Eksamen IN2090 Høsten 2025

Skriftlig eksamen i IN2090

2025 HØST

Tid: 12. desember 13:00 til 12. desember 19:00.

Sted: Silurveien

Tillatte hjelpemidler: Alle trykte og skrevne hjelpemidler tillatt.

Det er viktig at du leser denne forsiden nøye før du starter.

Håndtegning:

På denne eksamen er det oppgaver som krever håndtegning. Du bruker skisseark du får utdelt. Det er anledning til å bruke flere ark per oppgave. Se instruksjon for utfylling av skisseark i lenken under oppgavelinjen.

Struktur:

Eksamen består av 3 deler (maksimal poengsum i parentes):

- Modellering og realisering (30)
- SQL (50)
- Relasjonsmodellen, normalformer og dekomposisjon (20)

1 ER-modellering, relasjonsmodellen og relasjonsalgebra (30)

1.1 ER-modell (18)

Lag et ER-diagram basert på beskrivelsen i oppgaveteksten under. Sørg for å inkludere: entitetstyper, attributter, nøkler, relasjonstyper, kardinaliteter, flerverdiattributter og eventuelle sammensatte attributter. For hver relasjonstype, husk å oppgi kardinalitet og marker om hver side har total eller partiell deltakelse.

Du skal modellere informasjon som et fritidsbåt-firma skal bruke som grunnlag for å designe en database-løsning. Databasen skal inneholde informasjon om kunder, kundenes båter, og fakturaer som sendes etter at vedlikeholdsarbeid er utført. Lag ER-diagram ut fra følgende beskrivelse.

En kunde identifiseres med kundenummer. Kunden har et navn. I tillegg har kunden en postadresse og en fakturaadresse. Begge disse adressene er satt sammen av gatenavn, gatenummer, og postnummer. Kunden skal ha et unikt telefonnummer som kunden kan nås på. I tillegg kan kunden ha registrert en eller flere andre telefonnummer som kunden kan ringe inn fra for å åpne porten til en havn utenfor åpningstid. Merk at de ekstra telefonnumrene er knyttet til kunden, ikke til båten.

Firma har et register over båttyper. En båttype er identifisert ved et kortnavn, og inneholder dessuten data om modell og typebetegnelse. For eksempel kan BE6 være kortnavn for modellen BRIG EAGLE 6 med typebetegnelse RIB.

En kunde uten en båt kan være registrert i systemet. Men dersom en båt er registrert, må den ha en unik kunde som står som ansvarlig for båten. Den må også være registrert med én (og bare én) båttype. Flere båter kan ha samme båttype. En båt identifiseres med båttype og eksemplar, der eksemplar skiller ulike forekomster av samme båttype. For eksempel, hvis det er registrert to ulike båter av modell BRIG EAGLE 6, vil disse kunne identifiseres som BE6-1 og BE6-2. Hvis båten ligger i firmaets havn, skal det registreres hvilken bås båten ligger i. Båsen identifiseres med et nummer og har tilknyttet informasjon om bredde og dybde. En bås kan være tom eller ha én båt, men ikke flere båter på samme tid.

Båttypene hentes ned fra et eksternt register, så det vil alltid være mange flere båttyper i databasen enn båter.

Når firmaet avtaler å gjøre vedlikehold på en båt, opprettes en unik arbeidsordre, identifisert med et ordrenummer, med tilhørende arbeidsbeskrivelse. En arbdisordre gjelder bare for én båt. Når vedlikeholdet er ferdig, opprettes en faktura med unikt fakturanummer og fakturabeløp. Hver arbeidsordre gir opphav til nøyaktig én faktura, og hver faktura hører til nøyaktig én arbeidsordre. Det registreres så dato for når faktura og tilhørende arbeidsordre sendes til kunden som skal betale fakturaen (det trenger ikke være båtens eier).

