DB8

Felix Albrigtsen, Kristoffer Longva Eriksen, Jonas Haugland, Kristoffer Juelsen

1.1 Case

PVV er en studentorganisasjon som holder til i bergbygget på Gløshaugen. De kaller seg selv "nerder" og naturligvis har de mange bøker, filmer, spill, etc. som ligger på mange forskjellige plasser i lokalene sine. Per dags dato har de ikke noe system som holder oversikt over alt det forskjellige som ligger i hyllene deres, og dette er et stort behov for å lett kunne navigere seg rundt i labyrinten av fantastiske litterære verk og morsomme brettspill. Databasesystemet skal inneholde en oversikt over alle bøker og eventuelt brettspill, samt hvor de befinner, hvem som eier de forskjellige tingene og om de kan leies ut. I tillegg til utlån vil det også gjøre det lettere å plassere ting tilbake på riktig plass, og systemet kan lett oppdateres dersom en bok eller spill flytter på seg til en ny hylle. Med dette systemet er det flere enn bare en eier, PVV eier ikke alle bøkene, som tilsvarer at flere personer må kontaktes i det tilfelle at en person vil leie en bok som ikke er eid av PVV. Dette vil føre til noen problemer som ikke kan løses av et internt system siden individer må kontaktes for leie.

1.2 Egen relasjonsdatabase med eksempler i mysql

Relasjonsmodell

rom(id, navn, veibeskrivelse)

eier(id, navn, telefon, email)

kategori(id, navn, beskrivelse)

hylle(id, navn, rom_id*)

bok(id, tittel, forfatter, sitater, kategori_id*, hylle_id*, eier_id*)

Create Setninger

```
CREATE TABLE rom (
    id INT(20) AUTO_INCREMENT NOT NULL,
    navn VARCHAR(255) NOT NULL,
    veibeskrivelse VARCHAR(255) NOT NULL,
    PRIMARY KEY (id)
);

CREATE TABLE eier (
    id INT(20) AUTO_INCREMENT NOT NULL,
    navn VARCHAR(255) NOT NULL,
    telefon VARCHAR(16),
    email VARCHAR(255) NOT NULL,
    PRIMARY KEY (id)
);
```

```
CREATE TABLE kategori (
    id INT(20) AUTO_INCREMENT NOT NULL,
    navn VARCHAR(255) NOT NULL,
    beskrivelse VARCHAR(255) NOT NULL,
    PRIMARY KEY (id)
);

CREATE TABLE hylle (
    id INT(20) AUTO_INCREMENT NOT NULL,
    navn VARCHAR(255) NOT NULL,
    rom_id INT(20) NOT NULL,
    PRIMARY KEY (id),
    CONSTRAINT fk_hylle_rom FOREIGN KEY (rom_id) REFERENCES rom(id)
);

CREATE TABLE bok (
    id INT(20) AUTO_INCREMENT NOT NULL,
    tittel VARCHAR(255) NOT NULL,
    forfatter JSON,
    sitater JSON,
    kategori_id INT(20) NOT NULL,
    hylle_id INT(20) NOT NULL,
    eier_id INT(20) NOT NULL,
    PRIMARY KEY (id),
    CONSTRAINT fk_bok_kategori FOREIGN KEY (kategori_id) REFERENCES kategori(id),
    CONSTRAINT fk_bok_hylle FOREIGN KEY (hylle_id) REFERENCES hylle(id),
    CONSTRAINT fk_bok_eier FOREIGN KEY (eier_id) REFERENCES eier(id)
);
```

