create table Operator (

OperatorID int not null,

OEmail VARCHAR2(30) NOT NULL,

OPassword VARCHAR2(30) NOT NULL,

OMobileN varchar2(15) not null,

OName varchar2(30) not null,

OSex varchar2(6),

OBloodG varchar2(3) not null,

OAddress varchar2(30),

OAlternativeMN varchar2(15),

OperatorRegDate DATE NOT NULL,

Approval VARCHAR2(3) NOT NULL,

CONSTRAINT Operator\_PK1 PRIMARY KEY (OperatorID, OEmail),

CONSTRAINT Operator\_UI1 UNIQUE (OEmail),

CONSTRAINT Operator\_UI2 UNIQUE (OMobileN)

);

create table Donor (

DonorID int not null,

DMobileN varchar2(15) not null,

DName varchar2(30) not null,

DBloodG varchar2(3) not null,

DSex varchar2(6),

DAddress varchar2(30),

DEmail varchar2(30),

DAlternativeMN varchar2(15),

RegistrationDate date not null,

CONSTRAINT Donor\_PK PRIMARY KEY (DonorID),

CONSTRAINT Donor\_UI UNIQUE (DMobileN)

);

create table Donation (

DonationID int not null,

DonorID int not null,

DMobileN varchar2(15) not null,

DName varchar2(30) not null,

DBloodG varchar2(3) not null,

DSex varchar2(6),

DAddress varchar2(30),

DEmail varchar2(30),

DAlternativeMN varchar2(15),

DonationDate date not null,

BloodAvailable VARCHAR2(3) NOT NULL,

CONSTRAINT Donation\_PK PRIMARY KEY (DonationID),

CONSTRAINT Donation\_FK FOREIGN KEY (DonorID) REFERENCES DONOR(DonorID)

);

create table Request(

RequestId int not null,

RName varchar2(30) not null,

RMobileN varchar2(15) not null,

RBloodG varchar2(3) not null,

RSex varchar2(6),

RAddress varchar2(50),

REmail varchar2(30),

RAlternativeMN varchar(15),

RequestDate date NOT NULL,

RApproval VARCHAR2(3) NOT NULL,

CONSTRAINT Request\_PK PRIMARY KEY (RequestId)

);

create table BloodAssignment (

BAssignID INT NOT NULL,

DonationID INT NOT NULL,

RequestID INT NOT NULL,

BloodG VARCHAR2(3) NOT NULL,

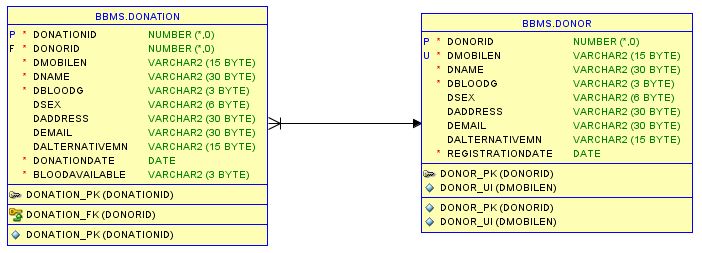
AssignDate DATE NOT NULL,

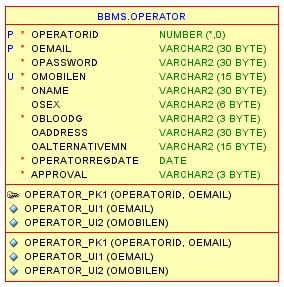
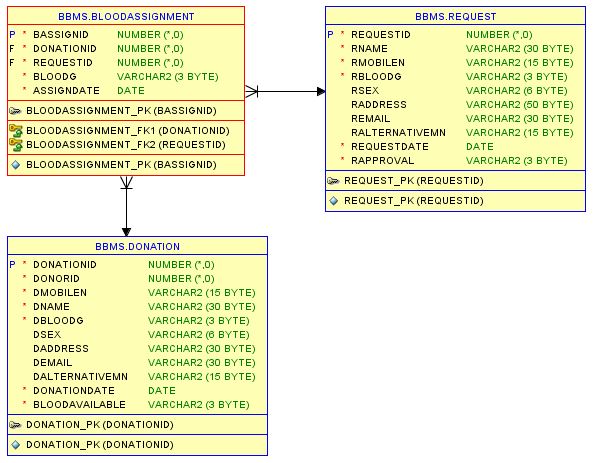
CONSTRAINT BloodAssignment\_PK PRIMARY KEY (BAssignID),

