TDT4145 Øving 3

Philip Ditlevsen, Yngve Tryggestad Larsen, Natalie Sørensen Forshaw

Februar 2021

1 Lage SQL-tabeller og legge inn data

a Bruker CASCADE ON UPDATE og DELETE, og referrerer til plasseringen den originale tabellen, som vist under

```
CREATE TABLE SjangerForFilm (
    FilmID INTEGER NOT NULL,
    SjangerID INTEGER,
    CONSTRAINT Sjanger_FK FOREIGN KEY (SjangerID) REFERENCES Film(SjangerID)
        ON UPDATE CASCADE
        ON DELETE CASCADE,
    CONSTRAINT Film_FK FOREIGN KEY (FilmID) REFERENCES Film(FilmID)
        ON UPDATE CASCADE
        ON DELETE CASCADE;
)
CREATE TABLE SkuespillerIFilm (
    FilmID INTEGER,
    SkuespillerID INTEGER,
    Rolle VARCHAR(40),
    CONSTRAINT S_FK FOREIGN KEY (SkuespillerID) REFERENCES Skuespiller(SkuespillerID)
        ON UPDATE CASCADE
        ON DELETE CASCADE,
    CONSTRAINT Film_FK FOREIGN KEY (FilmID) REFERENCES Film(FilmID)
        ON UPDATE CASCADE
        ON DELETE CASCADE;
)
```

b
```
CREATE TABLE Film (
    FilmID INTEGER NOT NULL,
    Tittel VARCHAR(40),
    Produksjonsår INTEGER,
    RegissørID Integer,
    Constraint Film_PK PRIMARY KEY FilmID,
)
CREATE TABLE Skuespiller(
```

```
    SkuespillerID INTEGER NOT NULL,
    Navn VARCHAR(40),
    Fødselsår INTEGER,
    Constraint Skuespiller_PK PRIMARY KEY SkuespillerID,
)
CREATE TABLE Regissør(
    RegissørID INTEGER NOT NULL,
    Navn VARCHAR(40),
    Constraint Regissør_PK PRIMARY KEY RegissørID,
)
CREATE TABLE Sjanger(
    SjangerID INTEGER NOT NULL,
    Navn VARCHAR(40),
    Beskrivelse VARCHAR(40),
    Constraint Sjanger_PK PRIMARY KEY SjangerID,
)
```

c
```
INSERT INTO Regissør VALUES(1, \Peyton Reed"), (2, "Tom Shadyac");
INSERT INTO Film VALUES(1, \Yes Man", 2008, 1);
INSERT INTO Skuespiller VALUES(1, \Jim Carrey", 1962);
INSERT INTO SkuespillerIFilm VALUES(1, 1, \Carl");
```

d
```
UPDATE Skuespiller
SET Navn = \James Eugene Carrey"
WHERE SkuespillerID = 1;
```

e
```
DELETE FROM Regissør WHERE RegissørID = 2;
```

2 Spørringer i SQL

a
```
SELECT * FROM film
```

b
```
SELECT skuespiller.navn FROM skuespiller WHERE Fødselsår $>$ 1960
```

c
```
SELECT skuespiller.Navn FROM skuespiller
WHERE Fødselsår $>$ 1979 AND Fødselsår $<$ 1990
ORDER BY skuespiller.Navn ASC
```

d
```
SELECT skuespillerifilm.Rolle, film.Tittel
FROM skuespillerifilm INNER JOIN skuespiller INNER JOIN film
WHERE skuespiller.Navn = "Morgan Freeman"
AND skuespillerifilm.SkuespillerID = 14 AND skuespillerifilm.FilmID = film.FilmID
```

e
```
SELECT film.Tittel
FROM skuespillerifilm INNER JOIN skuespiller INNER JOIN film INNER JOIN regissør
WHERE skuespiller.Navn = regissør.Navn
AND skuespillerifilm.FilmID = film.FilmID
AND film.RegissørID = regissør.RegissørID
```

```
  AND skuespillerifilm.SkuespillerID = skuespiller.SkuespillerID
```

f
```
SELECT COUNT(skuespiller.Navn) FROM skuespiller WHERE skuespiller.Navn LIKE 'C\%'
```

g
```
SELECT sjanger.Navn AS SJANGERNAVN, COUNT(film.Tittel) AS ANTALLFILMER
FROM sjanger NATURAL JOIN sjangerforfilm NATURAL JOIN film
WHERE sjangerforfilm.SjangerID = sjanger.SjangerID
AND sjangerforfilm.FilmId = film.FilmID
GROUP BY sjanger.Navn
```

