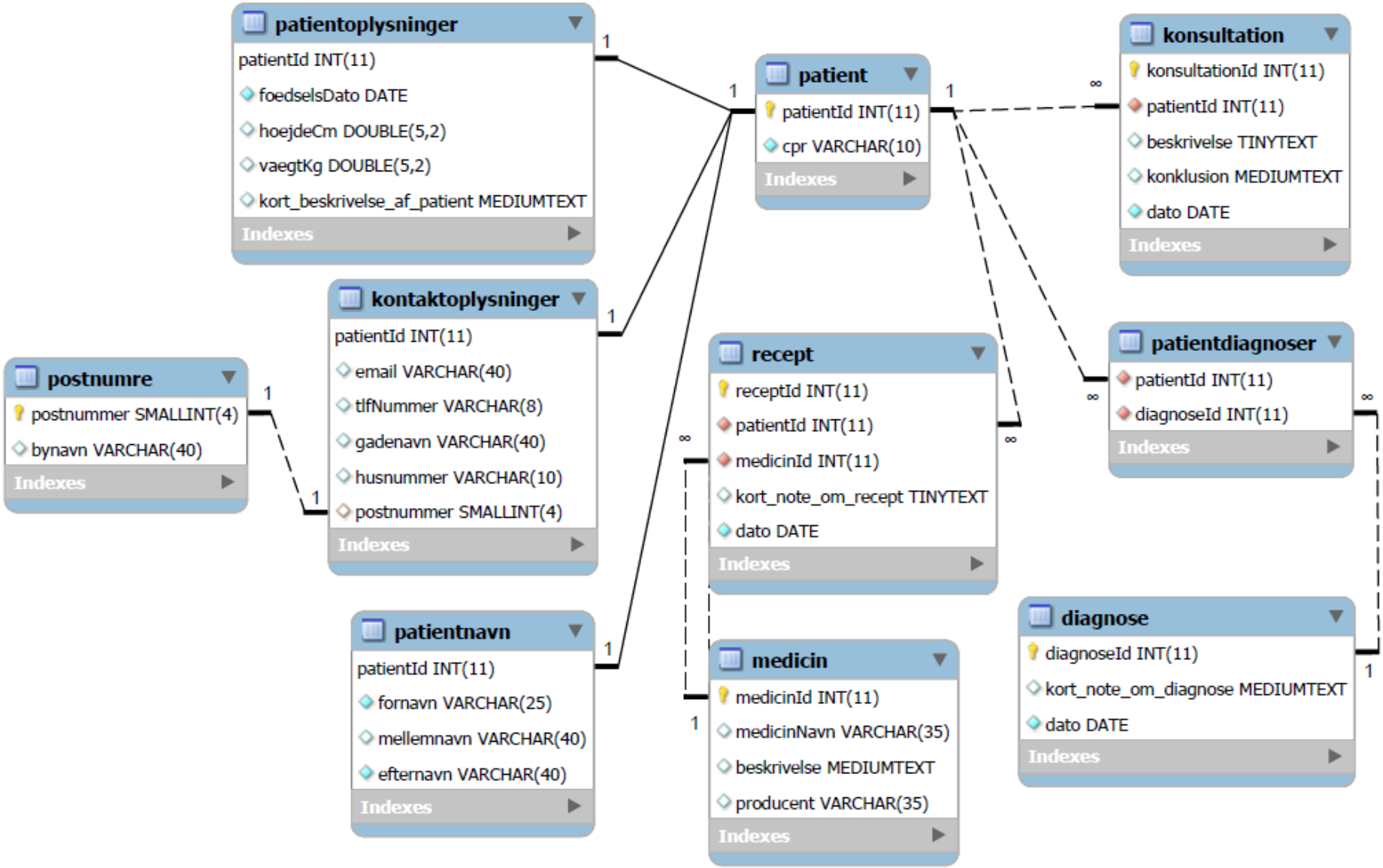
Entity relationship diagram



Forklaringer til ER og DDL

Normaliseringsformerne

Databasen er normaliseret efter de 3. første normaliseringsformer.

