TÖL303G

Gagnasafnsfræði

Snorri Agnarsson

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);
```

Hlutfyrirspurnir með n-dum

- Skrifa má:

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

- Sem sagt: Framleiðendur sem hafa framleitt Harrison Ford kvikmynd

Einfaldari lausn með venjulegri tengingu (join)

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

- Fáum reyndar sama nafn tvisvar því Movie taflan inniheldur tvær Harrison Ford kvikmyndir sem George Lucas framleiddi

Háðar hlutfyrirspurnir

- Í fyrri dæmum var útkoman úr innri (faldaðri) fyrirspurn fasti (föst tafla)
- En útkoman getur verið háð ytri fyrirspurn
- Innri fyrirspurnin getur vísað í dálk í ytri fyrirspurn

```
SELECT title
FROM Movie AS Old
WHERE year < ANY
        (SELECT year
         FROM Movie
         WHERE title = Old.title
        );
```

- Hvað er umdæmið fyrir Old?

Eilítið endurritað (töflunöfn)

```
SELECT Old.title
FROM Movie AS Old
WHERE Old.year < ANY
        (SELECT Movie.year
         FROM Movie
         WHERE Movie.title = Old.title);
```

eða (fyrir SQLite):

```
SELECT Old.title
FROM Movie AS Old
WHERE Old.year <
        (SELECT MAX(Movie.year)
         FROM Movie
         WHERE Movie.title = Old.title);
```

Hlutfyrirspurnir í FROM

- Hlutfyrirspurnir geta verið í FROM hluta – þurfa þá oft nafn til að geta tengt

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

Eilítið endurritað (töflunöfn)

```
SELECT MovieExec.name
FROM MovieExec,
        (SELECT Movie.producerC
         FROM Movie, StarsIn
         WHERE Movie.title=StarsIn.movieTitle AND
                 Movie.year=StarsIn.movieYear AND
                 StarsIn.starName='Harrison Ford') AS Prod
WHERE MovieExec.cert=Prod.producerC;
```

- Hér er auðveldara að sjá hvernig tengingarnar virka

Krossmargfeldi (join) í SQL

```
SELECT *
FROM Movie CROSS JOIN StarsIn;
```

jafngilt:

```
SELECT *
FROM Movie, StarsIn;
```

θ-join í SQL

```
SELECT title,starName
FROM Movie JOIN StarsIn ON
        title=movieTitle AND year=movieYear;
```

jafngilt

$$\pi_{title,starName}\left(Movie \bowtie_{title=movieTitle \wedge year=movieYear} StarsIn\right)$$

Natural Join í SQL

```
SELECT *
FROM MovieStar NATURAL JOIN MovieExec;
```

Með NATURAL JOIN notum við ekki ON lykilorðið (það er fyrir θ-join)

Hvaða gildi þarf að setja í MovieStar og MovieExec svo útkoman verði þessi?

Mel Gibson|Peekskill|M|1956-01-03|777|50000000

Breytingar á gögnum í gagnagrunni

1. Nýjar raðir settar í töflu
   INSERT skipun

2. Hluta af röðum eytt úr töflu
   DELETE skipun

3. Gildum dálka breytt í hluta töflu
   UPDATE skipun

Innsetning

INSERT INTO MovieStar VALUES(‘Mel Gibson‘,‘Peekskill‘,‘M‘,‘1956-01-03‘);

INSERT INTO MovieExec VALUES(‘Mel Gibson‘,‘Peekskill‘,777,50000000);

INSERT INTO MovieStar(gender,address,name,birthdate)
VALUES(‘M‘, ‘Peekskill‘,‘Mel Gibson‘,‘1956-01-03‘);

INSERT INTO MovieStar(gender,address,name)
VALUES(‘M‘, ‘Peekskill‘,‘Mel Gibson‘);

- Ef venslin hafa fleiri dálka en þeir sem við teljum upp þá fá aukadálkar sjálfgefin gildi
- Ef við teljum ekki upp dálka verða gildin að vera í sömu röð og í skilgreiningu töflunnar

Flóknari innsetning

- Við getum líka sett niðurstöðu fyrirspurnar inn í töflu

