**Azerbaijan State Oil and Industry University**

**Faculty:** Information Technology and Management

**Department:** Computer Engineering

**Group:** 652.20E

**Specialty:** Computer Engineer

**Subject:** Database Systems

**Topic:** Pharmacy Database

**Course Work**

**Student:** Nihad Asgarli Abaset

**Teacher:** Ibrahimova Ellada Nazim

**Department Director:** Rahimova Nazile Ali.

BAKI-2021

**Contents**

**Chapter 1. Commuter Train schedule database**

1.1 Introduction…………………………………………………………..3

1.2 ER diagram…………………………………………………………..4

1.3 Database diagram……………………………………………………5

**Chapter 2. Creating tables in the Commuter Train schedule database**

2.1 List of the tables…………………………………………………......6

2.2 Alter, drop columns…………………………………………………..7

2.3 SQL constraints……………………………………………………...8

**Chapter 3. SQL queries. Using SQL to Retrieve Information from tables.**

3.1 SQL select, order by, where command, update, delete, statements…15

3.2 The in, between, like, and, or, top, operators………………………..18

3.3 Aggregate functions………………………………………………....20

3.4 Joining tables………………………………………………………...22

3.5 SQL subqueries……………………………………………………...24

3.6 SQL Views……………………………………………………….…25

**Conclusion**……………………………………………………………………....27

**Pharmacy Database**

* 1. **Introduction**

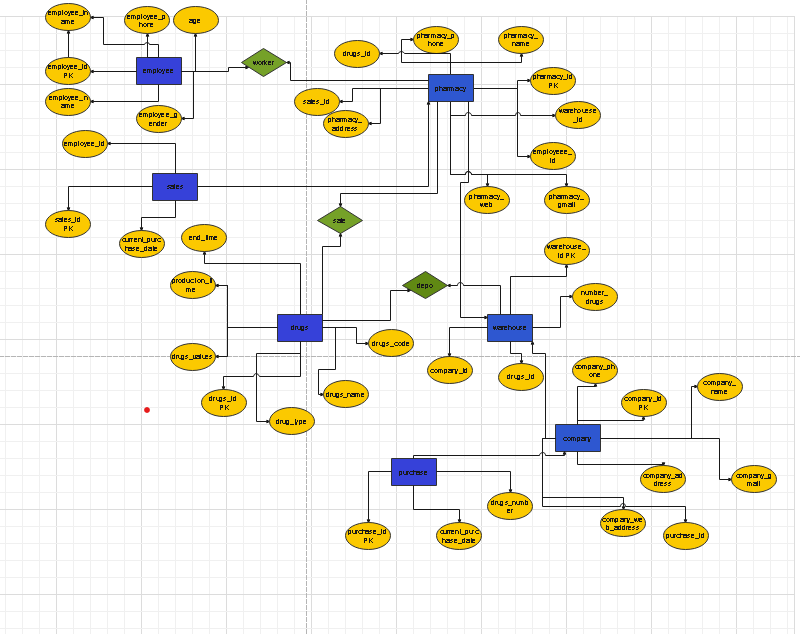
Trains are one of the most widely used transportation.

The main purpose of maintaining database for Commuter Train schedule is to reduce the manual errors involved in the booking and cancelling of tickets and make it convenient for customers and providers to maintain the data about their customers, also about transactions and about dates available at them. Due to automation many loopholes that exist in the manual maintenance of the records can be removed. The speed of obtaining and processing the data will be fast. For future expansion the proposed system can be web enabled so that clients can make various enquiries about trains between stations.

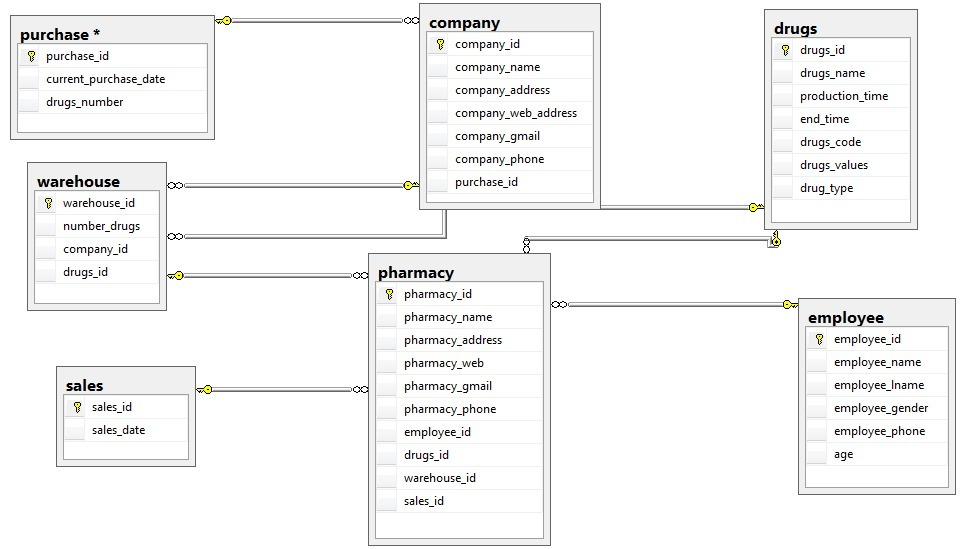
Due to this, sometimes a lot of problems occur and they are facing many disputes with customers. To solve the above problem, we design a data base which includes customer details, availability of dates in trains, transactions and their details.

