Lektionsøvelse 3

Hensigten med denne opgave er at normalisere datastrukturen gennem Normaliseringsmetoden.

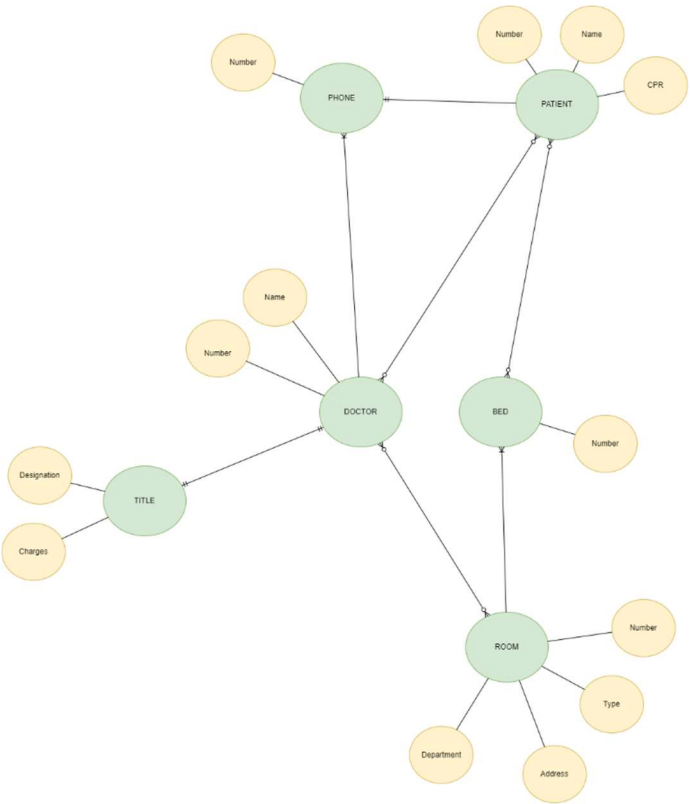
Normaliseringsmetoden er en metode, hvorpå vi finder relationer gennem kardinaliteter, og derefter dannes UML-Diagrammer hvor koden så er implementeret.

Trin 1: ER-Diagrammer

Vi starter allerførst med at tage et udkig på selve tabellen. Her kan det ses, at det ser følgende ud:

Doctor Number	Name	Room Address	Phone	Department Id	Designation	Charges per hour	Patient Number	Patient Name	CPR Number	Patient Phone	Room Number	Room Type	Bed Number
D1	Dr. Peterson	U45	12341234	Neurology	Professor	5000	P1	Jan	190582-1113	98769876	R2	Normal	B1
D1	Dr. Peterson	U45	12341324	Neurology	Professor	5000	P5	Peter	300175-2359	87658765	R2	Normal	B1
D1	Dr. Peterson	U45	12341234	Neurology	Professor	5000	P7	Jens	041298-1257	76547654	Null		Null
D2	Dr. Jensen	U32	24352435	Orthopedic	Professor	5000	P4	Ole	051165-9863	65436543	R2	Normal	B1
D2	Dr. Jensen	U32	23452345	Orthopedic	Professor	5000	P7	Jens	041298-1257	76547654	R4	Two Bed	B5
D2	Dr. Jensen	U32	23452435	Orthopedic	Professor	5000	P9	Anna	260792-1050	54325432	R4	Two Bed	B7
D4	Dr. Poetch	U186	34563456	ENT/Neurology	Assistant Professor	3000	P10	Dennis	150893-1151	43214321	Null		Null
D4	Dr. Poetch	U186	34563456	ENT/Neurology	Assistant Professor	3000	P1	Jan	190582-1113	98769876	R5	Special	B8
D5	Dr. Neurenheim	U150	45674567	Skin/Orthepoedic	Assistant Professor	3000	P12	Ahmed	010211-7853	32103210	Null		Null
D5	Dr. Neurenheim	U150	45674567	Skin/Orthepoedic	Assistant Professor	3000	P13	Annika	051285-8072	21092109	R6	Special	B9

Ude fra tabellen, kan vi danne en database hvorunder vi tilføjer de relevante attributter under databasetabellen. I tilfældet, kan det ses at vores database tabeller er de grønne cirkler og de tilhørende attributter er de gule cirkler til tabellen.