Løsningsforslag

En mulig korrekt ER-modell er som følger:

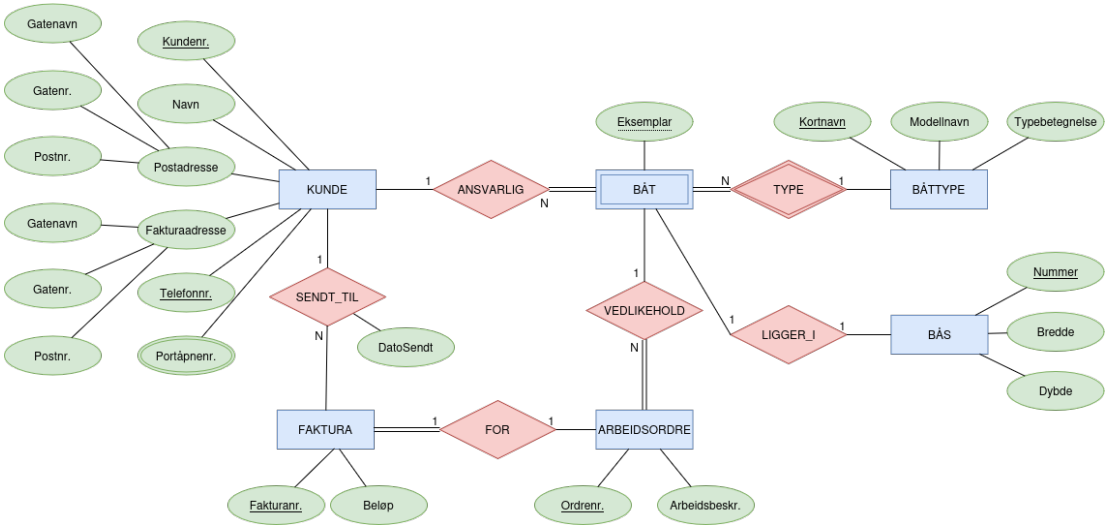


Figure 1: ER-modell

1.2 Oversettelse til relasjonsmodellen (7)

Oversett ER-modellen til et relasjonsskjema. Oppgi relasjoner (tabeller) med primærnøkler og fremmednøkler.

Løsningsforslag

Kunde(Kundenr, Navn, Post_gatenavn, Post_gatenummer, Post_postnummer, Faktura_gatenavn, Faktura_gatenummer, Faktura_postnummer, Hovedtelefon)

KN: {Kundenr}, {Telefonnr}

PN: {Kundenr}

Porttelefon(Kundenr, Telefon)

KN: {Kundenr, Telefon}

PN: {Kundenr, Telefon}

FN: Kundenr → Kunde(Kundenr)

Båttype(Kortnavn, Modell, Typebetegnelse)

KN: {Kortnavn}

PN: {Kortnavn}

Bås(BåsNr, Bredde, Dybde)

KN: {Nummer}

PN: {Nummer}

Båt(Kortnavn, Eksemplar, Kundenr, BåsNr)

KN: {Kortnavn, Eksemplar}

PN: {Kortnavn, Eksemplar}

FN1: Kortnavn → Båttype(Kortnavn)

FN2: Kundenr → Kunde(Kundenr)

FN3: BåsNr → Bås(BåsNr)

Arbeidsordre(Ordrenr, Kortnavn, Eksemplar, Arbeidsbeskrivelse)

KN: {Ordrenr}

PN: {Ordrenr}

FN: (Kortnavn, Eksemplar) → Båt(Kortnavn, Eksemplar)

Faktura(Fakturanr, Ordrenr, Beløp, Kundenr, DatoSendt)

KN: {Fakturanr}

PN: {Fakturanr}

FN1: Ordrenr → Arbeidsordre(Ordrenr)

FN2: Kundenr → Kunde(Kundenr)

1.3 Relasjonsalgebra (5)

I denne oppgaven skal du bruke relasjonsskjemaet du laget i forrige oppgave.

1. Skriv et relasjonsalgebrauttrykk som finner navn og kundenummer for alle kunder som bor i postnummer 1350.
2. Skriv et relasjonsalgebrauttrykk som finner navn og telefonnummer for alle kunder som har minst én båt.

