have an integrated systems with centralized sales data.

You are required to design a database system to capture and store the information coming from the cashier terminals in all the different stores and provide business and store managers with information to track sales, products and store performance.

- **The company has commercial point of sale terminals in multiples stores, each cashier terminal provides a file with the following invoice data structure:**
  - Invoice_Number: each invoice is assigned a unique consecutive number
  - Product_Number: each product has a unique number and it identifies the name, price, and vendor name and code of that product
  - Product_Name: each product has a label with a name
  - Vendor_Code: Each product vendor is assigned a code when registered
  - Sale_Date: The product sale date is recorded
  - Quantity_Sold: the number units sold for each product is recorded in the invoice
  - Product_Price: The sale price for each product is recorded in the invoice
  - Cashier_Number: The store assigns a cashier number to identify each store terminal
  - Store_Number: Each store has a store number assigned
- **Here you have a sample of the invoice data structure provided in a file by each one of the sale/cashier terminals. Assume the following:**
  - There are not repeating groups
  - An invoice number references more than one product (*Hint*: the table uses a composite primary key)
  - Any given product is supplied by a single vendor, but the vendor can supply many products:

Attribute Name	**Sample Value**	**Sample Value**	**Sample Value**	**Sample Value**	**Sample Value**
INV_NUMBER	127107	127107	127107	127108	127109
PROD_NUMBER	SG-224XW	AB-300LX	TP-954	SG-224XW	CD-834PR
PROD_NAME	Door Sign	2.5-in. plate	Gold Tape	Door Sign	Gloss Paper
VENDOR_CODE	110	110	205	110	157
VENDOR_NAME	SignCo, Inc.	SignCo, Inc.	BestTape, Inc.	SignCo, Inc.	PaperMill, Inc.
SALE_DATE	01-Jun-2019	01-Jun-2019	01-Jun-2019	01-Jun-2019	02-Jun-2019
QUANT_SOLD	5	12	2	1	8
PROD_PRICE	$9.95	$3.99	$29.99	$9.95	$7.50
CASHIER_NUM	12	12	12	12	12
STORE_NUM	27	27	27	27	27

1. **Do the analysis of the requirements and the data description provided above; you need to model and design the sales database, having this as a starting point:**

a. Write (draw) the relational schema (*Hint*: start with the single relation given)

b. Identify and discuss the functional dependencies (draw all the dependencies: full, partial, and transitive).

c. Create a database whose tables are at least in 2NF, showing the functional dependencies for each table. Draw the relational schema and the new functional dependencies.

d. Create a database whose tables are at least in 3NF, showing the functional dependencies for each table. Draw the relational schema and the new functional dependencies.

e. Draw the ER diagram. Include cardinality and participation. Use Chen's notation.

2. **Create the database *Sales.db* and all the tables, including the primary key and referential integrity constraints. Use SQL statements.**

3. **To import the sales data from a csv file (each sample value in the columns above is a line in the file), write a Java program that when called will read the data from one file and update the different database tables. Assume the file is uploaded already and ready to be open/read.**

   a. There are different ways to do it. *Hint*: *PreparedStatements* allow reuse of cached statements

   b. Note: The purpose in not to grade your Java coding skills. I'm expecting as a minimum that your use the best practices discussed in class to embed SQL

   c. You have at your disposal the class and checkpoint code examples

4. **The company needs to write another Java program to update an Inventory database system (keeps track of the overall product inventory of the company)**

   - Besides importing the data and updating the tables with the program from the previous question, you need to update the product inventory table on the Inventory database system
   - Design a SQL transaction that should be executed for each one of the product sales updates to ensure the consistency of the database (ACID transactions); there will be multiple processes running concurrently retrieving/updating that database
   - Write a program in Java to connect to that database (Inventory.db) and process the sales file to run the transactions and update the inventory of the products sold. The idea is to execute this program periodically to update the inventory more often and easier. (Note: you may expand and *blend* it with the previous program to do both things when reading the file)
   - The *inventory.db* has a product table that keeps the stock quantity (number of units in stock) for each product.
   - The structure of the Product table is:
     - Product (P#, label, vendor#, current_stock).
     - Current_stock is an integer and stores the number of units available, the other attributes are strings.
   - Write the program, implementing it using the best practices discussed in class, to ensure a reliable and secure system. Assume the database is a MySQL database.