**1.2 ER diagram**

The ER(Entity Relationship) diagram represents the model of Pharmacy schedule Entity. The entity-relationship diagram of Pharmacy schedule shows all the visual instrument of database tables and relation between sales, drugs, companies and etc. It used structure data and to define relationships between structured data groups of Pharmacy schedule functionalities. There are common features of Pharmacy schedule database such as reports and orders.



**1.3DatabaseDiagram**



**Creating tables Pharmacy database**

Tables are used to store data in the database. Creating a basic table involves naming the table and defining its columns and each column’s data type. The SQL CREATE TABLE statement is used to create a new table. The SQL Server INSERT INTO statement is used to add new rows of data to a table in the database.

**2.1 List of tables**

1) **pharmacy**

CREATE TABLE pharmacy(

pharmacy\_id int PRIMARY KEY,

pharmacy\_name varchar(20),

pharmacy\_address varchar(50),

pharmacy\_web varchar(50),

pharmacy\_gamil varchar(50),

pharmacy\_phone varchar(20),

employee\_id int foreign key references employee(employee\_id),

drugs\_id int foreign key references drugs(drugs\_id),

warehouse\_id int foreign key references warehouse(warehouse\_id) ,

sales\_id int foreign key references sales(sales\_id)

);

* pharmacy\_id - this is a unique ID of pharmacy
* pharmacy\_name -this is name of pharmacy
* pharmacy\_address -this is address of pharmacy
* pharmacy\_web -this is web of pharmacy
* pharmacy\_gamil -this is gender of admin
* pharmacy\_phone -this is address where admin works
* employee\_id- this is employee’s id which is referencing to the column employee\_id of the employee table
* drugs\_id- this is drugs’s id which is referencing to the column drugs\_id of the drugs table
* wharehouse\_id- this is wharehouse’s id which is referencing to the column wharehouse\_id of the employee table
* sales\_id- this is sales’s id which is referencing to the column sales\_id of the sales table

INSERT INTO pharmacy\_ (pharmacy\_id, pharmacy\_name, pharmacy\_address, pharmacy\_web,pharmacy\_gamil,pharmacy\_phone,employee\_id,drugs\_id,

wharehouse\_id,sales\_id) VALUES **(**1,’ yasil’,’ baki seher’,’ www.yasil.com, ’ yasil@gmail.com,’ 0516542222’,’ 2’)

**pharmacy**

****

**2.2 Alter, drop columns**

ALTER TABLE pharmacy

ADD employee\_id int

ADD is used to add columns into the existing table. Sometimes we may require to add additional information, in that case we do not require to create the whole database again, ADD comes to our rescue.

ALTER TABLE pharmacy

DROP COLUMN employee\_id

DROP COLUMN is used to drop column in a table. Deleting the unwanted columns from the table.

**2.3 SQL Constraints**

SQL Constraints are used to specify rules for the data in a table .Constraints are used to limit the type of data that can go into a table.

There is a primary key on the pharmacy\_id column of the pharmacy table. It means we cannot have two records having the same value of that field.

There is a foreign key on the head column of the pharmacy table. It means values match the pharmacy\_id column of the pharmacy\_ table.

With the following statement we add constraint to the position column of the pharmacy\_ table. It sets a default value for a column if no value is specified.