Løsningsforslag

1. $$\pi_{Kundenr,Navn}(\sigma_{Post_postnummer=1350}(Kunde))$$

2. $$\pi_{Navn,Hovedtelefon}(Kunde \bowtie_{Kunde.Kundenr=Båt.Kundenr} Båt)$$

2 SQL

Oppgavene i denne delen er om SQL. Alle oppgavene vil bruke det samme databaseskjemaet, og en beskrivelse av dette er vedlagt som PDF for hver oppgave.

Du må gjerne inkludere kommentarer i spørringene (gjøres med "–" i SQL). Dersom det er uklarheter eller tvetydigheter i oppgavebeskrivelsen, bruk sunn fornuft og skriv en kommentar om hvilke antagelser og tolkinger du gjør.

2.1 Flervalg (10)

Anta at databasen inneholder nøyaktig de dataene som er vist i eksempel-dataene i den vedlagte PDFen. For hver spørring under, oppgi hvor mange rader det er i resultatet. Hvert riktig svar gir 2 poeng, mens hvert uriktige svar gir -0.5 poeng. Blankt svar/ubesvart gir 0 poeng. Laveste poengsum på hele oppgaven er 0.

1.

```sql
SELECT *
FROM ansatt
WHERE navn LIKE '%i%';
```

2.

```sql
SELECT *
```

```
FROM prosjekt AS p1
     NATURAL JOIN prosjekt AS p2;
```

3.

```
SELECT *
FROM ansatt
WHERE navn != navn;
```

4.

```
SELECT *
FROM prosjekt
LIMIT 1
OFFSET 5;
```

5.

```
SELECT *
FROM ansatt AS a
WHERE NOT EXISTS (
  SELECT *
  FROM prosjekt AS p
  WHERE a.aid = p.leder
);
```

Løsning

1. 3
2. 5
3. 0
4. 0
5. 2

2.2 Oppgaver i dag (5)

Skriv en SQL-spørring som finner beskrivelse og navn på ansvarlig ansatt på alle oppgaver med frist i dag. *Hint:* Du kan bruke `current_date` for dagens dato.

Løsningsforslag

```
SELECT o.beskrivelse, a.navn
FROM oppgave AS o JOIN ansatt AS a ON (o.ansvarlig = a.aid)
WHERE o.frist = current_date;
```

2.3 Emneord (7)

Skriv en SQL-kommando som oppretter ny tabell med navn `emneord` som skal ha følgende kolonner:

- `oppgave` som er en fremmenøkkel til `oppgave(oid)`
- `ord` som har typen `text` og må ha en lengde på 3 eller mer (du kan bruke funksjonen `length` for å finne lengden av en `text`, f.eks. `length('hei') = 3`)

Skriv også en SQL-kommando som gjør at tabellen inneholder følgende data:

```
 oppgave |   ord
---------+----------
       1 | Struktur
       2 | Design
       2 | CSS
```

De to kommandoene skal kjøres som én transaksjon, altså hvis én feiler så feiler begge.

Løsningsforslag

```
BEGIN;

CREATE TABLE emneord (
    oppgave int REFERENCES oppgave(oid),
    ord text CHECK (length(ord) >= 3)
);

INSERT INTO emneord VALUES
(1, 'Struktur'),
(2, 'Design'),
(2, 'CSS');

COMMIT;
```

2.4 Oppgaver uten ansvarlige (5)

Skriv en SQL-kommando som setter ansvarlig lik prosjektlederen for prosjektet som oppgaven tilhører, for alle oppgaver med `NULL` som ansvarlig.

Løsningsforslag

```
UPDATE oppgave
SET ansvarlig = (
        SELECT leder
        FROM prosjekt
        WHERE pid = prosjekt
```

```
    )
WHERE ansvarlig IS NULL;
```

2.5 Prosjektlengde (7)

La lengden til et prosjekt være antall dager fra første frist til siste frist på oppgavene til prosjektet (husk at i SQL kan man ta differansen mellom to datoer ved å ta den ene minus den andre, og får da en `int` som sier hvor mange dager det er mellom datoene). La videre et stort prosjekt være et prosjekt med flere enn 3 medlemmer.

