

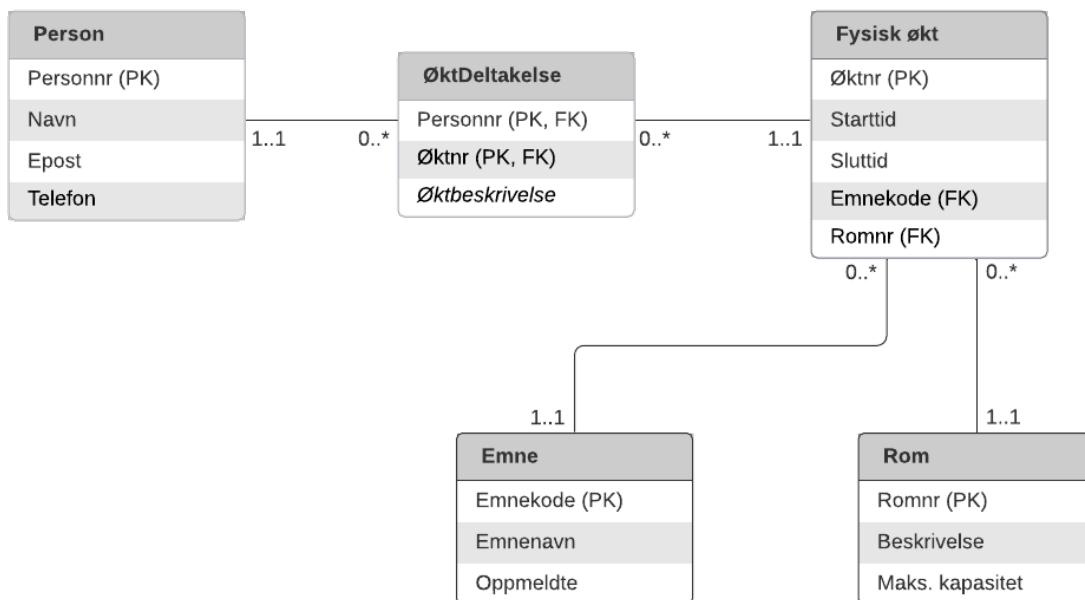
Oppgave 1 – modellering

Jeg ser at *person*, *fysisk økt*, *emne* og *rom* passer fint som hver sine entiteter. Jeg har også lagt til relevante attributter under hver av disse. I denne oppgaven benyttes det av UML notasjone.

Jeg kobler deretter de nevnte entitetene sammen etter primær- og fremmednøklene deres, og ser at person og fysisk økt har en mange-til-mange relasjon. Dette betyr i praksis at én person kan være registrert på flere økter, og hver økt kan involvere flere personer. Jeg oppretter derfor en koblingsentitet mellom disse to, og har valgt å kalle den for *økt deltagelse*.

Primærnøkkelen (også kalt kandidatnøkkelen) er satt sammen fra primærnøklene til de to involverte tabellene, som hver for seg blir fremmednøkler tilbake til disse. Jeg har også lagt til *øktbeskrivelse* som et attributt, hvor det er mulighet å spesifisere om økten var en forelesning eller øving. Gjennom økt deltagelse kan vi altså spore opp hvem som har deltatt i de spesifikke øktene. Det må minst være én person OG én spesifikk økt for hver økt deltagelse, derfor er deltagelsen fra økt deltagelse 1.

Multiplisiteten mellom *fysisk økt* og *emne* kan forklares som følgende: for hver fysisk økt må det minst finnes 1 emne, og kan maks tilhøre 1 emne. Hvert emne kan tilhøre flere økter, men må ikke tilhøre noen. Det samme gjelder for relasjonen mellom entitetene *fysisk økt* og *rom*. Hvert rom kan tilhøre flere økter, men må ikke tilhøre noen. Hver fysisk økt må det minst være 1 rom, og kan maks tilhøre 1 rom. Derfor er deltagelsen fra fysisk økt til tabellene 1.



