

Filmdatabasen

[ORM-versjon]

Ellen Munthe-Kaas

April 2010

Bakgrunn

I forbindelse med SQL-undervisningen skal studentene trene på å utføre spørringer mot en enkel relasjonell database. Inspirasjonen for denne testdatabasen er The Internet Movie Database (*imdb*) [1]. Kort fortalt er dette et nettsted der man kan finne informasjon om ca. 700000 filmer og 60000 TV-serier, 1,7 millioner personer som på en eller annen måte har tilknytning til filmindustrien, 11 millioner tilknytninger mellom personer og filmer osv¹.

Historikk

En god del av lærebokseksemplene bærer preg av å være kunstig enkle og/eller oppkonstruerte, noe som gjør dem relativt kjedelige å lære av. Derfor ble det i 2002² laget et utsnitt av *imdb* for å gi studentene en følelse av å jobbe med virkelige data. Dessverre viste det seg at Ifis daværende databaseserver ikke taklet relativt enkle spørringer med et fullt speil av *imdb*. Man valgte derfor å ta som utsnitt data om filmer som på en eller annen måte er knyttet til Frankrike. Denne første utgaven av databasen ble realisert på Sybase ASE 11.9.2 og var i bruk i kurset Inf212 (en forløper til Inf3100). Senere ble det besluttet at Ifi skulle migrere til Oracle, og databasen ble lagt over på Oracle 9.2i. Denne versjonen var i bruk i Inf3100 tilogmed våren 2007. I 2007 besluttet Ifi å migrere til Postgres; i den forbindelsen la vi opp en ny fullversjon³ av *imdb*.

Design av 2007-versjonen

Til 2007-versjonen ble det laget nye scripts for å parsere flatfilene som *imdb* tilbyr alle interessenter. Det må bemerkes at originalformatet er langt fra fornuftig, og opplagt ikke designet av databasemennesker.