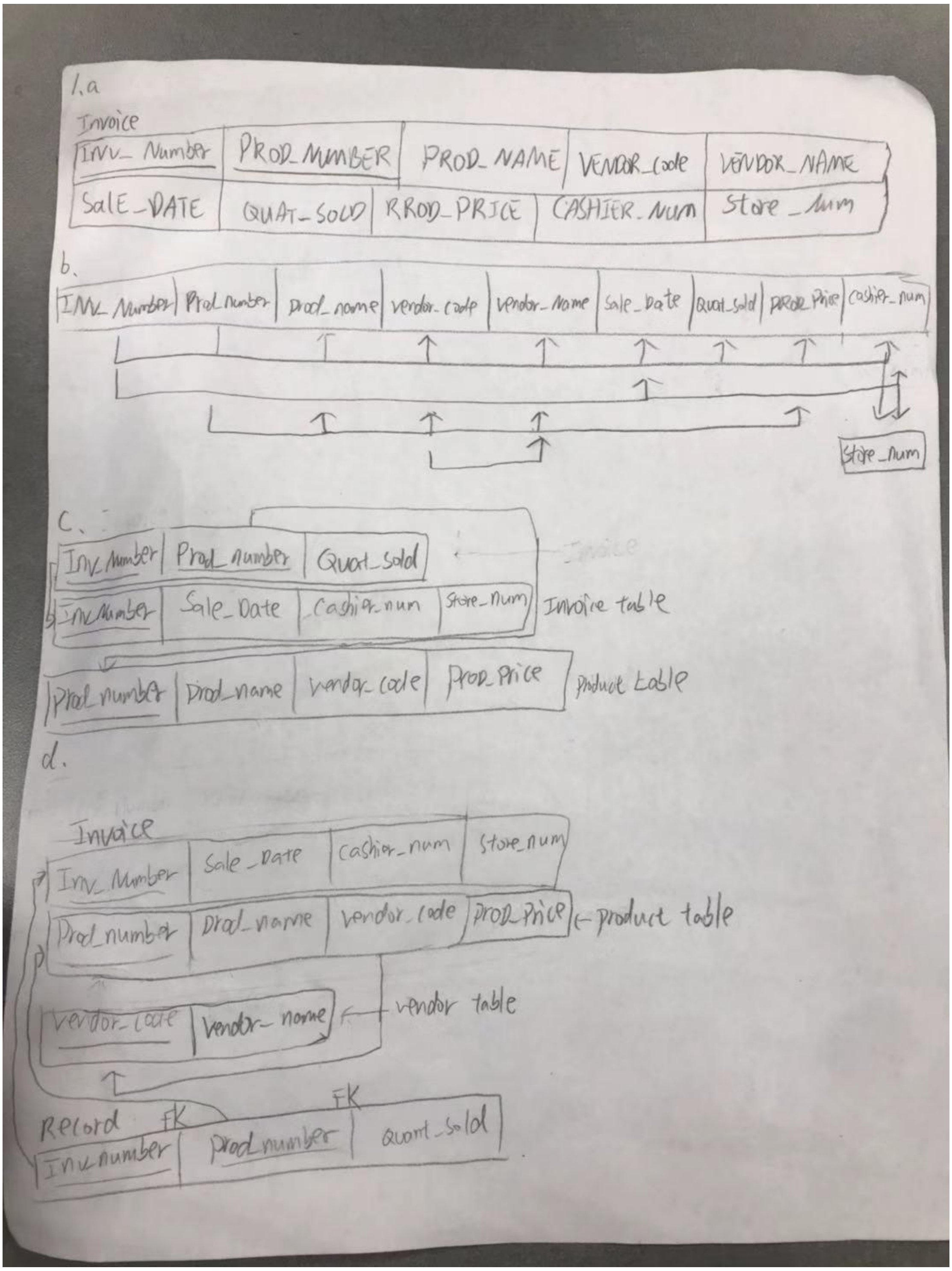
5. **There are different business processes running concurrently by different users: adding products that just arrived, removing defective items, shipping products to customers, updating prices, processing re-orders, and returns, among others. For the following schedules that access different data items like products and others more, determine:**
    a. **Is S1 cascadeless? Explain**
    b. **Is S2 a recoverable schedule? In what condition is not recoverable? Explain**
    c. **Is S3 serializable? If serializable write down the equivalent serial schedule(s).**

S_1 = r1(P); r2(M); r3(P);r1(M);r2(N);r3(N);w1(P);c1;w2(M);w3(N);w2(N);c3;c2;

S_2 = r1(P);r2(M);r1(M);r3(P);r3(N);w1(P);w3(N);r2(N);w2(M);w2(N);c1;c2;c3;

S_3 = r1(P);r2(M);r1(M);r3(P);r3(N);w1(P);c1;w3(N);c3;r2(N);w2(M);w2(N);c2;

1.



1.a

Invoice

Inv_ Number	PROD_NUMBER	PROD_NAME	VENDOR_code	VENDOR_NAME
SalE_DATE	QUAT_SOLD	RROD_PRICE	CASHIER_Num	Store _num

b.

INV_Number	Prod_number	prod_name	vendor_code	vendor_Name	Sale_Date	Quat_sold	PROD_Price	cashier_num

Store_num

C.