CONSTRAINT BloodAssignment\_FK1 FOREIGN KEY (DonationID) REFERENCES DONATION(DONATIONID),

CONSTRAINT BloodAssignment\_FK2 FOREIGN KEY (RequestID) REFERENCES REQUEST(REQUESTID)

);





Sequences:

CREATE SEQUENCE operatorIdSequence

MINVALUE 20001

START WITH 20001

INCREMENT BY 1;

CREATE SEQUENCE donorIdSequence

MINVALUE 1001

START WITH 1001

INCREMENT BY 1;

CREATE SEQUENCE donationIdSequence

MINVALUE 2001

START WITH 2001

INCREMENT BY 1;

CREATE SEQUENCE REQUESTIDSEQUENCE

MINVALUE 3001

START WITH 3101

INCREMENT BY 1;

CREATE SEQUENCE BASSIGNIDSEQUENCE

MINVALUE 5001

START WITH 5001

INCREMENT BY 1;

Procedure:

create or replace PROCEDURE getBloodListProc(

dBloodG IN DONATION.DBLOODG%TYPE,

res OUT SYS\_REFCURSOR

)

AS

BEGIN

OPEN res FOR

'SELECT DONATIONID, DONORID, DNAME, DSEX, DBLOODG, DMOBILEN, DEMAIL, DADDRESS, DALTERNATIVEMN, DONATIONDATE, BLOODAVAILABLE

FROM DONATION

WHERE DONATION.DBLOODG = dBloodG';

END;

create or replace PROCEDURE getDonorProc(

dMobile IN DONOR.DMOBILEN%TYPE,

dId OUT DONOR.DONORID%TYPE,

dName OUT Donor.DNAME%TYPE,

dBloodG OUT Donor.DBLOODG%TYPE,

dSex OUT Donor.DSEX%TYPE,

dAddress OUT Donor.DADDRESS%TYPE,

dEmail OUT Donor.DEMAIL%TYPE,

dAlternativeMN OUT Donor.DALTERNATIVEMN%TYPE,

dRegDate OUT DONOR.REGISTRATIONDATE%TYPE,

dMobileN OUT DONOR.DMOBILEN%TYPE

)

IS

BEGIN

SELECT DONORID, DNAME, DSEX, DBLOODG, DMOBILEN, DEMAIL, DADDRESS, DALTERNATIVEMN, REGISTRATIONDATE

INTO dId, dName, dSex, dBloodG, dMobileN, dEmail, dAddress, dAlternativeMN, dRegDate

FROM DONOR

WHERE DONOR.DMOBILEN = dMobile;

END;

create or replace PROCEDURE insertBloodAssignProc(

donationId IN BLOODASSIGNMENT.DONATIONID%TYPE,

requestId IN BLOODASSIGNMENT.REQUESTID%TYPE,

bloodG IN BLOODASSIGNMENT.BLOODG%TYPE

)

IS

CURSOR dIdVal IS SELECT bAssignIdSequence.NEXTVAL FROM DUAL;

bAssignID BLOODASSIGNMENT.BASSIGNID%TYPE;

BEGIN

OPEN dIdVal;

FETCH dIdVal INTO bAssignID;

INSERT INTO BLOODASSIGNMENT (BASSIGNID, REQUESTID, DONATIONID, BLOODG, ASSIGNDATE)

VALUES(bAssignID, requestId, donationId, bloodG, SYSDATE);

--UPDATE DONATION SET BLOODAVAILABLE = 'NO' WHERE DONATIONID = donationId;

CLOSE dIdVal;

END;

create or replace PROCEDURE insertDonationProc(

dMobile IN DONATION.DMOBILEN%TYPE,

dBloodG IN DONATION.DBLOODG%TYPE)

IS

CURSOR donorVal IS SELECT \* FROM DONOR WHERE DONOR.DMOBILEN = dMobile AND DONOR.DBLOODG = dBloodG;

