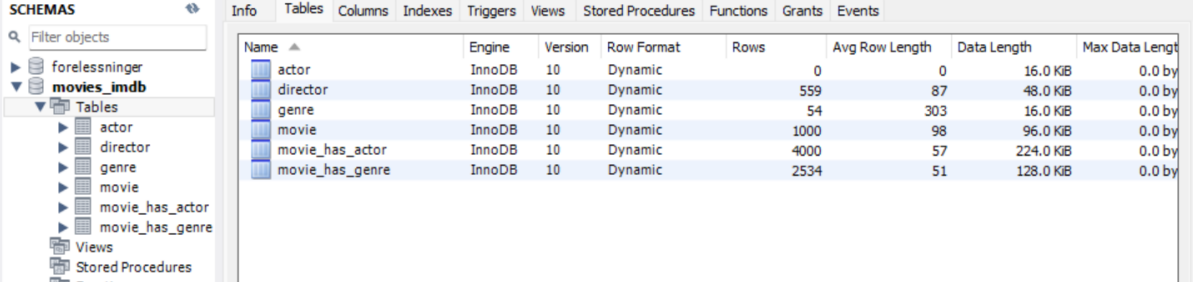


Skjema som er opprett og hvilke tabeller som ligger i skjemaet

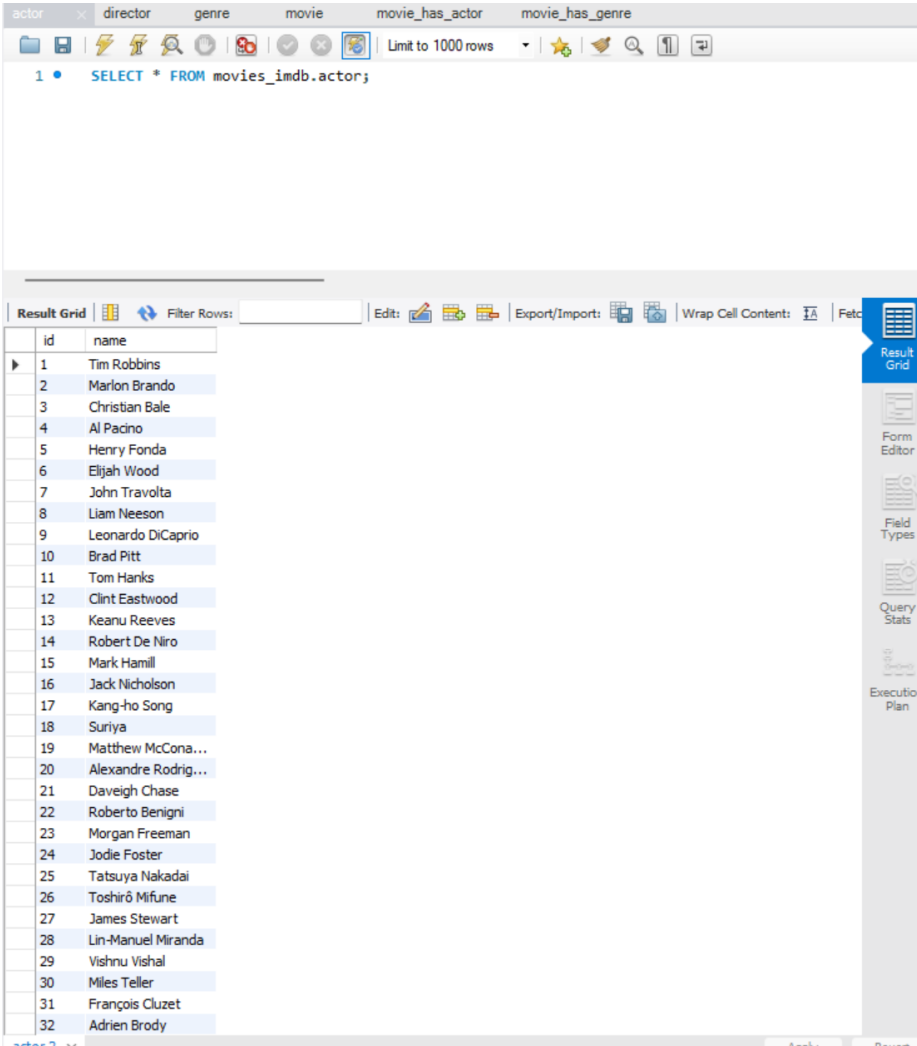


The screenshot shows the MySQL Workbench Schemas tab. On the left, a tree view shows the 'forelessninger' database containing a schema named 'movies_imdb'. Under 'movies_imdb', there are tables: 'actor', 'director', 'genre', 'movie', 'movie_has_actor', and 'movie_has_genre'. The main pane shows the 'Info' tab for the 'movies_imdb' database, displaying a table with columns: Name, Engine, Version, Row Format, Rows, Avg Row Length, Data Length, and Max Data Length.

