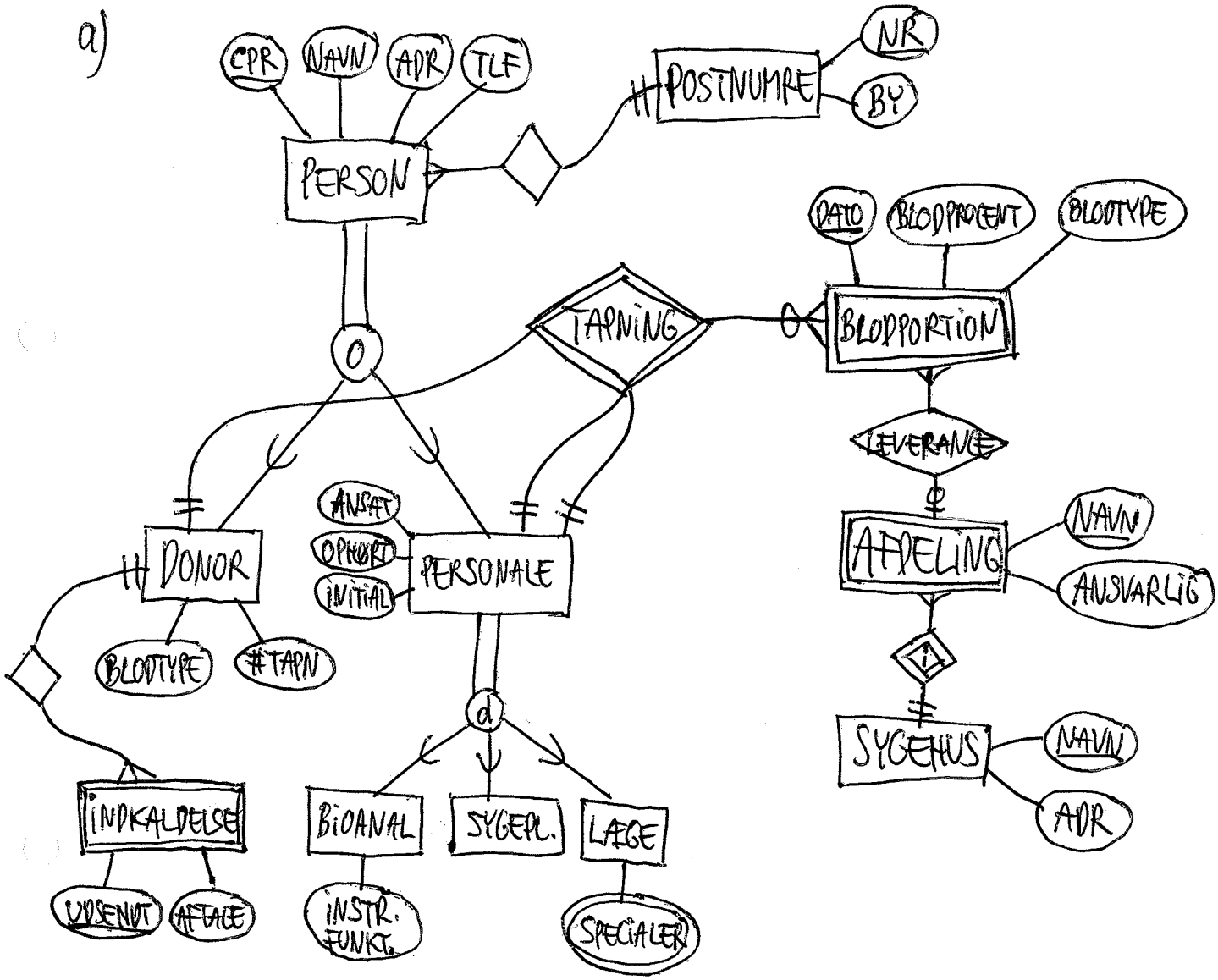
DATABASESYSTEMER 7/6 - 2005

OPGAVE 1

a)



b) VED BRUG AF METODE FRA FORELÆSNINGSSLIDES:

PERSON (CPR, NAVN, ADR, TLF, POSTNR) ← KAN EVT. INDEHOLDE "SUBTYPE DISCRIMINATOR"
POSTNUMRE (NR, BY)
DONOR (CPR, BLODTYPE, #TAPN)
PERSONALE (CPR, ANSAT, OPHØRT, INITIAL)
INDKALDELSE (CPR, UDSENDT, AFTALE)
BIOANAL (CPR, INSTRFUNKT.)
LÆGE (CPR, SPECIALE)

BLODPORTION (DONORCPR, PERSONALECPR1, PERSCPR2, DATO, BLODPROCENT, BLODTYPE, LEVERETTIL)
AFDELING (NAVN, SYGEHUS, ANSVARLIG)
SYGEHUS (NAVN, ADR)

