TÖL303G

Gagnasafnsfræði

Snorri Agnarsson

Venslalíkanið

- Gagnagrunnur: Safn af töflum
- Vensl: Tafla
- Eigindi: Dálkur í töflu
- n-d: Röð í töflu (borið fram „ennd“)

Venslalíkanið

- SQL er byggt á venslalíkaninu (relational model) og venslaalgebru (relational algebra)
- Venslalíkanið gefur okkur nákvæma merkingu á SQL setningum
- Venslaalgebran gerir okkur kleift að tala um jafngildar aðgerðir og endurskrifa fyrirspurnir

Önnur hlutverk SQL

- SQL er ekki aðeins fyrirspurnamál
- Auk fyrirspurna notum við SQL til að
  - Skilgreina ný vensl
  - Breyta gögnum
  - Setja upp skorður (constaints) og gikki (triggers)
  - Halda utan um notendur og öryggi
  - Stýra hreyfingum á gagnagrunnum fyrir marga notendur

Röðun

- Töflur (vensl) í SQL eru aldrei röðuð nema við biðjum sérstaklega um það
- Ástæðan er sú að vensl eru mengi (eða pokar), ekki röðuð gögn
- Einnig er dýrara að raða gögnum en að birta þau óröðuð
- Ef við viljum fá raðaða útkomu þá getum við notar ORDER BY á eftir WHERE:

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

Röðun

- Getum raðað eftir mörgum eigindum (dálkum)

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

- Til að fá öfuga (lækkandi) röð setjum við DESC (þýðir *descending*) á eftir eigindi (nafni dálks)

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

Mörg vensl

- Hingað til höfum við mestmegnis séð SELECT með einum venslum (einni töflu)
- Getum gert SELECT á margar töflur (vensl) samtímis

