DBMS Project Report

PES University

Database Management Systems

UE18CS252

Submitted By

PES2201800291 Deepali Raju

Write a summary: a few sentences from Introduction, a few sentences about the data model, a few sentences about the transactions, triggers and queries. And a sentence or two to conclude on the capabilities of your system.

A Courier Management System supports the high availability of courier services to the business and customers. The system is used for day to day activities like booking a courier , maintain hub details , company details and process data of company and others.

The data model comprises of the ER model.The entities of the ER model include courier customer, office,manager,shipment and delivery along with each of their attributes. The transactions are done in offline mode and we use online data for couriers.

Triggers occur due to an DML event. Some of the common DML events include INSERT, DELETE and UPDATE. For a courier management system specifically , INSERT can be used to add a new courier to be shipped at the database along with the address and other details of the place where it has to be delivered. DELETE can be used to remove this courier once it has been shipped and so on.

A structured query in SQL has three clauses : SELECT , FROM and WHERE.
The SELECT clause is used to display what we need here it can be used to display the courier that needs to be shipped , the FROM clause is used to specify from which table. The WHERE clause is used to indicate any specifications.

To conclude the Courier Management System helps in monitoring the information on transactions and delivery.

Introduction

<<Write the miniworld description for entities and the transactions of the system.>>

Customer Entity : The key attributes are customer_id and customer_username.
customer_id,customer_name,customer_mobile,customer_email,customer_username,
customer_password,customer_address.

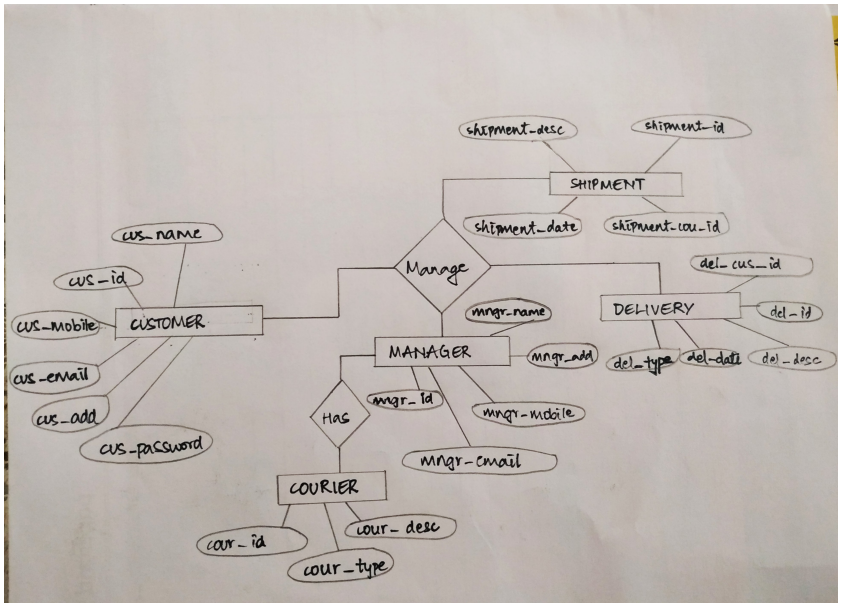
Courier Entity : The key attributes are courier_id,courier_product_id,courier_customer_id.
courier_id,courier_name,courier_description,courier_product_id,courier_customer_id.

Manager entity : The key attributes are manager_id and manager_username.
manager_id,manager_username,manager_name,manager_mobile,manager_email.

Shipment Entity : The key attribute is shipment_id.
shipment_id,shipment_courier_id,shipment_date,shipment_description.

Delivery Entity : The key attribute is delivery_id.
delivery_id,delivery_customer_id,delivery_address,delivery_date,delivery_description.