Oppgave 2 – SQL

A)

```
SELECT *
FROM deltaker
ORDER BY etternavn ASC, fornavn ASC;
```

DNr	Fornavn	Etternavn	EPost
15	Anders	Andersen	anders@andersen.no
6	Benny	Ball	benny@benny.no
10	Billy	Betong	billy@ppbb.no
4	Eva	Dahl	eva@dahl.no
12	Frida	Frosk	frida@ppbb.no
1	Hans	Hansen	hans@hansen.no
18	Svetlana	Iversen	svetlana@olsen.no
8	Hans	Jensen	hj@jensen.no
3	Jens	Jensen	jens@jensen.no
16	Julie	Jensen	julie@jensen.no

B)

```
SELECT fornavn, etternavn, epost
FROM deltaker
WHERE epost LIKE '%@ppbb.no';
```

fornavn	etternavn	epost
Sandra	Salamander	sandra@ppbb.no
Billy	Betong	billy@ppbb.no
Pelle	Parafin	pelle@ppbb.no
Frida	Frosk	frida@ppbb.no
Leon	Latex	leon@ppbb.no
Ragna	Rekkverk	ragna@ppbb.no

C)

```
SELECT Dagnr, SUM(måltidpris) AS 'Totalpris mat'
FROM måltid
GROUP BY dagnr;
```

Dagnr	Totalpris mat
1	278
2	278

D)

I denne oppgaven valgte jeg å benytte NATURAL LEFT JOIN (naturlig join) siden *deltaker*-tabellen og *forfatter*-tabellen har like kolonner (*DNr*). For å ta med alle deltakerne (rader) uavhengig om de er forfattere eller ikke, joiner jeg tabellen til venstre. Alle forfattere har fått tildelt et *PresNr*, og i dette tilfellet må jeg filtrere de som ikke er forfattere. Går frem med å finne rader med NULL-verdier.

```
SELECT dnr, fornavn, etternavn, epost
FROM deltaker NATURAL LEFT JOIN forfatter
WHERE presnr IS NULL;
```

dnr	fornavn	etternavn	epost
15	Anders	Andersen	anders@andersen.no
16	Julie	Jensen	julie@jensen.no
17	Igor	Olsen	igor@olsen.no
18	Svetlana	Iversen	svetlana@olsen.no

E)

For å hente antall temaer brukte jeg COUNT, og deretter grupperte jeg radene etter *DNr*. Da fikk jeg antall temaer per deltaker. Deretter sorterte dem etter synkende rekkefølge.

```
SELECT DNr, Fornavn, Etternavn, COUNT(temanr) AS 'Antall temaer'
FROM deltakertema NATURAL JOIN deltaker
GROUP BY dnr
ORDER BY COUNT(temanr) DESC;
```

DNr	Fornavn	Etternavn	Antall temaer
1	Hans	Hansen	5
10	Billy	Betong	5
7	Oline	Jensen	4
8	Hans	Jensen	3
11	Pelle	Parafin	3
12	Frida	Frosk	3
5	Ole	Olsen	3
2	Kari	Normann	2
4	Eva	Dahl	2
13	Leon	Latex	2

Jeg oppdager også at deltakere med DNr 14-18 ikke har markert noen temaer av interesse i deltakertema-tabellen. For å være helt sikker på at spørringen min er korrekt, brukte jeg følgende spørring for å teste:

```
SELECT *
FROM deltakertema
WHERE dnr = [Skriv inn ønsket DNr]
```

Deltaker nummer 13 har valgt temaer, men ikke deltaker nummer 14.

DNr	TemaNr
13	1
13	5

DNr	TemaNr
NULL	NULL

F)

```
INSERT INTO måltidbestilling (dnr, måltidstype, dagnr)
VALUES (1, 'Lunsj', 1),
(1, 'Middag', 1),
(2, 'Middag', 1),
(4, 'Lunsj', 2),
(8, 'Lunsj', 1);
```

G)

```
UPDATE deltaker
SET epost = 'svetlana@iversen.no'
WHERE dnr = 18;
```

H)

```
SELECT Etternavn, COUNT(*) AS 'Antall'
FROM deltaker
GROUP BY etternavn
HAVING COUNT(*) > 1;
```

Etternavn	Antall
Jensen	4
Olsen	2

I)

