Database Systems Term Project

CS 4347

Group 8

Jonathan Haerr
Shrey Saraswat
Rhed Santiago

Table of Contents

Problem Statement

The purpose of this project is to design a database system for ABC company. We start by developing an entity-relationship diagram that defines a company's individual entities, their attributes, and the relationships that connect the entities. Using the EER diagram, we then map the database into a relational database schema. We normalize the relations in this schema by creating dependency diagrams. Once this framework is established, we implement the database design using MySQL and perform a set of sample queries to demonstrate the integrity of our database system.

Individual Contributions

The breakdown for the contribution of the project is as follows:

Shrey:

- Helped create EER diagram
- Created and normalized dependency diagrams
- Helped edit code and write queries
- Helped write the report

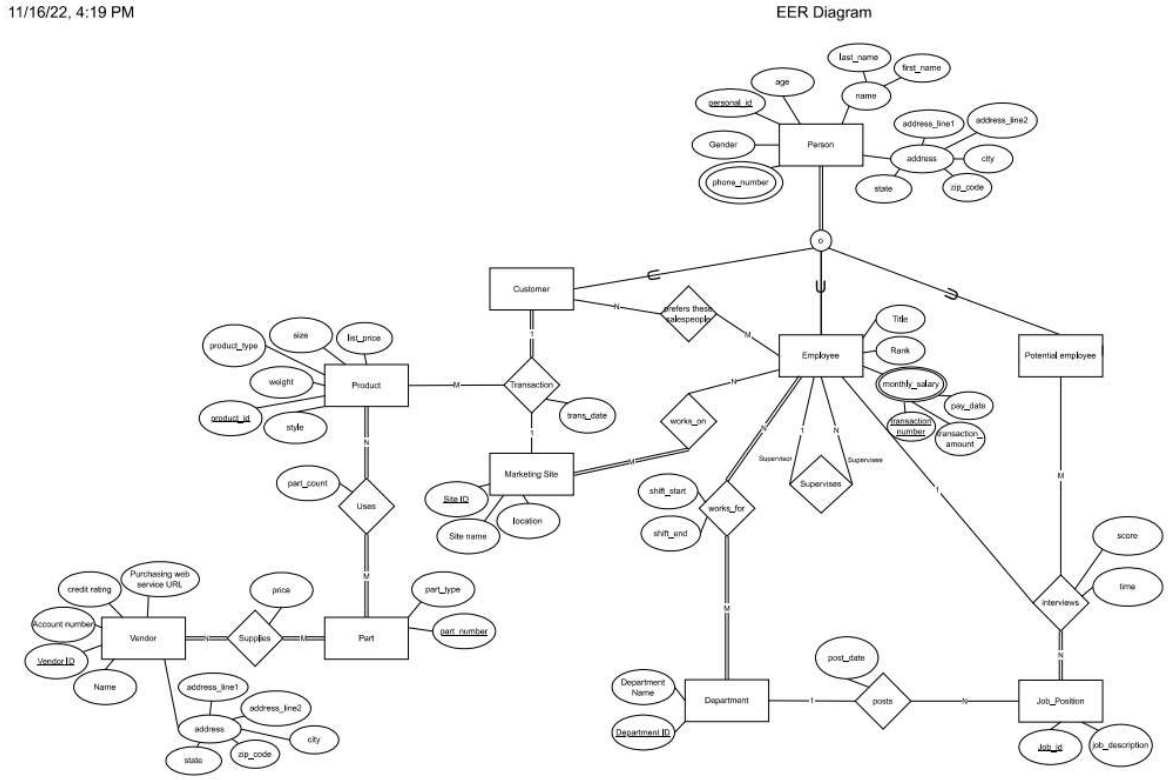
Jonathan:

- Helped create EER diagram
- Helped edit DB schema and dependencies
- Implemented database in MySQL; developed views and queries
- Helped write the report

Rhed:

- Helped create EER diagram
- Developed DB schema
- Helped edit code and write queries
- Helped write the report

EER Diagram



Assumptions

1.) In the company, the people can be divided into three subcategories, ‘Customer’, ‘Employee’, and ‘Potential Employee’, where a person must be in *at least one* of the subcategories. Salesperson is considered an employee of the company.