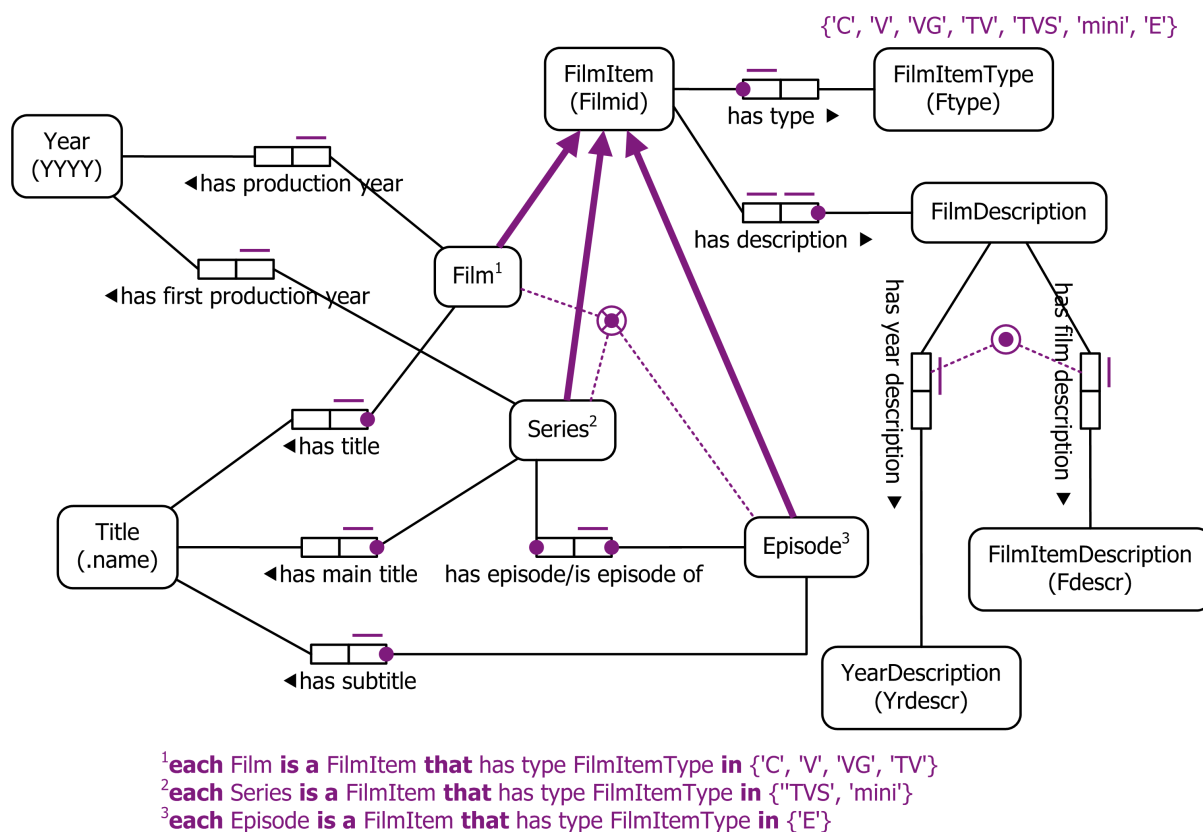
Filmer og TV-serier

2007-databasen omfatter ordinære kinofilmer såvel som TV-filmer og TV-serier mm. *FilmItem* omfatter alle typer "filmer", fra ordinære filmer til episoder i TV-serier. Dessuten inneholder den for hver gruppe av episoder som sammen danner en serie, en forekomst som representerer serien som sådann, og som man kan bruke til å finne tittelen på serien. Under de enkelte episodene finner vi derfor bare en episodetittel (subtittel), ikke seriens tittel.

¹ Tallene er fra april 2007.

² Primus motor bak opprettelsen av 2002-databasen var Igor Rafienko. David Ranvig konverterte *imdb*-filer til SQL insert-setninger. Rune Aske bidro til å få testet *imdb*-speilet.

³ 2007-versjonen er fullstendig hva gjelder filmer. Dessuten er alle data tatt med for de personkategoriene som er angitt under design av 2007-databasen lenger ned i dette skrevet, som skuespillere, regissører, produsenter, kostymedesignere, osv. Derimot er det fortsatt noen typer informasjon som ikke er tatt med, som produksjonsselskaper, distributører oa.



Figur 2: FilmItem og dens subbegreper

Forklaring på filmitemtyper:

- C vanlig kinofilm
- V videofilm
- VG videospill
- TV TV-film
- TVS TV-serie
- mini mini-TVserie
- E episode i en TV-serie eller mini-TVserie. Den tilhørende Series-forekomsten har type TVS eller mini.

filmdescription inneholder litt tilleggsinformasjon til hvert filmitem. Jeg har valgt å ha dette som en egen relasjon og ikke slå den sammen med filmitem, blant annet fordi dette er relativt lavkvalitets⁴ informasjon. For episoder er det likevel verdt å merke seg at informasjon om produksjonsår muligens står her. Det ligger ingen informasjon om episodenes produksjonsår på hver enkelt episode (dvs. det kan muligens være innplassert som del av episodetittelen), og forekomsten som representerer serien som sådann, inneholder bare informasjon om det året serien startet.

```
create table filmitem (  
  filmid int primary key,  
  filmtime varchar(4) not null  
);
```

⁴ Med "lavkvalitet" menes at informasjonen som fins, bærer noe preg av å være lite enhetlig og relativt tilfeldig.

```
create table film (  
  filmid int primary key references filmitem (filmid),  
  title text not null,  
  prodyear int  
);  
  
create index filmtitleindex on film (title);  
create index filmyearindex on film (prodyear);  
  
create table filmdescription (  
  filmid int primary key references filmitem (filmid),  
  year text,  
  filmdescr text,  
  check (year is not null or filmdescr is not null)  
);  
  
create table series (  
  seriesid int primary key references filmitem (filmid),  
  maintitle text not null,  
  firstprodyear int  
);  
  
create index seriesmaintitleindex on series (maintitle);  
  
create table episode (  
  episodeid int primary key references filmitem (filmid),  
  seriesid int not null references filmitem (filmid),  
  subtitle text not null,  
  foreign key (seriesid) references series (seriesid)  
);
```

