The database project
2022-2023

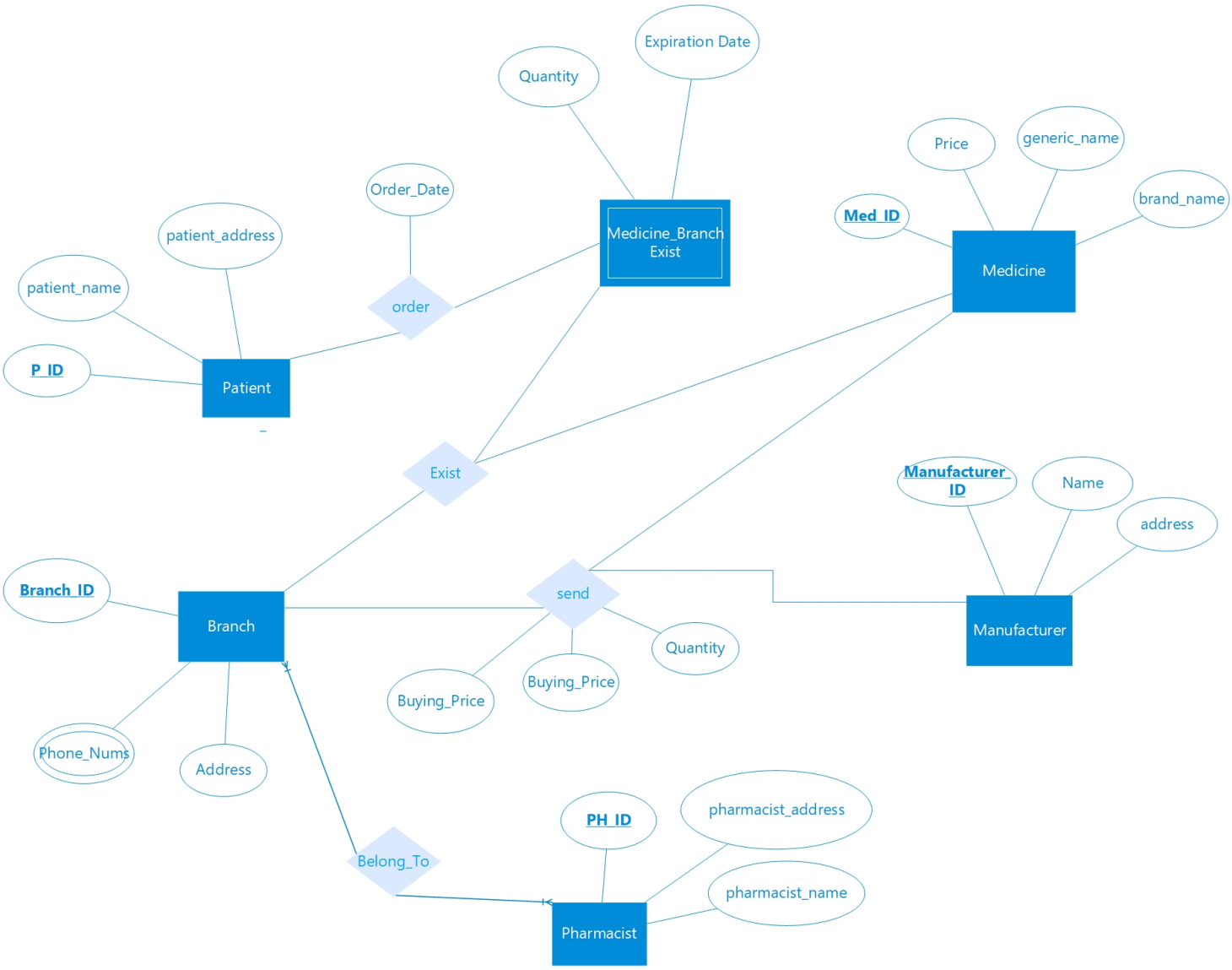
---

the project's title : Pharmacy Database

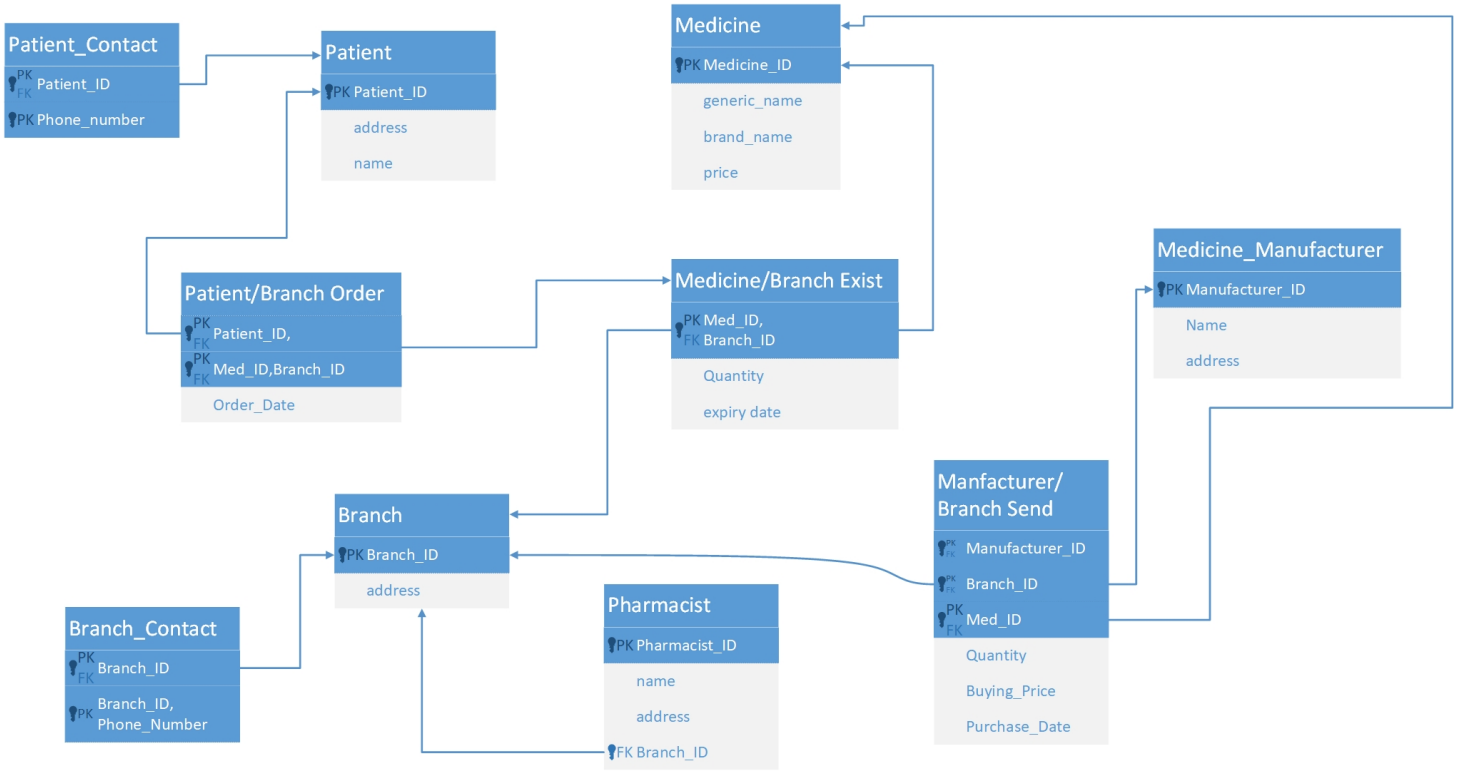
description:

---

entity diagram (ERD)



Relational Schema



SQL

1: Table Creation

1.1: Base Entities

```
Create table Patient(
patient_ID int primary key identity,
patient_address varchar(100),
patient_name varchar(868),
);

Create table Pharmacist(
pharmacist_ID int primary key identity,
pharmacist_name varchar(868),
pharmacist_address varchar(100),

branch_ID int,
);

Create table Medicine(
med_ID int primary key identity,
generic_name varchar(50),
brand_name varchar(50) not null,
price decimal (5,2),
);

Create table Branch(
branch_ID int primary key identity,
branch_address varchar(100),
);

Create table Manufacturer(
manufacturer_ID int primary key identity,
manufacturer_name varchar(50) unique not null,
manufacturer_address varchar(100),
);
```