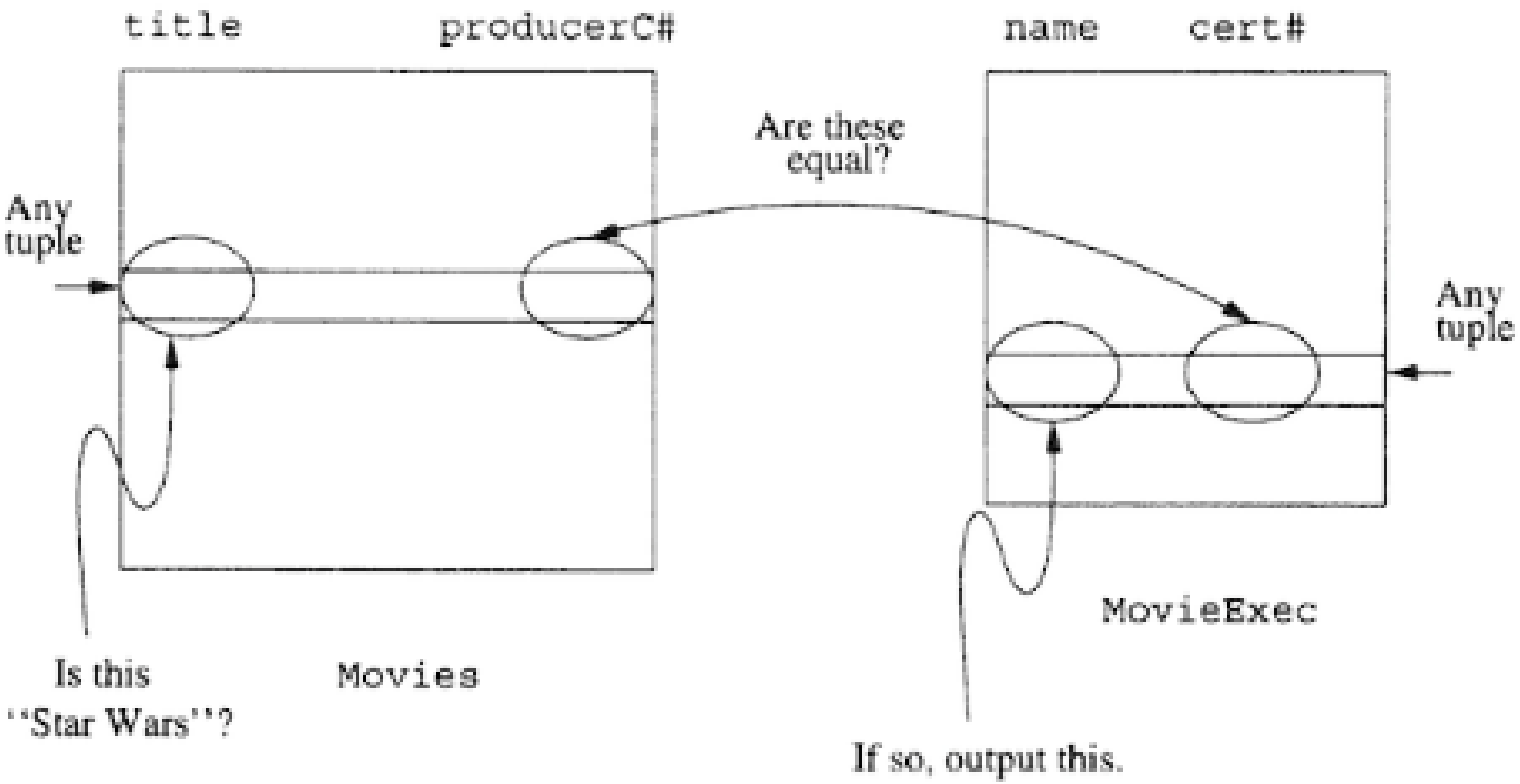
```
SELECT *
FROM Movie, MovieExec;
```

- Þegar við notum mörg vensl eru allar samsetningar af n-dum úr öllum töflunum búnar til, sem sagt **krossmargfeldið**
- Þetta er yfirleitt alls ekki það sem við viljum

Tengingar (join)

- Yfirleitt viljum við aðeins taka n-dir úr **krossmargfeldinu** sem passa saman á einhvern hátt

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


title
producerC#
name
cert#
Are these equal?
Any tuple
Any tuple
Is this "Star Wars"?
Movies
MovieExec
If so, output this.

Árekstur nafna

- Oft eru sömu (dálka)nöfn notuð í mörgum töflum fyrir eigindi

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

- Til að greina á milli þeirra getum við notað töflunafnið (venslanafnið) sjálft með punkti

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

Tengingar (join)

- Fyrirspurnin

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

Er jafngild fyrirspurninni

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

Tengingar (join)

- Fyrirspurn

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

- er einnig jafngild

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

- Einnig má skrifa INNER JOIN í stað JOIN
- Sjáum seinna fleiri afbrigði af JOIN

Vensl endurtekin

- Sömu vensl mega koma fyrir oftar en einu sinni, en þá þarf að gefa þeim ný nöfn 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 $\cup$, $\cap$ og – með UNION, INTERSECT og EXCEPT

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

- SQLite vill reyndar ekki leyfa þessa sviga, en styður samt INTERSECT á viðunandi hátt

SQLite afbrigði

- Í SQLite má skrifa

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

- eða (hér er subquery, þ.e. földuð fyrirspurn)

```
SELECT name FROM MovieStar
WHERE name IN
        (SELECT name FROM MovieExec);
```

- eða (hér eru subquery, þ.e. faldaðar fyrirspurnir)