Tilleggsinformasjon knyttet til filmer

For noen filmer fins det en oversikt over alternative titler (AlternativeFilmTitle). Dessuten kan det til hver film være angitt ett eller flere produksjonsland (FilmCountry) og språk (FilmLanguage). Country og Language omfatter henholdsvis alle land og alle språk. For noen filmer fins litt tilleggsinformasjon knyttet til hvert enkelt tilhørende språk, dette er representert ved forekomster i FilmLanguageInfo. En film kan være karakterisert ved en eller flere genre (FilmGenre). Dessuten kan det til en film finnes informasjon om filmens lengde ved visning i forskjellige land (RunningTime). Også her kan det finnes noe tilleggsinformasjon knyttet til en kjøretidsangivelse (RunningTimeInfo). Endelig tillater imdb interessenter å stemme på filmer, der man kan gi en film poeng fra 1 til 10. Antall stemmer og hvordan stemmene fordeler seg på de forskjellige poengangivelsene (Distribution), er tilgjengelig via forekomster i FilmRating. Dessuten er det (for noen av filmene) utarbeidet en rang basert på disse stemmene og noen tilleggs-kriterier. Vi har ikke tilgang på det nøyaktige antall stemmer i hver poengkategori; i stedenfor har vi totalantallet og en fordelingsnøkkel mellom poengkategoriene. Distribution er derfor realisert ved en streng på 10 tegn.

1.	tegn er kode for hvor mange stemmer som stemte 1 (dårligste karakter)
2.	"-
...	
10	"-
	10 (beste karakter)

Kodene er:

- "." ingen stemmer
- "0" 1-9%
- "1" 10-19%
- ...
- "9" 90-99%
- "*" 100%

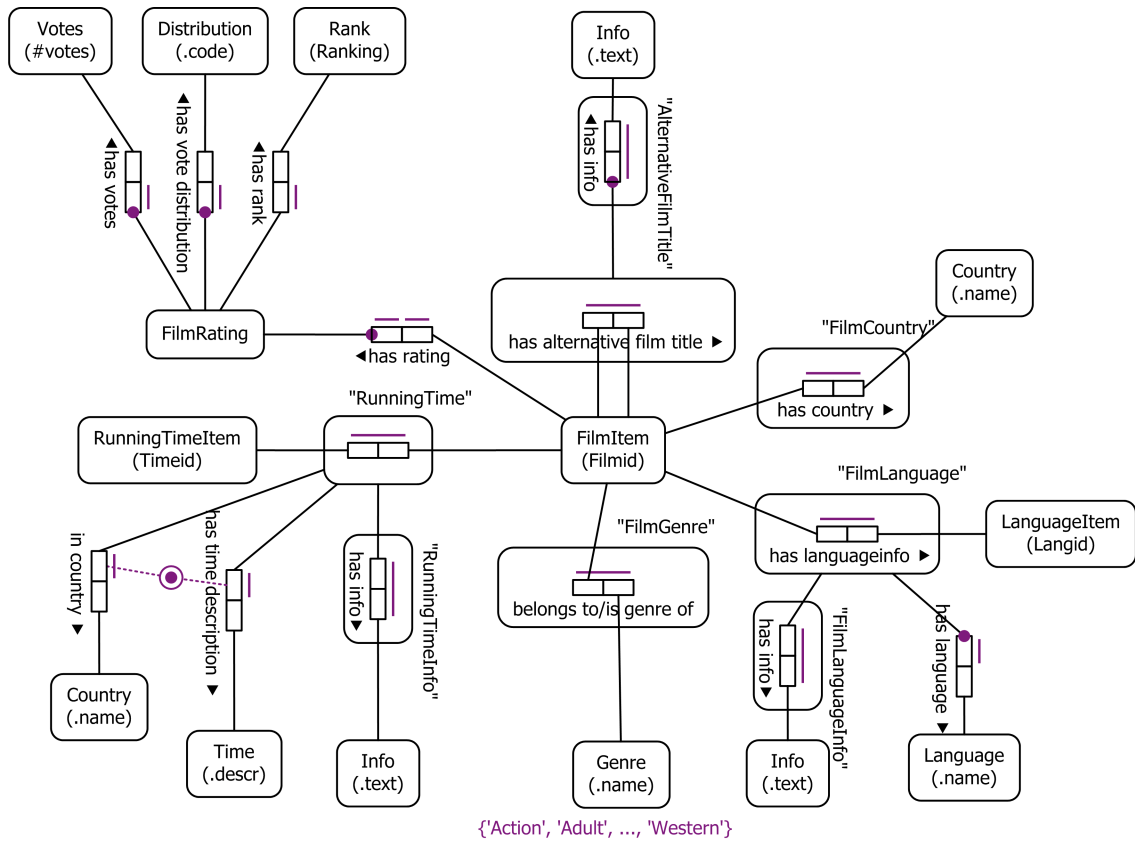
Rangen er vektet etter formelen $(v / (v+k)) * X + (k / (v+k)) * C$ hvor

