CP363: Databases I

Assignment #3

CP363, WLU, 2022

Instructor: Syed Nasir Danial

Due: Sunday, March 6th, 2022

Nishant Tewari
190684430
tewa4430@mylaurier.ca

Esha Panchal
190682410
panc2410@mylaurier.ca

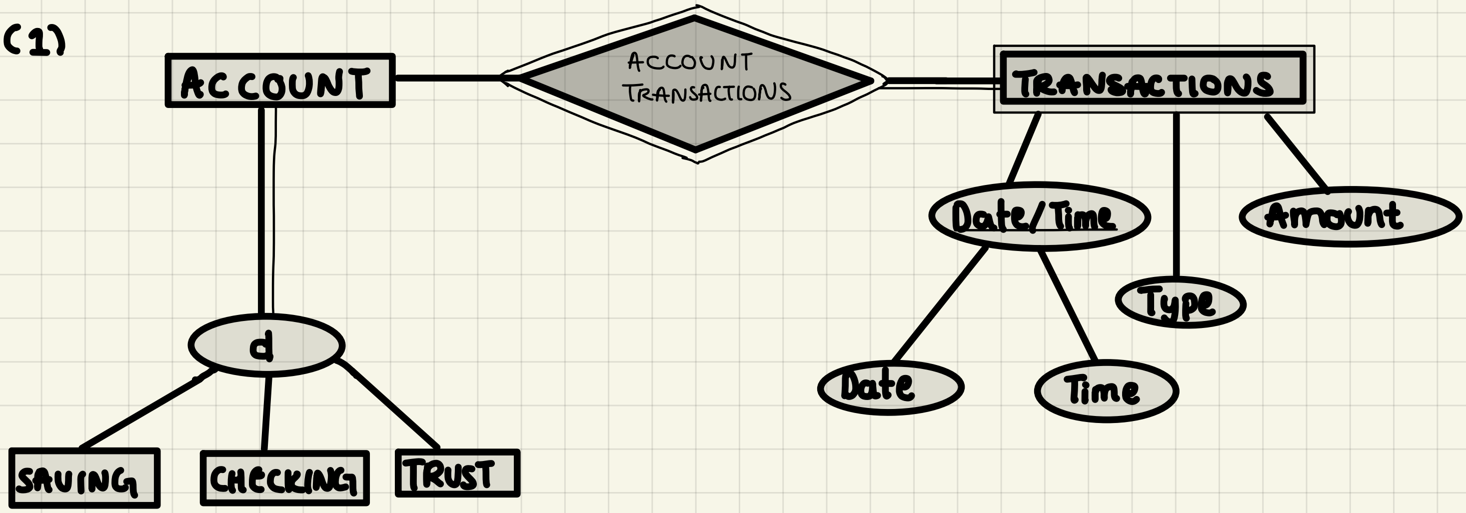
Kelvin Kellner
190668940
kell8940@mylaurier.ca