Name	Engine	Version	Row Format	Rows	Avg Row Length	Data Length	Max Data Length
actor	InnoDB	10	Dynamic	0	0	16.0 KiB	0.0 by
director	InnoDB	10	Dynamic	559	87	48.0 KiB	0.0 by
genre	InnoDB	10	Dynamic	54	303	16.0 KiB	0.0 by
movie	InnoDB	10	Dynamic	1000	98	96.0 KiB	0.0 by
movie_has_actor	InnoDB	10	Dynamic	4000	57	224.0 KiB	0.0 by
movie_has_genre	InnoDB	10	Dynamic	2534	51	128.0 KiB	0.0 by

Visning av innhold i de forskjellige tabellene

Actor



The screenshot shows the MySQL Workbench interface with the 'actor' table selected. The query editor shows the query: `SELECT * FROM movies_imdb.actor;`. The 'Result Grid' tab is active, displaying the data for the 'actor' table. The table has two columns: 'id' and 'name'. The data is listed in a grid with 32 rows. The 'actor' table is highlighted in the left sidebar.

id	name
1	Tim Robbins
2	Marlon Brando
3	Christian Bale
4	Al Pacino
5	Henry Fonda
6	Elijah Wood
7	John Travolta
8	Liam Neeson
9	Leonardo DiCaprio
10	Brad Pitt
11	Tom Hanks
12	Clint Eastwood
13	Keanu Reeves
14	Robert De Niro
15	Mark Hamill
16	Jack Nicholson
17	Kang-ho Song
18	Suriya
19	Matthew McConaughey
20	Alexandre Rodriguez
21	Daveigh Chase
22	Roberto Benigni
23	Morgan Freeman
24	Jodie Foster
25	Tatsuya Nakadai
26	Toshirô Mifune
27	James Stewart
28	Lin-Manuel Miranda
29	Vishnu Vishal
30	Miles Teller
31	François Cluzet
32	Adrien Brody

Director

actor		director	genre	movie	movie_has_actor	movie_has_genre
		Limit to 1000 rows				
1		SELECT * FROM movies_imdb.director;				
Result Grid		Filter Rows:				
		id	name			
1	▶	1	Frank Darabont			
2		2	Francis Ford Coppola			
3		3	Christopher Nolan			
4		4	Sidney Lumet			
5		5	Peter Jackson			
6		6	Quentin Tarantino			
7		7	Steven Spielberg			
8		8	David Fincher			
9		9	Robert Zemeckis			
10		10	Sergio Leone			
11		11	Lilly Wachowski			
12		12	Martin Scorsese			
13		13	Irvin Kershner			
14		14	Milos Forman			
15		15	Bong Joon Ho			
16		16	Sudha Kongara			
17		17	tia Lund			
18		18	Hayao Miyazaki			
19		19	Roberto Benigni			
20		20	Jonathan Demme			
21		21	George Lucas			
22		22	Masaki Kobayashi			
23		23	Akira Kurosawa			
24		24	Frank Capra			
25		25	Thomas Kail			
26		26	Ram Kumar			
27		27	Damien Chazelle			
28		28	ric Toledano			
29		29	Roman Polanski			
30		30	Ridley Scott			
31		31	Tony Kaye			
32		32	Bryan Singer			

director 3 × Apply Revert

Genre

actor director genre movie movie_has_actor movie_has_genre

Limit to 1000 rows

```
1 • SELECT * FROM movies_imdb.genre;
```

Result Grid Filter Rows:

	id	name
▶	1	Drama
	2	Crime
	3	Action
	4	Biography
	5	Drama
	6	Western
	7	Comedy
	8	Adventure
	9	Animation
	10	Horror
	11	Mystery
	12	Comedy
	13	Film-Noir
	14	Fantasy
	15	Horror
	16	Family
	17	Thriller
	18	Drama
	19	Crime
	20	Adventure
	21	Drama
	22	Romance
	23	Sci-Fi
	24	War
	25	Family
	26	Music
	27	Comedy
	28	Mystery
	29	Romance
	30	Biography
	31	Action
	32	Western

genre 2 x Apply Revert

Result Grid
Form Editor
Field Types
Query Stats
Execution Plan

Movie

actor

director

genre

movie

movie_has_actor

movie_has_genre

Limit to 1000 rows

1

•

SELECT * FROM movies_imdb.movies;