```
INSERT INTO Studio(name)
        SELECT DISTINCT studioName
        FROM Movie
        WHERE studioName NOT IN
                (SELECT name FROM Studio);
```

Eyðing raða

DELETE FROM Movie WHERE title=‘Gone With the Wind‘;

- Almennt:

DELETE FROM tafla WHERE skilyrði;

- Skilyrðið er eins og í SELECT
- Yfirleitt tiltekur skilyrðið gildi á lykli til að eyða einni röð

Uppfærsla

UPDATE Movie
SET length=123
WHERE title= ‘Star Wars‘ AND year=1977;

- Í SQLite höfum við raðarnúmer, **rowid**

UPDATE Movie SET length=0 WHERE length IS NULL;
UPDATE Movie SET length=0 WHERE rowid=3;

- Samanber niðurstöðu úr fyrirspurninni

SELECT rowid,* FROM Movie WHERE length IS NULL;

Endurtekin gildi

- Til að losna við endurtekin gildi úr útkomu notum við DISTINCT

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

- DISTINCT virkar bæði með einum dálki og stærri n-dum

DISTINCT kostnaður

- Notkun DISTINCT veldur því stundum að útkoman er röðuð
  - Annar útfærslumöguleiki er að nota tætitöflur (hash tables)
- Kostnaður við röðun er $\boldsymbol{O(n\ log\ n)}$
- Einfaldar SELECT skipanir kosta $\boldsymbol{O(n)}$
- Þegar hlutfyrirspurnir eru notaðar má oft komast af með eitt DISTINCT í einni hlutfyrirspurn (það borgar sig að fækka röðunum og minnka þær)

```
SELECT name
FROM MovieExec,
        (SELECT DISTINCT producerC FROM Movie) AS Prod
WHERE MovieExec.cert=Prod.producerC;
```

Mengjaaðgerðir

- Pokaaðgerðir (með réttum fjölda endurtekninga):

```
SELECT *
FROM
        (SELECT title FROM Movie) UNION ALL
        SELECT movieTitle AS title FROM StarsIn;
```

- Eða mengjaaðgerðir (án endurtekninga):

```
SELECT *
FROM
        (SELECT title FROM Movie) UNION
        SELECT movieTitle AS title FROM StarsIn;
```

- EXCEPT ALL og INTERSECT ALL eru ekki til í SQLite, en UNION ALL er þar

Hópaðgerðir

- SQL hefur aðgerðirnar COUNT, SUM, AVG, MIN, MAX, sem hægt er að beita á heila dálka
- SUM og AVG virka fyrir tölur, MIN og MAX með strengjum og tölum, COUNT með hverju sem er
- Nota má DISTINCT til að sleppa endurtekningum

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

COUNT(*)

- COUNT(*) telur allar raðir í niðurstöðu fyrirspurnar óháð innihaldi.
  - NULL gildi **eru** talin með
- COUNT(x) telur **ekki** NULL gildi

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

Hópun (Aggregation)

- Hópaðgerðirnar 5 má nota á sundurlæga hluta af niðurstöðu fyrirspurnarinnar með því að hópa saman niðurstöður

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

- Hér 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,...;
```

- Aðeins þeir dálkar sem taldir eru upp í GROUP BY hlutanum mega koma fyrir utan hópunaraðgerðar í SELECT hlutanum (hvers vegna?)
- Aðra dálkar má þó nota í hópaðgerðum í SELECT hluta
- SQLite fylgist ekki með því hvort þessum reglum er fylgt – prófið til dæmis

```
SELECT studioName,length
FROM Movie
GROUP BY studioName
```