NF1: handler om gruppering af data

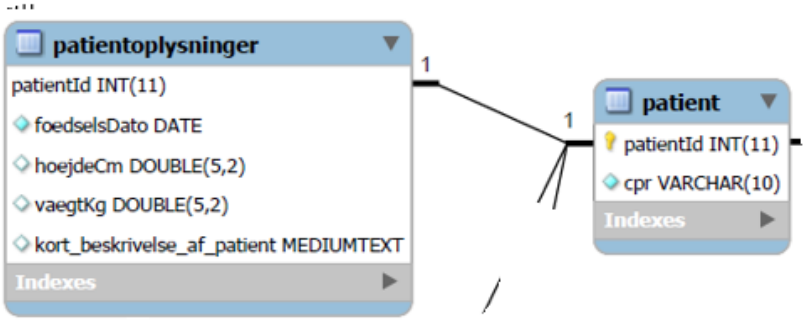
- Hver kolonne indeholder kun atomik data, dvs. der er ikke nogen mellemregninger.
  Dermed kommer fx alder ikke med, fordi den regnes ud fra fødselsdatoen.
  Hver celle har kun en værdi og værdierne i hver kolonne er samme datatype.
  Der er ikke nogen repetende grupper af data i kolonnerne.

NF2: handler om delvise afhængigheder

- Alle ikke nøgler kolonner er kun afhængige af hele PK

NF3: handler om transitive afhængigheder

- Ingen kolonne er afhængig af PK gennem en anden kolonne.

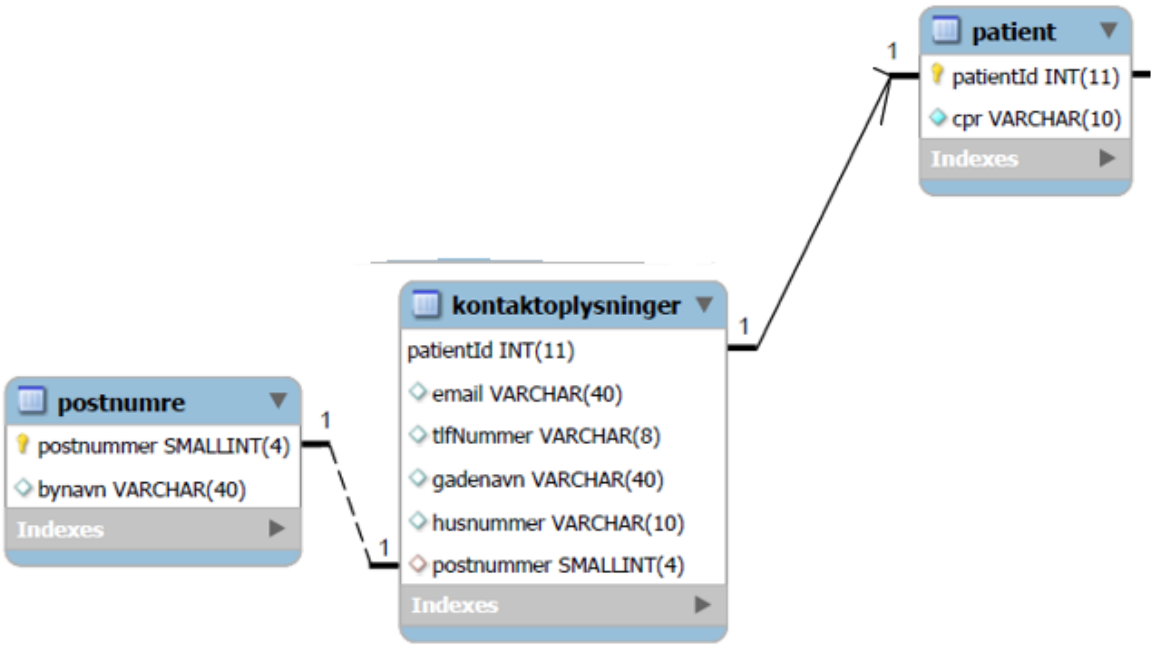


Relationen mellem patientoplysninger og patient er en-til-en, fordi en patien kun kan have et sæt oplysninger og oplysninger hører til én patient.
For at lave en-til-en -relationen, bruger man PK fra parent-tabel (patient) både som PK og FK i child-tabellen (patientOplysninger).

Når PK bliver slettet eller opdateret, gør man det samme med FK med cascade

```
CREATE TABLE patientOplysninger(
patientId INT PRIMARY KEY,
foedselsDato DATE NOT NULL,
hoejdeCm DOUBLE(5,2),
vaegtKg DOUBLE(5,2),
kort_beskrivelse_af_patient MEDIUMTEXT,
FOREIGN KEY (patientId) REFERENCES patient (patientId)  ON UPDATE CASCADE ON DELETE CASCADE
);
```