Setter inn data

```
-- Rom

INSERT INTO rom (navn, veibeskrivelse) VALUES ('gangen', 'rett til høyre');
INSERT INTO rom (navn, veibeskrivelse) VALUES ('koserommet', 'rett fram');
INSERT INTO rom (navn, veibeskrivelse) VALUES ('terminalrommet', 'inn i gangen,
til høyre, første venstre');
INSERT INTO rom (navn, veibeskrivelse) VALUES ('arbeidsrommet', 'inn i gangen, til
høyre, andre venstre');

-- Eiere

INSERT INTO eier (navn, telefon, email) VALUES ('Jonas Brothers', '+4798765432',
'jonas.brothers@feal.no');
INSERT INTO eier (navn, telefon, email) VALUES ('Jonas Jaeger', '+4767891234',
'jonas@jaeger.no');
INSERT INTO eier (navn, telefon, email) VALUES ('Programvareverkstedet',
'+4791298739', 'pvv@ntnu.no');
INSERT INTO eier (navn, telefon, email) VALUES ('RMS', '+4798127398',
'rms@pvv.ntnu.no');
```

```
-- Kategorier

INSERT INTO kategori (navn, beskrivelse) VALUES ('religion', 'kristendom,
hinduisme, vim etc.');
INSERT INTO kategori (navn, beskrivelse) VALUES ('fiction', 'oppfunnet
litteratur');
INSERT INTO kategori (navn, beskrivelse) VALUES ('fagstoff', 'fagstoff til fag og
andre ting');

-- Hyller

INSERT INTO hylle (navn, rom_id) VALUES ("spillhylla", 3);
INSERT INTO hylle (navn, rom_id) VALUES ("linuxhylla", 4);
INSERT INTO hylle (navn, rom_id) VALUES ("x86hylla", 4);
INSERT INTO hylle (navn, rom_id) VALUES ("dingshylla", 1);
INSERT INTO hylle (navn, rom_id) VALUES ("proghylla", 1);
INSERT INTO hylle (navn, rom_id) VALUES ("ved-siden-av-spillhylla", 3);
INSERT INTO hylle (navn, rom_id) VALUES ("katthylla", 2);

-- Bøker

INSERT INTO bok (tittel, forfatter, sitater, kategori_id, hylle_id, eier_id)
VALUES ("Clean Code", JSON_OBJECT('navn', 'Robert Martin', 'fodt', 1952),
JSON_ARRAY(), 3, 1, 1);

INSERT INTO bok (tittel, forfatter, sitater, kategori_id, hylle_id, eier_id)
VALUES ("1984", JSON_OBJECT('navn', 'George Orwell', 'fodt', 1903, 'dod', 1950),
JSON_OBJECT('sitat1', 'Hvis du lever i mørket, kan du kun se lysere tider'), 2, 4,
1);

INSERT INTO bok (tittel, forfatter, sitater, kategori_id, hylle_id, eier_id)
VALUES ("Linux Device Drivers", JSON_OBJECT('navn1', 'Allesandro Rubio', 'navn2',
'Jonathan Corbet'), JSON_ARRAY(), 3, 6, 3);

INSERT INTO bok (tittel, forfatter, sitater, kategori_id, hylle_id, eier_id)
VALUES ("Database Management Systems", JSON_OBJECT('navn', 'Ramakrishnan &
Gehrke'), JSON_OBJECT('sitat1', 'The SQL language has several aspects to it'), 3,
6, 4);

INSERT INTO bok (tittel, forfatter, sitater, kategori_id, hylle_id, eier_id)
VALUES ("Fundamental of database systems", JSON_OBJECT('navn', 'Elmasri og
Navathe'), JSON_OBJECT('sitat1', 'The unique clause specifies alternate keys',
'sitat2', 'A spacial index is used to organize objects into a set of buckets'), 3,
6, 2);
```

SELECT-spørringer

```
-- Hvilken hylle er 1984 i:
SELECT hylle.navn FROM hylle
INNER JOIN bok ON hylle.id=bok.hylle_id
WHERE bok.tittel = "1984";
```

```sql
-- Antall bøker i terminalrommmet:
SELECT COUNT(*) FROM bok
INNER JOIN hylle ON hylle.id = bok.hylle_id
INNER JOIN rom ON hylle.rom_id = rom.id
WHERE rom.navn = "terminalrommet";

-- Alle eiere med minst 2 bøker
SELECT eier.navn, COUNT(eier.navn) antall FROM bok
LEFT JOIN eier ON bok.eier_id = eier.id
GROUP BY bok.eier_id
HAVING antall > 1;

-- Antall hyller per rom
SELECT rom.navn, COUNT(*) AS antall_hyller FROM hylle
INNER JOIN rom ON rom.id = hylle.rom_id
GROUP BY hylle.rom_id;
```

1.3 Løsning med XML evt. JSON i MySQL

a) Sitater på listeform

I biblioteksystemet ønsker vi å ha sitater fra hver bok. Vi vil kun vise sitatene på informasjonssiden (i det hypotetiske brukergrensesnittet) for hver enkelt bok. Dataen skal aldri brukes utenfor konteksten av boken, og trenger derfor ikke å lagres utenfor bok-tabellen. Siden det finnes mellom én og mange sitater for hver bok er det vanskelig å lage kolonner/attributter i relasjonsdatabasen.

Et alternativ kunne vært å lage feltene sitat1, sitat2, sitat3 og sitat4 som attributter på boktypen i databasen. Dersom du kun har 2 sitater vil verdiene av sitat3 og sitat4 være NULL. Dette vil fungere, men du er begrenset til maks 4 sitater, og databasen vil ikke stoppe deg fra å lagre "feil".

Ved å bruke en egen tabell, for eksempel bok_sitat som bruker en fremmednøkkel til hver bok, vil vi ha en database på normalform, men det vil gi mer kompliserte select- og insert-setninger, enn om alt innholdet finnes i den ene boktabellen.

b) Forfatterobjekter, varierende format

Dette gjør vi for å unngå å lage ekstra tabeller for forfattere, da denne tabellen kan kreve veldig mange forskjellige attributter. Det er stor variasjon i hvor mye og hvilke data vi har om forfatteren, og for eksempel brettspill har kanskje ingen forfatter.