Hópun með WHERE

- Þegar WHERE er notað með hópun er fyrst valið á grundvelli WHERE klausunnar og síðan er gerð hópun
- Alla dálka í töflunum má nota í WHERE klausunni
- Hópunin gerist, sem sagt, eftir tengingar (join)

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

Hópun og NULL

- NULL er aldrei með í hópvirkjum, nema fyrir COUNT(*)
- NULL gildið getur myndað hóp – myndar þá einn 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
- Þar eð hópun á sér stað eftir WHERE þá er ekki hægt að nota hópaðgerðir (SUM,...) í WHERE
  - Ekki hægt:

    SELECT title, length FROM Movie WHERE length>AVG(length);
  - Aftur á móti er þetta hægt:

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

Hópun og HAVING

- Nota má HAVING til að beita skilyrðum á **heilan hóp**

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

- Sömu reglur gilda um HAVING og fyrir WHERE, en auk þess:
  - HAVING er einungis beitt á hópa
  - Einungis dálkar í GROUP BY mega koma fyrir, plús niðurstöður hópaðgerða

WITH klausa

- Þegar unnið er með flóknar fyrirspurnir er oft þægilegt að geta skilgreint og notað milliniðurstöður
- Notum WITH til að skilgreina milliniðurstöður sem við getum síðan notað í tengdri fyrirspurn, dæmi:

```
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);
```

- Virkar í nýrri útgáfum SQLite

Skorður (constraints) á venslum (Kafli 2.5)

- Skorður á venslum eru táknaðar í venslaalgebru á tvo vegu

1. Með tóma menginu, þ.e. skilyrði $R = \emptyset$ fyrir einhver vensl R
2. Með hlutmengi, þ.e. skilyrði $R \subseteq S$ fyrir einhver vensl R og S

- Skorður sem tákna má með tóma menginu má einnig tákna með hlutmengi og öfugt – við reiknum með að aðgerðir venslaalgebru séu til staðar:
  - $R \subseteq S$ gildir þá og því aðeins að $R - S = \emptyset$
  - $R = \emptyset$ gildir þá og því aðeins að $R \subseteq \emptyset$
- Við notum báðar aðferðir eftir hentugleika

Heilleikaskorður (referential integrity constraint)

```
SELECT starName
FROM StarsIn
WHERE starName NOT IN
        (SELECT name FROM MovieStar);
```

- Eru einhverjar stjörnur í einhverri kvikmynd sem ekki eru skráðir í kvikmyndastjörnutöflunni?
- Ef svo er þá brýtur það eðlilegar heilleikaskorður á gagnagrunninn sem krefst þess að allar stjörnur séu skráðar í stjörnutöflunni

Heilleikaskorður

- Á mengjamáli getum við skrifað

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

- eða

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

Lyklaskorður

- Ef við skilgreinum name sem lykil fyrir MovieStar venslin þá erum við að krefjast þess að name sé ólíkt fyrir allar n-dir

$$\sigma_C(M_1 \times M_2) = \emptyset$$

- þar sem M_1 stendur fyrir $\rho_{M_1(name,address,gender,birthdate)}(\text{MovieStar})$ og M_2 stendur fyrir $\rho_{M_2(name,address,gender,birthdate)}(\text{MovieStar})$ og C stendur fyrir

$$M_1.name = M_2.name \wedge$$
$$(M_1.address \neq M_2.address \vee$$
$$M_1.gender \neq M_2.gender \vee$$
$$M_1.birthdate \neq M_2.birthdate)$$

- Sem sagt: Engar raðir úr MovieStar hafa sama nafn en aðrar upplýsingar öðruvísi
- Með öðrum orðum: Allar raðir í MovieStar sem hafa sama nafn eru eins (sama röðin)
- Svipaðar skorður eiga við fyrir alla aðra **lykla** í öllum öðrum töflum