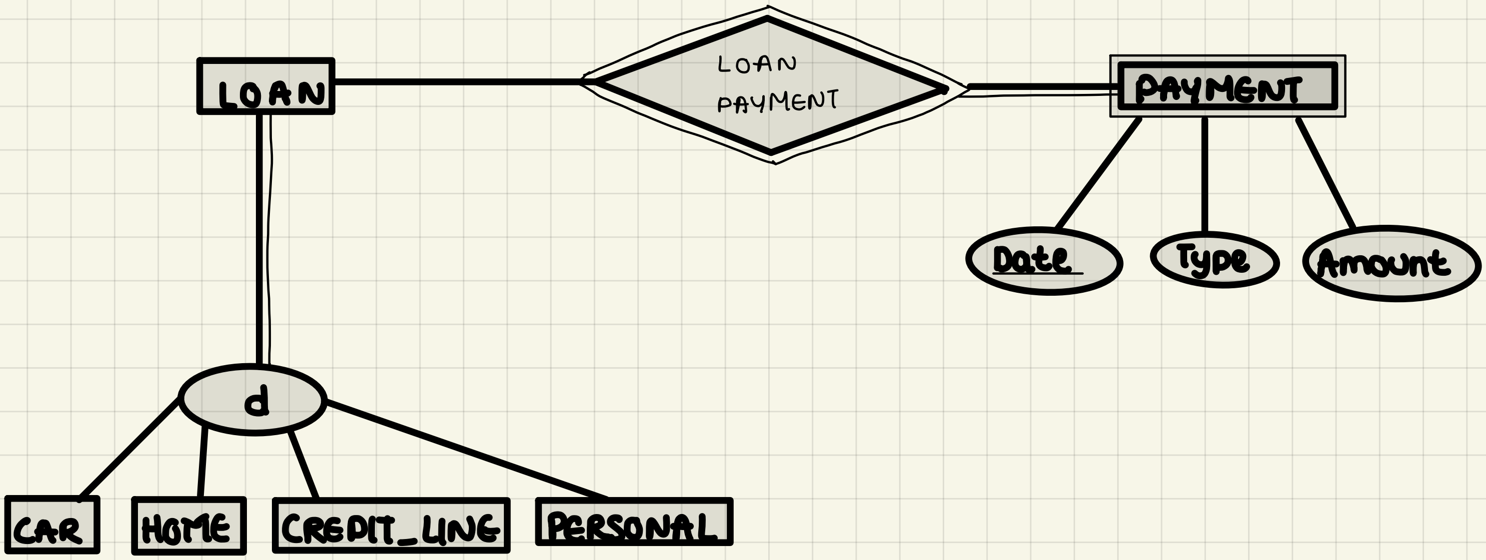
Chaitanya Bhavsar
190427300
bhav7300@mylaurier.ca

Dhari Gandhi
190500730
gand0730@mylaurier.ca

Jiten Aylani
190469460
ayla9460@mylaurier.ca

(1)


ACCOUNT
ACCOUNT TRANSACTIONS
TRANSACTIONS
Date/Time
Amount
Type
Date
Time
d
SAVING
CHECKING
TRUST


LOAN
LOAN PAYMENT
PAYMENT
Date
Type
Amount
d
CAR
HOME
CREDIT_LINE
PERSONAL

Q2

Ans To design a relational database schema for this database application, we must start off with understanding the attributes and it's role with functional dependencies.

Here are some key points to consider:

- Both SSN and Student Number have unique values for each student.
- Both name and code will have unique values for each department.
- The value of code number is unique for each course.

Let's come up with functional dependencies with the above information.

FD1 = {SSSN} → {SNAME, SNUM, SCADDR, SCPHONE, SPADDR, SPPHONE, BDATE, SEX, CLASS, MAJOR, MINOR, PROG}

FD2 = {SNUM} → {SNAME, SSSN, SCADDR, SCPHONE, SPADDR, SPPHONE, BDATE, SEX, CLASS, MAJOR, MINOR, PROG}

FD3 = {DEPTNAME} → {DEPTCODE, DEPTOFFICE, DEPTPHONE, DEPTCOLLEGE}

FD4 = {DEPTCODE} → {DEPTNAME, DEPTOFFICE, DEPTPHONE, DEPTCOLLEGE}

FD5 = {CNUM} → {CNAME, CDESC, CREDIT, LEVEL, CDEPT}

→ These are the functional dependencies that we have made from the proposed requirements using inference rules IR1 to IR3. The first two functional dependencies refer to the student attribute, with relation to student with either social security number or student number. Third and fourth functional dependencies are refering to the department with relation to the name or code. Fifth dependencies will define the course attribute and sixth dependencies section. Last dependencies talks about grade. Relations can be made via these dependendes. Course, grades, and section could be relations in 3NF and BCNF. Student and department are 3NF and BCNF. (possibility)

* Def. vary.

