Gagnasafnsfræði

Páll Melsted

2. sept

Venslalíkanið

gagnagrunnur == safn af töflum

vensl == tafla

eigindi == dálkur í töflunni

n-d == röð í töflunni (borið fram "ennd")

Fyrirspurnir í gagnagrunnum

Movie gagnagrunnurinn

```
title                       year        length
--------------------------  ----------  ----------
Pretty Woman                1990        119
The Man Who Wasn't There    2001        116
Logan's run                 1976
...
```

SQL fyrirspurn til að ná í gögnin

```sql
SELECT * FROM Movie;
```

Meira SQL

```sql
SELECT title, year
FROM Movie
WHERE length > 120;
```

```sql
SELECT *
FROM Movie
WHERE year > 15*length;
```

SELECT þekking

Hve margir hafa séð

```sql
SELECT title, year
FROM Movie
WHERE length > 120;
```

eða svipað áður?

En hvað með

$$\pi_{\text{title,year}}\left(\sigma_{\text{length}>120}(\text{Movie})\right)$$

Uppbygging SELECT

```
SELECT name1, name2, ...
FROM relation1, relation2, ...
WHERE <some condition>;
```

Við notum * sem styttingu á að fá allar n-dir

Tölum um meira í næstu viku hvernig við vinnum með mörg vensl í SELECT.

Málfræði og merking SQL

SQL er tiltölulega læsilegt, yfirleitt er hægt að skilja einfaldar SELECT fyrirspurnir.

Eftir því sem við lærum meira um SQL þurfum við að hafa í huga

- Málfræði SQL. Hvernig **má** skrifa SQL setningar
- Merkingarfræði SQL. Hvað **þýðir** SQL setningin

Málfræðin er einföld. Til að tala um merkingarfræðina þurfum við að hafa líkan.

Venslalíkanið

- SQL er byggt á venslalíkaninu og venslaalgebru.

- Venslalíkanið gefur okkur nákvæma merkingu á SQL setningum.

- Venslaalgebra leyfir okkur að tala um jafngildar aðgerðir og endurskrifa fyrirspurnir.

Önnur hlutverk SQL

SQL er meira en bara fyrirspurnir. Við notum SQL til að

- Skilgreina ný vensl
- Breyta gögnum
- Setja upp skorður og kveiki
- Halda utan um notendur og öryggi
- Stýra hreyfingum á gagnagrunni fyrir marga notendur

Dálkaval

Með vörpun/dálkavali (Projection) getum við valið hluta af eigindum úr venslum, endurnefnt og endurraðað

```
SELECT title AS name, length AS duration
FROM Movies;
```

Oft þægilegt að nota til að stytta sql fyrirspurnir

Dálkaval

Við getum líka reiknað út einfaldar segðir sem nýja dálka

```
SELECT vara AS nafn,
1.255*kostnadur + 1000 AS heimsendingarkostnadur ...
```

Val (á röðum)

Í **WHERE** getum við sett hvaða Boolean segð sem er, eins og í Java/C

1. = er notað fyrir samanburð ekki ==
2. <> er notað fyrir !=
3. strengir eru með einföldum gæsalöppum 'abc'
4. || er notað til að skeyta saman strengjum

SQL skipanir gera ekki greinarmun á há- og lágstöfum en strengjasamanburður á gildum gerir það.

Val

Útkoman úr Boolean segð í `WHERE` er yfirleitt `TRUE` eða `FALSE`. Þessum gildum er hægt að pússla saman með `AND`, `OR` og `NOT` eins og í Java/C. Venjulegar reglur gilda um svigasetningu t.d.

```
SELECT title
FROM Movie
WHERE (year > 1970 OR length < 90) AND studioName = 'MGM';
```

Strengjaleit

SQL er með sér virkja fyrir strengjaleit, `LIKE`

`s LIKE p` skilar `TRUE` ef s passar við mynstrið p.

Tveir stafir hafa sérstaka merkingu

- _ passar við hvaða einn staf sem er
- % passar við hvaða streng sem er, jafnvel tóman

```sql
SELECT title
FROM Movie
WHERE title LIKE 'Star ____'; -- fjögur _ í röð
```

Strengjaleit

Til að leita að ' eða % þarf að skrifa þá tvisvar

```
SELECT title
FROM Movie
WHERE title like '%''s%';
```

NULL

NULL er löglegt gildi í SQL getur haft mismunandi merkingu eftir samhengi.

