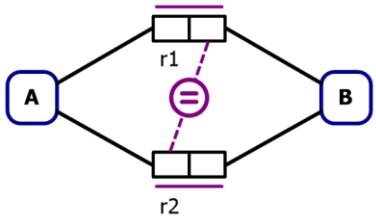
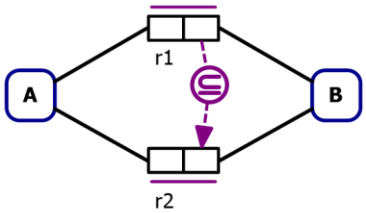
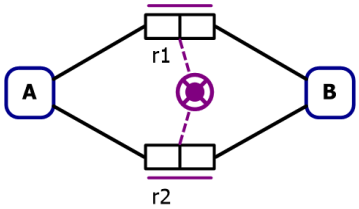
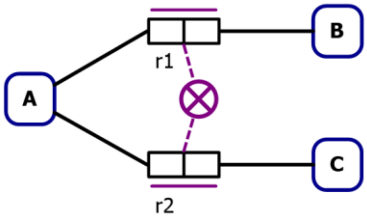
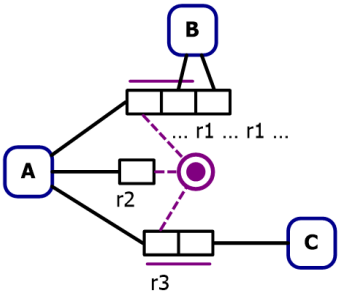
Løsningsforslag til eksamen i IN2090 – Databaser og datamodellering og INF1300 – Introduksjon til databaser

6. desember 2018 14:30 – 18:30 (4 timer)

1. Eksterne skranker (5%)

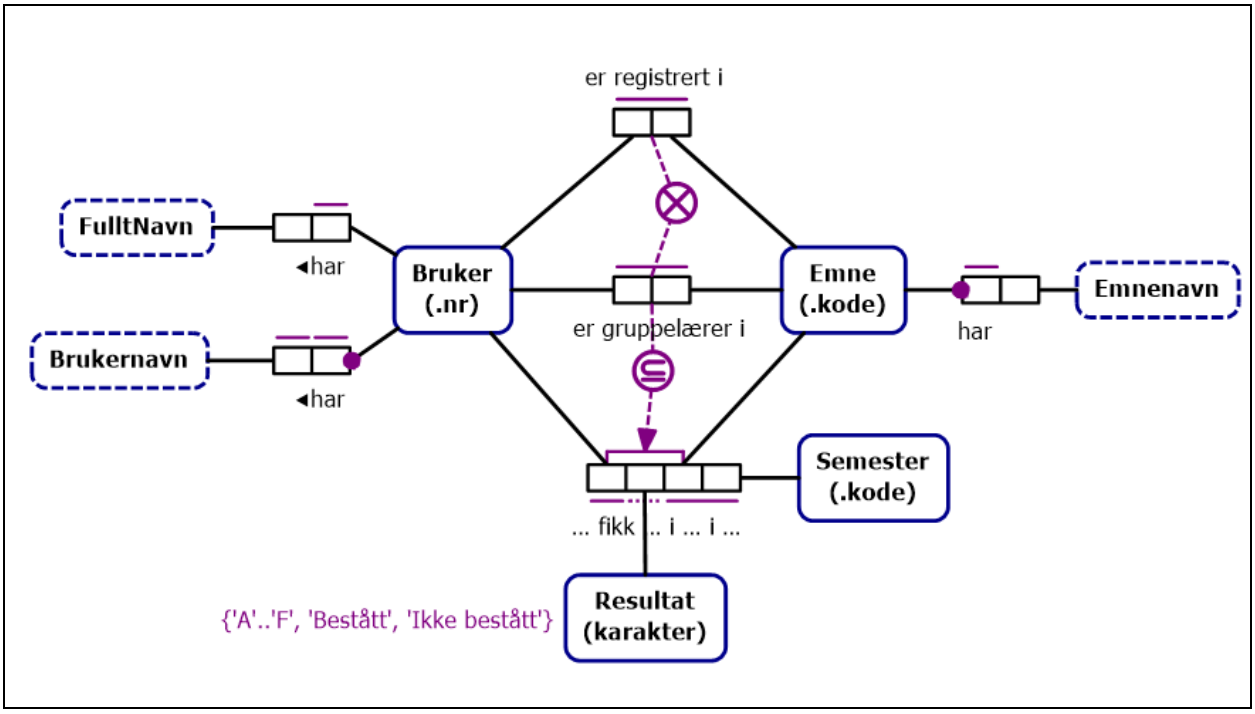
I modellene nedenfor (ORM2) skal du anta at alle begreper har en unik representasjon.