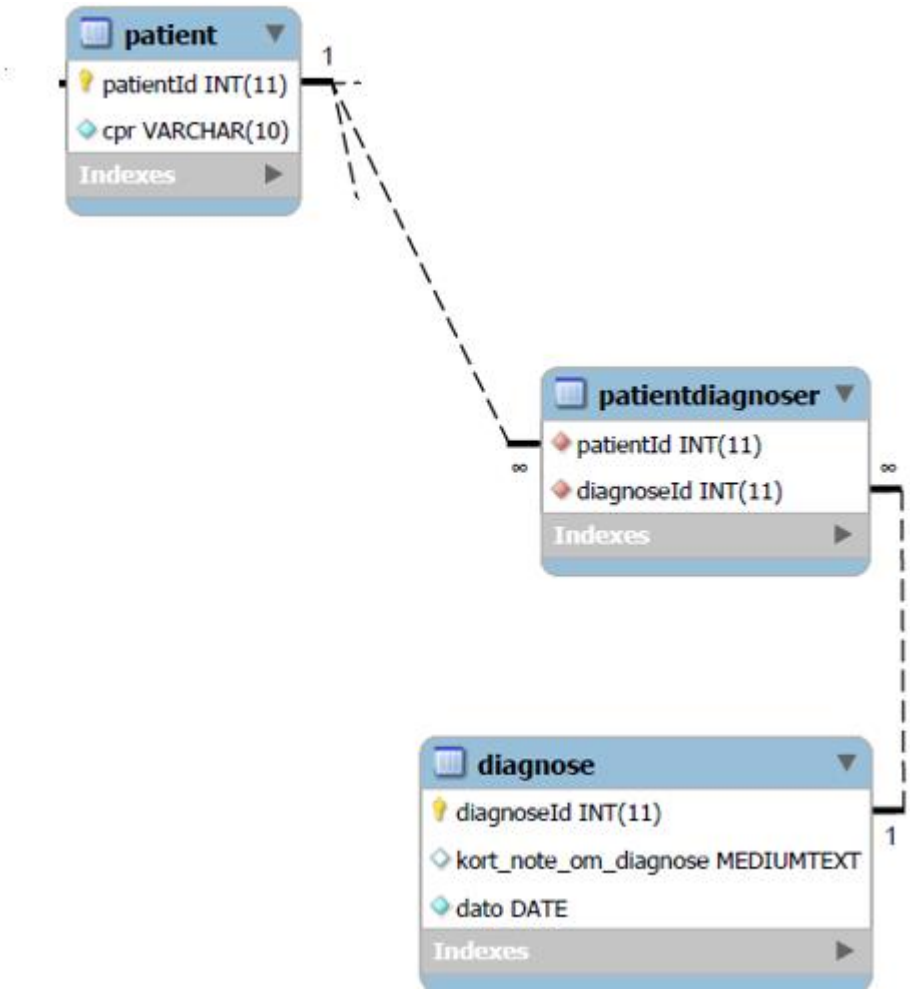
```
CREATE TABLE patient(
patientId INT PRIMARY KEY AUTO_INCREMENT,
cpr VARCHAR(10) UNIQUE NOT NULL
);
```



Kontaktoplysninger har en-til-en -relation både til patient og postnumre.
Der kan kun være knyttet ét postnummer til hver adresse og hver patient kan kun have én adresse.
Men her kan vi ikke bruge både postnummer, som er PK i postnumre, og patientId, som er PK i patient, som kombineret PK og FK, fordi det ville bryde 2. normaliseringsform.
Alle ikke-nøgler kolonner ville ikke være afhængige af hele PK, fordi fx gadenavn ville ikke være afhængig af postnummer, men vil kun høre til patientId.
Jeg giver kontaktoplysninger patientId som PK og FK, men postnummer kun som FK, men den skal være unik, sådan at den kun kan tage én gang den samme værdi.

```
CREATE TABLE postnumre(
postnummer SMALLINT(4) PRIMARY KEY,
bynavn VARCHAR(40),
CHECK (postnummer BETWEEN 1000 AND 2000)
);
```

```
CREATE TABLE kontaktOplysninger(
patientId INT PRIMARY KEY,
email VARCHAR(40),
tlfNummer VARCHAR(8),
gadenavn VARCHAR(40),
husnummer VARCHAR(10),
postnummer SMALLINT(4) UNIQUE,
FOREIGN KEY (patientId) REFERENCES patient (patientId) ON UPDATE CASCADE ON DELETE CASCADE,
FOREIGN KEY (postnummer) REFERENCES postnumre (postnummer) ON UPDATE CASCADE ON DELETE CASCADE
);
```