X = snittet for filmen (middeltall)

v = antall stemmer filmen har fått

k = minimum stemmer som kreves for å komme med på listen over de 250 beste (pr. i dag 1300 stemmer)

C = snittstemmer over hele datamengden (for inneværende datasett: 6,90)



Figur 3: Tilleggsinformasjon knyttet til filmer

```
create table alternativefilmtitle (
  filmid int references filmitem (filmid),
  akaid int references filmitem (filmid),
  info text not null
);

create index alternativefilmtitlefilmidindex
  on alternativefilmtitle (filmid);
create index alternativefilmtitleakaidindex
  on alternativefilmtitle (akaid);

create table filmcountry (
  filmid int references filmitem (filmid),
  country text,
  primary key (filmid, country)
);

create index filmcountryfilmidindex on filmcountry (filmid);

create table country (
  country text primary key
);

create table filmlanguage (
  filmid int references filmitem (filmid),
  langid int,
  language text not null,
  primary key (filmid, langid)
);
create index filmlanguagefilmidindex on filmlanguage (filmid);

create table filmlanguageinfo (
  filmid int not null,
  langid int not null,
  info text not null,
  foreign key (filmid, langid) references filmlanguage (filmid, langid)
);

create index filmlanguageinfofilmidlangidindex
  on filmlanguageinfo (filmid, langid);

create table language (
  language text primary key
);

create table filmgenre (
  filmid int references filmitem (filmid),
  genre text,
  primary key (filmid, genre)
);

create index filmgenrefilmidindex on filmgenre (filmid);
create index filmgenregenreindex on filmgenre (genre);

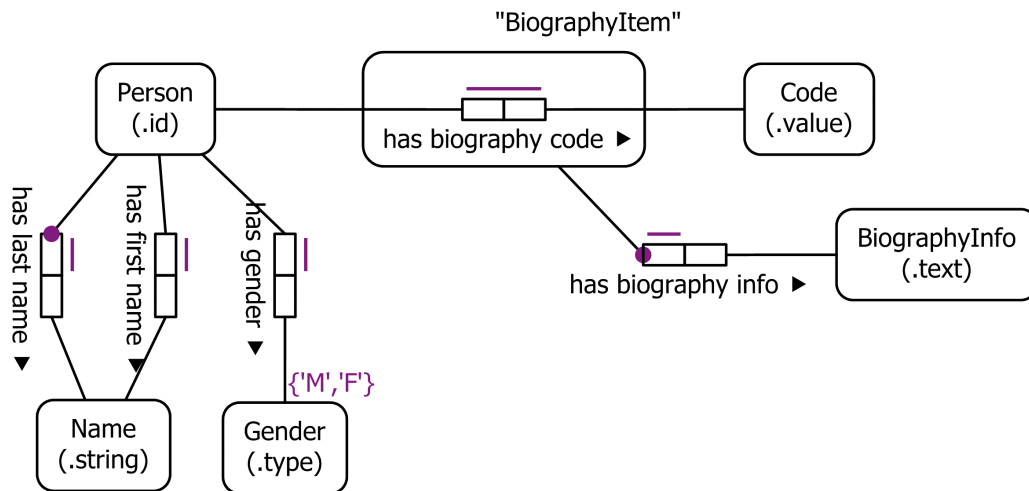
create table genre (
  genre text primary key
);
```

```
create table runningtime (  
  filmid int references filmitem (filmid),  
  timeid int,  
  time text,  
  country text,  
  primary key (filmid, timeid),  
  check (time is not null or country is not null)  
);  
  
create index runningtimefilmidindex on runningtime (filmid);  
  
create table runningtimeinfo (  
  filmid int not null,  
  timeid int not null,  
  info text not null,  
  foreign key (filmid, timeid) references runningtime (filmid, timeid)  
);  
  
create index runningtimeinfotimeidindex on runningtimeinfo (filmid,  
timeid);  
  
create table filmrating (  
  filmid int primary key references filmitem (filmid),  
  votes int not null,  
  distribution char(10) not null,  
  rank float(3)  
);
```