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

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, aðskilin með kommum

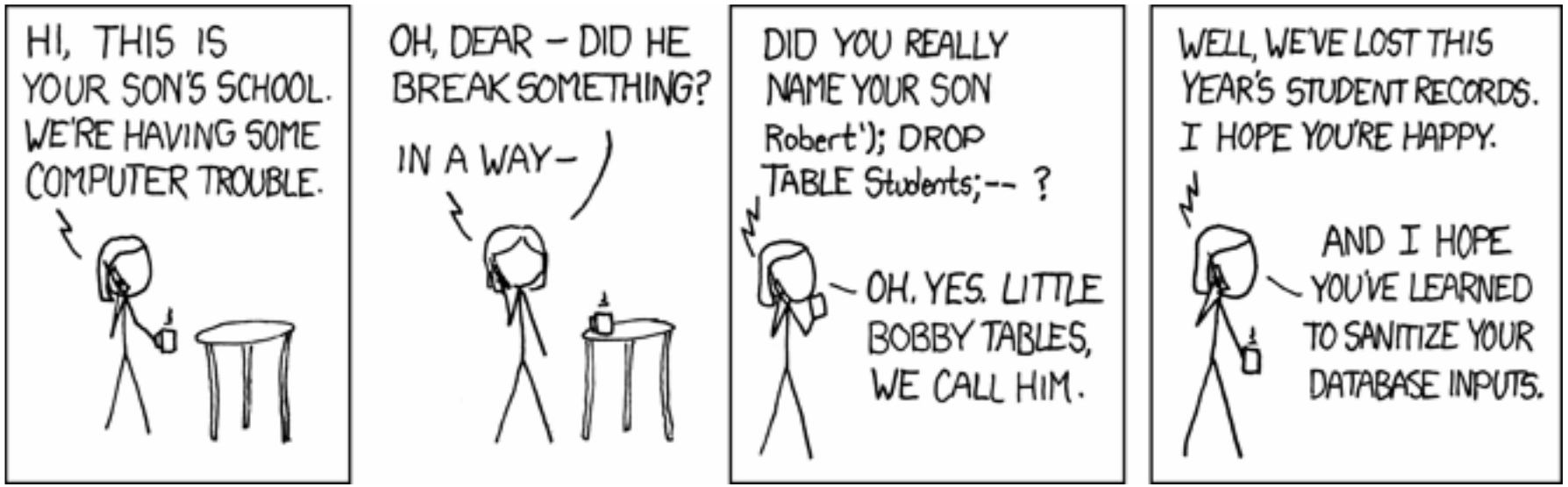
SQL tög

- Strengir: CHAR(n) og VARCHAR(n), svipuð merking en oftast geymdir á mismunandi hátt
- BOOLEAN fyrir Boolsk gildi
- INT fyrir heiltölur
- FLOAT fyrir fleytitölur
- DATE, TIME og TIMESTAMP fyrir tímasetningar
  - Sjá t.d. https://en.wikibooks.org/wiki/SQL_Dialects_Reference/Data_structure_definition/Data_types/Date_and_time_types
- SQLite tekur ekki mark á tögun, þurfum ekki að tiltaka tag í SQLite

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

Breytingar á gagnagrunni

- Hendum burt töflu og öllum gögnum hennar með DROP TABLE



- http://xkcd.com/327/
- Sérstök SQLite skipun (oft þægileg): DROP TABLE IF EXISTS t;

Breytingar á gagnagrunni

- ALTER TABLE getur bætt við dálkum (eigindum, *attribute*) eða eytt þeim
- ADD til að bæta við dálki:

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

- DROP til að eyða dálki:

```
ALTER TABLE MovieStar DROP address;
```

Sjálfgefin gildi

- Getum skilgreint sjálfgefin (default) gildi fyrir dálk
  - Ef ekkert sjálfgefið gildi er tilgreint þá er sjálfgefið gildi NULL

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

- Hentugt þegar bæta skal við dálki

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

Lyklar

- Við getum skilgreint dálka sem lykla
- Leyfilegt er að hafa fleiri en einn lykil
- Leyfilegt er að einn lykill spanni fleiri en einn dálk
- Hafa má í mesta lagi einn PRIMARY KEY
- Dálkar sem eru í PRIMARY KEY eða eru skilgreindir UNIQUE verða að vera mismunandi (sem n-d-ir) í mismunandi röðum töflunnar
- PRIMARY KEY má aldrei vera NULL

Lyklar

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

Lyklar

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

- Mismunandi kvikmyndir mega hafa sama title og mega hafa sama year, en mega ekki hafa bæði sama title og sama year
- title og year mega ekki vera NULL