2.) Address is a single string attribute that will always be written in a (Street name, City, State, Country) format.

3.) When a customer makes an order they can only purchase one product. Transaction_id was added as a surrogate key.

4.) The Person relation will be used in 2nd normal form rather than 3rd for database functionality.

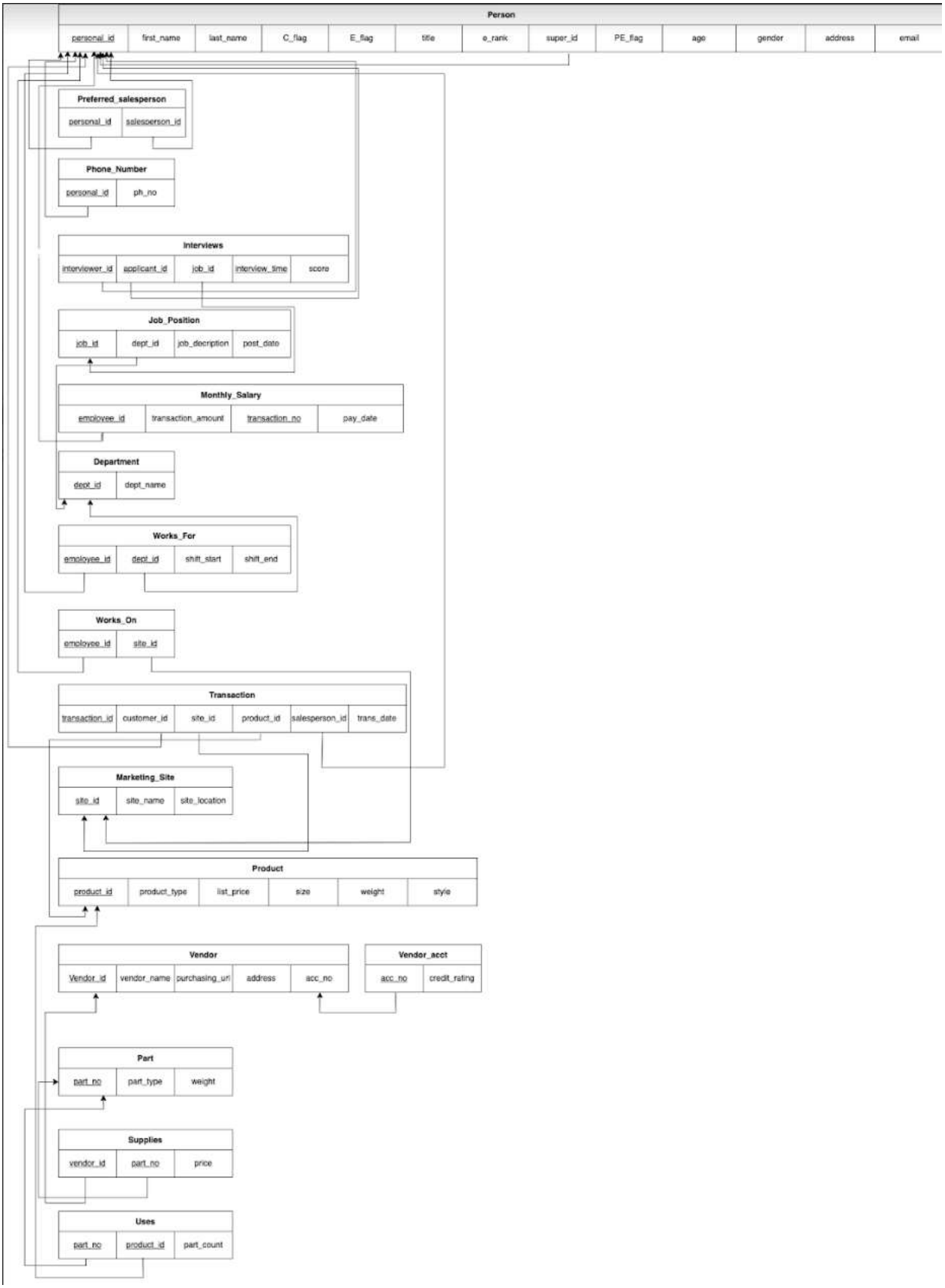
5.) For View 4 we are assuming that the view should return the sum of the purchase cost of all the parts for a given product_id.