STUDENT

SNUM	SSSN	SCADDR	SCPHONE	SPADDR	SPPHONE	BDATE	SEX	CLASS	PROG	MAJOR	MINOR

DEPARTMENT

DEPTCODE	DEPTCODE	DEPTOFFICE	DEPTPHONE	DEPTCOLLEGE

COURSE

CNUM	CNAME	CDESC	CREDIT	LEVEL	CDEPT

SECTION

SECCOURSE	SECNUM	SEMESTER	YEAR	INSTRUCTORNAME

GRADES

SECCOURSE	SECNUM	SEMESTER	YEAR	SNUM	GRADE

We will observe the foreign keys as:

STUDENT.MAJOR → DEPARTMENT.DEPTCODE

STUDENT.MINOR → DEPARTMENT.DEPTCODE

COURSE.CDEPT → DEPARTMENT.DEPTCODE

SECTION.SECCOURSE → COURSE.CNUM

GRADES.(SECCOURSE, SEMESTER, YEAR, SECNUM) →
SECTION.(SECCOURSE, SEMESTER, YEAR, SECNUM)

GRADES.SNUM → STUDENT.SNUM

A)

Relation:

CAR_SALE(CarID, Option_type, Option_Listprice, Sale_date, Discounted_price)

Functional Dependencies (FDs):

- CarID → Sale_date
- Option_type → Option_Listprice
- CarID, Option_type → Discounted_price

When looking at the FDs, one might notice that CarID and Option_type are not present in the RHS of any of them. Noticing this, we try to search for {CarID, Option_type}$^+$.

Using the three given FDs, we can determine that:

{CarID, Option_type}$^+$ = {CarID, Option_type, Option_Listprice, Sale_date, Discounted_price}

⇒{CarID, Option_Type} is the candidate key for the relation CAR_SALE.

A relationship is considered 3NF, if it is first 2NF and there are no transitive dependencies. A transitive dependency is when you have some dependency: part of key → non-key.

CAR_SALE has CarID → Sale_date, which is a transitive dependency asCarID is part of the candidate key and Sale_date is a non-key, meaning that the relation CAR_SALE is not 3NF.

Now, for a relation to be 2NF it must first be 1NF and every non key attribute must be fully Functionally Dependent on the primary key.

In the first FD, we are given: CarID (a prime attribute) → Sale_date (a non-prime attribute). The superkey is {CarID, Option_type}. Since CarID is not the superkey, and Sale_date is a non-prime attribute, as Sale_date is not fully FD on only CarID. This is only a partial dependency, and thus, the relation is not 2NF.

∴ The relation is neither 3NF, nor 2NF.

B)

Relation:

R(Doctor#, Patient#, Date, Diagnosis, Treat_code, Charge)

From the outline text we can extrapolate:

Functional Dependencies (FDs):

Doctor#, Patient#, Date → Diagnosis, Treat_code, Charge
Treat_code → Charge

We can see that the given relationship is already in 2NF, as there are no partial dependencies. However, this relation is <u>not</u> 3NF, as Charge is a non-key attribute that is dependent on another non-key attribute, Treat_code.

We could normalise this relation to 3NF to reduce data redundancy of the Charge attribute, as this attribute is fixed and will not change per patient. The new relation would be as follows…

R1(Doctor#, Patient#, Date, Diagnosis, Treat_code)
R2(Treat_code, Charge)

- Doctor#, Patient#, Date → Diagnosis, Treat_code
- Treat_code → Charge

C)

Relation:

TRIP(trip_id, start_date, cities_visted, cards_used)

Here is a sample population for the data in the TRIP table before normalisation:

trip_id	**start_date**	**cities_visited**	**cards_used**
1	04/11/2022	Waterloo, Toronto, Ajax, Ottawa	MasterCard (9811)
2	04/21/2022	Ottawa, Montreal	MasterCard (9811), Visa (1423)

Functional Dependencies (FDs):

- trip_id → start_date

Multivalued Dependency (MVDs):

- trip_id → → cities_visited, cards_used
- start_date → → cities_visited, cards_used

trip_id determines start_date, cities_visited and cards_used may repeat for a particular trip_id or start_date, they are independent from one another and also have multiple values.