Data Model



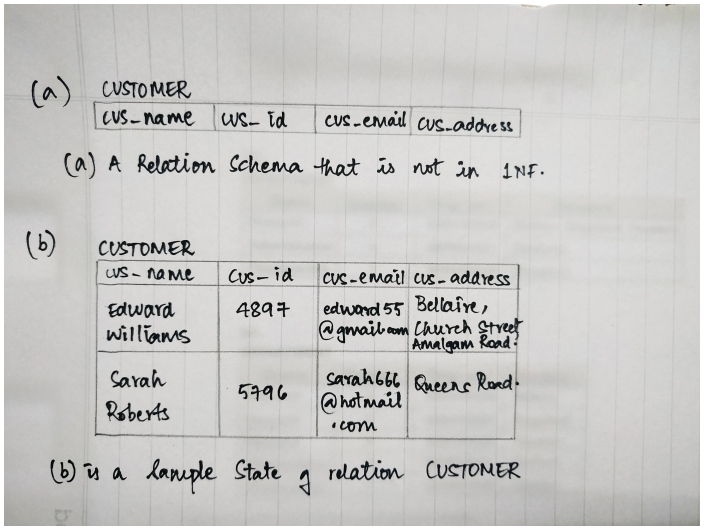
FD and Normalisation

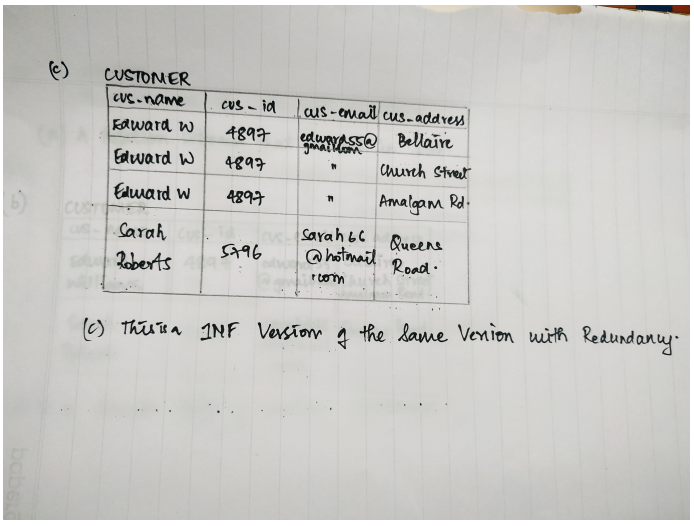
Identify the FDs and Normal forms

Some of the functional dependencies are :
- cus_id->cus_name
- cus_name->cus_mobile
- cour_id->cour_desc
- mngr_id->mngr_name
- mngr_name->mngr_address
- cour_id->shipment_cour_id
- shipment_id->shipment_decs
- cus_id->del_cus_id and so on.

Normal Forms :

1NF : A relation schema R is said to be in normal form if it disallows composite attributes.

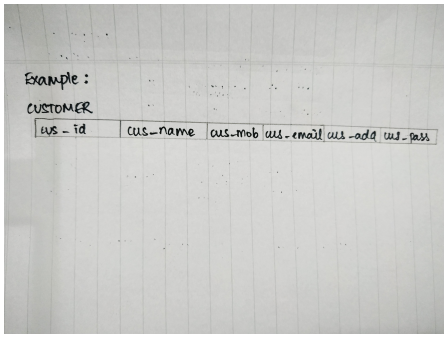




2NF :

A relation schema is said to br in 2NF if every non prime attribute is fully functionally dependant on the primary key.

Example :



Let us take the example of the customer entity here,

Functional Dependancy 1 :
cus_id->{cus_name,cus_mobile}
Here , cus_id->cus_name holds and cus_id->cus_mobile also holds and hence it is not fully functionally dependant.

Functional Dependency 2 :
del_cus_id->{del_data,del_type}
Here , both do not hold hence it is fully functionally dependant.

3NF :

A relation schema is said to be in 3NF if it is in 2NF and it should not be transitively functionally dependant.

Example :
Let cour_id->shipment_id (X->Y) and shipment_id->shipment_cour_id(Y->Z).
Hence this is transitively functionally dependant and not in 3NF whereas cus_id->cus_email is not transitively dependant.