Skrive en spørring som finner lengden til store prosjekter.

Løsningsforslag

```
WITH
  store_prosjekter AS (
    SELECT p.pid, p.navn
    FROM prosjekt AS p JOIN medlem AS m ON (p.pid = m.prosjekt)
    GROUP BY p.pid, p.navn
    HAVING count(*) > 3
  )
SELECT p.navn, max(o.frist) - min(o.frist) AS lengde
FROM store_prosjekter AS p JOIN oppgave AS o ON (p.pid = o.prosjekt)
GROUP BY pid, navn;
```

2.6 Prosent ferdig (8)

Skriv en SQL-kommando som oppretter et VIEW med navn `prosjektstatus` som inneholder navn på prosjektet og hvor mange prosent av oppgavene på prosjektet som er ferdige (altså `100 * (F/T)`, hvor `F` er antall ferdige oppgaver på prosjektet og `T` er antall oppgaver på prosjektet totalt). Denne siste kolonnen skal ha navn `prosent_ferdig`. Resultatet skal være sortert på `prosent_ferdig` fra høyest til lavest.

Løsningsforslag

```
CREATE VIEW prosjektstatus AS
WITH
  antall AS (
    SELECT prosjekt, count(ferdig) AS f, count(*) AS t
    FROM oppgave
    GROUP BY prosjekt
  )
SELECT p.navn,
       100 * (f::float/t) AS prosent_ferdig
FROM antall
```

```
     JOIN prosjekt AS p ON (prosjekt = p.pid)
ORDER BY prosent_ferdig DESC;
```

2.7 Timeregnskap (8)

Vi antar at det tar en ansatt 7 timer å gjennomføre én oppgave de er ansvarlige for. Videre antar vi at en ansatt som leder et prosjekt bruker 2 timer med oppfølging per oppgave som er del av prosjektet. Skriv en SQL-spørring som finner ut hvor mange timer hver ansatt jobber, samt navnet på den ansatte. Merk at det kan være ansatte som ikke jobber noen timer, og disse skal også med i resultatet (da enten med `0` eller `NULL` som antall timer).

Løsningsforslag

```
WITH
  timer AS (
    SELECT ansvarlig AS ansatt, 7 AS ant_timer
    FROM oppgave
    UNION ALL
    SELECT p.leder AS ansatt, 2 AS ant_timer
    FROM prosjekt AS p JOIN oppgave AS o ON (p.pid = o.prosjekt)
  )
SELECT a.navn, sum(t.ant_timer) AS timer_totalt
FROM ansatt AS a
     LEFT JOIN timer AS t ON (a.aid = t.ansatt)
GROUP BY a.aid, a.navn;
```

3 Relasjonsmodellen, normalformer og dekomposisjon (20)

3.1 Funksjonelle avhengigheter og instanser (4)

Vi betrakter relasjonen:

$$R(A, B, C, D)$$

med følgende funksjonelle avhengigheter:

- $A \to B$
- $B \to C$

Nedenfor ser du fire instanser av R. Avgjør om instansen bryter med noen av FDene. Du får 1 poeng per riktig svar, -1 poeng per galt svar, og 0 poeng for ubesvart. Minimum poengsum for hele oppgaven er 0 poeng.

Bryter instansen noen av FD-ene?

Instans I

A	B	C	D
1	x	y	9
1	x	y	8
2	w	z	7

Instans II

A	B	C	D
1	x	y	9
1	x	z	8
2	w	z	7

Instans III

A	B	C	D
1	x	y	9
1	w	y	9
2	w	y	7

Instans IV

A	B	C	D
1	x	y	9
1	NULL	y	9

Løsningsforslag

Instans I: NEI

Instans II: JA

Instans III: JA

Instans IV: JA

Altså, bare **Instans I** bryter ikke noen FD-er.