Jeg har valgt å besvare denne oppgaven med én tabell. Denne tabellen er veldig lik *måltidbestilling*-tabellen. Deltakerne har to alternativer å velge mellom (transport ved ankomst eller hjemreise), derfor har jeg tatt i bruk datatypen ENUM i dette tilfellet. *Dnr* og *transport* er primærnøklene, da vi må ha en kombinasjon av disse to for å finne *dagnr*. Dette er forutsett at deltakerne ikke reiser til og fra begge dagene (da det i praksis er mer logisk å sjekke inn på et hotell over natten), for hvis det er tilfellet må siste kolonnen også gjøres om til en primærnøkkelse – slik som det har blitt gjort i *måltidbestilling*-tabellen.

```
CREATE TABLE Transport (
DNr INT NOT NULL,
Transporttype ENUM('Hjemreise', 'Ankomst'),
DagNr INT NOT NULL,
PRIMARY KEY (DNr, Transporttype)
);
```

```
INSERT INTO transport
VALUES (3, 'Ankomst', 1),
(3, 'Hjemreise', 1),
(5, 'Ankomst', 2);
```

```
SELECT *
FROM transport;
```



DNR	Transporttype	DagNr
3	Hjemreise	1
3	Ankomst	1
5	Ankomst	2

J)

Jeg tar utgangspunkt i *presentasjon*-tabellen da den inneholder de fleste attributtene oppgaven spør etter, og har en tilknytning til de resterende attributtene. *Temanavn* kan hentes via *temanr* fra *tema*-tabellen. *Fornavn* og *etternavn* kan hentes via *dnr* fra *deltaker*-tabellen. Siste attributtet som må hentes «eksternt» er *antplasser* via *romnr* fra *rom*-tabellen.

I den første view'en henter jeg kun ut attributtene innad tabellen. Formaterer *starttid* slik som oppgaven ber om, og *DATE_ADD* blir brukt på *varighet*-kolonnen. I den andre view'en joiner jeg tabellene og trekker inn de resterende attributtene i tabellen. Siste finish med formatering på *sluttid*. I den tredje, og siste view'en sammensetter jeg attributtene ved bruk av *CONCAT* og sorterer tabellen.

RomNr	Tittel	Temanavn	Presenteres av	Tidspunkt	Antall plasser
A1	Feasability of Optimizations Requiring Bounded...	Performance and Optimization	Hans Hansen	24. November kl 09.45-10.05	100
A1	Evaluation of graph algorithm frameworks for m...	Performance and Optimization	Kari Normann	24. November kl 10.15-10.35	100
F1	IT students perceptions of mandatory coursework	IT didactics	Jens Jensen	24. November kl 09.45-10.05	50
F1	Introducing ePortfolios to IT students: The supp...	IT didactics	Eva Dahl	24. November kl 10.15-10.35	50
F1	Teaching AI Ethics: Observations and Challenges	IT didactics	Ole Olsen	24. November kl 10.45-11.05	50
F1	The Live Programming Lecturing Technique: A...	IT didactics	Benny Ball	24. November kl 11.15-11.35	50
F2	INERTIA AND CHANGE IN TRANSFORMATIO...	Digital transformation	Oline Jensen	25. November kl 09.45-10.05	40
F2	DIGITAL TRANSFORMATION UNDER A PAND...	Digital transformation	Hans Jensen	25. November kl 10.15-10.35	40
F2	Exploring the Impact of Mob Programming on th...	Digital transformation	Sandra Salamander	25. November kl 10.45-11.05	40
F2	Exploring the Hiring Process of a Norwegian Mu...	Digital transformation	Billy Betong	25. November kl 11.15-11.35	40

```
-- Plukk ut kolonner fra presentasjon-tabellen og lag view
```

```
CREATE OR REPLACE VIEW presentasjon_view
AS
SELECT dnr, temanr, tittel, romnr,
DATE_FORMAT(starttid, '%d. %M kl %H.%i-') AS Starttid,
DATE_ADD(Starttid, INTERVAL 20 MINUTE) AS 'Sluttid'
FROM presentasjon;
```

```
SELECT *
FROM presentasjon_view;
```

```
-- Join tabellene og lag nytt view
```

```
CREATE OR REPLACE VIEW pres_preconcat_view
AS
SELECT Tittel, pview.RomNr, antplasser, starttid, DATE_FORMAT(sluttid, '%H.%i') AS Sluttid, fornavn,
etternavn, Temanavn
FROM presentasjon_view pview JOIN deltaker ON pview.dnr = deltaker.dnr
JOIN tema ON pview.temanr = tema.temanr
JOIN rom ON pview.romnr = rom.romnr;
```

```
SELECT *
FROM pres_preconcat_view
```

```
-- Sett sammen kolonnene (concat) og sorter, så lage siste view
```

```
CREATE OR REPLACE VIEW presentasjon_informasjon_view
AS
SELECT RomNr, Tittel, Temanavn, CONCAT(Fornavn, " ", Etternavn) AS 'Presenteres av',
CONCAT(Starttid, Sluttid) AS Tidspunkt, antplasser AS 'Antall plasser'
FROM pres_preconcat_view
ORDER BY romnr ASC, Tidspunkt ASC;
```

```
SELECT *
FROM presentasjon_informasjon_view;
```