CURSOR dIdVal IS SELECT donationIdSequence.NEXTVAL FROM DUAL;

dVal donorVal%ROWTYPE;

--dDonationDate DATE;

dDonationID DONATION.DONATIONID%TYPE;

BEGIN

OPEN donorVal;

OPEN dIdVal;

FETCH dIdVal INTO dDonationID;

--dDonationDate := SYSDATE;

--LOOP

FETCH donorVal INTO dVal;

--EXIT WHEN donorVal%NOTFOUND;

--IF dVal.DONORID>0 THEN

IF dVal.DMOBILEN = dMobile AND dVal.DBLOODG = dBloodG THEN

INSERT INTO DONATION (DONATIONID,DONORID,DMOBILEN, DNAME, DBLOODG, DSEX, DADDRESS, DEMAIL, DALTERNATIVEMN, DONATIONDATE, BLOODAVAILABLE)

VALUES (dDonationID, dVal.DONORID, dVal.DMOBILEN, dVal.DNAME, dVal.DBLOODG, dVal.DSEX, dVal.DADDRESS, dVal.DEMAIL, dVal.DALTERNATIVEMN, SYSDATE,'YES');

END IF;

--END LOOP;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

dbms\_output.put\_line('No such customer!');

CLOSE donorVal;

CLOSE dIdVal;

END;

create or replace PROCEDURE insertDonorProc(

dMobile IN Donor.DMOBILEN%TYPE,

dName IN Donor.DNAME%TYPE,

dBloodG IN Donor.DBLOODG%TYPE,

dSex IN Donor.DSEX%TYPE,

dAddress IN Donor.DADDRESS%TYPE,

dEmai IN Donor.DEMAIL%TYPE,

dAlternativeMN IN Donor.DALTERNATIVEMN%TYPE)

IS

CURSOR dIdVal IS SELECT donorIdSequence.NEXTVAL FROM DUAL;

dDonorID Donor.DONORID%TYPE;

BEGIN

OPEN dIdVal;

FETCH dIdVal INTO dDonorID;

--idVal := donorIdSequence.nextval;

INSERT INTO Donor ("DONORID","DMOBILEN", "DNAME", "DBLOODG", "DSEX", "DADDRESS", "DEMAIL", "DALTERNATIVEMN", "REGISTRATIONDATE")

VALUES (dDonorID, dMobile, dName, dBloodG, dSex, dAddress, dEmai, dAlternativeMN, SYSDATE);

COMMIT;

END;

create or replace PROCEDURE insertOperatorProc(

oEmail IN OPERATOR.OEmail%TYPE,

oPassword IN OPERATOR.OPassword%TYPE,

oMobile IN OPERATOR.OMobileN%TYPE,

oName IN OPERATOR.OName%TYPE,

oBloodG IN OPERATOR.OBloodG%TYPE,

oSex IN OPERATOR.OSex%TYPE,

oAddress IN OPERATOR.OAddress%TYPE,

oAlternativeMN IN OPERATOR.OAlternativeMN%TYPE)

IS

CURSOR oIdVal IS SELECT operatorIdSequence.NEXTVAL FROM DUAL;

dOperatorID OPERATOR.OperatorID%TYPE;

BEGIN

OPEN oIdVal;

FETCH oIdVal INTO dOperatorID;

--idVal := donorIdSequence.nextval;

INSERT INTO OPERATOR (OPERATORID,OEMAIL,OPASSWORD,OMOBILEN, ONAME, OBLOODG,OSEX, OADDRESS,OALTERNATIVEMN, OPERATORREGDATE, APPROVAL)

VALUES (dOperatorID, oEmail, oPassword, oMobile, oName, oBloodG, oSex, oAddress, oAlternativeMN, SYSDATE, 'NO');

COMMIT;

END;

create or replace PROCEDURE insertRequestProc(

rMobile IN REQUEST.RMOBILEN%TYPE,

rName IN REQUEST.RNAME%TYPE,

rBloodG IN REQUEST.RBLOODG%TYPE,

rSex IN REQUEST.RSEX%TYPE,

rAddress IN REQUEST.RADDRESS%TYPE,

rEmai IN REQUEST.REMAIL%TYPE,

rAlternativeMN IN REQUEST.RALTERNATIVEMN%TYPE)

