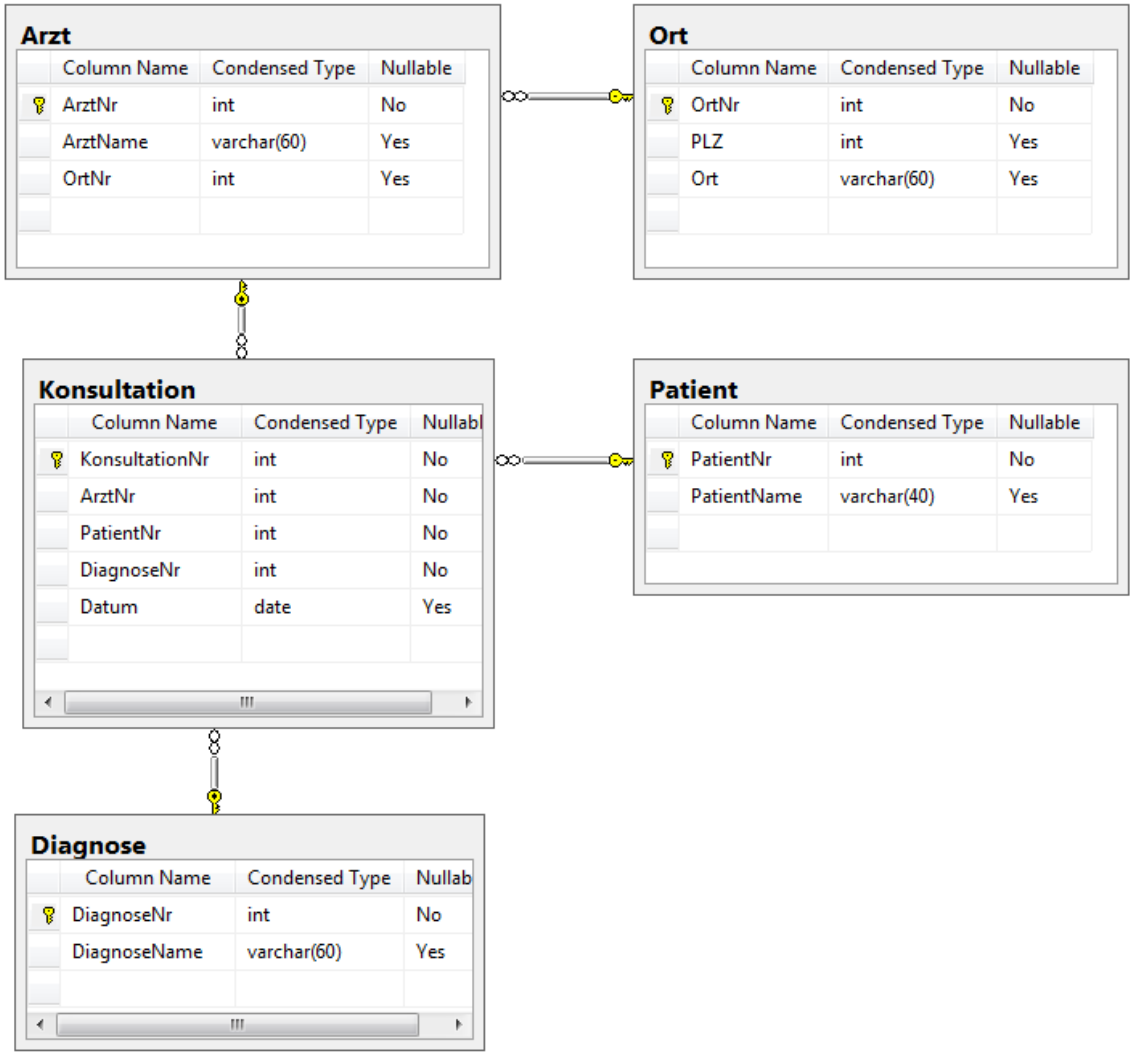
Aufgaben: Views / Stored Procedures (Lösungen)

Datenmodell



Implementieren Sie das obige Datenmodel unter Ihrem Benutzer-Account in der SQL-Server Datenbank.