6.) Part number will be unique for each part type sold by different vendors.

7.) Product price, size, weight, and style depend only on the product number, so these attributes can vary within the same product style.

DB Schema

We used the EER diagram that we created and mapped it to a database schema, giving us the schema below. It is important to note that this is our finalized schema diagram after our design was normalized. The main relations affected by normalization changes were the Vendor relation and the Person relation. After normalizing all of them to 3NF, we chose to represent Person relationship in a lower form, leaving it in 2NF for the functionality purposes in MySQL implementation.


Person
personal_id
first_name
last_name
C_flag
E_flag
title
e_rank
super_id
PE_flag
age
gender
address
email
Preferred_salesperson
personal_id
salesperson_id
Phone_Number
personal_id
ph_no
Interviews
interviewer_id
applicant_id
job_id
interview_time
score
Job_Position
job_id
dept_id
job_decription
post_date
Monthly_Salary
employee_id
transaction_amount
transaction_no
pay_date
Department
dept_id
dept_name
Works_For
employee_id
dept_id
shift_start
shift_end
Works_On
employee_id
site_id
Transaction
transaction_id
customer_id
site_id
product_id
salesperson_id
trans_date
Marketing_Site
site_id
site_name
site_location
Product
product_id
product_type
list_price
size
weight
style
Vendor
Vendor_id
vendor_name
purchasing_url
address
acc_no
Vendor_acct
acc_no
credit_rating
Part
part_no
part_type
weight
Supplies
vendor_id
part_no
price
Uses
part_no
product_id
part_count

Normalization Process

Now that we have developed our EER diagram and schema, we will go through the process of standardizing the schema into the third normal form.

First Normal Form

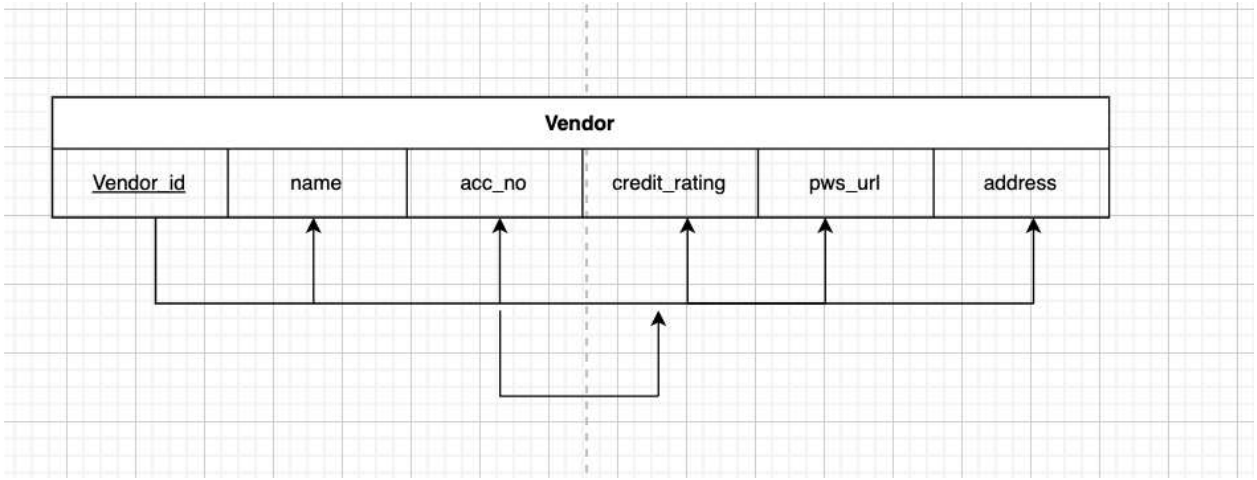
First normal form disallows the use of composite attributes, multivalued attributes, and nested relations. Our relation schema does not have any of these, so our relation schema is already in first normal form.