Inv_Number	Prod_number	Quat_sold

Inv_Number	Sale_Date	Cashier_num	Store_Num

Invoice table

Prod_number	Prod_name	vendor_code	Prop_Price

product table

d.

Invoice

Inv_Number	Sale_Date	cashier_num	Store_num

Prod_number	Prod_name	vendor_code	PROD_Price

← product table

vendor_code	Vendor_name

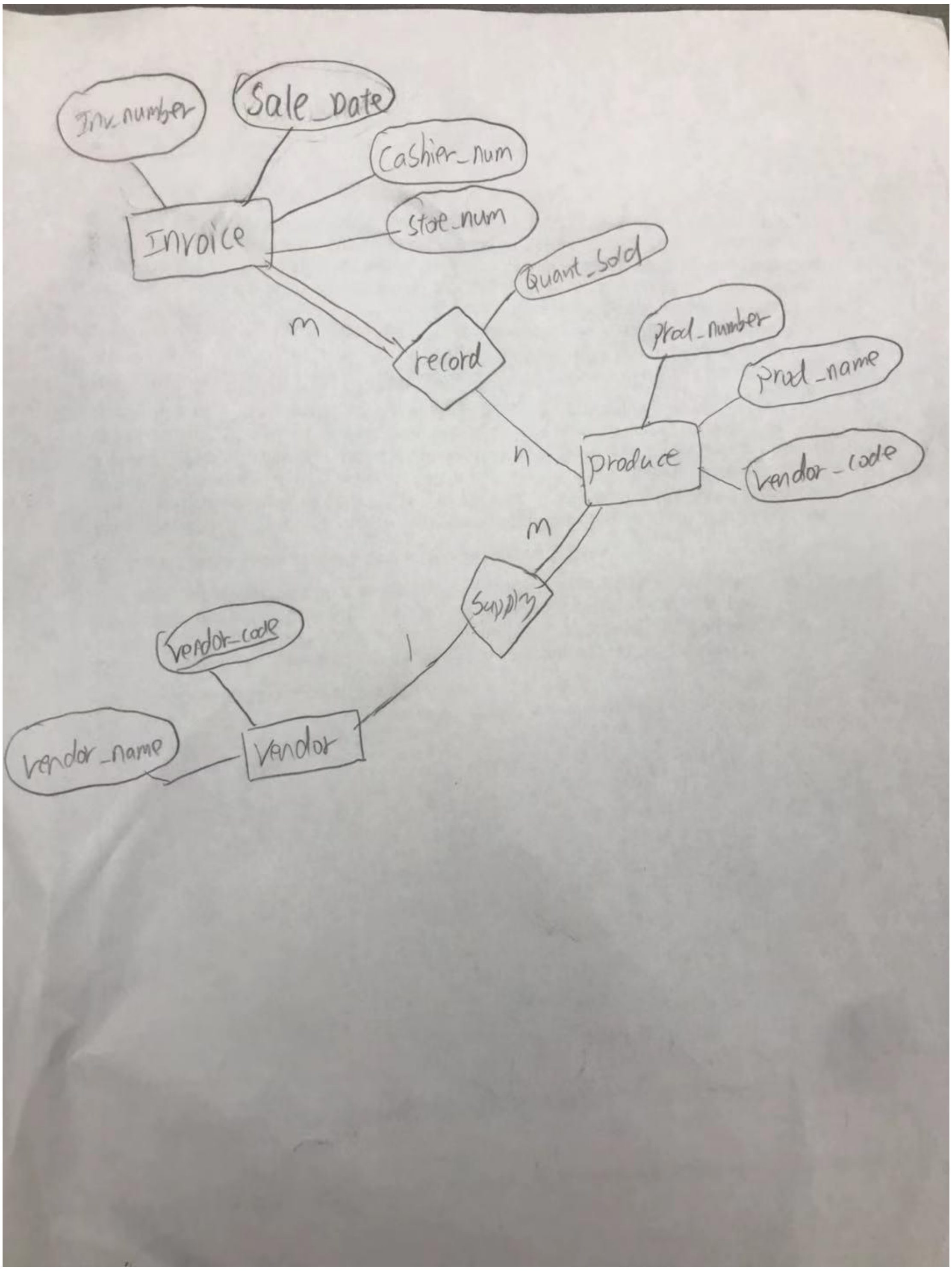
← vendor table

Record

FK

FK

Inv_number	Prod_number	Quant_sold



2.
Create invoice table
Invoice_number varchar(6) not null
Sale_date date
Cashier_number integer
Store_number integer
Primary key (invoice_number)

```
Create product table
Prod_number varchar(10) not null
Prod_name varchar(10) not null
Prod_price double not null
Vendor_code integer
Primary key(prod_number)

Create vendor table
Vendor-code integer not null
Vendor_name varchar(10) not null
Primary key(vendor_code)
Foreign key (vendor_code) reference vendor (vendor_code)

Create record table
Invoice_number varchar(6) not null
Prod_number varchar(10) not null
Quant_sold integer
Primary key(invoice_number, prod_number)
Foreign key(invoice_number) reference invoice (invoice_number)
Foreign key(prod_number) reference product(prod_number)
```