Personer

Alle personer har et etternavn, de fleste (men ikke alle) også fornavn. Kjønn er bare tatt med for skuespillere, siden disse i den opprinnelige databasen er delt i actors og actresses og det derfor lar seg gjøre å bestemme kjønnstilhørighet. For øvrige personer er det ingen enkel måte å få tak i rett kjønn på. For noen ganske få personnavn går navnene igjen både som actor og actress; for disse er kjønn derfor ikke angitt. Jeg har ikke forsøkt å parsere i fornavn og mellomnavn, så under hasFirstName ligger alt som ikke er karakterisert som etternavn. Under BiographyItem fins relativt lange tekster under diverse koder. Disse kodene sier noe om hva den biografiske teksten omtaler, noen av dem (med min gjetning på hva de representerer) er følgende:

- RN: Real name
- TR: Trade
- DB: Date of birth
- DD: Date of death
- NK: Nickname
- BG: Background
- BY: Biographer
- SP: Spouse
- HT: Height
- OW: Other work
- CV: Curriculum vitae
- QU: Quotations



Figur 4: Person

```
create table person (
  personid int primary key,
  lastname text not null,
  firstname text not null,
  gender char(1),
  check (gender = 'M' or gender = 'F');
);
```

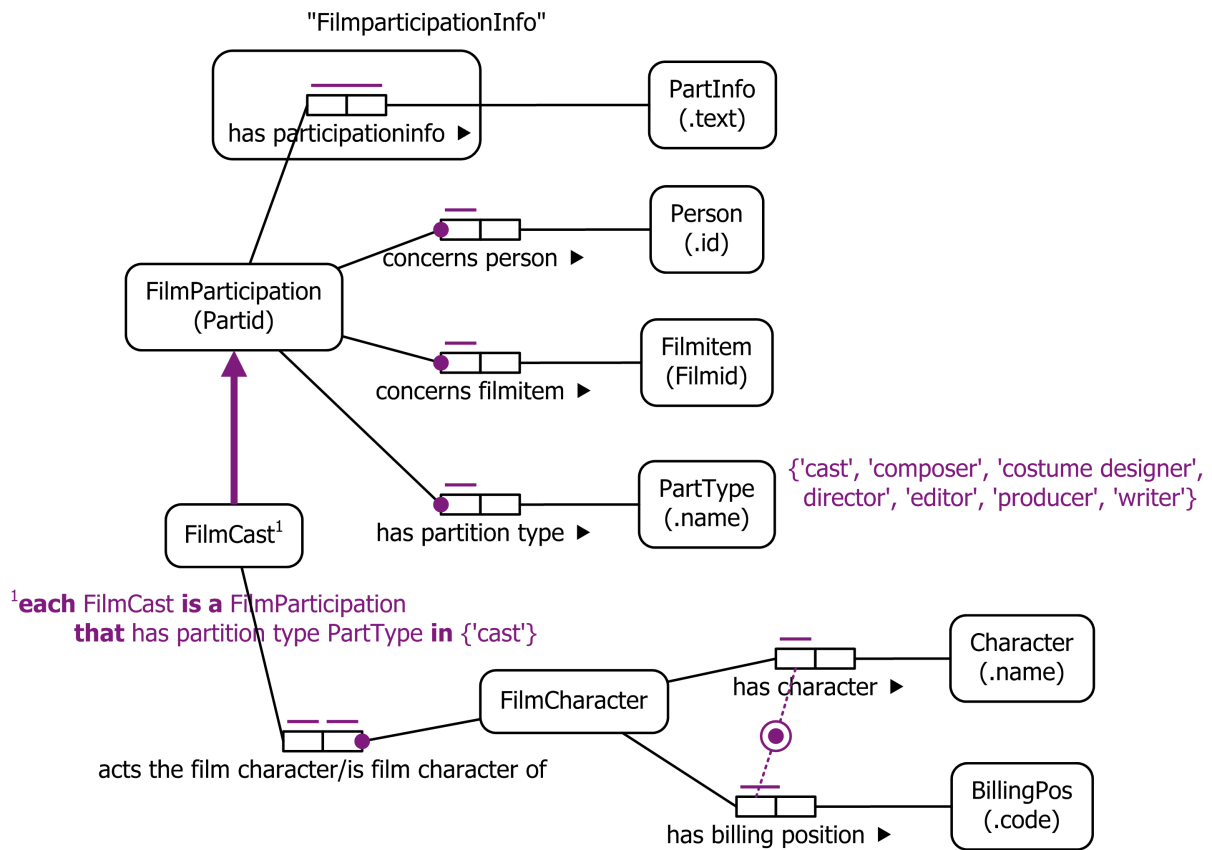
```
create index personlastnameindex on person (lastname);
```

```
create table biographyitem (
  personid int references person (personid),
  code char(2),
  description text not null,
  primary key (personid, code)
);
```