Góðir lyklar

- Hvað er gott að nota sem lykil?
- Viljum tryggja að ekki séu mismunandi gögn undir sama lykli
- Dæmi um lykla sem skilgreina manneskjur
  - Notendavalið userid
  - tölvupóstfang
  - SSN (social security number, BNA)
  - Kennitala (Ísland)
  - Sjálfvirk heiltala

Venslaalgebra

- Venslaalgebra vinnur með vensl, sem eru mengi, ekki pokar
- SQL aðgerðum má lýsa í venslaalgebru ef töflurnar innihalda ekki endurteknar raðir
- Við munum oftast ekki hafa áhyggjur af endurteknum röðum
  - Við getum komið í veg fyrir að þær komi fyrir
- Vensl yfir mengi A, B, C eru hlutmengi af $A \times B \times C$
- SQL aðgerðunum UNION, INTERSECT og EXCEPT má lýsa með samsvarandi mengjaaðgerðum, en athugið að til þess að $R \cup S$ sé vitrænt þarf að tryggja að í R og S séu sömu eigindi (dálkar), í sömu röð og með sama nafni

Dálkaval (*projection*, ofanvarp)

- Ef R eru vensl má búa til ný vensl með því að velja suma dálka úr R, mögulega í annarri röð
- Virkinn π (gríska pí) virkar þannig að $\pi_{A_1,A_2,\ldots,A_n}(R)$ velur dálka $A_1, A_2, \ldots, A_n$ úr R
- Til dæmis gefur $\pi_{title,year}(Movie)$ niðurstöðuna á næstu blaðsíðu
- Sama og

```
SELECT title, year FROM Movie
```

title	year
Pretty Woman	1990
The Man Who Wasn't There	2001
Logan's run	1976
Star Wars	1977
Empire Strikes Back	1980
Star Trek	1979
Star Trek: Nemesis	2002
Terms of Endearment	1983
The Usual Suspects	1995
Gone With the Wind	1938

Dálkaval (*projection*, ofanvarp)

- Hins vegar gæfi $\pi_{length}(Movie)$

```
length
------
119
116
124
111
132
106
238
```

- Ástæðan er að algebran vinnur með mengi, ekki poka – endurtekin gildi detta út – einnig eru NULL gildi ekki endilega til í venslaalgebru

Val (*selection*)

- Valvirkinn σ (gríska sigma) tekur vensl R sem viðfang og skilar nýjum venslum $\sigma_C(R)$ þar sem C er eitthvert skilyrði
- Útkoman er með sömu dálkanöfn og R og hefur aðeins þær n-dir (raðir) sem uppfylla skilyrðið C
- Útkoman úr $\sigma_{length \geq 100}(Movie)$ er á næstu blaðsíðu
- Svipað og útkoman úr

```
SELECT * FROM Movie WHERE length>=100;
```

title	year	length	inColor	studioName	producerC
Pretty Woman	1990	119	1	Disney	999
The Man Who Wasn't There	2001	116	0	USA Entert	777
Star Wars	1977	124	1	Fox	555
Empire Strikes Back	1980	111	1	Fox	555
Star Trek	1979	132	1	Paramount	444
Star Trek: Nemesis	2002	116	1	Paramount	321
Terms of Endearment	1983	132	1	MGM	123
The Usual Suspects	1995	106	1	MGM	999
Gone With the Wind	1938	238	1	MGM	123

Tenging við SQL

- Fyrirspurnir í einstökum venslum á sniðinu

```
SELECT A1,A2,...,An
FROM R
WHERE C;
```

- má skrifa á algebrusniði sem

$$\pi_{A_1,A_2,\ldots,A_n}\left(\sigma_C(R)\right)$$

- En hvað með mörg vensl?