Er plasseringen av den eksterne skranken gyldig eller ugyldig?

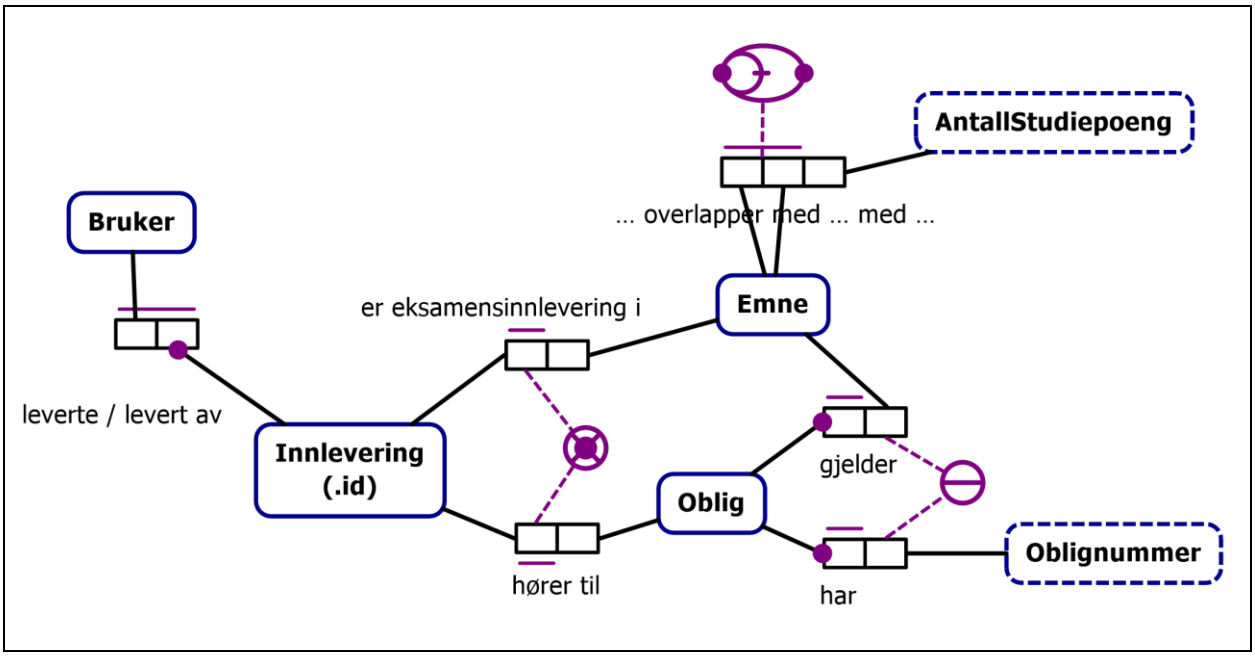
	**Ugyldig** ❌
	**Gyldig** ✔
	**Ugyldig** ❌
	**Ugyldig** ❌
	**Gyldig** ✔

2. Modellering i ORM (35%)

Del 1



Del 2



3. Realisering (10%)

Himmellegeme(navn, [oppdaget_årstall], [i_bane_rundt])
Romfartøy(navn, oppskytningsdato, status)
Besøk(himmellegeme, romfartøy, dato, oppdragstype)

Fremmednøkler:

- Himmellegeme(i_bane_rundt) → Himmellegeme(navn)
- Besøk(himmellegeme) → Himmellegeme(navn)
- Besøk(romfartøy) → Romfartøy(navn)

Realiseringen gir også relasjonene Dato(ymd) og År(tall). Disse relasjonene kan undertrykkes (slettes), og er derfor ikke obligatoriske å ha med. Studenten må selv finne et passende navn for relasjonen Besøk. Navnene på attributtene kan være annerledes, gitt at de i tilstrekkelig grad er beskrivende.