- Notum **NULL** þegar gildi vantar
- ekkert gildi á við

NULL má nota í hvaða segð sem er

- **NULL** með hvaða gildissegð, streng eða tölu, gefur útkomuna
- **NULL** með samanburðarvirkja gefur boolean útkomuna **UNKNOWN**

NULL

NULL getur komið fyrir sem gildi í n-d en aldrei sem fasti.

Til að athuga hvort gildi sé NULL þarf að nota IS virkjann

```
SELECT title,length FROM Movie WHERE length = NULL;
```

skilar engu, en

```
SELECT title,length FROM Movie WHERE length IS NULL;
```

skilar

```
Logan's run|
```

UNKNOWN

UNKNOWN virkar eins og eitthvað sem við vitum ekki hvort er **TRUE** eða **FALSE**

Getum blandað saman við boolean gildi, t.d. er

```
UNKNOWN OR TRUE = TRUE
```

```
UNKNOWN AND FALSE = FALSE
```

því þá skiptir engu hvert gildið er í **UNKNOWN**.

Hins vegar er

```
UNKNOWN OR FALSE = UNKNOWN
```

Með **WHERE** birtast aðeins niðurstöður sem eru **TRUE**, ekki **FALSE** eða **UNKNOWN**

Röðun

Vensl í SQL eru aldrei röðuð, nema að við biðjum sérstaklega um það.

Ástæðan er sú að SQL lýsir mengi, ekki röðuðum gögnum. Að auki er dýrara að raða gögnum heldur en að birta þau óröðuð.

Ef við viljum fá gögnin röðuð getum við notað **`ORDER BY`** á eftir **`WHERE`**

```
SELECT title, length
FROM Movie
WHERE length IS NOT NULL
ORDER BY length;
```

Röðun

Getum raðað eftir mörgum eigindum

```
SELECT title, length
FROM Movie
ORDER BY length,title;
```

Til að fá öfuga röð setjum við **DESC** (descending) á eftir eigindum.

Mörg vensl

Hingað til höfum við aðeins séð SELECT með einum venslum.

```
SELECT *
FROM Movie, MovieExec;
```

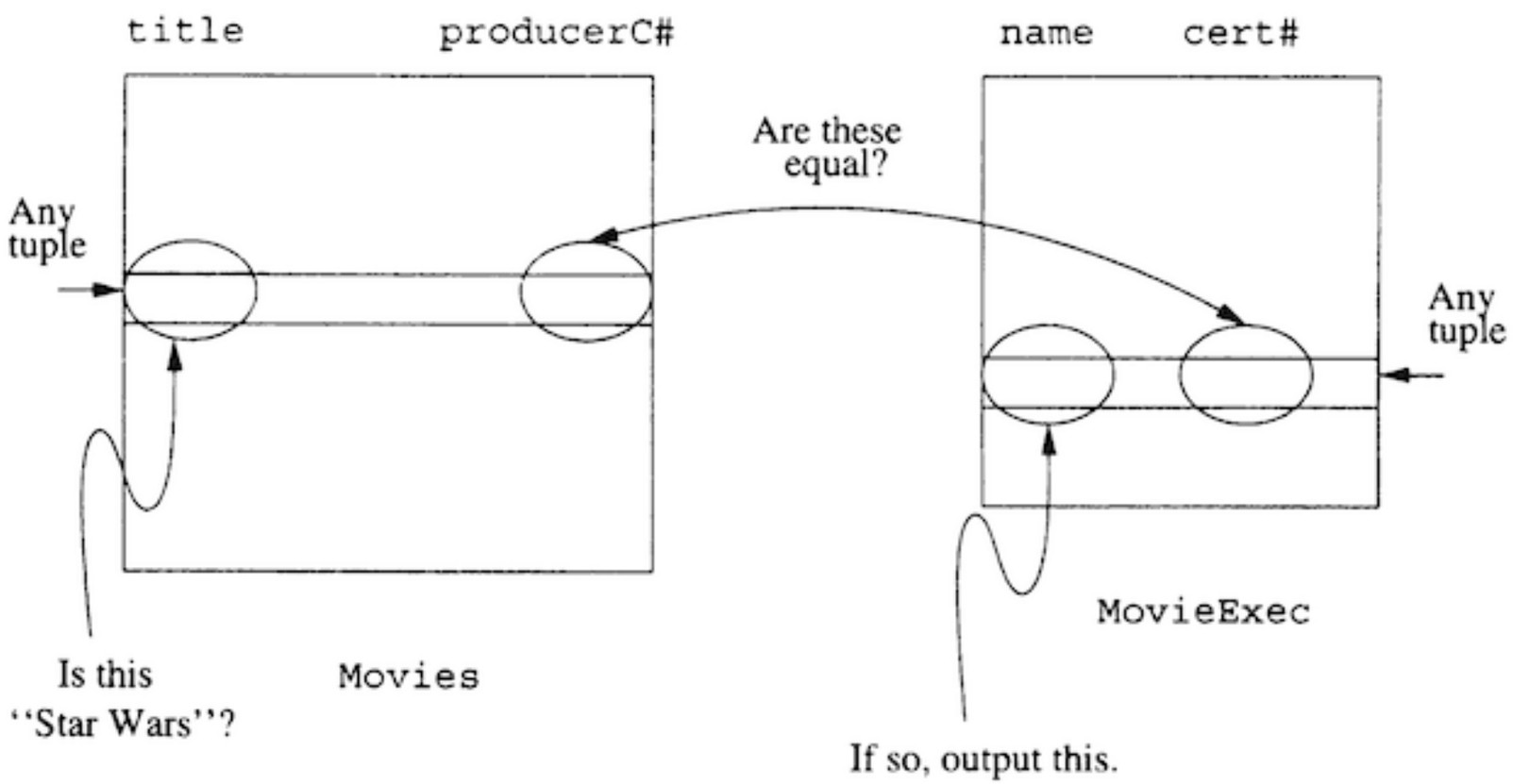
Þegar við notum mörg vensl eru allar samsetningar af n-dum úr öllum töflum búnar til.