Mörg vensl

- Til að vinna með fleiri en eina töflu í einu, til dæmis R og S, skilgreinum við (kross)margfeldi vensla, $R \times S$
- $R \times S$ er mengi allra samsetninga af n-dum í R við n-dir í S
- Dálkarnir í $R \times S$ halda nöfnum sínum – ef R og S hafa dálka með sama nafni, A, þá heita nýju dálkarnir í $R \times S$ nöfnunum $R.A$ og $S.A$

Mörg vensl

- SQL fyrirspurnin

```
SELECT A1,A2,...,An
FROM R1,R2,...,Rk
WHERE C
```

- verður þá

$$\pi_{A_1,A_2,\ldots,A_n}\left(\sigma_C(R_1 \times R_2 \times \cdots \times R_k)\right)$$

Endurnefning vensla (taflna) og eiginda (dálka)

- Til að endurnefna töflu og dálka notum við ρ virkjann (gríska rho)

$$\rho_{S(A_1,A_2,\ldots,A_n)}(R)$$

- býr til ný vensl S með sömu gögn og venslin R, en dálkarnir heita $A_1, A_2, \ldots, A_n$ (reiknum með að R hafi n dálka)

- Ef við viljum aðeins endurnefna venslin skrifum við

$$\rho_S(R)$$

Join dæmi – tenging vensla

- SQL fyrirspurnina

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

- má skrifa sem

$$\rho_{R(pname)}\left(\pi_{name}\left(\sigma_{title='Star\ Wars'\ \wedge\ producerC=cert}(Movie \times MovieExec)\right)\right)$$

Join

- Í tengingunni á undan tengdum við á dálkum producerC og cert og báðir dálkar fóru í niðurstöðuvenslin
- Oft eða oftast hafa báðir dálkar sama nafn og við viljum tengja á jöfnum gildum, eins og hér:

R

A	B
1	2
3	9

S

B	C	D
2	5	6
4	7	8
9	10	11

$\sigma_{R.B=S.B}(R \times S)$

A	R.B	S.B	C	D
1	2	2	5	6
3	9	9	10	11

Theta join – þeta tenging

- Fyrra sniðið á tengingu sem við sáum á undan er almennara þar eð C getur verið hvaða skilyrði sem er
- Við höfum sérstakan virkja fyrir slíka tengingu, kallaður þeta-join
- Táknið er $\bowtie_\theta$ þar sem θ er skilyrðið sem áður var í virkjanum σ_θ

R

A	B
1	2
3	9

S

B	C	D
2	5	6
4	7	8
9	10	11

$R \bowtie_{R.B=S.B} S$

A	R.B	S.B	C	D
1	2	2	5	6
3	9	9	10	11

Natural join

- Eðlileg tenging (natural join), $R \bowtie S$, (takið eftir: ekkert θ, ekkert skilyrði) tengir saman R og S með því að velja á sameiginlegum dálkum og sleppa endurteknum gildum

R

A	B
1	2
3	9

S

B	C	D
2	5	6
4	7	8
9	10	11

$R \bowtie S$

A	B	C	D
1	2	5	6
3	9	10	11

- Takið eftir að miðröðin í S kemur ekki fram í útkomunni

Natural join

- Eðlileg tenging (natural join), $R \bowtie S$, (takið eftir: ekkert θ, ekkert skilyrði) tengir saman R og S með því að velja á sameiginlegum dálkum og sleppa endurteknum gildum

R

A	B
1	2
3	9

S

B	C	D
2	5	6
4	7	8
9	10	11

$R \bowtie S$

A	B	C	D
1	2	5	6
3	9	10	11

- Takið eftir að miðröðin í S kemur ekki fram í útkomunni

Theta join – Þeta tenging

- Natural join má þá skrifa sem

$$R \bowtie S = \pi_L\big(\sigma_C(R \times S)\big)$$

- þar sem C er skilyrðið sem velur dálkagildin jöfn og L eru dálkarnir án endurtekninga – e.t.v. má bæta við endurnefningu hér (ρ-aðgerð), ef við viljum vera smámunasöm:

$$R \bowtie S = \rho_{T(D)}\left(\pi_L\big(\sigma_C(R \times S)\big)\right)$$

- þar sem T er nafn venslanna sem koma út og D eru dálkanöfnin (eigindanöfnin) í þeim venslum