ALTER TABLE pharmacy

ADD CONSTRAINT CH\_email

CHECK(pharmacy\_gmail LIKE '%@mail.ru' or pharmacy\_gmail LIKE '%@gmail.com' or pharmacy\_gmail LIKE '%@yahhoo.com')

**2)** **drugs**

CREATE TABLE drugs(

drugs\_ID int PRIMARY KEY,

drugs\_name varchar ,

production\_time date,

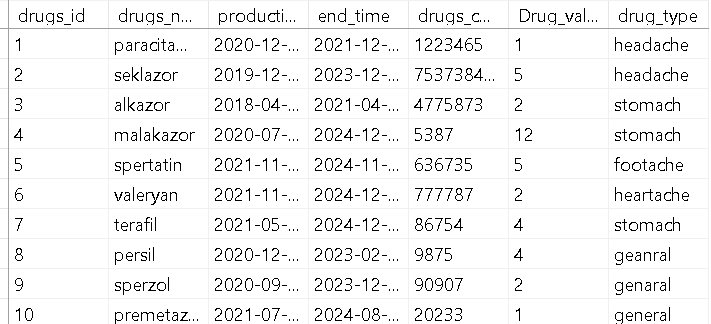
end\_time date,

drugs\_code int,

drug\_values char(10),

drug\_type varchar (20))

**drugs**



* drugs\_id - this is unique id for drugs
* drugs\_name - this is name of pharmacy
* production\_time -this is production time of drugs
* end\_time - this is end time of drugs
* drugs\_code - this is barcode of drugs
* Drug\_values - this is values of drugs
* drug\_type- this is type of drugs

**3)warehouse**

CREATE TABLE warehouse(

warehouse\_id int PRIMARY KEY,

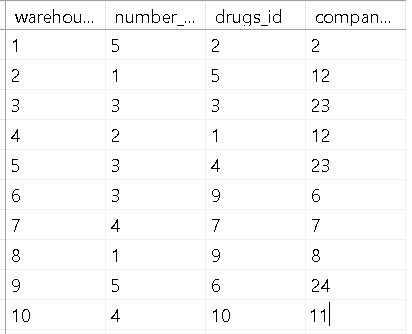
number\_drugs int,

drugs\_id int FOREIGN KEY REFERENCES drugs(drugs\_id)

company\_id int FOREIGN KEY REFERENCES company(company\_id),

);

**warehouse**

****

* warehouse\_id -this is unique ID of warehouse
* number\_drugs - this is number of drugs
* drugs\_id - this is name of wharehouse which is referencing to the column
* company\_id - this is name of wharehouse which is referencing to the column

**4)company**

create table company (

company\_id int primary key ,

company\_name varchar(20),

company\_address varchar (20),

company\_web\_address varchar (50),

company\_gmail varchar(50),

company\_phone int ,

purchase \_id int FOREIGN KEY REFERENCES purchase(purchase\_id)

);

**company**



* company\_id - this is unique ID of each company
* company\_name - this is name of company
* company\_address –this is address of each company
* company\_web\_address–this is web address of each company
* company\_phone - this is number of company
* purchase\_id - this is name of company which is referencing to the column

sp\_rename 'compan.company\_address','company\_web\_address';

sp\_rename is used to rename column.

SELECT\*FROM company

**5)sales**

CREATE TABLE sales(

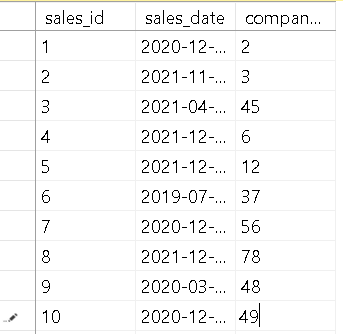
sales\_id int primary key,

sales\_date date,

employee\_id int foreign key references employee(employee\_id)

)

**sales**



* sales\_id - this is unique id of sales
* sales\_date- this is date of sales
* employee\_id - determines which employee is sold

**6)purchase**

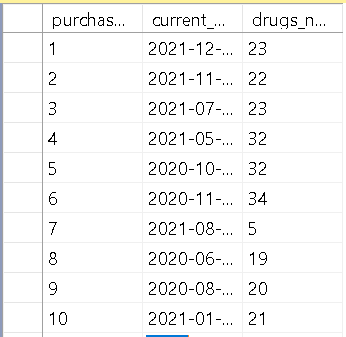
CREATE TABLE purchase (

purchase\_id int primary key,

current\_purchase\_date date,

drugs\_number int)

**purchase**

****

* purchase\_id- this is unique ID of purchase
* current\_purchase\_date - now shows the received date
* drugs\_number- certain drugs of number

**7)employee**

CREATE TABLE employee(

employee\_id int primary key,

employee\_name varchar(10),

employee\_lname varchar(15),

employee\_dob date,

employee\_gender varchar(5),

employee\_phone varchar(20)

age int

)