Oppgave 3 – normalisering

utstyr_ansatt	FK		PK	Innkjøpspris	Innkjøpsdato	Type
	Ansattnr	Fnavn				
123456	Jens	Jensen	55555555	PC	10000	2019-01-02
123456	Jens	Jensen	55555555	Mobil	9000	2019-01-02
123456	Jens	Jensen	55555555	Stol	3490	2018-12-12
234567	Kari	Normann	66666666	Mac	13900	2017-05-05
234567	Kari	Normann	66666666	Mobil	9900	2019-05-05
234567	Kari	Normann	66666666	Stol	3900	2017-07-05
234567	Kari	Normann	66666666	Headsett	2900	2017-08-05

Steg 1:

Oppgaveteksten etterspør normalisering av begge tabellene til 3.normalform. I tabellene ovenfor er det redundans. For eksempel i tabell *utstyr_ansatt* er kolonnene *ansattnr*, *fnavn*, *enavn* og *tlfnr* gjentakende (også kalt dobbellagring). For å unngå dette starter jeg med å splitte *utstyr_ansatt*-tabellen i to og fjerner redundansen.

Utstyr	PK		PK		FK	
	Utstyr	Innkjøpspris	Innkjøpsdato	Type	Ansattnr	
PC	10000	2019-01-02	Lenovo	123456		
Mobil	9000	2019-01-02	iPhone	123456		
Stol	3490	2018-12-12	Zareto	123456		
Mac	13900	2017-05-05	Macbook Pro	234567		
Mobil	9900	2019-05-05	Samsung	234567		
Stol	3900	2017-07-05	Watford	234567		
Headsett	2900	2017-08-05	Boss	234567		

Ansatt_filial	PK		Tlfnr	Rom	Sted	Etasje	Adresse
	Ansattnr	Fnavn	Enavn				
123456	Jens	Jensen	55555555	12	Oslo	1	Smalveien 1
234567	Kari	Normann	66666666	12	Oslo	1	Smalveien 2
345678	Ole	Olsen	77777777	22	Oslo	2	Smalveien 3
445544	Lise	Olsen	88888888	Gløtt	Bergen	5	Brygga 2
554455	Per	Persen	88668866	Gløtt	Bergen	5	Brygga 3
989898	Eva	Jensen	45454545	Regn	Bergen	5	Brygga 4
323232	Nils	Nilsen	23343223	Regn	Bergen	5	Brygga 5

Steg 2:

Nå står *ansattnr* igjen i tabellen som fremmednøkkel. Det ingen gjentakende dataelementer i tabellen lenger, med unntak av FK. Én ansatt kan ha flere utstyr, men per utstyr kan kun tilhøre én ansatt.

Jeg ønsker nå å splitte *ansatt_filial* i to tabeller. Dette er fordi det er mer praktisk å kunne legge til en ny filial uten å måtte lage en «dummybruker», og hvis alle radene fra *ansatt* noen gang blir slettet, beholder databasen fortsatt dataene til filialene.