h
```
SELECT skuespiller.Navn FROM skuespiller
INNER JOIN skuespillerifilm
ON skuespiller.SkuespillerID = skuespillerifilm.skuespillerID
INNER JOIN film
ON film.filmID = skuespillerifilm.filmID
WHERE film.tittel = "Ace Ventura: Pet Detective" AND skuespiller.navn
NOT IN
(SELECT skuespiller.Navn FROM skuespiller
    INNER JOIN skuespillerifilm
    ON skuespiller.SkuespillerID = skuespillerifilm.skuespillerID
    INNER JOIN film
    ON film.filmID = skuespillerifilm.filmID
    WHERE film.tittel = "Ace Ventura: When Nature Calls")
```

i
```
SELECT Tittel, film.filmID, AVG(fødselsår) AS gjennomsnittlig
From film INNER JOIN skuespillerifilm
ON film.filmid=skuespillerifilm.filmid INNER JOIN skuespiller
ON skuespiller.skuespillerid = skuespillerifilm.skuespillerid
GROUP BY Tittel
HAVING gjennomsnittlig > (SELECT AVG(fødselsår) FROM skuespiller)
```

3 Flere spørringer i relasjonsalgebra

a

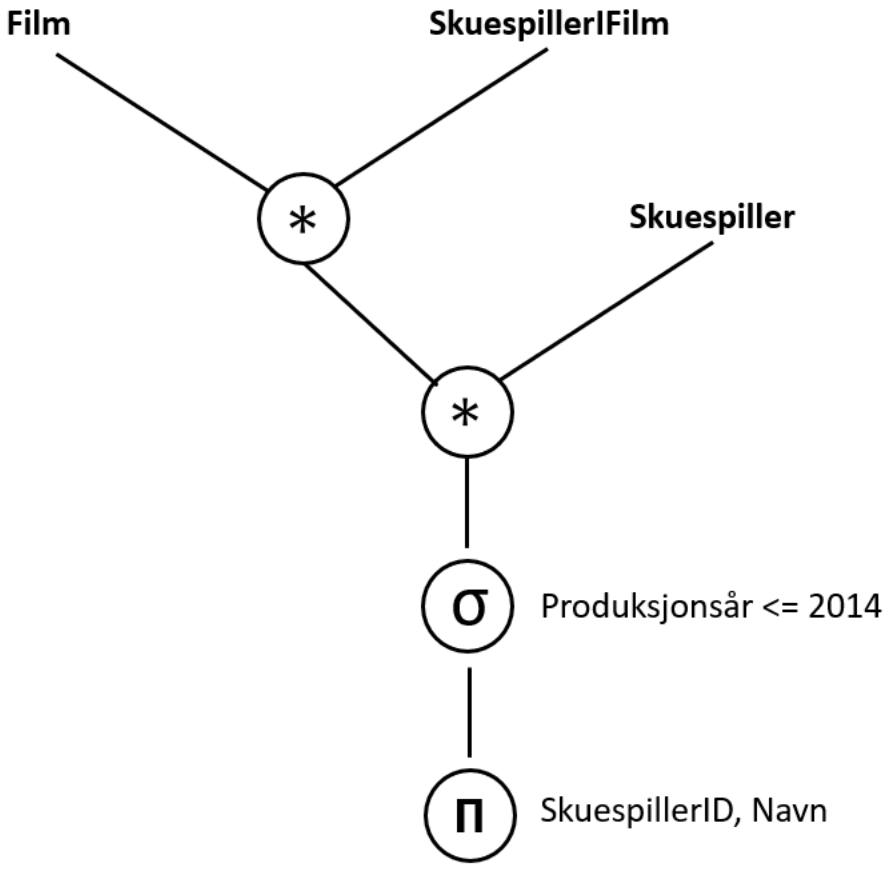

Figure 1: Finner skuespillerID og navn på de skuespillere som ikke har spilt i en film produsert etter 2014.