Second Normal Form

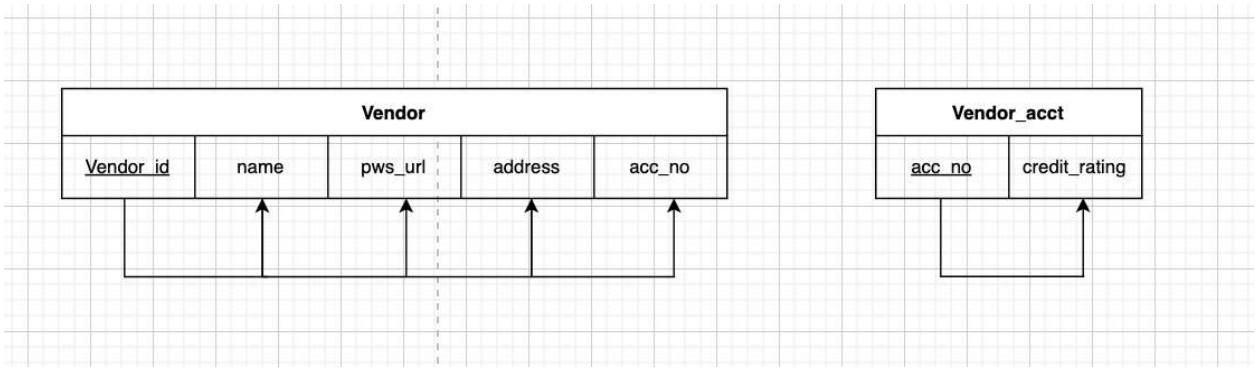
Second normal form is when every non-prime attribute A in R is fully dependent on the primary key. We know that if a PK is a single attribute, we do not need to apply a test and the relation is in second normal form. Looking through our dependency diagrams we see that we are operating within the paradigms of second normal form. Therefore, our relation schema is in second normal form without us having to make any changes.

Third Normal Form

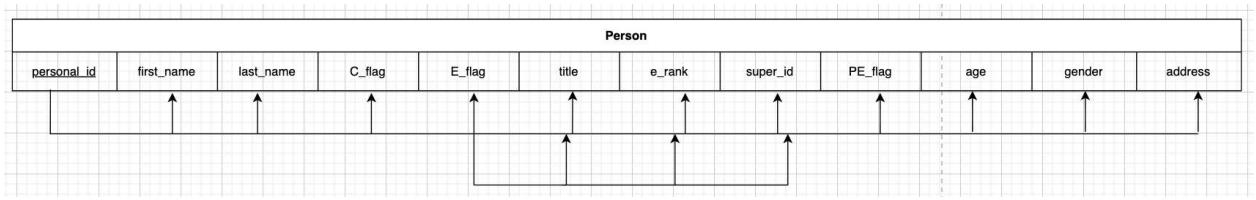
A relation schema is said to be in third normal form if it is in 2NF and no non-prime attribute A in R is transitively dependent on the primary key. We did have a couple violations of the third normal form in our relational schema for the vendor relation that can be seen below:



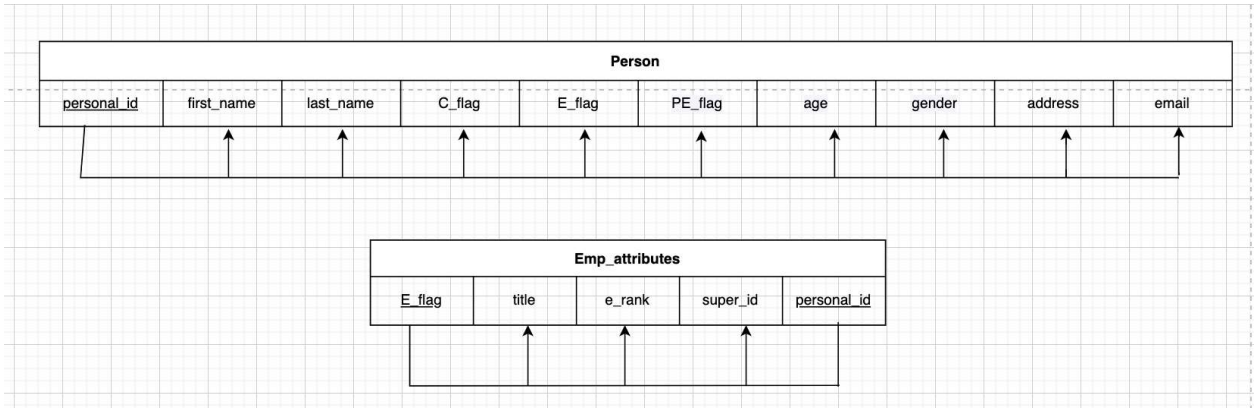
We can see here that the attribute credit_rating is dependent on acc_no, which indicates credit_rating is dependent on acc_no (stated in our assumptions). To fix this we created a Vendor_acct table and separated the attributes that were causing violation of 3NF as seen below:



We also see that our Person relation is not in 3NF as title, e_rank, and super_id have a dependence on E_flag. This is a violation of 3NF.



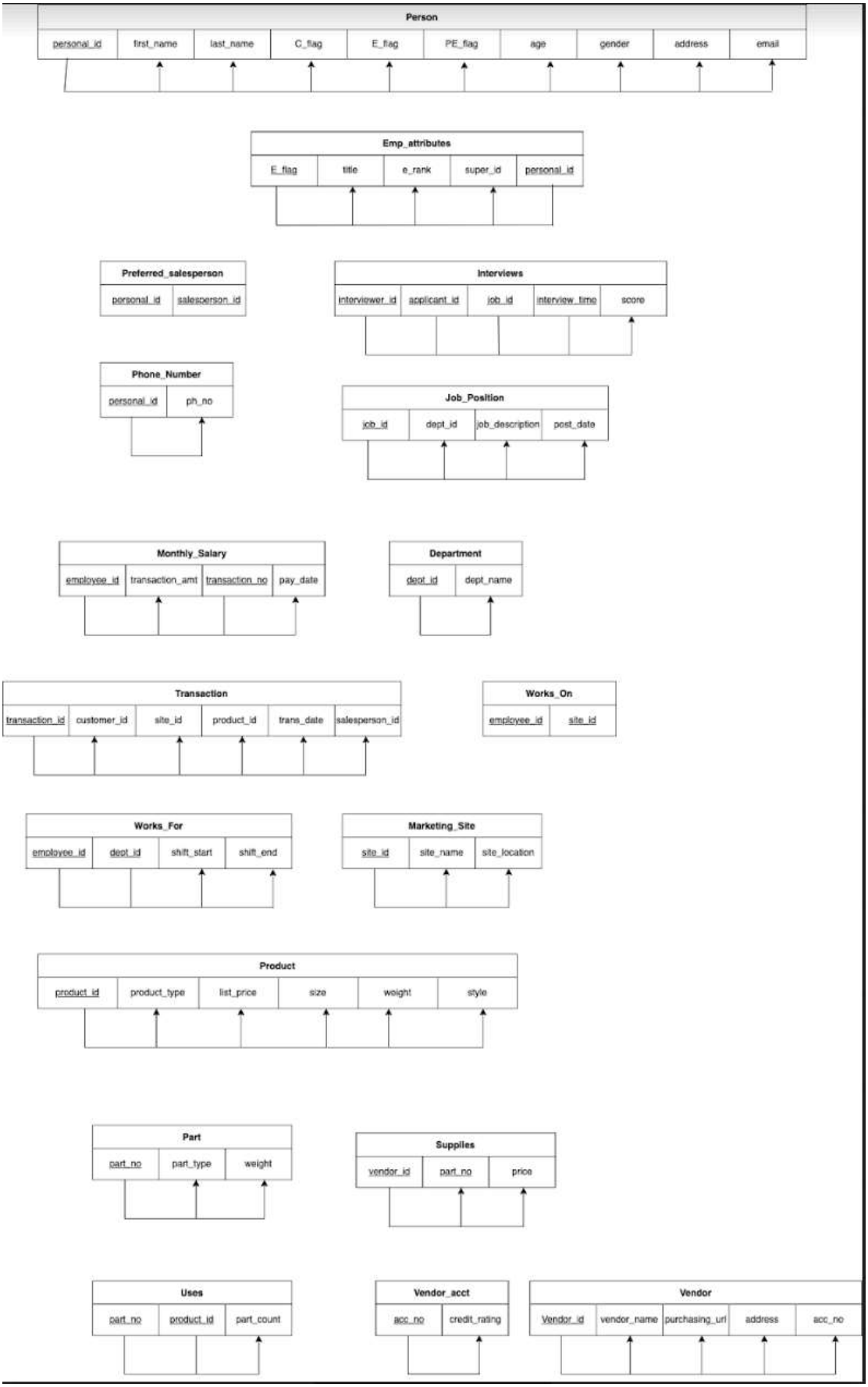
To fix the violation mentioned above we make the changes that can be seen below:



After making these changes our relational schema is now in 3NF. It is important to note that in our DB schema, we are using our person relation in 2NF rather than 3NF for better database functionality.

Dependency Diagrams

Below we have the dependency diagrams created for all relations of our DB schema, which were used throughout the normalization process:


Person
personal_id
first_name
last_name
C_flag
E_flag
PE_flag
age
gender
address
email
Emp_attributes
E_flag
title
e_rank
super_id
personal_id
Preferred_salesperson
personal_id
salesperson_id
Interviews
interviewer_id
applicant_id
job_id
interview_time
score
Phone_Number
personal_id
ph_no
Job_Position
job_id
dept_id
job_description
post_date
Monthly_Salary
employee_id
transaction_amt
transaction_no
pay_date
Department
dept_id
dept_name
Transaction
transaction_id
customer_id
site_id
product_id
trans_date
salesperson_id
Works_On
employee_id
site_id
Works_For
employee_id
dept_id
shift_start
shift_end
Marketing_Site
site_id
site_name
site_location
Product
product_id
product_type
list_price
size
weight
style
Part
part_no
part_type
weight
Supplies
vendor_id
part_no
price
Uses
part_no
product_id
part_count
Vendor_acct
acc_no
credit_rating
Vendor
Vendor_id
vendor_name
purchasing_url
address
acc_no