To normalise the table for TRIPS, we must first change 1NF form by eliminating all repeating groups, breaking all values in the row to atomic level. We must also identify the primary_key, which in this case is trip_id. The resulting table would look like this:

trip_id	**start_date**	**cities_visited**	**cards_used**
1	04/11/2022	Waterloo	MasterCard (9811)
1	04/11/2022	Toronto	MasterCard (9811)
1	04/11/2022	Ajax	MasterCard (9811)
1	04/11/2022	Ottawa	MasterCard (9811)
2	04/21/2022	Ottawa	MasterCard (9811)
2	04/21/2022	Ottawa	Visa (1423)
2	04/21/2022	Montreal	MasterCard (9811)
2	04/21/2022	Montreal	Visa (1423)

To normalise further into 2NF, we must eliminate all partial dependencies. Here we can see that cities_visited and cards_used are functionally dependent on either trip_id or start_date. There are no partial dependencies. However, start_date is functionally dependent on trip_id. This means that cities_visited and cards_used are transitively dependent on trip_id through start_date, but also directly dependent on trip_id. We can normalise from 2NF to 3NF by splitting into three tables, effectively removing all partial dependencies and transitive dependencies in one swoop. The resulting relation will be as follows:

Relations:

- TRIP_DATE(trip_id, start_date)
- TRIP_CITIES(trip_id, cities_visited)
- TRIP_CARS(trip_id, cards_used)

Functional Dependencies (FDs):

- trip_id → start_date
- trip_id → → cities_visited
- trip_id → → cards_used

trip_id	**start_date**
1	04/11/2022
2	04/21/2022

trip_id	**cities_visited**
1	Waterloo
1	Toronto
1	Ajax
1	Ottawa
2	Ottawa
2	Montreal

trip_id	**cards_used**
1	MasterCard (9811)
2	MasterCard (9811)
2	Visa (1423)

D)

Relation:

DiskDrive(serialNumber, manufacturer, model, batch, capacity, retailer)

As requested, here are the FDs given as text in the outline…

Functional Dependencies (FDs):

i. serialNumber, manufacturer → model, batch, capacity, retailer
ii. model → manufacturer
iii. batch, manufacturer → model
    as two manufacturers may use the same batch number
iv. model → capacity
    as model codes are unique across all manufacturers

E)

Relations:

ORDER(O#, Odate, Cust#, Total_amount)

ORDER_ITEM(O#, I#, Qty_ordered, Total_price, Discount%)

After applying a natural join on ORDER and ORDER_ITEM, we would get…

Relation:

ORDER(O#, Odate, Cust#, Total_amount, I#, Qty_order, Total_price, Discount%)

Functional Dependencies (FDs):

- O# → Odate, Cust#, Total_amount
- O#, I# → Qty_ordered, Total_price, Discount%

The relation has a candidate key of {O#, I#}. As Qty_ordered, Total_price, Discount% are partial dependencies, relation is not in 2NF. Since the relation is not 2NF, it cannot be 3NF.

```sql
CREATE TABLE Orders(
    order_id int NOT NULL,
    order_date datetime NOT NULL,
    customer_id int NOT NULL,
    total_amount DECIMAL (11,2) NOT NULL,
    item_id int NOT NULL,
    qty_ordered int NOT NULL,
    total_item_price DECIMAL (11,2) NOT NULL,
    discount_percent DECIMAL (11,2) NOT NULL
);

INSERT INTO orders VALUES( 1, current_timestamp, 1, 10.00, 123, 2, 10.00, 0.00 );
INSERT INTO orders VALUES( 2, current_timestamp, 1, 25.00, 123, 5, 25.00, 50.00 );
INSERT INTO orders VALUES( 3, current_timestamp, 3, 50.00, 789, 20, 50.00, 0.00 );
```

```sql
SELECT * FROM orders LIMIT 100;
```

	order_id int(11)	order_date datetime	customer_id int(11)	total_amount decimal(11,2)	item_id int(11)	qty_ordered int(11)	total_item_price decimal(11,2)	discount_percent decimal(11,2)
1	1	2022-03-06 21:51:35	1	10.00	123	2	10.00	0.00
2	2	2022-03-06 21:51:36	1	25.00	123	5	25.00	50.00
3	3	2022-03-06 21:51:37	3	50.00	789	20	50.00	0.00

