Gagnasafnsfræði

Páll Melsted

18. okt

Skorður

SQL getur skilgreint skorður á eiginleika í töflum. Algengustu skorðurnar sem við þurfum á að halda eru heilleikaskorður, þ.e. við viljum tryggja að tengingar á milli tafla vísi ekki út í loftið.

Við höfum séð skorðurnar `PRIMARY KEY` og `UNIQUE` fyrir eiginleika, sem tryggja að gildin séu ólík og ekki `NULL` í tilviki `PRIMARY KEY`

NOT NULL

Í sumum tilfellum viljum við ekki leyfa að geyma NULL gildi, t.d. fyrir price í tölvugagnagrunninum, þetta er gert með því að skilgreina dálka sem NOT NULL.

Við getum sett skilyrði á eftir skilgreiningu á dálki í gagnagrunni,

```
CREATE TABLE T (
  A TYPE [skilyrði],
  B INT NOT NULL
  id INT PRIMARY KEY,
  C INT UNIQUE
);
```

Foreign Key

Tengingar á milli tafla hafa verið óformlegar hingað til. Á milli Movies og StarsIn venslanna tengir (title,year) töflurnar en það er ekki tryggt að gildin passi.

```
CREATE TABLE Movie (
  title varchar(25),
  year int,
  length int,
  inColor int,
  studioName varchar(15),
  producerC varchar(3)
);
CREATE TABLE StarsIn (
  movieTitle varchar(30),
  movieYear int,
  starName varchar(30)
);
```

Primary key

Fyrst setjum við (title,year) sem primary key

```
CREATE TABLE Movie (
  title varchar(25),
  year int,
  length int,
  inColor int,
  studioName varchar(15),
  producerC varchar(3),
  PRIMARY KEY(title,year)
);
```

Til að tryggja að þetta verði alltaf lykill.

Foreign key

```
CREATE TABLE StarsIn (
  movieTitle varchar(30),
  movieYear int,
  starName varchar(30) REFERENCES MovieStar(name),
  FOREIGN KEY (movieTitle, movieYear)
    REFERENCES Movie(title,year)
);
```

Hér er starName tengt við MovieStar.name og hægt að skilgreina beint á eftir dálknum.

Samsetti lykillinn (movieTitle, movieYear) vísar svo í lykilinn (title,year) í Movie. Þar sem hann er samsettur úr tveimur dálkum þarf að skilgreina hann í sinni eigin línu með FOREIGN KEY.

Þurfum að setja `PRAGMA foreign_keys = ON;` í sqlite, eldri útgáfur eru ekki með ytri lyklum (foreign key)

Til hvers?

Með því að skilgreina aðallykil (Primary Key) tryggjum við að dálkarnir séu lyklar og að það sé eingöngu hægt að setja gögn inn sem uppfylla þessi skilyrði.

Ytri lykill á milli tafla tryggir að við fáum aldrei tilvísun út í loftið. Það er ekki hægt að setja inn röð í StarsIn sem vísar ekki í einhverja mynd sem er nú þegar til.

. . .

Þessi skilyrði segja gagnagrunninum hver **fastayrðing gagna** er og gagnagrunnurinn sér um að viðhalda henni.

Movie gagnagrunnurinn

Lögum núna gagnagrunninn. Við getum fengið afrit af grunninum með `.dump` skipuninni

```
sqlite3 f8.db .dump > f8.sql
```

Skilar gagnagrunninum, bæði töfluskema og gögnum, sem SQL skipunum.

Hvenær?

Skorður eru skilgreindar

- Með skemanu sem hluti af CREATE TABLE T(…)
  - Yfirleitt athugað í lok fjöldainnsetningar
- Seinna með ALTER TABLE (ekki hægt í sqlite)
  - Athugað á þeim gögnum sem eru nú þegar]

Í hver skipti sem við breytum gögnum eru skorður athugaðar.

- INSERT, ný gildi verða að uppfylla skorður
- UPDATE, sama
- DELETE, ytri lyklar verða ennþá að vísa á lykla í töflu

Getum seinkað því að athuga á skorðum þar til við ljúkum við margar aðgerðir í einu (Transaction) með því að bæta `DEFERRABLE INITIALLY DEFERRED` við `REFERENCES`

Röð innsetninga

Ef við keyrum

```
INSERT INTO StarsIn(movieTitle, movieYear, starName)
  VALUES ('Glengarry Glen Ross', 1992, 'Alec Baldwin');
```

Í f8b.db þá kvartar gagnagrunnurinn.

