DAT151DB – OBLIG1

Adrian Mortensen
DAT151

Høgskulen på Vestlandet

Innhold

Oppgave 1

All kildekode output kan finnes på github som sql filer.
Samt kjørte kommandoer og print av output.

1. Oppsett av tabeller

```
1. DROP TABLE IF EXISTS testTable;
2. DROP TABLE IF EXISTS person;
3. DROP TABLE IF EXISTS bedrift;
4.
5. CREATE TABLE bedrift(
6.     bedriftsId varchar(4),
7.     navn varchar(8),
8.     CONSTRAINT bedriftPK PRIMARY KEY (bedriftsId)
9. );
10.
11.   CREATE TABLE person(
12.       enId varchar(4),
13.       navn varchar(8),
14.       alder int,
15.       bedriftsId varchar(4),
16.       CONSTRAINT bedriftFK FOREIGN KEY (bedriftsId) REFERENCES
   bedrift(bedriftsId),
17.       CONSTRAINT personPK PRIMARY KEY (enId)
18.   );
19.
20.   CREATE TABLE testTable(
21.       testId varchar(4),
22.       navn varchar(8),
23.       CONSTRAINT testPK PRIMARY KEY (testId)
24.   );
```

2. Legge til data i tabellene.

```
1. INSERT INTO bedrift VALUES('abcd','Bedrift1');
2. INSERT INTO bedrift VALUES('acdc','Bedrift2');
3. INSERT INTO person VALUES('admo','Adrian',24,'abcd');
4. INSERT INTO person VALUES('nono','Noen',26,'abcd');
5. INSERT INTO person VALUES('pers','Person',30,'acdc');
6. INSERT INTO person VALUES('andr','Andre',60,'abcd');
7. INSERT INTO testTable VALUES('admo','Adrian');
```

3. Prøve kommandoer i MariaDB

a. Endring av «engine»

Endrer først engine på «person» tabellen til «InnoDB» dette går fint og blir gjennomført. Prøver deretter og endre «person» tabellen til «MyISAM» her feiler operasjonen siden «person» inneholder foreign keys noe «MyISAM» ikke vil ta stilling til. Bruker derfor «testTable» og kjører endringen til «MyISAM» på den uten problemer.

b. Vis index

```
> SHOW INDEX FROM person;
Return: +--------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+
        | Table  | Non_unique | Key_name | Seq_in_index | Column_name | Collation | Cardinality | Sub_part | Packed | Null | Index_type |
        +--------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+
        | person |          0 | PRIMARY  |            1 | enId        | A         |           4 |     NULL | NULL   |      | BTREE      |
        | person |          1 | bedriftFK |           1 | bedriftsId  | A         |           4 |     NULL | NULL   | YES  | BTREE      |
        +--------+------------+----------+--------------+-------------+-----------+-------------+----------+--------+------+------------+
```

Output fra kjøring kan finnes på github og lagt ved. under "commands.sql"

Ting av interesse i denne outputen er spesielt rundt fremmed nøkkelen «BedriftFK» Den trenger ikke være unik og kan være «NULL».

c. Analyser tabellen

```
> ANALYZE TABLE person;
Return: +-----------------+---------+----------+----------+
        | Table           | Op      | Msg_type | Msg_text |
        +-----------------+---------+----------+----------+
        | privBase.person | analyze | status   | OK       |
        +-----------------+---------+----------+----------+
```

Output fra kjøring kan finnes på github og lagt ved. under "commands.sql"

Denne kommandoen analyserer tabellen for å finne nøkkeldistribusjon for tabellen (MariaDB, u.d.).

d. Sjekk av tabell

```
> CHECK TABLE person;
Return: +-----------------+-------+----------+----------+
        | Table           | Op    | Msg_type | Msg_text |
        +-----------------+-------+----------+----------+
        | privBase.person | check | status   | OK       |
        +-----------------+-------+----------+----------+
```

Output fra kjøring kan finnes på github og lagt ved. under "commands.sql"

Denne kommandoen sjekker for feil i tabellen (MariaDB, u.d.).
Som vist ser alt greit ut i denne tabellen.

e. Reparer tabellen

Denne kommandoen reparerer en muligens korrupt tabell. Men fungerer bare for noen av *«storage engine'ene»* (MariaDB, u.d.).