Executive Summary

The Food Delivery management system tends to be a required database system for today's generation. Due to the convenience of food delivery, many students would prefer the service since it saves time and effort in meal prep. This is one of the reasons for selecting the food delivery management system for database development. This DBMS system is important for businesses and stores that offer food delivery services and help understand every aspect of data management related to the service industry. The system would provide management functions for customers, delivery drivers, and also restaurant staff. The report also consists of an ER diagram of the DBMS in order to view the relations between entities and attributes. The food delivery DBMS has the potential to leave a large impact on the industry, as the food delivery industry is one of the highest rising services around the world, this system can have a positive impact on users and consumers. This system may also create a positive impact for restaurants since it will be easier to analyze data related to menu items that were most liked, least liked, consistent customers, etc. providing feedback to management. Overall, this should provide usable, helpful infrastructure for innovating the restaurant ordering process.

Introduction

The following document outlines the overall requirements of the proposed food delivery system. The proposal is meant to be a general outline with the specific requirements to come later in other documentation if needed. The outline provides and illustrates the database management system and certain functionalities and representations of the system. It is a rough guideline and may need revisions as the project progresses and new requirements emerge.

System Overview:

The project proposed is a DBMS (Database Management System) for operating a food delivery service. The system will provide functionality for restaurant staff, customers, and drivers alike. It will allow restaurants to set up menus and manage orders, allow customers to view menus and submit orders to restaurants, and allow drivers to take on the delivery of any order, among other minor supporting features. The system will support all CRUD (Create-Read-Update-Delete) actions for when related to menus, orders, and deliveries. The system has the potential to be linked with frontend systems, however this is beyond the scope of this implementation.

Data Entry & Storage:

This is the ER Diagram for the food ordering system that outlines the relations between entities and their attributes are outlined as well.

(4)

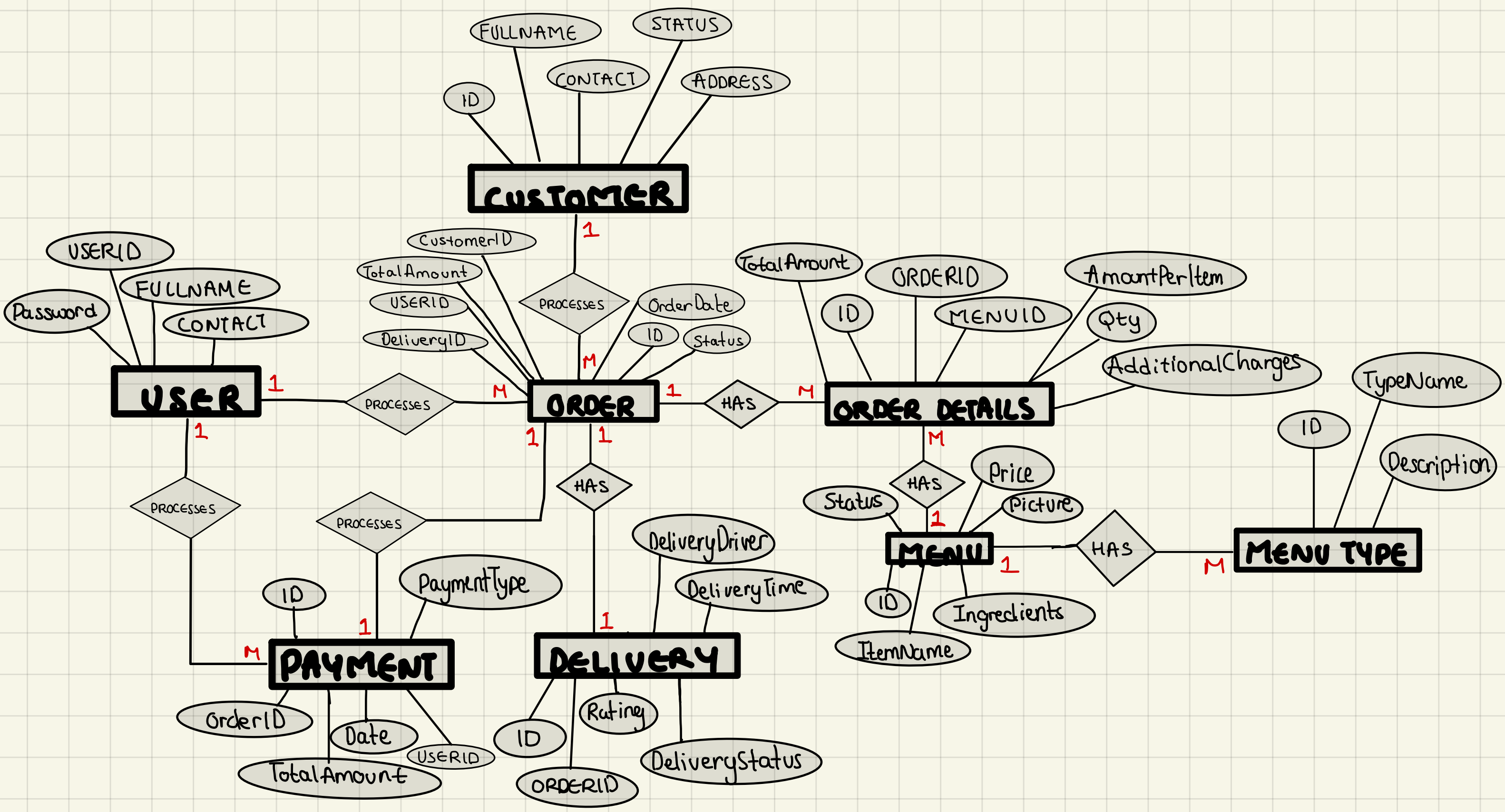
ER DIAGRAM FOR AN ONLINE FOOD ORDERING MANAGEMENT SYSTEM