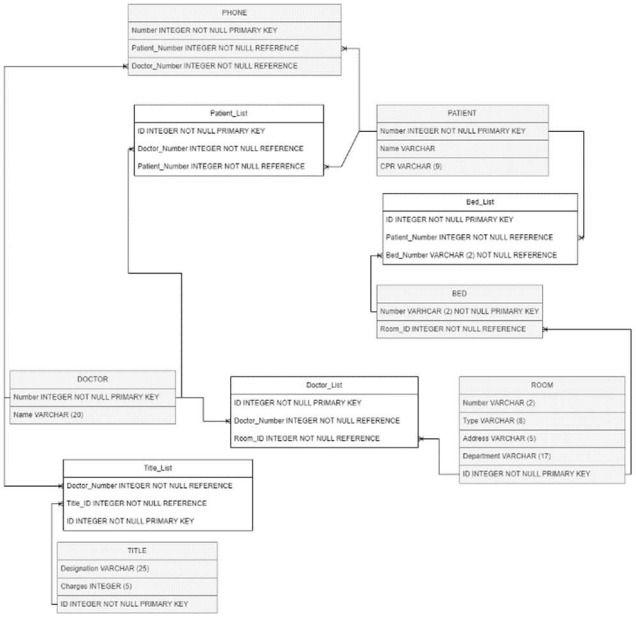
Den måde vi har fundet attributterne på er, ved at spørge os selv om forholdet mellem attributten og tabellen. Eksempelvis kan vi tage udgangspunkt i selve DOCTOR-Tabellen, hvor vi har tænkt følgende:

- *Kan der være en Doktor have flere telefonnumre?*
    - *Ja, det kan det godt.*
- *Kan der være flere Doktor_Number til en Doktor?*
    - *Nej, det er ikke muligt.*

Vi har stillet disse spørgsmål fordi vi vil gerne hjælpe os selv med at finde en løsning og om det er relevant for vores ER-Diagram dannelse.

Trin 2: UML-Diagrammer

Nu er vi gået videre til den næste opgave, hvor vi har dannet UML-Diagrammer til normalisering af vores kode. Her skal det understreges, at UML-Diagrammer optræder som det næste skridt for ER-Diagrammet, så det betyder at vi begynder nu at danne en liste imellem to tabeller for at kunne skabe en mere logisk og tydelig relationssamenhæng.



Vi kan kigge tilbage på Patient_Tabellen, hvor vi kigger på relationen og her kan det ses at Patient_List fungerer som en forbindelse i relationen mellem Patient og Phone & Doctor. I stedet for at sætte cardinalities mellem to tabeller, har vi nu lavet List som kan håndtere disse.

Så her kan vi se, at vi har oprettet en ID som er vores Primary Key fordi Number fra de to andre tabeller er deres individuelle Primary Keys. Derfor har vi oprettet ID som er primært en INTEGER Primary Key i Lists.

Vi forbinder Patient_Number PRIMARY KEY med Reference til Foreign Key tilhørende den samme tabelnavn og attribut. Husk, at tage hensyn til kardinalitetsforholdet mellem to tabeller. Dette betyder, at vi kan stille spørgsmål til forholdet mellem Doctor > Patient_List > Patient. *Er der en Doktor til mange Patienter? (Ja).*

Derfor sætter vi 1 til mange forhold på fra Doctor og over til Patient_List så vi kan skabe en relation mellem Doctor og Patient tilbage ved de grønne og gule ER-Diagrammer.

HUSK ALTID AT PRIMARY KEY PLEJER NÆSTEN ALTID AT VÆRE EN INTEGER, DOG MED UNDTAGELSE AF HVORDAN DEN ER DEFINERET. EKSEMPELVIS NAVN KAN VÆRE PRIMARY KEY MED VARCHAR.

FOREIGN KEY ELLER REFERENCER PLEJER NÆSTEN ALTID AT VÆRE INTEGER OG SKAL VÆRE NOT NULL FORDI DET ER DER HVOR VI AFGØR OM VORES FORHOLD I TABELLEN I RÆKKER.

Vi bruger den samme metode og princip mellem de andre tabeller og de områder hvor vi ikke har en List_tabel skal vi bare implementere cardinalitets metode direkte mellem tabellerne. Nu kan vi gå videre til at lave SQL-kodelægning.

Trin 3: Anlægning af SQL-Kode

For at kunne lave SQL-koden er det en anbefalet ting om at starte med de tabeller som ikke har en reference men som er kun tabeller. Derefter kan man bevæge sig over imod list som har referencer.