Prøvde først å kjøre på en *InnoDB*-tabell.

```
> REPAIR TABLE person; --InnoDB table
+-----------------+--------+----------+-------------------------------------------------------------+
| Table           | Op     | Msg_type | Msg_text                                                    |
+-----------------+--------+----------+-------------------------------------------------------------+
| privBase.person | repair | note     | The storage engine for the table doesnt support repair      |
+-----------------+--------+----------+-------------------------------------------------------------+
```

Output fra kjøring kan finnes på github og lagt ved. under "commands.sql"

Som vist vil ikke dette kjøre på denne *«enginen»* Prøvde dermed på *testTable* som er en MyISAM tabell fra tidligere kommandoer.

```
> REPAIR TABLE testTable;
Return: +---------------------+--------+----------+----------+
        | Table               | Op     | Msg_type | Msg_text |
        +---------------------+--------+----------+----------+
        | privBase.testTable | repair | status   | OK       |
        +---------------------+--------+----------+----------+
```

Output fra kjøring kan finnes på github og lagt ved. under "commands.sql"

På en MyISAM-tabell fungerte alt fint.

f. Optimaliser tabellen

OPTIMIZE TABLE kommandoen har to hoved funksjoner. Enten brukes den til å defragmentere tabellen, eller for å oppdatere InnoDB fulltext indexen (MariaDB, u.d.).

Defragmentering burde brukes om man har slettet store deler av en tabell eller gjort store endringer på en tabell med variabel lengde på rekkene (MariaDB, u.d.).

Oppdatering av InnoDB fulltext indexen må spesifiseres at skal gjøres ettersom denne oppdateringen ikke skjer med engang ved endring; Siden denne operasjonen kan ta lang tid og er ressurskrevende (MariaDB, u.d.).

g. Sjekksum av tabell

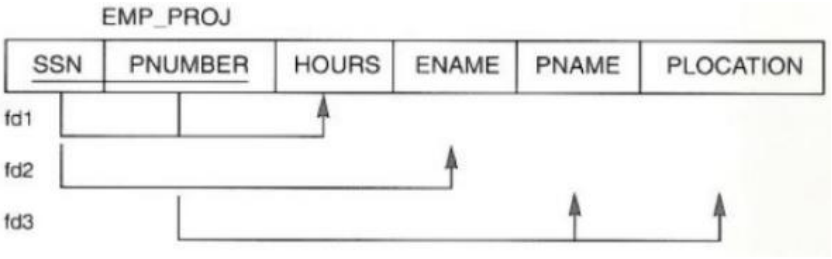
Denne kommandoen rapporterer på sjekksummen på en tabell, slik at man for eksempel kan sjekke om to tabeller er like eller om det er gjort en endring på tabellen (MariaDB, u.d.). Se output i «*commands.sql filen*»

h. Innochecksum program

Innochecksum programmet vill man kjøre for å sjekke om tabell filen er korrupt eller om det er noe feil på den. Programmet sjekker om den nåværende checksummen mot en allerede lagret checksum.

Oppgave 2

1. Forståelse av oppgaven



Min forståelse fra dette diagrammet er at *HOURS* er avhengig av *SSN* og *PNUMBER*, ENAME er Avhengig av bare *SSN* og *PNAME, PLOCATION* er avhengig av *PNUMBER*.

EMP_PROJ

SSN	PNUMBER	HOURS	ENAME	PNAME	PLOCATION
123456789	1	32.5	Smith,John B.	ProductX	Bellaire
123456789	2	7.5	Smith,John B.	ProductY	Sugarland
666884444	3	40.0	Narayan,Ramesh K.	ProductZ	Houston
453453453	1	20.0	English,Joyce A.	ProductX	Bellaire
453453453	2	20.0	English,Joyce A.	ProductY	Sugarland
333445555	2	10.0	Wong,Franklin T.	ProductY	Sugarland
333445555	3	10.0	Wong,Franklin T.	ProductZ	Houston
333445555	10	10.0	Wong,Franklin T.	Computerization	Stafford
333445555	20	10.0	Wong,Franklin T.	Reorganization	Houston
999887777	30	30.0	Zelaya,Alicia J.	Newbenefits	Stafford
999887777	10	10.0	Zelaya,Alicia J.	Computerization	Stafford
987987987	10	35.0	Jabbar,Ahmad V.	Computerization	Stafford
987987987	30	5.0	Jabbar,Ahmad V.	Newbenefits	Stafford
987654321	30	20.0	Wallace,Jennifer S.	Newbenefits	Stafford
987654321	20	15.0	Wallace,Jennifer S.	Reorganization	Houston
888665555	20	null	Borg,James E.	Reorganization	Houston

I oppgaven er det beskrevet at ENAME kan ansees som en atomisk verdi. Dermed ser vi at dataen er atomisk.

2. Besvarelse

Siden alle kolonnene er atomiske, er vi i første grad normalisering 1NF. Men siden *ENAME*, *PNAME* og *PLOCATION* bare er avhengig av 1 av halvdelene i primærnøkkelen er det ikke 2NF.

Lager derfor 2 tabeller ekstra og flytter disse kolonnene ut av EMP_PROJ tabellen.