IS

CURSOR rIdVal IS SELECT requestIdSequence.NEXTVAL FROM DUAL;

rRequestID REQUEST.REQUESTID%TYPE;

BEGIN

OPEN rIdVal;

FETCH rIdVAl INTO rRequestID;

INSERT INTO REQUEST ("REQUESTID","RMOBILEN", "RNAME", "RBLOODG", "RSEX", "RADDRESS", "REMAIL", "RALTERNATIVEMN", "REQUESTDATE","RAPPROVAL")

VALUES (rRequestID, rMobile, rName, rBloodG, rSex, rAddress, rEmai, rAlternativeMN, SYSDATE, 'NO');

COMMIT;

END;

create or replace PROCEDURE updateDonationProc(

donatId IN NUMBER

)

AS

BEGIN

UPDATE DONATION SET BLOODAVAILABLE = 'NO' WHERE DONATIONID = donatId AND BLOODAVAILABLE = 'YES';

COMMIT;

END;

create or replace PROCEDURE updateOperatorProc(

opId IN NUMBER

)

AS

BEGIN

UPDATE OPERATOR SET APPROVAL='YES' WHERE OPERATORID = opId AND APPROVAL = 'NO';

COMMIT;

END;

create or replace PROCEDURE updateRequestProc(

reqId IN NUMBER

)

AS

BEGIN

UPDATE REQUEST SET RAPPROVAL = 'YES' WHERE REQUESTID = reqId AND RAPPROVAL = 'NO';

COMMIT;

END;

Function:

create or replace FUNCTION getDonationIdFunc (

bloodG IN VARCHAR2,

available IN VARCHAR2

)

RETURN SYS\_REFCURSOR

AS

result\_List SYS\_REFCURSOR;

BEGIN

OPEN result\_List

FOR SELECT \* FROM DONATION WHERE DBLOODG = bloodG AND BLOODAVAILABLE = available;

RETURN result\_List;

END;

create or replace FUNCTION getDonationListFunc (

available IN VARCHAR2

)

RETURN SYS\_REFCURSOR

AS

result\_List SYS\_REFCURSOR;

BEGIN

OPEN result\_List

FOR SELECT \* FROM DONATION WHERE BLOODAVAILABLE = available;

RETURN result\_List;

END;

create or replace FUNCTION getDonorFunc (

mobile IN VARCHAR2

)

RETURN SYS\_REFCURSOR

AS

result\_List SYS\_REFCURSOR;

BEGIN

OPEN result\_List

FOR SELECT \* FROM DONOR WHERE DMOBILEN = mobile;

RETURN result\_List;

END;

create or replace FUNCTION getOperatorListFunc (

available IN VARCHAR2

)

RETURN SYS\_REFCURSOR

AS

result\_List SYS\_REFCURSOR;

BEGIN

OPEN result\_List

FOR SELECT \* FROM OPERATOR WHERE APPROVAL = available;

RETURN result\_List;

END;

create or replace FUNCTION getRequestFunc (

mobile IN VARCHAR2

)

RETURN SYS\_REFCURSOR

AS

result\_List SYS\_REFCURSOR;

BEGIN

OPEN result\_List

FOR SELECT \* FROM REQUEST WHERE RMOBILEN = mobile;

RETURN result\_List;

END;

create or replace FUNCTION getRequestListFunc

RETURN SYS\_REFCURSOR

AS

result\_List SYS\_REFCURSOR;

BEGIN

OPEN result\_List

FOR SELECT \* FROM REQUEST;

RETURN result\_List;

END;

create or replace FUNCTION getSellDataFunc

RETURN SYS\_REFCURSOR

AS

result\_List SYS\_REFCURSOR;

BEGIN

OPEN result\_List

FOR SELECT \* FROM BLOODASSIGNMENT;

RETURN result\_List;

END;

create or replace FUNCTION getUserPasswordFunc (

userN IN VARCHAR2

)

RETURN SYS\_REFCURSOR

AS

result\_List SYS\_REFCURSOR;

BEGIN

OPEN result\_List

FOR SELECT \* FROM OPERATOR WHERE OEMAIL = userN AND APPROVAL = 'YES';

RETURN result\_List;

END;