Forklaring:

- Instans I: $A = 1$ gir konsistent $B = x$ og $C = y$. $B = x$ gir konsistent $C = y$. (NEI)
- Instans II: $B = x$ gir både $C = y$ og $C = z$, altså bryter $B \rightarrow C$. (JA)
- Instans III: $A = 1$ gir $B = x$ og $B = w$, altså bryter $A \rightarrow B$. (JA)
- Instans IV: $A = 1$ gir $B = x$ og $B = NULL \rightarrow$ bryter $A \rightarrow B$. (JA)

3.2 Nøkler og normalformer (5)

Vi bruker samme relasjon og FDer som forrige oppgave, altså:

$$R(A, B, C, D)$$

med følgende funksjonelle avhengigheter:

- $A \rightarrow B$
- $B \rightarrow C$

a) Finn alle kandidatnøkler i R.

b) Bestem hvilken høyeste normalform (1NF, 2NF, 3NF eller BCNF) R er i. Begrunn kort.

Løsningsforslag

a) Kandidatnøkkel:

- Fra $A \rightarrow B \rightarrow C$ får vi at A bestemmer B og C.
- D bestemmes ikke av noen FD-er.
- Dermed må nøkkelen inkludere D.

Kandidatnøkkel:
$\{A, D\}$

b) Normalform:

- 1NF: Ja (atomære verdier).
- 2NF: **Brytes**, fordi $A \rightarrow B$ og $A \rightarrow C$ er avhengigheter fra del av sammensatt nøkkel $\{A, D\}$.
- Dermed også ikke 3NF og ikke BCNF.

Høyeste normalform: 1NF.

3.3 Tapsfri dekomponering (6)

Vi bruker igjen samme relasjon og FDer som forrige oppgave, altså:

$$R(A, B, C, D)$$

med følgende funksjonelle avhengigheter:

- $A \rightarrow B$
- $B \rightarrow C$

Bruk standardalgoritmen for tapsfri dekomponering.

a) Identifiser en FD som bryter BCNF.

b) Utfør første trinn av BCNF-dekomponeringen av R, slik at R blir dekomponert til to relasjoner R_1 og R_2 ved å bruke FDen du identifiserte i a). Oppgi de nye relasjonene med atributter.

c) Er denne dekomponeringen tapsfri? Begrunn kort.

Løsningsforslag

a) FD-en $A \to B$ bryter BCNF siden A ikke er supernøkkel.

b) Første trinn av BCNF-dekomponering:

- $R_1(A, B, C)$
- $R_2(A, D)$

c) Tapsfrihet:

Ja. A er fellesattributt og nøkkel i R_1.
Dette tilfredsstiller tapsfrihetskriteriet.

3.4 Stramme inn med flere funksjonelle avhengigheter (5)

Vi bruker igjen samme relasjon og FDer som forrige oppgave, altså:

$$R(A, B, C, D)$$

med følgende funksjonelle avhengigheter:

- $A \to B$
- $B \to C$

Vi ønsker å beholde relasjonen $R(A, B, C, D)$ uten å dekomponere den, men i stedet legge til flere funksjonelle avhengigheter slik at R blir i BCNF.

Foreslå ett konkret sett med nye funksjonelle avhengigheter som kan legges til de opprinnelige ($A \to B$ og $B \to C$) slik at R kommer i BCNF. Begrunn kort at den resulterende mengden av funksjonelle avhengigheter faktisk gir BCNF.

Løsningsforslag

Et mulig sett med FD-er som gjør R til BCNF:

- $B \to A$
- $A \to D$

Med disse får vi:

- $A^* = \{A, B, C, D\}$ ($A \to B \to C$ og $A \to D$)

- $B^* = \{A, B, C, D\}$ ($B \to A$, $B \to C$ og $A \to D$)

Dermed er både A og B supernøkler.
Alle ikke-trivielle FD-er har supernøkkel på venstresiden, altså R er i BCNF.

Andre sett som gjør A og B til supernøkler kan også godtas.