Teil 1: Datenmodell implementieren

A1.1

Erstellen Sie die Textdatei CREATE_TABLES.SQL mit den CREATE TABLE Befehlen.

```
siehe SQL-Script Datei
```

A1.2

Erstellen Sie die Beziehungen zwischen den Tabellen (referenzielle Integrität).

```
alter table Arzt
add constraint fk_arzt_ort foreign key  (OrtNr)
       references Ort (OrtNr)

alter table Konsultation
    add constraint fk_konsultation_arzt foreign key  (ArztNr)
       references Arzt (ArztNr)
```

A1.3

Fügen Sie die Testdaten in die Tabellen ein. Prüfen Sie ob die referenzielle Integriät erfüllt wird.

```
siehe SQL-Script Datei
```

A1.4

Erstellen Sie eine View welche alle Patienten mit Diagnose 'ANGINA' inkl. den Arzt-Namen listet.

```
create view VIEW_ARZT_PATIENT_ANGINA
as
select p.PatientNr, p.PatientName, a.ArztNr, a.ArztName, d.DiagnoseName
from patient p inner join konsultation k
            on p.PatientNr = k.PatientNr
        inner join arzt a
            on k.ArztNr = a.ArztNr
        inner join Diagnose d
            on d.DiagnoseNr = k.DiagnoseNr
        where upper(d.DiagnoseName) = 'ANGINA'
```

Teil 2: Stored Procedures

A2.1

Erstellen Sie die gespeicherte Prozedur ***scores***, welche die Summe der 5 übergebenen Parameter (Datentyp smallint) berechnet und als Output Parameter zurückliefert.

```
CREATE PROCEDURE scores
      @score1 smallint,
      @score2 smallint,
      @score3 smallint,
      @score4 smallint,
      @score5 smallint,
      @mySum smallint OUTPUT
AS
BEGIN
      SELECT @mySum = (@score1 + @score2 + @score3 + @score4 + @score5)
END

-- Aufruf
DECLARE @SumScore smallint
EXEC scores 10, 9, 8, 8, 10, @SumScore OUTPUT
SELECT 'Die Summe ist: ', @SumScore
GO
```

A2.2

Verwenden Sie für den obigen ***scores*** Prozeduraufruf benannte Parameter in der Form *Parametername = Wert*

```
DECLARE @SumScore smallint
EXEC scores @score1 = 10,
                  @mySum = @SumScore OUTPUT,
                  @score3 = 8,
                  @score2 = 9,
                  @score4 = 8,
                  @score5 = 10
SELECT 'Die Summe ist: ', @SumScore
GO
```

A2.3

Erstellen Sie nach oben stehendem Vorbild eine gespeicherte Prozedur, welche entweder die Summe oder der Mittelwert der übergebenen Parameter berechnet. Ein Parameter bestimmt welche Funktion berechnet wird!

```
CREATE PROCEDURE scores_
@i,
@score1 smallint,
@score2 smallint,
@score3 smallint,
@score4 smallint,
@score5 smallint,
@myAvg int OUTPUT
AS
If @i=1
SELECT @myAvg =(@score1 + @score2 + @score3 + @score4 + @score5)
Else
SELECT @myAvg =(@score1 * @score2 * @score3 * @score4 * @score5)
```

A2.4

Erstellen Sie eine Stored Procedures die die Anzahl Konsultationen einer bestimmten Diagnose ermittelt.

```
CREATE PROCEDURE AnzahlKonsultationen @DiagnoseName nvarchar(30)
AS
    Select count(*) as Anzahl from KONSULTATION k inner join DIAGNOSE d
             on k.DiagnoseNr = d.DiagnoseNr
      where d.DiagnoseName = @DiagnoseName

exec AnzahlKonsultationen 'Angina'
```

A2.5

Erstellen Sie eine Stored Procedure welche alle Aerzte listet.

```
CREATE PROCEDURE ArztListe
AS
    Select * from Arzt
go

exec ArztListe
```

A2.6

Erstellen Sie eine Stored Procedures die eine neue Ortschaft (PLZ, Ort) einfügt.

```
CREATE PROCEDURE InsertOrt @PLZ int,  @Ort nvarchar(50)
AS
    Insert into Ort (PLZ, ORT )
      Values( @PLZ, @Ort )
go
exec InsertOrt 3000, 'Bern'
```