PK	FK		
	Utstyr	Type	Ansattnr
PC	Lenovo	123456	
Mobil	iPhone	123456	
Stol	Zareto	123456	
Mac	Macbook Pro	234567	
Mobil	Samsung	234567	
Stol	Watford	234567	
Headsett	Boss	234567	

PK	FK		
	Innkjøpsdato	Innkjøpspris	Ansattnr
2019-01-02	10000	123456	
2019-01-02	9000	123456	
2018-12-12	3490	123456	
2017-05-05	13900	234567	
2019-05-05	9900	234567	
2017-07-05	3900	234567	
2017-08-05	2900	234567	

Ansatt	PK		Postnr	FK	
	Ansattnr	Fnavn	Enavn	Tlfnr	
	123456	Jens	Jensen	55555555	1111
	234567	Kari	Normann	66666666	1111
	345678	Ole	Olsen	77777777	1111
	445544	Lise	Olsen	88888888	2222
	554455	Per	Persen	88668866	2222
	989898	Eva	Jensen	45454545	2222
	323232	Nils	Nilsen	23343223	2222

	PK				
Filial_kontor	Postnr	Sted	Adresse	Etasje	Rom
	1111	Oslo	Smalveien 1	1	12
	1111	Oslo	Smalveien 1	2	22
	2222	Bergen	Brygga 2	5	Gløtt
	2222	Bergen	Brygga 2	5	Regn

Steg 3:

I dette steget har jeg tilføyd et nytt attributt, «postnr». Da *sted*, *adresse* og *etasje* er transitivt avhengige av *ansattnr* via *postnr* og *rom* – splitter jeg tabellen. *Ansatt*-tabellen inneholder nå kun attributter som er funksjonelt avhengig av primærnøkkelen, som er *ansattnr*. Det vil også si at *ansattnr* er determinant for attributtene i tabellen. Attributtet *postnr* er nå en fremmednøkkel i *ansatt*-tabellen og primærnøkkel i *filial_kontor*-tabellen. Ved å dele det opp slik gir det en bedre struktur i databasen. Per ansatt kan kun tilhøre én filial, men én filial kan ha flere ansatte. Til tross for tabellsplittingen er det fortsatt redundans i *filial_kontor*-tabellen, så jeg velger å splitte tabellen videre.

Utstyr-tabellen er også splittet videre fordi en tabell i 3NF skal alle ikke-primærnøkkel attributter kun være funksjonelt avhengige av primærnøkkelen sin.

Filial	PK		
	Postnr	Sted	Adresse
	1111	Oslo	Smalveien 1
	2222	Bergen	Brygga 2

Kontor	PK		FK
	Rom	Etasje	Postnr
	12	1	1111
	22	2	1111
	34	5	2222
	56	5	2222

Steg 4:

Tabellene *filial* og *kontor* er nå egne entiteter. *Postnr* er satt som primærnøkkel, da *sted* er funksjonelt avhengig av *postnr*. *Adresse* er derimot funksjonelt avhengig av kandidatnøkkelen *postnr* og *sted* kombinert, siden databasen må ha både det ene og det andre for å finne adressen. Det er ikke lenger dobbellagring i *filial*-tabellen. *Rom* er satt som primærnøkkel i *kontor*-tabellen da det er unikt. Av hensiktsmessige grunner har jeg gitt dataene i *rom* samme datatype (jeg valgte å benytte av datatypen INT). Her kan hver filial ha flere kontorer, mens per kontor kan kun tilhøre én filial.

Konklusjon:

Løsningen min oppfyller 3.normalform fordi den er på 2NF. Det er ingen ikke-primærnøkkel attributt som har en transitiv avhengighet til en primærnøkkel. Det vil si alle ikke-primærnøkkel attributter er kun funksjonelt avhengige av sine primærnøkler. Det er heller ingen redundans i løsningen utenom fremmednøkler og attributter som ikke kan stå i en egen tabell med én kolonne (f.eks *etasje*)

Oppgave 4 – diverse

Ofte bruker tunge spørninger (gjerne spørninger som involverer mange tabeller og betingelser) lengre tid å utføre. For å se om en spørring er tung å gjennomføre kan man se under duration på worckbench. En av løsningene for å optimalisere ytelsen på spørninger i databasen på er å ta i bruk indekser. Én indeks er nok til å gjøre spørringen hundre ganger raskere. Dette er fordi den optimaliserer WHERE-clause delen av spørringen.

Er databasen for treg er denormalisering, som er det omvendte av normalisering, også en løsning. Dette er fordi en full normalisering kan gi et tregere databasesystem. Flere tabeller fører til flere likekoblinger. Denormalisering vil altså si at for å oppnå et mer effektivt system kan man akseptere en viss redundans (dobbella gring).