OPGAVE 2

a) KANDIDATNØGLER: {id, sygdom}, {sygdom, by}, {sygdom, postnr}.

b) FLG. FUNKTIONELLE AFHÆNGIGHEDER HAR IKKE EN KANDIDATNØGLE PÅ VENSTE SIDE:

id → adresse, postnr, by
postnr → by
sygdom → speciale

DA postnr OG by BEGGE ER EN ÆGTE DELMÆNGDE AF EN KANDIDATNØGLE, GIVER KUN FJERNE id→adresse OG sygdom→ speiale ANLEDNING TIL DEKOMPOSITION:

Adr(id, adresse)
Spec(sygdom, speciale)
Behandler2(id, postnr, by, sygdom)

OPGAVE 3

a) 1 RETURNERER ALLE BEHANDLERE AF HUNDEGALSKAB I POSTNR 1000 (IFØLGE OPG. 2 ER DER HØJST ÉN).

2 RETURNERER ALLE SPECIALER SOM DEN PÅGÆLDENDE BEHANDLER HAR.

b)
```
SELECT speciale, COUNT(*), COUNT(DISTINCT sygdom)
FROM Behandler
GROUP BY speciale;
```

c)
```
SELECT id
FROM Patient
WHERE sygdom NOT IN (SELECT sygdom
                     FROM Behandler
                     WHERE id = behandler_id);
```

d)
```
CREATE TABLE Patient2(
    id INT, sygdom CHAR(32), behandler_by CHAR(16),
    FOREIGN KEY (sygdom, behandler_by) REFERENCES Behandler(sygdom, by));

UPDATE Patient
SET behandler_id=NULL
WHERE sygdom NOT IN (SELECT sygdom FROM Behandler
                     WHERE id=behandler_id);
```

ÆNDRING AF PATIENT KAN UNDGÅS VED FØRST AT LAVE EN KOPI

```
INSERT INTO Patient2 VALUES
    (SELECT Patient.id, Patient.sygdom, Behandler.by
     FROM Patient, Behandler
     WHERE behandler_id = Behandler.id);
```

OPGAVE 4

a) PÅ ISOLERINGSNIVEAU READ COMMITTED KAN ÆNDRINGER LAVET AF ANDRE TRANSAKTIONER FOREKOMME MELLEM TO SQL SÆTNINGER.

TRANSAKTIONEN FINDER FRIE SÆDER I SÆTNING 3, OG ÉT AF DEM KAN SÅGTENS BLIVE BOOKET INDEN SÆTNING 7.

b) PÅ "REPEATABLE READ" OG "SERIALIZABLE" KAN INGEN LÆST VÆRDI ÆNDRE SIG UNDER TRANSAKTIONEN. DERFOR KAN Seats IKKE OPDATERES MELLEM SÆTNING 3 OG 7.

VED "SNAPSHOT ISOLATION" VIL TRANSAKTIONEN ABORTE HVIS DATA, DEN HAR SKREVET TIL, ER BLEVET ÆNDRET UNDERVEJS AF EN ANDEN TRANSAKTION.

DOBBELTBOOKING ER ALTSÅ UMULIG I ALLE 3 TILFÆLDE.

OPGAVE 5

a) INDEKS PÅ speciale HJÆLPER IKKE, DA speciale IKKE ER DEL AF NOGEN BETINGELSE.

INDEKS PÅ id HJÆLPER IKKE – id FOREKOMMER KUN i FORBINDELSE MED "IN" i BETINGELSEN. (EN SMART DBMS KUNNE MÅSKE UDNYTTE ET INDEKS PÅ id, HVIS SUBQUERYEN i 2 HAR ET LILLE RESULTAT).

INDEKS PÅ postnr ELLER sygdom ER EN GOD IDÉ, DA BEGGE BLIVER BRUGT i HØJT GRAD SELEKTIVE SELECTS i 1 SÅVEL SOM 2.

b) idx1 GØR BÅDE 1 OG 2 HURTIGERE. (SÆRLIGT 1)

idx2 GØR 1 OG SUBQUERYEN i 2 HURTIGERE (VS. 2 LIDT HURTIGERE)

idx3 HJÆLPER IKKE PÅ 1, OG FORMENTLIG IKKE PÅ 2
(EN SMART DBMS KUNNE BRUGE idx3 VED 2 HVIS SUBQUERY HAR LILLE RES.)

idx4 HJÆLPER BÅDE i 1 OG 2

c)

FØRSTE INDEKS PÅ {sygdom, postnr} (idx2)

ANDET INDEKS PÅ {postnr}

GØR 1 MEGET HURTIG. OG KAN BRUGES AF SUBQUERY i 2.

SELEKTIV ATTRIBUT i 2

(EN SMART DBMS KUNNE UDNYTTE {postnr, id} HVIS SUBQUERYEN GIVER ET LILLE RESULTAT).