På Næste kan man se, hvordan koden egentligt er skrevet oppe.

```sql
-- Nu dannes tabellerne udefra UML-Diagrammet.

CREATE TABLE PHONE(
    NUMBER INTEGER PRIMARY KEY,
    PATIENT_NUMBER INTEGER NOT NULL REFERENCES PATIENT,
    DOCTOR_NUMBER INTEGER NOT NULL REFERENCES DOCTOR
);

INSERT INTO PHONE(NUMBER, PATIENT_NUMBER, DOCTOR_NUMBER)
VALUES (12341234,NULL,'D1');

INSERT INTO PHONE(NUMBER, PATIENT_NUMBER, DOCTOR_NUMBER)
VALUES (12341324,NULL,'D1');

INSERT INTO PHONE(NUMBER, PATIENT_NUMBER, DOCTOR_NUMBER)
VALUES (24352435,NULL,'D2');

INSERT INTO PHONE(NUMBER, PATIENT_NUMBER, DOCTOR_NUMBER)
VALUES (23452345,NULL,'D2');

INSERT INTO PHONE(NUMBER, PATIENT_NUMBER, DOCTOR_NUMBER)
VALUES (34563456,NULL,'D4');

INSERT INTO PHONE(NUMBER, PATIENT_NUMBER, DOCTOR_NUMBER)
VALUES (45674567,NULL,'D5');

INSERT INTO PHONE(NUMBER, PATIENT_NUMBER, DOCTOR_NUMBER)
VALUES (98769876,'P1',NULL);

INSERT INTO PHONE(NUMBER, PATIENT_NUMBER, DOCTOR_NUMBER)
VALUES (87658765,'P5',NULL);