Þetta er yfirleitt ekki það sem við viljum

Tengingar (join)

Yfirleitt viljum við aðeins taka n-dir sem passa saman á einhvern hátt.

```
SELECT name
FROM Movie, MovieExec
WHERE title = 'Star Wars' AND producerC = cert;
```



Tengingar (join)

Hér er tengingin fengin í **WHERE** hlutanum

```
SELECT name
FROM Movie, MovieExec
WHERE title = 'Star Wars' AND producerC = cert;
```

Sömu fyrirspurn er hægt að skrifa sem

```
SELECT name
FROM Movie JOIN MovieExec ON producerC = cert
WHERE title = 'Star Wars';
```

í stað **JOIN** er líka hægt að skrifa **INNER JOIN**

Árekstur nafna

Oft eru sömu nöfn notuð fyrir eigindi í venslum

```
MovieStar(name, address, gender, birthdate)
MovieExec(name, address, cert, netWorth)
```

Til að greina á milli þeirra getum við notað venslanafnið sjálft með punkti

```sql
SELECT MovieStar.name, MovieExec.name
FROM MovieStar, MovieExec
WHERE MovieStar.address = MovieExec.address;
```

Vensl endurtekin

Sömu vensl mega koma fyrir oftar en einu sinni, en þá verðum við að gefa þeim nafn með **AS**.

```
SELECT M1.title,M1.year,M2.year
FROM Movie AS M1, Movie AS M2
WHERE M1.year <> M2.year AND M1.title = M2.title;
```

Mengjaaðgerðir

SQL styður mengjaaðgerðirnar, ∪, ∩ og − með **UNION**, **INTERSECT** og **EXCEPT**

```
(SELECT name
FROM MovieStar)
INTERSECT
(SELECT name
FROM MovieExec);
```

Svigar virka ekki í sqlite (hægt að redda með subqueries)

Skilgreining á gagnagrunnum

Skilgreining á gagnagrunni er líka hluti af SQL.
Til að skilgreina nýja töflu notum við **CREATE TABLE**

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

Teljum upp eigindi og tög aðskilið með kommum.