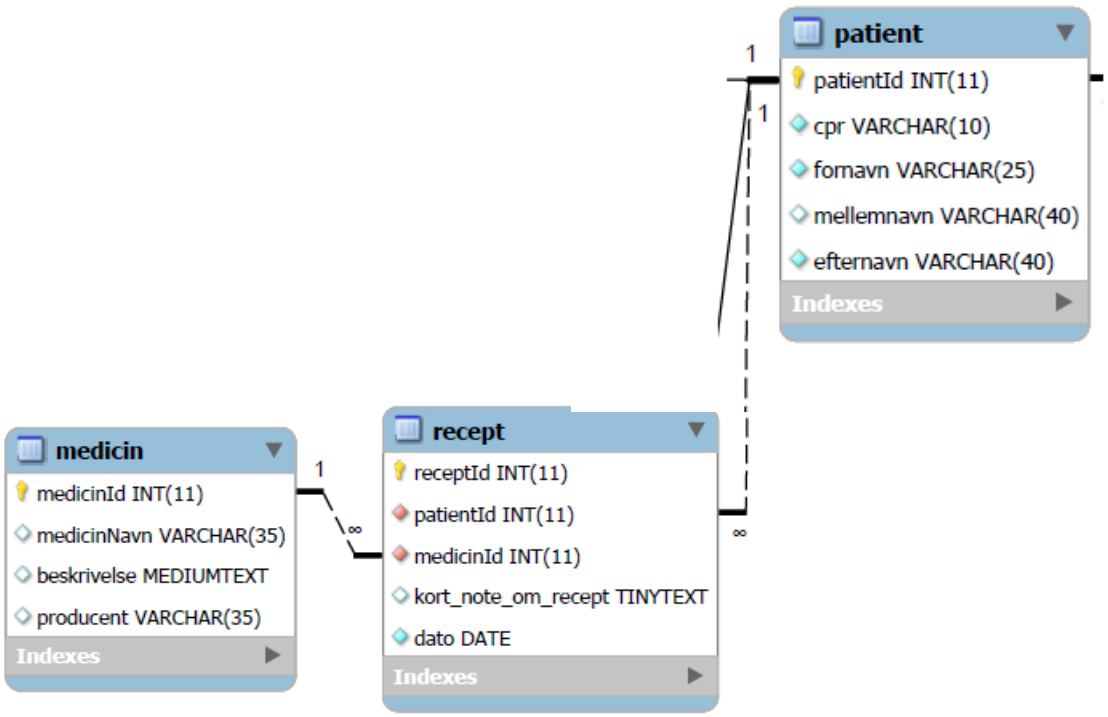


Relationen mellem patient og diagnose er mange-til-mange, som man ikke kan søge på. Så man laver en mellemtabel (junction), som patient og diagnose har en-til-mange -relation til. Patientdiagnose får PK fra både patient og diagnose som FK.

```
CREATE TABLE diagnose(
diagnoseId INT PRIMARY KEY AUTO_INCREMENT,
kort_note_om_diagnose MEDIUMTEXT,
dato DATE NOT NULL
);

CREATE TABLE patient(
patientId INT PRIMARY KEY AUTO_INCREMENT,
cpr VARCHAR(10) UNIQUE NOT NULL
);

CREATE TABLE patientDiagnoser(
patientId INT NOT NULL,
diagnoseId INT NOT NULL,
FOREIGN KEY (diagnoseId) REFERENCES diagnose (diagnoseId) ON UPDATE CASCADE ON DELETE CASCADE,
FOREIGN KEY (patientId) REFERENCES patient (patientId) ON UPDATE CASCADE ON DELETE CASCADE
);
```



Relationen mellem medicin og recept er en til mange og relationen mellem recept og patient er mange til en.
Så en recept laves kun til en slags medicin, men medicin kan høre til mange forskellige recepter.
En patient kan have mange recepter, men en recept kan kun tilhøre til en patient.
Recepten får både medicinId og patientId som FK.

```
CREATE TABLE medicin(
medicinId INT PRIMARY KEY AUTO_INCREMENT,
medicinNavn VARCHAR(35),
beskrivelse MEDIUMTEXT,
producent VARCHAR(35)
);
```

```
CREATE TABLE patient(
patientId INT PRIMARY KEY AUTO_INCREMENT,
cpr VARCHAR(10) UNIQUE NOT NULL
);
```

```
CREATE TABLE recept(
receptId INT PRIMARY KEY AUTO_INCREMENT,
patientId INT NOT NULL,
medicinId INT NOT NULL,
kort_note_om_recept TINYTEXT,
dato DATE NOT NULL,
FOREIGN KEY (patientId) REFERENCES patient (patientId) ON UPDATE CASCADE ON DELETE CASCADE,
FOREIGN KEY (medicinId) REFERENCES medicin (medicinId) ON UPDATE CASCADE ON DELETE CASCADE
);
```