```

```
33 INSERT INTO PHONE(NUMBER, PATIENT_NUMBER,
   DOCTOR_NUMBER)
34 VALUES (76547654,'P7',NULL);
35
36 INSERT INTO PHONE(NUMBER, PATIENT_NUMBER,
   DOCTOR_NUMBER)
37 VALUES (65436543,'P4',NULL);
38
39 INSERT INTO PHONE(NUMBER, PATIENT_NUMBER,
   DOCTOR_NUMBER)
40 VALUES (54325432,'P9',NULL);
41
42 INSERT INTO PHONE(NUMBER, PATIENT_NUMBER,
   DOCTOR_NUMBER)
43 VALUES (43214321,'P10',NULL);
44
45 INSERT INTO PHONE(NUMBER, PATIENT_NUMBER,
   DOCTOR_NUMBER)
46 VALUES (32103210,'P12',NULL);
47
48 INSERT INTO PHONE(NUMBER, PATIENT_NUMBER,
   DOCTOR_NUMBER)
49 VALUES (21092109,'P13',NULL);
50
51
52 CREATE TABLE PATIENT_LIST(
53     ID INTEGER NOT NULL PRIMARY KEY,
54     DOCTOR_NUMBER INTEGER NOT NULL REFERENCES DOCTOR,
55     PATIENT_NUMBER INTEGER NOT NULL REFERENCES
   PATIENT
56 );
57
58 INSERT INTO PATIENT_LIST(ID, DOCTOR_NUMBER,
   PATIENT_NUMBER)
59 VALUES (1,'D1','P1');
60
61 INSERT INTO PATIENT_LIST(ID, DOCTOR_NUMBER,
   PATIENT_NUMBER)
62 VALUES (2,'D1','P5');
63
64 INSERT INTO PATIENT_LIST(ID, DOCTOR_NUMBER,
```

```
64 PATIENT_NUMBER)
65 VALUES (3,'D1','P7');
66
67 INSERT INTO PATIENT_LIST(ID, DOCTOR_NUMBER,
   PATIENT_NUMBER)
68 VALUES (4,'D2','P4');
69
70 INSERT INTO PATIENT_LIST(ID, DOCTOR_NUMBER,
   PATIENT_NUMBER)
71 VALUES (5,'D2','P7');
72
73 INSERT INTO PATIENT_LIST(ID, DOCTOR_NUMBER,
   PATIENT_NUMBER)
74 VALUES (6, 'D2','P9');
75
76 INSERT INTO PATIENT_LIST(ID, DOCTOR_NUMBER,
   PATIENT_NUMBER)
77 VALUES (7, 'D4','P10');
78
79 INSERT INTO PATIENT_LIST(ID, DOCTOR_NUMBER,
   PATIENT_NUMBER)
80 VALUES (8, 'D4','P1');
81
82 INSERT INTO PATIENT_LIST(ID, DOCTOR_NUMBER,
   PATIENT_NUMBER)
83 VALUES (9, 'D5','P12');
84
85 INSERT INTO PATIENT_LIST(ID, DOCTOR_NUMBER,
   PATIENT_NUMBER)
86 VALUES (10, 'D5','P13');
87
88 CREATE TABLE PATIENT(
89     NUMBER INTEGER NOT NULL PRIMARY KEY,
90     NAME VARCHAR,
91     CPR VARCHAR (9)
92 );
93
94 INSERT INTO PATIENT(NUMBER, NAME, CPR)
95 VALUES ('P1','Jan','190582-1113');
96
97 INSERT INTO PATIENT(NUMBER, NAME, CPR)
```

```
 98 VALUES ('P5','Peter','300175-2359');
 99
100 INSERT INTO PATIENT(NUMBER, NAME, CPR)
101 VALUES ('P7','Jens','041298-1257');
102
103 INSERT INTO PATIENT(NUMBER, NAME, CPR)
104 VALUES ('P4','Ole','051165-9863');
105
106 INSERT INTO PATIENT(NUMBER, NAME, CPR)
107 VALUES ('P9','Anna','260792-1050');
108
109 INSERT INTO PATIENT(NUMBER, NAME, CPR)
110 VALUES ('P10','Dennis','150893-1151');
111
112 INSERT INTO PATIENT(NUMBER, NAME, CPR)
113 VALUES ('P12','Ahmed','010211-7853');
114
115 INSERT INTO PATIENT(NUMBER, NAME, CPR)
116 VALUES ('P13','Annika','051285-8072');
117
118 SELECT * FROM PATIENT;
119
120 CREATE TABLE BED_LIST(
121     ID INTEGER NOT NULL PRIMARY KEY,
122     PATIENT_NUMBER INTEGER NOT NULL REFERENCES
    PATIENT,
123     BED_NUMBER VARCHAR (2) NOT NULL REFERENCES BED
124 );
125
126 INSERT INTO BED_LIST(ID,PATIENT_NUMBER,BED_NUMBER)
127 VALUES (1,'P1','B1');
128
129 INSERT INTO BED_LIST(ID,PATIENT_NUMBER,BED_NUMBER)
130 VALUES (2,'P5','B1');
131
132 INSERT INTO BED_LIST(ID,PATIENT_NUMBER,BED_NUMBER)
133 VALUES (3,'P7',NULL);
134
135 INSERT INTO BED_LIST(ID,PATIENT_NUMBER,BED_NUMBER)
136 VALUES (4,'P4','B1');
137
```

```
138 INSERT INTO BED_LIST(ID,PATIENT_NUMBER,BED_NUMBER)
139 VALUES (5,'P7','B5');
140
141 INSERT INTO BED_LIST(ID,PATIENT_NUMBER,BED_NUMBER)
142 VALUES (6,'P9','B7');
143
144 INSERT INTO BED_LIST(ID,PATIENT_NUMBER,BED_NUMBER)
145 VALUES (7,'P10');