**employee**



* employee\_id- this is unique ID of the employe
* employee\_name- looks at his name
* employee\_lname- shows the second name
* employee\_gender- indicates gender
* employee\_phone- shows the contact number
* age-show employee of age

**SQL Queries. Using SQL to Retrieve Information from tables**

**3.1 SQL select, order by, where command, update, delete statements**

1.This query is to find all information about employee with id

SELECT\*FROM employee

WHERE employee\_id=5;



2.This query is used to sort employee’s and company’s data with ascending to the age. The ASC command is used to sort data returned in ascending order. The following SQL statements selects some columns from the “employee” and “company” table, sorted by the “age” column:

SELECT e.employee\_name as employee, c.company\_name as company

FROM employee e,company c

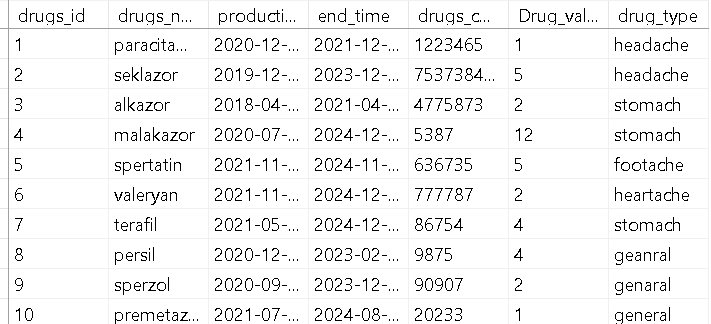
WHERE e.employee\_id=c.company\_id

ORDER BY e.age ASC;



3. The UPDATE statement is used to modify the existing records in a table. The WHERE clause specifies which record(s) that should be updated.

Before UPDATE:

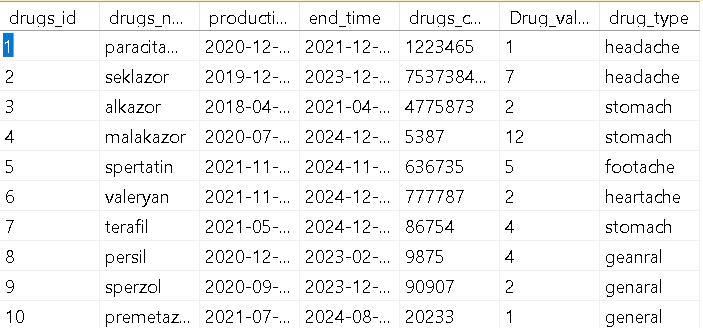


UPDATE drugs

SET Drug\_values=7

WHERE drugs\_id=2;

After UPDATE:



4. The DELETE statement is used to delete existing records in a table. The WHERE clause specifies which record(s) should be deleted. If you omit the WHERE clause, all records in the table will be deleted!



DELETE FROM employee

WHERE employe\_id=6;

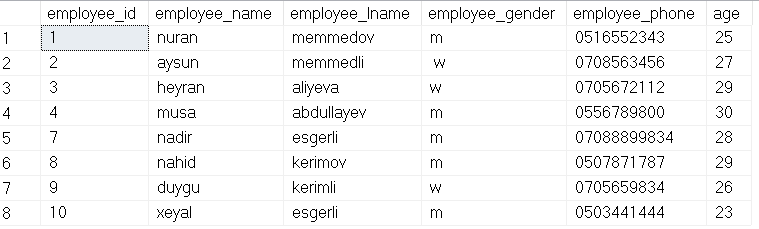


**3.2 The in, between, like, and, or, top, operators**

1. shows us employees between the ages of 20-30.

SELECT \* FROM employee

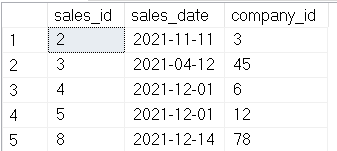
WHERE age 20 between 20 and 30



2.The BETWEEN operator selects values within a given range. The values can be numbers, text, or dates. In this survey, we will show which company are sold in 2021.

SELECT \* FROM sales

WHERE sales\_date BETWEEN '2021-01-01' AND '2021-12-31';



3. The IN operator allows you to specify multiple values in a WHERE clause. The IN operator is a shorthand for multiple OR conditions. Below we present the data of companies 5,7,9.

SELECT \* from company

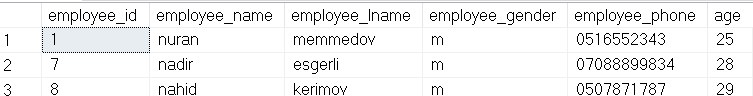
where company\_id in (5,7,9)



4. The LIKE operator is used in a WHERE clause to search for a specified pattern in a column. The following query gives the names of employees starting with n.