MySQL Implementation

Create Database

```
drop schema if exists company;
create database company;
use company;

create table person(
      personal_id int(10),
    first_name varchar(255),
    last_name varchar(255),
    c_flag boolean,
    e_flag boolean,
    title varchar(255),
    e_rank varchar(255),
    super_id int(10),
    pe_flag boolean,
    age int check (age < 65 or age is null),
    gender varchar(255),
    address varchar(255),
    email varchar(255),
    primary key (personal_id),
    foreign key (super_id) references person(personal_id),
       check ((e_flag is false and title is null and e_rank is null and
       super_id is null)
                 OR (e_flag is true and title is not null and e_rank is
       not null))
       );
create table preferred_salesperson(
      personal_id int(10),
      salesperson_id int(10),
      foreign key (personal_id) references person(personal_id),
      foreign key (salesperson_id) references person(personal_id)
);
create table phone_no(
      personal_id int(10),
      phone_no int(10),
      foreign key (personal_id) references person(personal_id)
);
create table monthly_salary(
      employee_id int(10),
      transaction_amount double,
      transaction_no int,
      pay_date date,
      primary key (employee_id,transaction_no),
```

```
      foreign key (employee_id) references person(personal_id)
);
create table department(
      dept_id int,
      dept_name varchar(255),
      primary key (dept_id)
);
create table job_position(
      job_id int,
      dept_id int,
      job_description varchar(225),
      post_date date,
      primary key (job_id),
      foreign key (dept_id) references department(dept_id)
);
create table interviews(
      interviewer_id int(10),
      applicant_id int(10),
      job_id int,
      interview_time datetime,
      score double check (score between 0 and 100),
      foreign key (interviewer_id) references person(personal_id),
      foreign key (applicant_id) references person(personal_id),
      foreign key (job_id) references job_position(job_id)
);
create table works_for(
      employee_id int(10),
      dept_id int,
      shift_start time,
      shift_end time,
      foreign key (employee_id) references person(personal_id),
      foreign key (dept_id) references job_position(dept_id),
      check (shift_start < shift_end)
);
create table marketing_site(
      site_id int,
      site_name varchar(255),
      site_location varchar(255),
      primary key (site_id)
);
create table works_on(
      employee_id int(10),
      site_id int,
      foreign key (employee_id) references person(personal_id),
      foreign key (site_id) references marketing_site(site_id)
);
create table product(
```

```
      product_id int,
      product_type varchar(255),
      list_price double,
      size double,
      weight double,
      style varchar(255),
      primary key (product_id)
);
create table transaction(
      transaction_id int,
      product_id int,
      customer_id int(10),
      site_id int,
      salesperson_id int(10),
      trans_date datetime,
      foreign key (customer_id) references person(personal_id),
      foreign key (product_id) references product(product_id),
      foreign key (site_id) references marketing_site(site_id),
      foreign key (salesperson_id) references person(personal_id),
      primary key (transaction_id)
);
create table vendor(
      vendor_id int,
      vendor_name varchar(255),
      acc_no int unique,
      purchasing_url varchar(255),
      address varchar(255),
      primary key (vendor_id)
);
create table vendor_account(
      acc_no int,
      credit_rating double,
      foreign key (acc_no) references vendor(acc_no)
);
create table part(
      part_no int,
      part_type varchar(255),
      weight double,
      primary key (part_no)
);
create table supplies(
      vendor_id int,
      part_no int,
      price double,
      foreign key (vendor_id) references vendor(vendor_id),
      foreign key (part_no) references part(part_no)
);
```

```
create table uses(
      part_no int,
      product_id int,
      part_count int,
      foreign key (part_no) references part(part_no),
      foreign key (product_id) references product(product_id)
);
```

Creating Views

```
create view view1 as
      select employee_id, avg(transaction_amount) as avg_salary
    from monthly_salary
    group by employee_id;
create view view2 as
      select applicant_id, job_id, count(score > 60) as pass_count
    from interviews
    group by applicant_id, job_id;
create view view3 as
      select p.product_type, count(t.product_id)
    from transaction t
      join product p on p.product_id = t.product_id
    group by p.product_type;
create view view4 as
      select u.product_id, sum(u.part_count*s.price) as cost
    from uses u
    join supplies s on u.part_no = s.part_no;
```