ENTITIES Attributes

- **USER** ⟶ USERID, FULLNAME, EMAIL, PHONENUM, PASSWORD
- **CUSTOMER** ⟶ ID, FULLNAME, EMAIL, PHONENUM, PASSWORD, STATUS, ADDRESS
- **MENU** ⟶ ID, ItemName, Price, Picture, Ingredients, Status, Type
- **MENU TYPE** ⟶ ID, TypeName, Description
- **ORDER** ⟶ ID, OrderDate, TotalAmount, Status, CustomerID, USERID, DeliveryID
- **ORDER DETAILS** ⟶ ID, ORDERID, MENUID, AmountPerItem, Qty, TotalAmount, AdditionalCharge
- **PAYMENT** ⟶ ID, ORDERID, TotalAmount, PaymentType, Date, USERID
- **DELIVERY** ⟶ ID, ORDERID, DeliveryDriver, Rating, DeliveryStatus, DeliveryTime

= Primary key

= Foreign key



Reasons For Choosing:

Digital food delivery services have proven to be useful for consumers, especially for University students who may not always have the time to cook and meal prep in their busy schedules. Many members of our group use these services regularly, and they have become even more vital since the start of the COVID-19 pandemic. Since we have a personal connection with this service and understand how it works, we decided that it would be the perfect choice for consideration in a database project. Not to mention, apps like Uber and Skipthedishes have a lot of complexity to them, and a range of types of uses, making it an excellent challenge and full-fledged idea.

Importance / Impact:

We live in a generation where there is a huge rise and great demand for convenience, which is why it's very clear why food delivery apps are very important in today's society. With the addition of many apps and the rise of technology, we can see the rise of food delivery has become a major trend among consumers in today's world. It was estimated that the global food delivery service industry generated over $82 billion in gross revenue. It's said that this amount will be increasing as time passes, with the rise of technology. With COVID-19 having a major impact in the sales of dine-in options at restaurants, there has been an increase in profits in this industry. In today's time, digital methods are the best way to find restaurants and to order food. This allows the user to really get a chance to research ahead instead of walking in. In fact, around 90% of customers will conduct online research before getting a chance to visit the restaurant. So since social media and the internet takes so much effect on their decisions, ordering in via the internet is not so different. When the user orders from online, there is a really high chance of tailoring the menu to current trends and introducing new items that haven't been tried before. Ordering from a food delivery service not only allows you to order something new, but to allow for a degree of customization. Such interfaces allow the user to switch out the input options like ordering extra sauces, upgrading, and swapping certain items for another. Overall, this generation has really proven a great demand and has allowed for these food delivery apps to progress and play a huge role in this generation.