Gagnasafnsfræði

Páll Melsted

16. sept

Endurtekin gildi

Ef við viljum losna við endurtekin gildi er hægt að nota DISTINCT

```
SELECT DISTINCT name
FROM MovieExec, Movie, StarsIn
WHERE cert = producerC AND
    title = movieTitle AND
    year = movieYear AND
    starName = 'Harrison Ford';
```

DISTINCT er oftast notað með einum dálki en virkar með n-dum.

DISTINCT kostnaður

Til að hægt sé að reikna út niðurstöðuna fyrir DISTINCT verður að raða töflunni. Röðun tekur **O(nlog(n))** fyrir n raðir á meðan flestar SELECT fyrirspurnir taka **O(n)** tíma.

Í hlutfyrirspurnum er oft hægt að komast hjá því að framkvæma DISTINCT þar til í síðustu fyrirspurn.

Mengjaaðgerðir

Vensl í SQL eru pokar af n-dum en í venslaalgebru eru mengi.
Undantekningin frá þessari reglu eru mengjaaðgerðirnar UNION, INTERSECT og EXCEPT

Áður en aðgerðin er framkvæmd er niðurstöðunni breytt í mengi, rétt eins og DISTINCT hefði verið notað, og aðgerðin síðan framkvæmd með mengjareglum.

```
SELECT title FROM Movie
UNION ALL
SELECT movieTitle AS title from StarsIn;
```

Ef við viljum nota pokaaðgerðir og halda réttum fjölda er hægt að nota ALL útgáfur af aðgerðunum.

EXCEPT ALL og INTERSECT ALL eru ekki til í sqlite :(

Hópvirkjar

SQL hefur 5 virkja sem hægt er að beita á dálka í venslum.

1. COUNT
2. SUM
3. MIN
4. MAX
5. AVG

```
SELECT SUM(length),SUM(DISTINCT length),
       AVG(length),MIN(length),MAX(length)
FROM Movie;
```

SUM og AVG virka með tölum. MIN og MAX með strengjum og tölum. COUNT virkar með hverju sem er. Það er hægt að nota DISTINCT með dálkum til að forðast endurtekningar.

COUNT(*)

COUNT(*) telur fjölda rađa í niđurstöđunni óháđ innihaldi. NULL gildi telja líka með. COUNT(x) telur ekki með NULL gildi.

```
SELECT COUNT(length),COUNT(DISTINCT length),COUNT(*)
FROM Movie;
```

Hópun (Aggregation)

Hópvirkjarnir 5 geta verið notaðir á hluta af niðurstöðunni með því að hópa saman niðurstöður.

```
SELECT studioName, SUM(length)
FROM Movie
GROUP BY studioName;
```

Þá mynda allar raðir með sama gildi fyrir studioName hlutvensl sem hópvirkjanum SUM er beitt á. Niðurstaðan verður vensl með eina röð fyrir hvern hóp.

Hópun

```
SELECT ..
FROM R
GROUP BY A,B..;
```

Í fyrirspurninni mega einungis þeir dálkar sem koma fyrir í GROUP BY hlutanum vera í SELECT hluta. Aðrir dálkar mega bara koma fyrir í hópvirkjum

Hópun með WHERE

Þegar WHERE er notað með hópun er fyrst farið í gegnum select skipunina eins og venjulega og hópunin er framkvæmd síðast, þ.e. eftir að where skilyrðunum hefur verið beitt. Allir dálkar í töflunum mega koma fyrir í WHERE hluta

```
SELECT name, SUM(length)
FROM MovieExec, Movie
WHERE producerC=cert
GROUP BY name;
```

Hópun og NULL

- NULL er aldrei notað í hópvirkjum, t.d. SUM,MIN.. (nema COUNT(*))
- NULL gildið getur myndað hóp
- Þegar hópvirkja er beitt á tóman lista verður niðurstaðan NULL
- nema fyrir COUNT, þá er niðurstaðan 0.

Hópun og skilyrði

Í SELECT skipun er hægt að takmarka niðurstöður með where skilyrðum. En þar sem hópun á sér stað eftir where þá er ekki hægt að nota hópvirkja í WHERE.

```
SELECT title,length
FROM movie
WHERE length > AVG(length);
```

Notum hlutfyrirspurn

```
SELECT title,length
FROM movie
WHERE length > (SELECT AVG(length) FROM Movie);
```

Hópun og HAVING

Þegar beita þarf skilyrðum á hóp, en ekki einstök gildi, þá notum við HAVING

```
SELECT studioName, SUM(length)
FROM Movie
GROUP BY studioName
HAVING SUM(length) > 150;
```

Sömu reglur gilda um HAVING eins og um WHERE, en að auki

- HAVING er bara beitt á hópa
- bara GROUP BY dálkar mega koma fyrir
- og niðurstöður hópvirkja

WITH

Þegar við vinnum með flóknar fyrirspurnir er oft þægilegt að geta vísað í útreiknaðar milliniðurstöður, t.d. hlutfyrirspurnir. Þetta er gert t.d. með WITH lykilorðinu

```
WITH M as (SELECT model,price from laptop
           union select model,price from pc
           union select model,price from printer)
SELECT M.model
FROM M
WHERE M.price = (select max(price) from M);
```

Hér getum við vísað tvisvar í niðurstöðuna M án þess að endurtaka hlutfyrirspurnina. Þetta virkar bara í sqlite >3.8.3

Skorður á venslum (Kafli 2.5)

Skorður á venslum eru táknaðar í venslaalgebru á tvo vegu

1. Með tóma menginu, t.d. $R = \emptyset$
2. Með hlutmengi, t.d. $R \subseteq S$

Bæði eru jafngild en við munum nota tómamengisritháttinn.

Heilleikaskorður (referential integrity constraint)

```
SELECT starName
FROM starsIn
WHERE starName NOT IN (SELECT name FROM moviestar);
```

Einhverjir eru skráðir leikarar en koma ekki fyrir í leikaratöflunni!

Við segjum að þetta brjóti heilleikaskorður (referential integrity) þarsem allir leikarar í StarsIn eiga að koma fyrir í movieStar.

Heilleikaskorður

Á mengjamáli myndum við skrifa

$$\pi_{starName}(StarsIn) \subseteq \pi_{name}(MovieStar)$$

eða

$$\pi_{starName}(StarsIn) - \pi_{name}(MovieStar) = \emptyset$$

Lyklaskorður

Ef við skilgreinum (title,year) sem lykil fyrir Movie venslin þá verður (title,year) að vera ólíkt fyrir allar n-dir

$\sigma_{M1.title=M2.titleANDM1.year=M2.yearANDNOT(M1.length=M2.lengthAND...)}(M1 \times M2) = \emptyset$

Við getum skrifað þetta sem val þar sem ekki eru til tvær ólíkar n-dir með sama title og year.

þar sem M1 er stytting á

$$\rho_{M1(...)}(Movie)$$

Gildisskorður

Fyrir takmörkuð gildi er hægt að telja upp möguleika. Gender í MovieStar er geymt sem CHAR(1) en við tökum bara mark á F og M. Til að skrifa þetta sem skorður þá notum við aftur val

$$\sigma_{gender \neq 'F' AND gender \neq 'M'}(MovieStar) = \emptyset$$