Filmdeltakelse

Personer deltar i filmer under forskjellige funksjoner. For øyeblikket er disse *cast*, *composer*, *costume designer*, *director*, *editor*, *producer*, *writer*. Tilknytninger til en film er identifisert ved forekomster i *FilmParticipation*. En person kan delta i en og samme film som f.eks. både skuespiller (*cast*) og regissør (*director*), eller inneha mer enn én rolle i en film, derfor kan det være to eller flere forekomster i *FilmParticipation* som vedrører samme *Person/FilmItem/PartType*-trippel.

For alle filmdeltakelser (skuespillere og andre) kan det finnes en eller flere informasjonsbiter om filmdeltakelsen (*FilmParticipationInfo*). Dette er relativt tilfeldig informasjon av lav kvalitet. For skuespillere fins i tillegg informasjon om de rollene de har i en film, typisk i form av navnet på den personen de spilte (*Character*). Dessuten kan det for skuespillere finnes informasjon om 'billingposisjon' (*BillingPos*), som sier noe om hvor fremtredende rollen er. Vi undertrykker begrepet *FilmCast*.



Figur 5: Filmdeltakelse

```
create table filmparticipation (
  partid int primary key,
  personid int not null references person (personid),
  filmid int not null references filmitem (filmid),
  parttype text not null
);

create index filmparticipationpersonidindex
  on filmparticipation (personid);
create index filmparticipationfilmidindex
  on filmparticipation (filmid);

create table filmcharacter (
  partid int primary key references filmparticipation (partid),
  filmcharacter text,
  billingpos int,
  check (filmcharacter is not null or billingpos is not null)
);

create table filmparticipationinfo (
  partid int not null references filmparticipation (partid),
  info text not null
);

create index filmparticipationinfopartidindex
  on filmparticipationinfo (partid);
```


Eksempler

Eksemplene er kjørt på en versjon av filmdatabasen som ble lastet ned i april 2007.

Luc Bessons deltakelse i filmen The Fifth Element:

```
brukernavn=> select x.partid, x.personid, x.filmid, x.parttype
brukernavn-> from filmparticipation x, person p, film f
brukernavn-> where
brukernavn->   p.lastname = 'Besson' and
brukernavn->   p.firstname = 'Luc' and
brukernavn->   f.title = 'Fifth Element, The' and
brukernavn->   x.personid = p.personid and
brukernavn->   x.filmid = f.filmid;
```

```
  partid | personid | filmid | parttype
-----+-----+-----+-----
   781285 |    89222 | 237127 | director
  1009544 |    89222 | 237127 | writer
  1009560 |    89222 | 237127 | writer
  1009576 |    89222 | 665467 | writer
  1009592 |    89222 | 665467 | writer
(5 rows)
```

```
brukernavn=>
```

Milla Jovovich's roller i The Fifth Element:

```
brukernavn=> select filmcharacter
brukernavn-> from filmcharacter
brukernavn-> where partid = 19580594 or partid = 19580610;
  filmcharacter
```

```
-----
  Leeloo
  Leeloo
(2 rows)
```

```
brukernavn=>
```

Det fins følgende tilleggsinformasjon om disse filmdeltakelsene:

```
brukernavn=> select partid, info
brukernavn-> from filmparticipationinfo
brukernavn-> where partid = 19580594 or partid = 19580610;
  partid | info
```

```
-----+-----
  19580610 | voice
(1 row)
```

```
brukernavn=>
```