146
147 INSERT INTO BED_LIST(ID, PATIENT_NUMBER, BED_NUMBER)
148 VALUES (8,'P1','B8');
149
150 INSERT INTO BED_LIST(ID, PATIENT_NUMBER, BED_NUMBER)
151 VALUES (9,'P12');
152
153 INSERT INTO BED_LIST(ID, PATIENT_NUMBER, BED_NUMBER)
154 VALUES (10,'P13','B9');
155
156 CREATE TABLE BED(
157     NUMBER VARCHAR (2) NOT NULL PRIMARY KEY,
158     ROOM_ID INTEGER NOT NULL REFERENCES ROOM
159 );
160
161 INSERT INTO BED(NUMBER, ROOM_ID)
162 VALUES ('B1',1);
163
164 INSERT INTO BED(NUMBER, ROOM_ID)
165 VALUES ('B5',2);
166
167 INSERT INTO BED(NUMBER, ROOM_ID)
168 VALUES ('B7',3);
169
170 INSERT INTO BED(NUMBER, ROOM_ID)
171 VALUES ('B8',4);
172
173 INSERT INTO BED(NUMBER, ROOM_ID)
174 VALUES ('B9',5);
175
176 CREATE TABLE ROOM(
177     NUMBER VARCHAR (2),
178     TYPE VARCHAR (8),
```

```
179     ADDRESS VARCHAR (5),
180     DEPARTMENT VARCHAR (17),
181     ID INTEGER NOT NULL PRIMARY KEY
182 );
183
184 INSERT INTO ROOM(NUMBER, TYPE, ADDRESS, DEPARTMENT,
    ID)
185 VALUES ('R2','Normal','B1','Neurology',1);
186
187 INSERT INTO ROOM(NUMBER, TYPE, ADDRESS, DEPARTMENT,
    ID)
188 VALUES ('R2','Normal','B1','Orthopedic',2);
189
190 INSERT INTO ROOM(NUMBER, TYPE, ADDRESS, DEPARTMENT,
    ID)
191 VALUES ('R4','Two Bed','B5','Orthopedic',3);
192
193 INSERT INTO ROOM(NUMBER, TYPE, ADDRESS, DEPARTMENT,
    ID)
194 VALUES ('R4','Two Bed', 'B7','Orthopedic',4);
195
196 INSERT INTO ROOM(NUMBER, TYPE, ADDRESS, DEPARTMENT,
    ID)
197 VALUES ('R5','Special','B8','ENT/Neurology',5);
198
199 INSERT INTO ROOM(NUMBER, TYPE, ADDRESS, DEPARTMENT,
    ID)
200 VALUES ('R6','Special','B9','Skin/Orthopedic',6);
201
202 CREATE TABLE DOCTOR_LIST(
203     ID INTEGER NOT NULL PRIMARY KEY,
204     DOCTOR_NUMBER INTEGER NOT NULL REFERENCES DOCTOR
    ,
205     ROOM_ID INTEGER NOT NULL REFERENCES ROOM
206 );
207
208 INSERT INTO DOCTOR_LIST(ID, DOCTOR_NUMBER,ROOM_ID)
209 VALUES (1,'D1',1);
210
211 INSERT INTO DOCTOR_LIST(ID,DOCTOR_NUMBER,ROOM_ID)
212 VALUES (2,'D2',2);
```

```
213
214 INSERT INTO DOCTOR_LIST(ID,DOCTOR_NUMBER,ROOM_ID)
215 VALUES (3,'D4',3);
216
217 INSERT INTO DOCTOR_LIST(ID,DOCTOR_NUMBER,ROOM_ID)
218 VALUES(4,'D5',4);
219
220 CREATE TABLE DOCTOR(
221     NUMBER INTEGER NOT NULL PRIMARY KEY,
222     NAME VARCHAR (20)
223 );
224
225 INSERT INTO DOCTOR(NUMBER, NAME)
226 VALUES ('D1','Dr. Peterson');
227
228 INSERT INTO DOCTOR(NUMBER, NAME)
229 VALUES ('D2','Dr. Jensen');
230
231 INSERT INTO DOCTOR(NUMBER, NAME)
232 VALUES ('D4','Dr. Poetch');
233
234 INSERT INTO DOCTOR(NUMBER, NAME)
235 VALUES ('D5','Dr. Neurenheim');
236
237 CREATE TABLE TITLE_LIST(
238     DOCTOR_NUMBER INTEGER NOT NULL REFERENCES DOCTOR
    ,
239     TITLE_ID INTEGER NOT NULL REFERENCES TITLE,
240     ID INTEGER NOT NULL PRIMARY KEY
241 );
242
243 INSERT INTO TITLE_LIST(DOCTOR_NUMBER, TITLE_ID, ID)
244 VALUES ('D1',1,1);
245
246 INSERT INTO TITLE_LIST(DOCTOR_NUMBER, TITLE_ID, ID)
247 VALUES ('D2',2,2);
248
249 INSERT INTO TITLE_LIST(DOCTOR_NUMBER, TITLE_ID, ID)
250 VALUES ('D4',3,3);
251
252 INSERT INTO TITLE_LIST(DOCTOR_NUMBER, TITLE_ID, ID)
```

```
253 VALUES ('D5',4,4);
254
255 CREATE TABLE TITLE(
256     DESIGNATION VARCHAR (25),
257     CHARGES INTEGER (5),
258     ID INTEGER NOT NULL PRIMARY KEY
259 );
260
261 INSERT INTO TITLE(DESIGNATION, CHARGES,ID)
262 VALUES('Professor',5000, 1);
263
264 INSERT INTO TITLE(DESIGNATION, CHARGES, ID)
265 VALUES('Assistant Professor', 3000, 2);
266
267
268
```