Nauðsynlegar aðgerðir

- Eftirfarandi 6 aðgerðir (virkjar) **duga** í venslaalgebru:

$$\cup, -, \sigma, \pi, \times, \rho$$

- Allar aðrar aðgerðir má skilgreina og skrifa út frá þessum, til dæmis: $\cap$, $\bowtie$, $\bowtie_\theta$ og (eins og við sjáum kannski seinna) $\ltimes$, $\rhd$, $\div$, $\overline{\bowtie}$, $\bowtie\!\!\!-$ og $\overline{\bowtie}$
- SQL gagnagrunnar geta notað venslaalgebru til að einfalda fyrirspurnir á grunni reglna sem gilda um venslaalgebru
- Til dæmis gildir eftirfarandi ef engir dálkar í S koma fyrir í C:

$$\sigma_C(R \times S) = \sigma_C(R) \times S$$

Hlutfyrirspurnir – faldaðar fyrirspurnir (nested queries)

- Í SQL skila allar fyrirspurnir nýjum venslum
  - Þessi vensl má svo nota í aðrar SQL fyrirspurnir og koma þá fyrir sem hlutfyrirspurnir, þ.e. faldaðar fyrirspurnir í stærra samhengi – fyrirspurnir inni í fyrirspurnum
- Þrjár helstu leiðir til að nota hlutfyrirspurnir
  1. Földuð fyrirspurn skilar einu gildi – þetta gildi má þá bera saman við önnur gildi í WHERE hluta
  2. Földuð fyrirspurn skilar venslum – hægt er að nota þessi vensl eins og önnur í WHERE hluta
  3. Földuð fyrirspurn getur komið fyrir í FROM hluta með nafni til að nota fyrir niðurstöðu

Hlutfyrirspurnir með stöku gildi

```
SELECT name
FROM MovieExec
WHERE cert =
        (SELECT producerC
        FROM Movie
        WHERE title = ‘Star Wars‘
        );
```

- Hér skilar faldaða fyrirspurnin gildinu 555 – niðurstaðan er því eins og úr fyrirspurninni

```
SELECT name FROM MovieExec WHERE cert=555;
```

Hlutfyrirspurnir með venslum (1/2)

- Niðurstaðan úr fyrirspurn er yfirleitt vensl
- Þegar venslin eru með einn dálk má nota eftirfarandi virkja
  1. EXISTS (er niðurstaða földuðu fyrirspurnarinnar ekki tóm):
     - Dæmi:

       SELECT a FROM t1 WHERE EXISTS (SELECT * FROM t2 WHERE ...)

       og:

       SELECT name FROM MovieExec
       WHERE EXISTS (SELECT * FROM Movie WHERE cert=ProducerC);
  2. s IN (er s í niðurstöðu földuðu fyrirspurnarinnar):
     - Dæmi:

       SELECT a FROM t1 WHERE t1.b IN (SELECT t2.b FROM t2 WHERE ...)

       og:

       SELECT name FROM MovieExec
       WHERE cert IN (SELECT producerC FROM Movie);

Hlutfyrirspurnir með venslum (2/2)

3. s > ALL (er s stærra en **öll** gildin í niðurstöðu földuðu fyrirspurnarinnar, einnig má nota aðra samanburði – virkar ekki í SQLite):
   - Dæmi:

     SELECT a FROM t1 WHERE t1.b > ALL (SELECT c FROM t2 WHERE ...)

     og:

     SELECT title FROM Movie
     WHERE length > ALL (SELECT length FROM Movie WHERE cert=555);
4. s > ANY (er s stærra en **eitthvert** gildi í niðurstöðu földuðu fyrirspurnarinnar , einnig má nota aðra samanburði – virkar ekki í SQLite):
   - Dæmi:

     SELECT a FROM t1 WHERE t1.b > ANY (SELECT c FROM t2 WHERE ...)

     og:

     SELECT name FROM Movie
     WHERE length > ANY (SELECT length FROM Movie WHERE cert=555);

Svipað í SQLite

```
SELECT title
FROM Movie
WHERE length >
        (SELECT MAX(length)
        FROM Movie
        WHERE producerC=555);
```

og

```
SELECT title
FROM Movie
WHERE length >
        (SELECT MIN(length)
        FROM Movie
        WHERE producerC=555);
```