. . .

Við þurfum fyrst að sjá til þess að myndin sé til, síðan að setja inn gildið.

```
INSERT INTO Movie(title,year, length, inColor, StudioName, producerC)
  VALUES ('Glengarry Glen Ross', 1992, 100,1,'GGR',333);

INSERT INTO StarsIn(movieTitle, movieYear, starName)
  VALUES ('Glengarry Glen Ross', 1992, 'Alec Baldwin');
```

Ytri lyklar og breytingar

Ef við höfum ytri lykil frá S.B sem vísar í R.A hvaða breytingar gætu brotið skilyrðið?

. . .

- INSERT inn í S
- UPDATE á S.B
- DELETE úr R
- UPDATE á R.A

Breytingar?

Hvernig eigum við að höndla breytingarnar

- INSERT inn í S
  - villa!
- UPDATE á S.B
  - villa!
- DELETE úr R eða UPDATE á R.A
  - villa?
  - setja gildin í S.B sem NULL?
  - eyða út gildunum í S.B?

Brotin tilvísun

Þegar gildi í dálki sem vísað er á í ytra lykli er breytt er þrennt í boði

1. Skila villu. Þetta er `RESTRICT` og sjálfgefið
2. Setja tilvísun sem `NULL`, `SET NULL`
3. Breyta/eyða tilvísun, `CASCADE`

Það er hægt að breyta hegðun fyrir `DELETE` og `UPDATE` óháð.

```
CREATE TABLE Movie (
 ...
 cert INT REFERENCES MovieExec(cert)
     ON DELETE SET NULL
     ON UPDATE CASCADE
)
```

- Ef við breytum númeri fyrir MovieExec.cert þá breytast raðirnar í Movie með.
- Ef við eyðum MovieExec röð þá fær cert gildið NULL

Jaðartilfelli

```
CREATE TABLE T (
    A int, B int, C int,
    PRIMARY KEY (A,B),
    FOREIGN KEY (B,C) REFERENCES T(A,B) ON DELETE CASCADE );
```

Hvað gerist ef við eyðum fyrstu röðinni?

```
A B C
-----
1 1 1
2 1 1
3 2 1
4 3 2
5 4 3
6 5 4
7 6 5
8 7 6
```

. . .

Öll taflan hverfur!

Almennar skorður

Við getum athugað gildi með því að nota `CHECK` á eftir skilgreiningu á dálki.

```sql
CREATE TABLE MovieExec (
  name varchar(30),
  address varchar(30),
  cert int PRIMARY KEY,
  netWorth int CHECK (netWorth > 1000000)
);
```

Sumir gagnagrunnar leyfa hlutfyrirspurnir í CHECK en Sqlite og PostgreSQL leyfa það ekki.

Almennar skorður

Við getum blandað dálkum í skorðum

```sql
CREATE TABLE MovieStar (
  name varchar(30) PRIMARY KEY,
  address varchar(30),
  gender varchar(1) CHECK (gender in ('F','M')),
  birthdate varchar(10),
  CONSTRAINT RightTitle CHECK (gender = 'F' OR name NOT LIKE 'Ms.%')
);
```

Fastayrðingar

SQL skilgreinir fastayrðingar (assertion) sem eru mjög öflugar en fáir gagnagrunnar styðja :(

```sql
CREATE ASSERTION NoColorMoviesWithPoorMovieExecs
CHECK (NOT EXISTS (
  SELECT * FROM Movie,MovieExec
  WHERE cert=producerC AND inColor=1 AND netWorth < 1000000
  ));
```

Gikkir (triggers)

Skorður takmarka hvað má fara inn í gagnagrunn. Gikkir geta lagað gögnin til svo að skorðunar passi.

```
CREATE TRIGGER NetWorthTrigger
AFTER UPDATE OF netWorth ON MovieExec
FOR EACH ROW
WHEN (OLD.netWorth > NEW.netWorth)
BEGIN
  UPDATE MovieExec
  SET netWorth = OLD.netWorth
  WHERE cert=NEW.cert;
END;
```

Í sqlite eru gikkir bara til í einfaldri útgáfu og vísa til nýju og gömlu raðanna með OLD og NEW. Í INSERT er NEW bara til og í DELETE er OLD bara til.