Tög

SQL skilgreinir nokkur tög

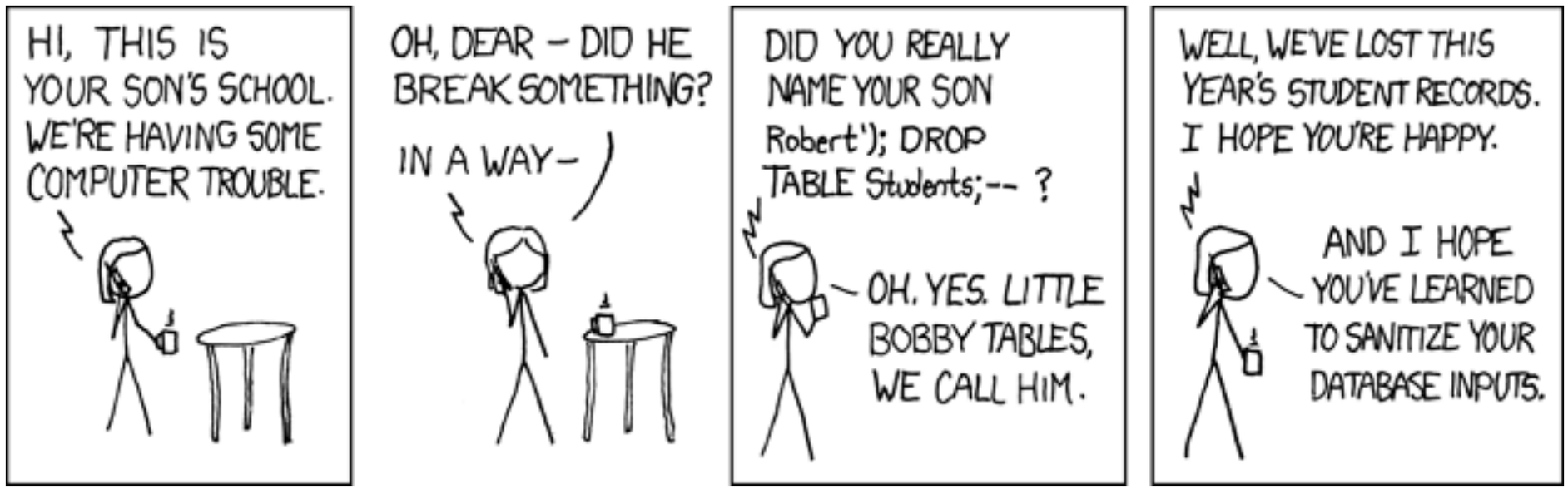
1. Strengir, **`CHAR(n)`** og **`VARCHAR(n)`**, merkingarlega séð er enginn munur en bæði geyma strengi af lengd í mesta lagi n.
2. **`BOOLEAN`** fyrir boolean gildi
3. **`INT`** fyrir heiltölur
4. **`FLOAT`** fyrir rauntölur
5. **`DATE`**, **`TIME`** og **`TIMESTAMP`** fyrir tíma, sjá http://bit.ly/1CtmpsG

sqlite er ekki strangt á tögunum, getum sett texta inn í int

```
CREATE TABLE tmp (x INT);
INSERT INTO tmp VALUES ('hallo!');
SELECT * FROM tmp;
```

Breytingar á gagnagrunni

Hendum burtu töflu og öllum gögnum með **DROP TABLE**



http://xkcd.com/327/

Breytinar á gagnagrunni

`ALTER TABLE` getur bætt við eða tekið út dálka í töflum

- `ADD` til að bæta við nýjum dálk
- `DROP` til að taka út dálk

```
ALTER TABLE MovieStar ADD phone CHAR(16);
ALTER TABLE MovieStar DROP address;
```

Sjálfgefin gildi

Getum skilgreint sjálfgefin gildi. Ef ekkert er sett inn fyrir n-d þá verður gildið **NULL**, getum breytt þessu með **DEFAULT** á eftir taginu.

```
CREATE TABLE Person (
  ...
  gender CHAR(1) DEFAULT '?',
  ...
);
```

Mjög hentugt þegar bæta þarf nýjum dálk

```
ALTER TABLE MovieStar ADD phone CHAR(16) DEFAULT 'unlisted';
```

Lyklar

Við getum skilgreint dálka sem lykla. Annað hvort einn dálk eða fleiri.

`PRIMARY KEY` og `UNIQUE` setja skorður á gildi að þau verða að vera ólík fyrir hverja n-d. Að auki má

`PRIMARY KEY` aldrei fá gildið `NULL`

```
CREATE TABLE MovieStar (
  name varchar(30) PRIMARY KEY,
  address varchar(30),
  gender varchar(1),
  birthdate varchar(10)
);
```

Lyklar

Ef við þurfum að skilgreina fleira en eitt eigindi sem lykil þá verður að gera það í sér línu.

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

Þá má title vera endurtekið, year endurtekið en saman verða þau að vera einstök og aldrei **NULL**.

Góðir lyklar

Hvað er gott að nota sem lykil? Við verðum að tryggja að við setjum aldrei gögn í gagnagrunninn með sama gildi í lykil.

- Notendavalið userid
- email?
- SSN (BNA)
- Kennitala (Ísland)
- Sjálfvirk heiltala

Næsta vika

Lesið kafla 2.4, vandlega (sleppum 2.4.13)

Lesið kafla 6.3, 6.5 (bíðum með 6.4 þar til í næstu viku)

Heimadæmi 2 á mánudag á gradescope

Gradiance verkefni

Heimadæmi 3 detta inn á þriðjudaginn.