Grunnen til at det er to filmid'er for The Fifth Element, er at det fins en VG-utgave av filmen også:

```
brukernavn=> select f.filmid, title, prodyear, filmttype
brukernavn-> from film f, filmitem i
brukernavn-> where
brukernavn->   f.title = 'Fifth Element, The' and
brukernavn->   f.filmid = i.filmid;
```

```
filmid | title | prodyear | filmtyp
-----+-----+-----+-----
237127 | Fifth Element, The | 1997 | C
665467 | Fifth Element, The | 1998 | VG
(2 rows)
```

brukernavn=>

Alle roller som Milla Jovovich har spilt, med tilhørende filmtittel⁵:

```
brukernavn=> select f.title, c.filmcharacter
brukernavn-> from filmparticipation x, person p, film f, filmcharacter c
brukernavn-> where
brukernavn->   x.parttype = 'cast' and
brukernavn->   p.lastname = 'Jovovich' and
brukernavn->   p.firstname = 'Milla' and
brukernavn->   x.personid = p.personid and
brukernavn->   x.filmid = f.filmid and
brukernavn->   x.partid = c.partid;
```

```
title | filmcharacter
-----+-----
.45 | Kate
AFI's 100 Years... 100 Cheers: America's Most Inspiring Movies | Herself
Cannes: Through the Eyes of the Hunter | Herself
Chaplin | Mildred Harris
Claim, The | Lucia
Corporate Malfeasance | Herself
Dazed and Confused | Michelle Burroughs
Dummy | Fangora
Fifth Element, The | Leeloo
Fifth Element, The | Leeloo
Game Babes | Herself
Game Over: 'Resident Evil' Reanimated | Herself
He Got Game | Dakota Burns
House on Turk Street, The | Erin
Kuffs | Maya Carlton
Making and Meaning of 'We Are Family', The | Herself
Messenger: The Story of Joan of Arc, The | Joan of Arc
Million Dollar Hotel, The | Eloise
Night Train to Kathmandu, The | Lily McLeod
Playing Dead: 'Resident Evil' from Game to Screen | Herself
Resident Evil | Alice
Resident Evil: Apocalypse | Alice
Resident Evil: Extinction | Alice
Return to the Blue Lagoon | Lilli
Star Element, The | Herself
Starz on the Set: Ultraviolet | Herself
Teen Vid II | Herself
Trailer for a Remake of Gore Vidal's Caligula | Druscilla
Two Moon Junction | Samantha Delongpre
Ultraviolet | Violet
VH1/Vogue Fashion Awards | Herself
You Stupid Man | Nadine
Zoolander | Katinka
(33 rows)
```

brukernavn=>

⁵ Resultatsettet er gjengitt med mindre font for å slippe å brette linjene i resultatsettet.

Dette er imidlertid bare kinofilmene hennes. Milla Jovovich har også spilt i TV-serier, og slik får vi tak i disse:

```
brukernavn=> select s.maintitle, e.subtitle, c.filmcharacter
brukernavn-> from filmparticipation x, person p,
brukernavn-> episode e, filmcharacter c, series s
brukernavn-> where
brukernavn-> x.parttype = 'cast' and
brukernavn-> p.lastname = 'Jovovich' and
brukernavn-> p.firstname = 'Milla' and
brukernavn-> x.personid = p.personid and
brukernavn-> x.filmid = e.episodeid and
brukernavn-> x.partid = c.partid and
brukernavn-> e.seriesid = s.seriesid;
```

maintitle	subtitle	filmcharacter
4Pop	Nuo surkeat Hollywood-pelit (#3.6)	Herself
Grand journal de Canal+, Le	(2005-05-20)	Herself
Harald Schmidt Show, Die	(2002-03-19)	Herself
HBO First Look	The Messenger: The Story of Joan of Arc	Herself
HypaSpace	(#5.39)	Herself
HypaSpace	(#5.40)	Herself
HypaSpace	(#5.42)	Herself
HypaSpace	(#5.44)	Herself
HypaSpace	(#5.45)	Herself
King of the Hill	Get Your Freak Off (#7.1)	Serena
Late Late Show with Craig Ferguson, The	(#2.104)	Herself
Married with Children	Fair Exchange (#4.6)	Yvette
Paradise	Childhood's End (#1.8)	Katie
Parker Lewis Can't Lose	Pilot (#1.1)	Robin Fecknowitz
Tout le monde en parle	(2002-03-23)	Herself
V Graham Norton	(#1.47)	Herself

(16 rows)
brukernavn=>

Referanser

[1] The Internet Movie Database, <http://www.imdb.com/>