Ser nå slik ut:

SSN	PNUMBER	HOURS

SSN	ENAME

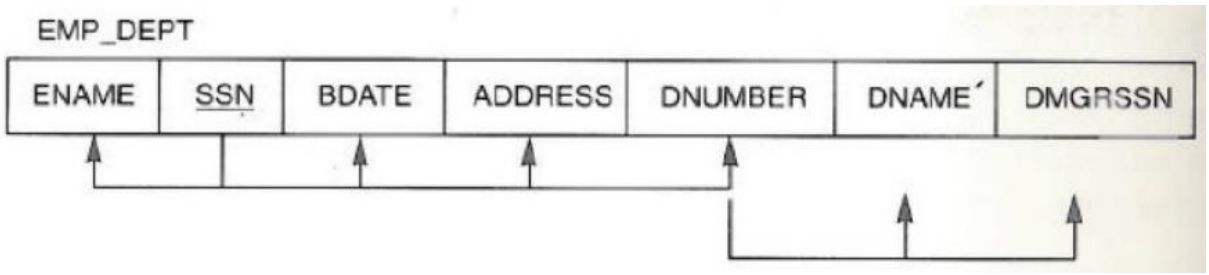
PNUMBER	PNAME	PLOCATION

Dette oppfyller både 2NF og 3NF.

Siden alle attributtene som ikke er nøkler er direkte avhengig av hele primærnøkkelen og ingen av attributtene har en transitiv knytning til nøklene.

Oppgave 3

1. Forståelse av oppgaven



Jeg forstår denne oppgaven som at alt er avhengig av SSN men DNAME og DMGRSSN er avhengig av DNUMBER som da er avhengig av SSN

EMP_DEPT

ENAME	SSN	BDATE	ADDRESS	DNUMBER	DNAME	DMGRSSN
Smith,John B.	123456789	09-JAN-55	731 Fondren,Houston,TX	5	Research	333445555
Wong,Franklin T.	333445555	08-DEC-45	638 Voss,Houston,TX	5	Research	333445555
Zelaya, Alicia J.	999887777	19-JUL-58	3321 Castle,Spring,TX	4	Administration	987654321
Wallace,Jennifer S.	987654321	19-JUN-31	291 Berry,Bellaire,TX	4	Administration	987654321
Narayan,Ramesh K.	666884444	15-SEP-52	975 FireOak,Humble,TX	5	Research	333445555
English,Joyce A.	453453453	31-JUL-62	5631 Rice,Houston,TX	5	Research	333445555
Jabbar,Ahmad V.	987987987	29-MAR-59	980 Dallas,Houston,TX	4	Administration	987654321
Borg,James E.	888665555	10-NOV-27	450 Stone,Houston,TX	1	Headquarters	888665555

All data er atomisk ettersom vi skulle se på ENAME og ADDRESS som atomiske felt.

2. Besvarelse

Ettersom all data er atomisk, er i hvert fall i 1NF. Siden all data er knyttet til primærnøkkelen er vi også i 2NF. Men siden DNAME OG DMGRSSN er knyttet transitivt til nøkkelen igjennom DNUMBER splitter vi disse ut av tabellen og lager en ny. Dermed vil modellen nå være:

ENAME	SSN	BDATE	ADDRESS	DNUMBER (FK)

DNUMBER	DNAME	DMGRSSN

Er nå i 3NF siden alle ikke nøkkel-attributter er nå ikke-transitivt bundet til primærnøkkelen i tabellen

Oppgave 4

a) Schema

Jeg gikk ut ifra at vi kunne lage flere tabeller enn de 3 navngitte. Dermed blir mine tabeller:

GRADES

GradeID	YEAR	CourseID (FK)	StudentID (FK)

FACULTY

fCode	fName	fPhoneNr	fAddress

COURSE

cCode	cName	cYear

DEPARTMENT

DepID	DepName	fCode(FK)

COURSE_SCHEDULE

SchedID	cYear	TeacherID(fk)	cCode(fk)

TEACHER

TeacherID	Name	fCode(FK)

PROGRAM

ProgID	ProgName

STUDENT

StudentID	BirthID	sName	CurAddress	sPhoneNr	homAddress	bDate	Gender	sYear	level	Faculty(fk)	Program(fk)	DepID(fk)

b) Implementering til MariaDB

Velger å skrive en .sql fil og source den til databasen kildekoden finnes på github og lagt ved.

C) Optimalisering av spørringer

SELECT s.SNAME, f.FNAME FROM STUDENT s, FACULTY f where s.FCODE=f.FCODE;

Denne vil da behøve en join for å finne fname fra student. Derfor kunne det vært ideelt å ha denormalisert student tabellen til å inneholde fname for å gjøre denne spørringen raskere.

SELECT COUNT(*) FROM STUDENT WHERE FCODE = 'FIN';

Denne spørringen vil være grei siden all informasjon finnes i student tabellen.

SELECT DISTINCT CYEAR FROM COURSE_SCHEDULE;

Denne spørringen vil også være rask siden alt som trengs er i tabellen.

Referanser

MariaDB, u.d. *MariaDB Knowledge Base.* [Internett]
Available at: https://mariadb.com/kb/en/
[Funnet 2020].