DDL

```
CREATE TABLE CUSTOMER(
ID INT                    NOT NULL,
NAME VARCHAR(20)  NOT NULL,
ADDRESS CHAR(25),
EMAIL VARCHAR(40),
MOBILE INT              NOT NULL,
PASSWORD VARCHAR(10),
PRIMARY KEY(ID)
 );

CREATE TABLE MANAGER(
ID INT                     NOT NULL,
NAME VARCHAR(20)  NOT NULL,
ADDRESS CHAR(25),
MOBILE INT              NOT NULL,
EMAIL VARCHAR(40),
PRIMARY KEY(ID)
);

CREATE TABLE COURIER(
ID INT                     NOT NULL,
TYPE CHAR(25),
DESC CHAR(40),
PRIMARY KEY(ID)
);

CREATE TABLE SHIPMENT(
ID INT                     NOT NULL,
DATE INT                NOT NULL,
COUR_ID INT            NOTNULL,
DESC CHAR(40),
PRIMARY KEY(ID,COUR_ID)
);

CREATE TABLE DELIVERY(
ID INT                    NOT NULL,
DESC CHAR(40),
TYPE CHAR(25),
DATE INT                NOT NULL,
CUS_ID INT             NOT NULL,
PRIMARY KEY(ID,CUS_ID)
);

ALTER TABLE customer ADD CONSTRAINT cust_fk FOREIGN KEY(cour_id)
REFERENCES courier(cour_id) DEFERRABLE INITIALLY IMMEDIATE;

ALTER TABLE courier ADD CONSTRAINT cour_fk FOREIGN KEY(mngr_id)
REFERENCES manager(mngr_id) DEFERRABLE INITIALLY IMMEDIATE;
```

Triggers

```
CREATE CONSTRAINT TRIGGER check_update
AFTER UPDATE
ON customer
[ FROM courier ]
{ NOT DEFERRABLE | [ DEFERRABLE ] { INITIALLY IMMEDIATE | INITIALLY DEFERRED } }
FOR EACH ROW
[ WHEN ( NEW.customer is distinct from OLD.customer) ]
EXECUTE PROCEDURE check_update( customer)
```

SQL Queries

Correlated Nested Queries

To display customers whose address is Queens Road.

```
SELECT
customer_name,
customer_id,
customer_address

FROM
customer C1

WHERE
customer_address IN(

SELECT
C2.customer_address="Queens Road"

FROM
customer C2

WHERE
C2.customer_id=C1.customer_id

GROUP BY
C2.customer_id
)
ORDER BY
customer_id,
customer_name;
```

To display all deliveries of type medicine.

```
SELECT
delivery_id,
delivery_type,

FROM
delivery d1

WHERE
delivery_type IN (

SELECT delivery_type="medicine"
FROM delivery d2
WHERE d2.delivery_id = d1.delivery_id
GROUP BY d2.delivery_id
)
ORDER BY
delivery_id;
```

Aggregate Queries

To find customers who have places at least 2 shipments per year.

```
SELECT
customer_id,
YEAR(shipment_date)
COUNT(shipment_id) shipment_count

FROM
shipment S
customer C

GROUP BY
customer_id
```

```
YEAR(shipment_date)

HAVING
COUNT(shipment_id) >=2

ORDER BY
customer_id;
```

Outer Join Query

To find customers whose item isn't shipped.

```
SELECT
customer_name,
shipment_desc,

FROM
customer C
shipment S

FULL OUTER JOIN shipment S on S.shipment_id=C.customer_id

WHERE shipment_desc is NULL;
```

Conclusion

There is a one to one and one to many relationships between shipment,manager,delivery and courier.
All entities are normalised and have been reduced of duplicates of records.
Each entity has a primary key.
Some of the limitations is that the courier management system could have been more intricate and detailed.
Some of the improvements is that we choose to include more entities and facilities to make it more user friendly.