1.2 : MultiValue attributes

```
Create table Patient_Contact(
patient_ID int,
phone_number varchar(15),
primary key(patient_ID,phone_number),
);

Create table Branch_Contact(
branch_ID int,
phone_number varchar(15),
primary key(branch_ID,phone_number),
);
```

1.3: Relationships

```
Create table Patient_Order(
patient_ID int,
med_ID int,
 int,
order_date date,
primary key(patient_ID,med_ID,branch_ID),
);

Create table Med_Exist(
med_ID int,
branch_ID int,
quantity int,
expiryDate date,
primary key(med_ID,branch_ID)
);

Create table Manufacturer_Send(
manufacturer_ID int,
branch_ID int,
med_ID int,
med_quantity int,
buying_price decimal(5,2),
purchase_date date,
primary key(manufacturer_ID,branch_ID,med_ID),
);
```

2.Defining Constraints (PK, FK)

```
alter table Pharmacist
add constraint Pharmacist_Branch_FK foreign key(branch_ID) references Branch;


alter table Med_Exist
add constraint med_exsitingMed_FK foreign key(med_ID) references Medicine;

alter table Med_Exist
add constraint exsitingMed_branch_FK foreign key(branch_ID) references Branch;

--Patient Order from Existing_Medicine in a branch
alter table Patient_Order
add constraint patient_order_FK foreign key(patient_ID) references patient;

alter table Patient_Order
add constraint order_existingMed_FK foreign key(med_ID,branch_ID) references Med_Exist;


alter table Manufacturer_Send
add constraint manufacturer_Send_FK foreign key(manufacturer_ID) references Manufacturer;

alter table Manufacturer_Send
add constraint Send_Medicine_FK foreign key(med_ID) references Medicine;

alter table Manufacturer_Send
add constraint Medicine_Branch_FK foreign key(branch_ID) references Branch;


alter table Patient_Contact
add constraint patientContact_patient foreign key(patient_ID) references Patient;

alter table Branch_Contact
add constraint branchContact_Branch foreign key(branch_ID) references Branch;
```

3.Queries.

3.1 : selecting the name and number of different medicines in each branch (retrieve data using join).

```
select B.branch_ID, M.brand_name , quantity
from Med_Exist as ME
join Medicine as M
on M.med_ID = ME.med_ID
join Branch as B
on B.branch_ID = ME.branch_ID;
```

3.2: selecting Pharmacists names and which branch they work on (retrieve data using subqueries).

```
select Ph.pharmacist_name, branch_ID
from Pharmacist as Ph
where Ph.branch_ID IN(
select B.branch_ID from
Branch as B
)
```

4.Aggregate Functions.

4.1 : Pharmacist expenses.

```
select SUM(buying_price) as 'Pharmacist expenses'
from Manufacturer_Send;
```

4.2 : Pharmacist revenues

```
select SUM(Price) as 'Pharmacist revenues'
from Orders;
```

4.3: Average medicine price

```
select AVG(price) as 'Average Medicine Price'
from Medicine;
```

4.4: (retrieve data using a group by and having).

4.4.1 : certain Medicine expenses.

```
select med_ID,SUM(buying_price) as 'Medicine expenses'
from Manufacturer_Send
group by med_ID;
```

4.4.2: Certain Medicine revenues.

```
select Medicine,SUM(Price) as 'revenues'
from Orders
group by(Medicine);
```

5.Views.

```
Create view Orders
as
select patient_name 'Customer Name', brand_name 'Medicine', price  as 'Price', branch_address as 'Pharmacy Branch'
from Patient_Order as PO
join Patient as P
    on PO.patient_ID = P.patient_ID
join Branch as B
    on B.branch_ID = PO.branch_ID
join Med_Exist as ME
    on ME.med_ID = PO.med_ID AND ME.branch_ID = B.branch_ID
join Medicine as M
    on M.med_ID = ME.med_ID
```

6.Stored procedures.

```
create proc OrderMedicine(
@patientID int,
@medID int,
@branchID int,
@orderDate date
)
as
begin

Insert into Patient_Order
Values (@patientID,@medID,@branchID,@orderDate);

end
```