SELECT \* FROM employee

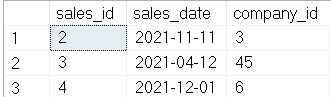
WHERE employee\_name LIKE 'n%';



5.SELECT TOP clause is used to specify the number of records to return.

SELECT TOP(50) PERCENT \* FROM sales

WHERE sales\_date LIKE '2021%\_\_\_\_';



**3.3 Aggregate functions**

1.The COUNT() function returns the number of rows that matches a specified criterion. we showed how many women workers there are

SELECT count(employee\_gender)

as employee\_women from employee

where not employee\_gender='m'



2.The AVG() function returns the average value of a numeric column. Let's show what is the average age of women workers

SELECT avg (age)

as age\_average from employee

where not employee\_gender='m'



3.The MIN() function returns the smallest value of the selected column. The MAX() function returns the largest value of the selected column. Let's find out the maximum number of older employees among the total employees

SELECT MAX(age) AS "the oldest\_employee"

FROM employee;



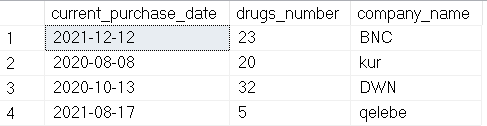
**3.4 Joining tables**

1.The INNER JOIN keyword selects records that have matching values in both tables. The following survey asks how many drugs were purchased on which purchase date and from which company

SELECT current\_purchase\_date, drugs\_number,company\_name

FROM purchase p INNER JOIN company c

ON p.purchase\_id=c.purchase\_id;

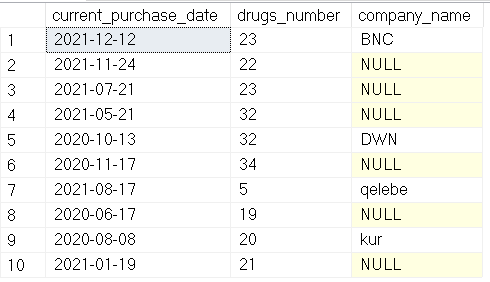


2. The LEFT JOIN keyword returns all records from the left table, and the matching records from the right table. He result is 0 records from the right side, if there is no match. We need to determine the company names of all the drugs purchased at the same time

SELECT current\_purchase\_date, drugs\_number,company\_name

FROM purchase p LEFT JOİN company c

ON p.purchase\_id=c.purchase\_id;

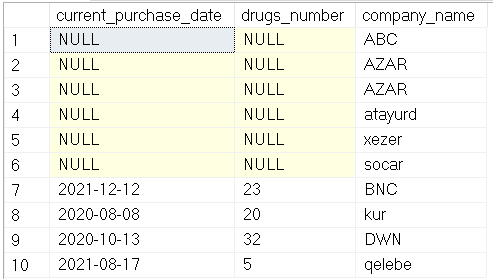


3.The RIGHT JOIN keyword returns all records from the right table, and the matching records from the left table.

SELECT current\_purchase\_date, drugs\_number,company\_name

FROM purchase p RİGHT JOİNcompany c

ON p.purchase\_id=c.purchase\_id;



**3.5 SQL Subqueries**

1. The following survey shows when a drug called terafil was sold

select sales\_date from sales

where drugs\_id=

(select drugs\_id from drugs

where drugs\_name='terafil')

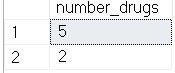


2. We showed him how many medicines for headaches are in stock

select number\_drugs from warehouse

where drugs\_id in (select drugs\_id

from drugs where drug\_type='headache')



**3.6 SQL Views**

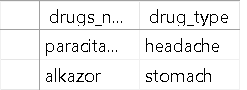
1. A view is a virtual table based on the result-set of an SQL statement. In the following statement I create view from Reservation table. It contains ID of customers, ID of tickets and accommodation date which are not used.

CREATE VIEW [non-used\_sales]

AS SELECT drugs\_name,drugs\_type

FROM drugs

WHERE end\_time BETWEEN '2021-01-01' AND '2021-12-31';



**Conclusion**

Every pharmacy faces certain problems. For example, there may be problems such as lack of medication on time, lack of medication on time. We use a database to solve these problems. This makes it easier for both customers and employees. Knowing the time of each employee. Every company wants to know how many drugs it has taken and to which pharmacy. In addition, it is easy and quick to learn certain information through surveys.