For en gitt bok kan altså forfatter-feltet inneholde for eksempel

```json
{
    "name": "George Orwell",
    "birthname": "Eric Arthur Blair",
    "year_born": 1903,
    "year_dead": 1950,
}
```

eller, like gyldig:

```
{
    "navn": "Don Rosa",
    "imageUrl":
"https://upload.wikimedia.org/wikipedia/commons/thumb/c/c4/Don_Rosa_in_Helsinki_20
14_C_IMG_2764.JPG/1024px-Don_Rosa_in_Helsinki_2014_C_IMG_2764.JPG"
}
```

Her vil JSON være mer hensiktsmessig enn simple datatyper, da dataformatet ikke er statisk eller uniformt. Kanskje vi ønsker å hente forfatterinfo fra forskjellige API-er og kilder på internett, uten å bruke tid på å normalisere og formatere dataen.

c) Eksempel på spørring

```
SELECT tittel, sitater FROM bok;
```

1.4 Løsning med NoSQL

Som nevnt i besvarelsen til oppgave 1.3 finnes det flere grunner til å ikke bruke en ren relasjonsdatabase til databasesystemet vårt.

I en dokumentdatabase vil JSON-datatypen være likt som alle andre tabellene, der det ikke er noen forhåndsdefinert format og attributt-sett. På programvareverkstedet har vi mange forskjellige verker, for eksempel tegneserier, bøker, manga, DVD-er, PlayStation™-spill, brettspill og lignende typer, som alle bør indekseres i biblioteksystemet. Om vi tar med dette i vurderingen vil en dokumentdatabase være til hjelp, da hver tuppel i databasen kan trenge forskjellig metadata.

I et større databasesystem vil vi også få bedre ytelse og redundans av et distribuert databasesystem(DBMS), for eksempel MongoDB, i motsetning til en sentral MySQL-tjener.

Likevel, i det forenklede tilfellet vi har implementert over, med kun bøker, kategorier og lokasjoner, er en relasjonsdatabase god for å sikre forutsigbarhet og brukervennlighet. Det blir enkelt å lage å lage web-grensesnitt, og all informasjonen kan presenteres på en forhåndsdefinert måte.

Demo-output:

MariaDB [idatt2103_ov8]> -- Hvilken hylle er 1984 i:

MariaDB [idatt2103_ov8]> `SELECT hylle.navn FROM hylle INNER JOIN bok ON hylle.id=bok.hylle_id WHERE bok.tittel = "1984";`

navn
dingshylla

1 row in set (0.001 sec)

MariaDB [idatt2103_ov8]> -- Antall bøker i terminalrommmet:

MariaDB [idatt2103_ov8]> `SELECT COUNT(*) FROM bok INNER JOIN hylle ON hylle.id = bok.hylle_id INNER JOIN rom ON hylle.rom_id = rom.id WHERE rom.navn = "terminalrommet";`

COUNT(*)
3

1 row in set (0.011 sec)

MariaDB [idatt2103_ov8]> -- Alle eiere med minst 2 bøker

MariaDB [idatt2103_ov8]> `SELECT eier.navn, COUNT(eier.navn) antall FROM bok LEFT JOIN eier ON bok.eier_id = eier.id GROUP BY bok.eier_id HAVING antall > 1;`

navn	antall
Jonas Brothers	2

1 row in set (0.001 sec)

MariaDB [idatt2103_ov8]> -- Antall hyller per rom

MariaDB [idatt2103_ov8]> `SELECT rom.navn, COUNT(*) AS antall_hyller FROM hylle INNER JOIN rom ON rom.id = hylle.rom_id GROUP BY hylle.rom_id;`

navn	antall_hyller
gangen	2
koserommet	1
terminalrommet	2
arbeidsrommet	2

4 rows in set (0.001 sec)

MariaDB [idatt2103_ov8]> `SELECT tittel, sitater FROM bok;`

tittel	sitater
Clean Code	[]
1984	["Hvis du lever i mørket, kan du kun se lysere tider"]
Linux Device Drivers	[]
Fundamental of database systems	["The unique clause specifies alternate keys", "A spacial index is used to organize objects into a set of buckets"]

4 rows in set (0.001 sec)

MariaDB [idatt2103_ov8]> `SELECT tittel, JSON_VALUE(forfatter, '$.navn') AS forfatternavn FROM bok HAVING forfatternavn IS NOT NULL;`

tittel	forfatternavn
Clean Code	Robert Martin
1984	George Orwell
Fundamental of database systems	Elmasri og Navathe

3 rows in set (0.001 sec)