3.

```
Import java.sql*

Public class invoice{
  Static final String JDBC_DRIVER = “com.mysql.jbdc.Driver”;
  Static final String DB_URL = “jdbc:mysql://localhost/EMP”;
  Static final String USER = “username”;
  Static final String PASS = “password”;
  Public static void main(String [] args) {
  Connection conn = null;
  PreparedStatement stmt = null;
  ResultSet rSet = null;
  Try
{ Class.forName(JBDC_DRIVER);
 System.out.println(“ Connecting to database...);
Conn = DriverManager.getConnection(DB_URL,USER,PASS);
Open file
Read a line
Get first line value in an array =[] invoiceNumber
Get second value in an array=[] ProductNumber
Get third value in an array=[] PrdouctName
Get fourth value in an array=[] vendorCode
Get fifth value in an array=[] VendorName
Get sixth value in an array=[] SaleDate
Get seventh value in an array=[] QuantitySold
Get eighth value in an array=[] ProductPrice
Get ninth value in an array=[] cashierNumber
Get tenth value in an array=[] StoreNumber

while(invoice.number has next){
Int x = 0;
String sql = “insert into Invoice(inv_number, sale_date, casheir_num, Store_num) values(?,?,?,?)”;
Stmt = conn.preparestatement(sql);
Stmt.setString(1,InvoiceNumber[x]);
Stmt.setString(2,SaleDate[x]);
Stmt.SetInteger(3,cashierNumber[x]);
Stmt.setInteger(4,StoreNumber[x]);
X++;
}

while(invoice.number has next){
Int x = 0;
String sql2 = “insert into Product(pro_number, prod_name, vendor_code, Prod_price) values(?,?,?,?)”;
Stmt = conn.preparestatement(sql2[x]);
Stmt.setString(1,productNumber[x]);
Stmt.setString(2,ProductName[x]);
Stmt.setInteger(3,vendor_code[x]);
Stmt.setdouble(4, Prod_price[x]);
X++;
}

while(invoice.number has next){
Int x = 0;
String sql3 = “insert into Vendor(Vendor_code, Vendor_name) values(?,?)”;
Stmt = conn.preparestatement(sql3[x]);
Stmt.setInteger(1, vendorCode[x]);
Stmt.setString(2, vendorName[x]);
X++;
}
while(invoice.number has next){
Int x = 0;
String sql4 = “ insert into Record(Inv_number, Prod_number, Quant_sold) values(?,?,?)”
Stmt.conn.preparestatement(sql4);
Stmt.setString(1,invoiceNumber[x]);
Stmt.setString(2,productNumber[x]);
Stmt.setInteger(3,quantitySold[x]);
X++;
}

Rset = stmt.executeQuery():
}
Catch (exception ex)
{
Handle error
}
finally
{
if(rSet!= null) { rSet.close(); }
if(stmt != null) { stmt.close(); }
if(conn!= null) { conn.close(); }
}
}
```

4.

```
Import java.sql*

Public class invoice{
  Static final String JDBC_DRIVER = “com.mysql.jbdc.Driver”;
  Static final String DB_URL = “jdbc:mysql://localhost/EMP”;
  Static final String USER = “username”;
  Static final String PASS = “password”;
  Public static void main(String [] args) {
  Connection conn = null;
  PreparedStatement stmt = null;
  ResultSet rSet = null;
  Try
{ Class.forName(JBDC_DRIVER);
 System.out.println(“ Connecting to database...);
Conn = DriverManager.getConnection(DB_URL,USER,PASS);
Open file
Read a line
Get value quantitySold
Get value Prod_number
String sql = “Begin transaction stock
         Update Product
         Set current_stock = Current_stock - quantitysold
         Where p# = prod_number
If error Then goto undo; end if.
Commit;
Go to finish:
               Undo: rollback;
                     Finish: End transaction;
Stmt.conn.preparestatement(sql);
Rset = stmt.executeQuery():
}
Catch (exception ex)
{
Handle error
}
finally
{
if(rSet!= null) { rSet.close(); }
if(stmt != null) { stmt.close(); }
if(conn!= null) { conn.close(); }
}
}
```

5.

S1: Cascadeless
Because there is no read after changes(write).

S2: non-recoverable
Because W3(n) is before W2 but C2 is before C3.

S3: serializable
R1(P);R1(M);R1(P);C1;R3(P);R3(N);W3(N);C3;R2(M);R2(N);W2(M);W2(N);C2;