Result Grid

Filter Rows:

Edit:

Export/Import:

Wrap Cell Content:

Fetch

</

Movie has actor

actordirectorgenremovie**movie_has_actor**movie_has_genre

Limit to 1000 rows

1 • SELECT * FROM movies_imdb.movie_has_actor;

Result Grid

Filter Rows:

Edit: Export/Import: Wrap Cell Content: Fetch

	movie_id	actor_id
▶	1	1
	510	1
	824	1
	884	1
	2	2
	76	2
	312	2
	453	2
	3	3
	37	3
	65	3
	157	3
	221	3
	606	3
	622	3
	781	3
	785	3
	839	3
	957	3
	2	4
	4	4
	109	4
	166	4
	404	4
	422	4
	528	4
	613	4
	656	4
	816	4
	830	4
	856	4
	977	4

movie_has_actor 2 x

Apply

Revert

Result Grid

Form Editor

Field Types

Query Stats

Execution Plan

Movie has genre

actor director genre movie movie_has_actor movie_has_genre x

Limit to 1000 rows

```
1 • SELECT * FROM movies_imdb.movie_has_genre;
```

Result Grid Filter Rows: Edit: Export/Import: Wrap Cell Content: Fetch

	genre_id	movie_id
▶	1	1
	1	2
	1	3
	1	4
	1	5
	1	6
	1	7
	1	8
	1	10
	1	11
	1	12
	1	14
	1	16
	1	18
	1	19
	1	20
	1	21
	1	22
	1	24
	1	25
	1	26
	1	27
	1	28
	1	30
	1	31
	1	32
	1	33
	1	35
	1	36
	1	37
	1	38
	1	39

e_has_genre 1 x Apply Revert

Result Grid
Form Editor
Field Types
Query Stats
Execution Plan

Skriftlig forklaring av datamodellen

Oppstart

For å opprette databasen `movies_imdb` startet jeg med å opprette et skjema kalt `movies_imdb`. Neste steg var så å kjøre spørringer innefor dette skjemat for å opprette mine 6 tabeller som datamodellen skal inneholde. Tabellene var opprettet uten innfylt informasjon. Jeg opprette disse tabellene i følgende rekkefølge: `director`, `movie`, `actor`, `movie_has_actor`, `genre` og tilsatt `movie_has_genre`.

Tabeller

- “**director**” er satt opp med to kolonner, *id* og *name*, primærnøkkelen her er *id* og det er ingen fremmednøkkel satt opp for denne tabellen.

- “**movie**” er satt opp med 9 kolonner, *id*, *tittel*, *year*, *runtime*, *imdb_rating*, *metascore*, *votes*, *gross* og *director_id* og er den største tabellen vi har. Her er *id* igjen primærnøkkelen og *director_id* er brukt som fremmednøkkel koblet opp mot ``movies_imdb`.`director` (`id`)`.

- “**actor**” er satt opp med 2 kolonner, *id* og *name*, hvor *id* er primærnøkkelen og her er det heller ingen fremmednøkkel satt opp.

- “**movie_has_actor**” er satt opp med 2 kolonner, *movie_id* og *actor_id*, hvor begge er primærnøkler og fremmednøkler. *actor_id* er koblet opp mot ``movies_imdb`.`actor` (`id`)` og *movie_id* er koblet opp mot ``movies_imdb`.`movie` (`id`)`.

- “**genre**” er satt opp med 2 kolonner, *id* og *name*, hvor *id* er primærnøkkelen og her igjen, er det ingen fremmednøkkel satt opp.

- “**movie_has_genre**” er satt opp med to kolonner, *id* og *name*, hvor begge er primærnøkler og fremmednøkler. *genre_id* er koblet opp mot ``movies_imdb`.`genre` (`id`)` og *movie_id* er koblet opp mot ``movies_imdb`.`movie` (`id`)`.

Innfilling

Neste steg nå er å fylle inn alle tabellene. Jeg har valgt å fylle tabellene i følgende rekkefølge: `director`, `movie`, `genre`, `actor`, `movie_has_genre` og tilslutt `movie_has_actor`. Jeg valgte denne rekkefølgen da ga mest mening for meg, ut i fra hvordan primærnøkklene er satt opp i de forskjellige tabellene, ved å ta tabellen uten egen primærnøkkel først etter fulgt av tabellen(ene) med oppsatt kobling.