b

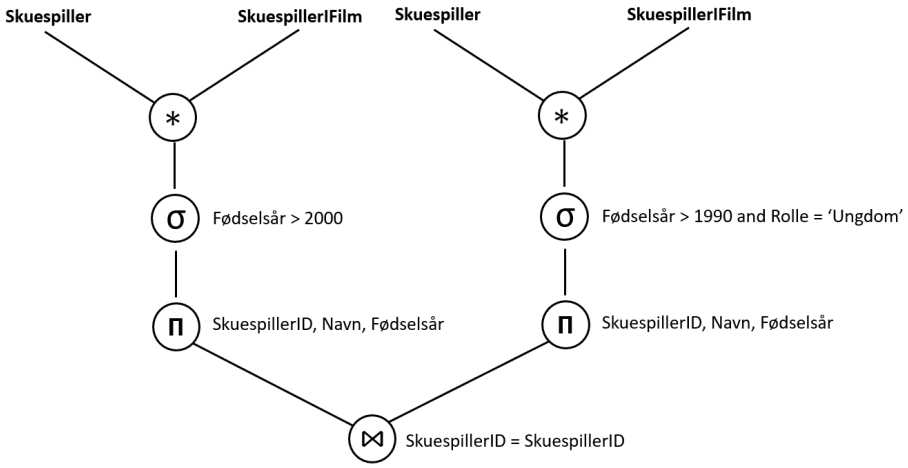

Figure 2: Henter skuespillerID, navn og fødselsår på skuespillerne som er født senere enn 2000, og de skuespillerne som er født senere enn 1990 og har spilt i en film som en ungdom (Rolle = "Ungdom").

c

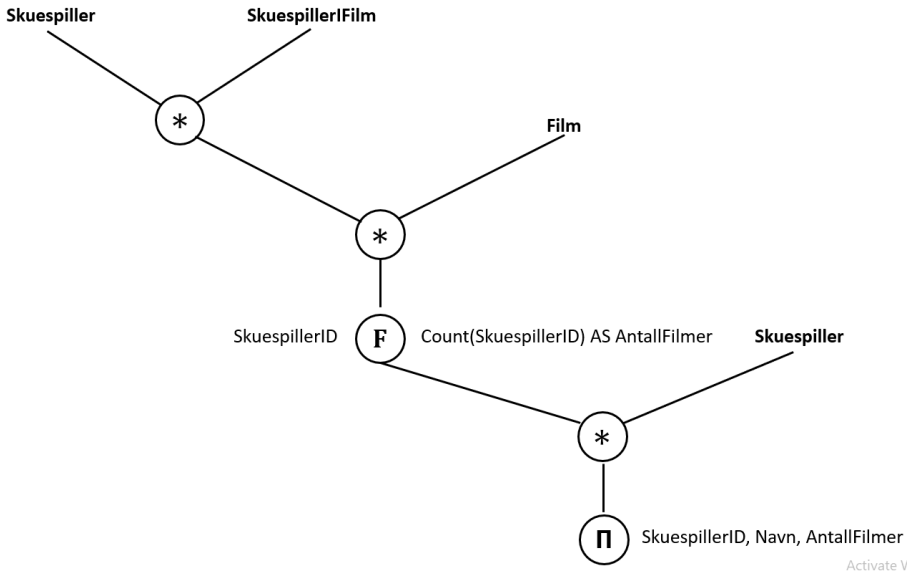

Figure 3: For hver skuespiller finner skuespillerID, navnet og antall filmer vedkommende har spilt i.

d

4 Introduksjon til normaliseringsteori

a 10 ganger

PersonID	Navn	Telefonnr	FakultetsID
270393	Ola Johansen	73735667	1
313874	Kari Vintermo	73739023	1
241257	Bernt Nilsen	73731234	1
935784	Olav Foss	73738471	1
345481	Per Høyder	73739021	1
134876	Åse Bekkerud	73746617	2

Figure 4: Person

b

FakultetsID	Fakultetskode	Fakultetsnavn	Fakultetsbygg
1	EDI	Institutt for energi og datainnovasjon	Oasen
2	IØA	Institutt for økonomiske arkitekturer	Solbakken

Figure 5: Fakultet

c

5 Funksjonelle avhengigheter, nøkler og tilluk-ning

a $A \to C$ er umulig fordi $a1 \to c1$ og $a1 \to c2$.

$AB \to C$ er umulig fordi $a1b1 \to c1$ og $a1b1 \to c2$.

$D \to C$ er umulig fordi $d2 \to c2$ og $d2 \to c4$.

A kan ikke være en kandidatnøkkel fordi $A \to C$ ikke stemmer med tabellen.

b A+ = AC

D+ = D

BC+ = BCD

AB+ = ABCD