4. SQL, create (5%)

Skriv en CREATE-spørring for tabellen orders med passende datatyper for attributtene, slik at tabellen blir som beskrevet, inkludert primær- og fremmednøkler.

```sql
CREATE TABLE orders (
    id INT PRIMARY KEY,
    customer_id INT NOT NULL REFERENCES customers(id),
    shipping_addr VARCHAR NOT NULL,
    time_entered TIMESTAMP NOT NULL,
    status VARCHAR NOT NULL
);
```

5. SQL, view (5%)

Definer et view delayed_orders(id, customer_id, time_entered) som inneholder alle ordrer som har status 'delayed'.

```sql
CREATE VIEW delayed_orders(id, customer_id, time_entered) AS
    SELECT id, customer_id, time_entered
    FROM orders
    WHERE status = 'delayed';
```

6. SQL, kunder og ordrer (5%)

Skriv en spørring som returnerer alle kunder der navnet begynner med «Ola», samt deres ordrer hvis de har noen. Ta også med kunder som ikke har kjøpt noe. Resultatet skal inneholde kundens navn og adresse, samt ordrenes id og status, sortert etter registreringstidspunkt slik at de siste registrerte kundene kommer først.

```sql
SELECT name, address, o.id, o.status
FROM customers c
  LEFT JOIN orders o ON c.id = o.customer_id
WHERE name LIKE 'Ola%'
ORDER BY c.time_registered DESC;
```

7. SQL, aggregering (5%)

Skriv en spørring som finner antall ordrer per kunde for hver forskjellig status. Skriv ut kundens id og navn, samt antall og status.

```sql
SELECT c.id, c.name, COUNT(*), o.status
FROM customers c
  INNER JOIN orders o ON c.id = o.customer_id
GROUP BY c.id, c.name, o.status;
```

8. SQL, kunder med flere ordrer (5%)

Skriv en spørring som finner alle kunder som har minst to ordrer med forskjellig shipping_address. Skriv ut kundenes id og navn.

```sql
-- med self-join
SELECT DISTINCT c.id, c.name
FROM customers c
  INNER JOIN orders o1
    ON c.id = o1.customer_id
  INNER JOIN orders o2
    ON o1.customer_id = o2.customer_id AND o1.shipping_addr != o2.shipping_addr;

-- med gruppering og HAVING
SELECT c.id, c.name
FROM customers c
  INNER JOIN orders o ON c.id = o.customer_id
GROUP BY c.id, c.name
HAVING COUNT(DISTINCT shipping_addr) > 1;
```

9. SQL, produkter bestilt oftest (10%)

Skriv en spørring som returnerer alle kunder som har bestilt de produktene som har vært bestilt oftest (altså de produktene som finnes på flest forskjellige ordrer). Skriv ut kundenes id.

```sql
WITH num_sold AS (
  SELECT product_id, COUNT(order_id) as num
  FROM order_items
  GROUP BY product_id
)
SELECT DISTINCT o.customer_id
FROM num_sold
  INNER JOIN order_items oi ON num_sold.product_id = oi.product_id
  INNER JOIN orders o ON oi.order_id = o.id
WHERE num_sold.num = (SELECT MAX(ns.num) FROM num_sold ns);
```

10. SQL, total kostnad (10%)

Skriv en spørring som finner antall enheter og total kostnad for alle produkter solgt til kunder i USA, hvor minst 10 enheter har blitt solgt. Skriv ut produktenes id og navn, samt antall enheter og total kostnad.

```sql
SELECT
  oi.product_id,
  p.name,
  SUM(quantity) AS antall,
  SUM(quantity*unit_price) AS kostnad
FROM order_items oi
  INNER JOIN orders o ON oi.order_id = o.id
  INNER JOIN customers c ON o.customer_id = c.id
  INNER JOIN products p ON oi.product_id = p.id
WHERE c.country = 'USA'
GROUP BY oi.product_id, p.name
HAVING SUM(quantity) >= 10;
```

11. Relasjonsteori (5%)

a. Hvilke attributter i tabellen *customers* er kandidatnøkler?

id og {pnr, fdate}

b. Hvilken normalform er tabellen *customers* på? Begrunn svaret ditt.

Vi har følgende ikke-trivielle FD-er:

- id → pnr, fdate, name, address, country, time_registered
- {pnr, fdate} → id, name, adderss, country, time_registered

I begge FD-ene er venstre side en supernøkkel. Dermed er begge FD-ene på BCNF.

Tabellen er derfor på **BCNF**.

Det er også greit å si at venstre side er en kandidatnøkkel.