Answering Queries

```
-- 1. Return the ID and Name of interviewers who participate in interviews
where the interviewee's name is "Hellen Cole" arranged for job "11111".
select i.interviewer_id, p.last_name
from interviews i
join person p on i.interviewer_id = p.personal_id
where job_id = 11111
      and applicant_id = (
            select personal_id
        from person
        where first_name = 'Hellen'
                and last_name = 'Cole');

-- 2. Return the ID of all jobs which are posted by department "Marketing"
in January 2011.
select job_id
from job_position
```

```
where dept_id = (
      select dept_id
    from department
    where dept_name = 'Marketing')
    and month(post_date) = 1 and year(post_date) = 2011;

-- 3. Return the ID and Name of the employees having no supervisees
select personal_id, last_name
from person
where e_flag = TRUE
      and personal_id not in (
            select distinct super_id
        from person);

-- 4. Return the Id and Location of the marketing sites with no sale
records during March 2011.
select site_id, site_location
from marketing_site
where site_id not in (
      select site_id
    from transaction
    where month(trans_date) = 3 and year(trans_date) = 2011);

-- 5. Return the job's id and description, which does not hire a suitable
person one month after it is posted.
select job_id, job_description
from job_position
where job_id not in (
      select j.job_id
    from (
            select * from interviews
            natural join job_position
            having max(interview_time) between post_date and
date_add(post_date, interval 1 month)) j
      group by j.job_id
    having count(j.interview_time) >= 5
            and avg(j.score) > 70);

-- 6. Return the ID and Name of the salespeople who have sold all product
types whose price is above $200.
select personal_id, last_name
from person
where personal_id in (
      select salesperson_id
    from transaction
    where (
            select distinct product_type
```

```
            from product
            where list_price > 200)
            in (
            select distinct p.product_type
            from product p join transaction t on p.product_id =
t.product_id
            ));

-- 7. Return the department's id and name, which has no job post during
1/1/2011 and 2/1/2011.
-- (I'm assuming this is asking about posts on 1/1 or 2/1, as opposed to
between 1/1 and 2/1.)
select dept_id, dept_name
from department
where dept_id not in (
      select dept_id
    from job_position
    where post_date = '2011-01-01' or post_date = '2011-02-01');

-- 8. Return the ID, Name, and Department ID of the existing employees who
apply for job "12345".
select w.employee_id, p.last_name, group_concat(w.dept_id) as dept_id
from works_for w
join person p on w.employee_id = p.personal_id
where w.employee_id in (
      select applicant_id
    from interviews
    where job_id = '12345');

-- 9. Return the best seller's type in the company (sold the most items).
select p.product_type
from transaction t
join product p on p.product_id = t.product_id
group by p.product_type
order by count(t.transaction_id) desc limit 1;

-- 10. Return the product type whose net profit is highest in the company
(money earned minus the part cost).
select p.product_type
from uses u
inner join supplies s on u.part_no = s.part_no
inner join product p on p.product_id = u.product_id
inner join transaction t on t.product_id = p.product_id
group by p.product_id
order by (p.list_price - sum(u.part_count*s.price)) *
count(t.transaction_id) desc limit 1;
```

```
-- 11. Return the name and id of the employees who have worked in all
departments after being hired by the company.
select last_name, personal_id
from person
where personal_id in (
      select employee_id
    from works_for
    group by employee_id
    having count(distinct dept_id) = (
            select count(distinct dept_id)
        from works_for));

-- 12. Return the name and email address of the interviewee who is
selected.
select last_name, email
from person
where personal_id in (
      select distinct applicant_id
    from interviews
    group by applicant_id, job_id
    having count(interview_time) >= 5
            and avg(score) > 70);

-- 13. Retrieve the names, phone numbers, and email addresses of the
interviewees selected for all the jobs they apply for.
select p.last_name, group_concat(phone_no.phone_no) as phone_no, p.email
from person p
inner join phone_no on p.personal_id = phone_no.personal_id
where pe_flag = TRUE
      and p.personal_id not in (
            select applicant_id
            from interviews
            group by applicant_id, job_id
            having count(interview_time) < 5
                  or avg(score) > 70);

-- 14. Return the employee's name and id whose average monthly salary is
the highest in the company.
select m.employee_id, p.last_name
from monthly_salary m
join person p on m.employee_id = p.personal_id
group by m.employee_id
order by avg(transaction_amount) desc limit 1;

-- 15. Return the ID and Name of the vendor who supplies part whose name
is "Cup" and weight is smaller than 4 pounds, and the price is lowest
among all vendors.
```

```
select v.vendor_id, v.vendor_name
from vendor v
inner join supplies s on v.vendor_id = s.vendor_id
inner join part p on p.part_no = s.part_no
where p.part_type = 'Cup'
      and p.weight < 4
having min(s.price);
```