Gildisskorður

- Fyrir takmarkað úrval gilda er hægt að telja upp möguleika
- gender dálkurinn í MovieStar er skilgreindur sem eins stafs strengur, en ef til vill leyfum við aðeins ‘M‘ eða ‘F‘
- Við gætum þá skrifað skorður

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

Vinstri ytri tengingar (left outer join)

- Ef R og S eru vensl með einhver sameiginleg eigindi þá er vinstri ytri tenging R og S (left outer join):

$$R⟕S = (R \bowtie S) \cup \left(\left(R - \pi_{r_1,r_2,\ldots,r_n}(R \bowtie S)\right) \times \{(\bot, \ldots, \bot)\}\right)$$

- þar sem $r_1, r_2, \ldots, r_n$ eru eigindi R sem ekki koma fyrir í S og mengið $\{(\bot, \ldots, \bot)\}$ er eins staks vensl yfir þau eigindi R sem ekki koma fyrir í S
- Áhrifin eru þau að $R⟕S$ inniheldur a.m.k. eina röð fyrir sérhverja röð í R
- Ef röð vantar í $R \bowtie S$ fyrir einhverja tiltekna röð R þá mun $R⟕S$ samt innihalda röð sem samsvarar þeirri röð, en í þeirri röð verða eiginleikarnir sem eðlilega kæmu frá S með gildi $\bot$ (samsvarar NULL)

Hægri ytri tengingar (right outer join)

- Ef R og S eru vensl með einhver sameiginleg eigindi þá er hægri ytri tenging R og S (right outer join):

$$R \rtimes\!\!\!\!\!\!\subset S = (R \bowtie S) \cup \left(\{(\bot, \ldots, \bot)\} \times \left(S - \pi_{s_1, s_2, \ldots, s_m}(R \bowtie S) \right) \right)$$

- þar sem $s_1, s_2, \ldots, s_m$ eru eigindi S sem ekki koma fyrir í R og mengið $\{(\bot, \ldots, \bot)\}$ er eins staks vensl yfir þau eigindi S sem ekki koma fyrir í R
- Áhrifin eru þau að $R \rtimes\!\!\!\!\!\!\subset S$ inniheldur a.m.k. eina röð fyrir sérhverja röð í S
- Ef röð vantar í $R \bowtie S$ fyrir einhverja tiltekna röð S þá mun $R \rtimes\!\!\!\!\!\!\subset S$ samt innihalda röð sem samsvarar þeirri röð, en í þeirri röð verða eiginleikarnir sem eðlilega kæmu frá R með gildi $\bot$ (samsvarar NULL)

Fullar ytri tengingar (full outer join)

- Ef R og S eru vensl með einhver sameiginleg eigindi þá er full ytri tenging R og S (full outer join):

$$R ⟗ S = (R ⟕ S) \cup (R ⟖ S)$$

- Í Ullman og Widom eru notuð önnur tákn í stað ⟗, ⟖ og ⟕
- Þið megið nota þau tákn ef ykkur hentar, en þau eru svona:

  ⟗ samsvarar $\overset{\circ}{\bowtie}_L$ ⟖ samsvarar $\overset{\circ}{\bowtie}_R$ ⟕ samsvarar $\overset{\circ}{\bowtie}$

- Setja má skilyrði með ytri tengingum á svipaðan hátt og með innri tengingum ($\bowtie$, θ-tengingar)
  - þá er ekki nauðsynlegt að R og S hafi sameiginleg eigindi

Hönnun gagnagrunna – Database Design

- Tengsl milli gagna ákvarða hvernig er „best“ að hanna gagnagrunna
  Data relationships decide the best way to design databases

1. Hvernig á að skipuleggja gagnagrunn í töflur
   How to organize a database with multiple tables/relations
2. Hvernig hægt er að bera kennsl á galla í hönnun gagnagrunns
   Recognizing design flaws in databases
3. Hvernig hægt er að laga lélegan gagnagrunn
   How to fix a bad design
4. Formlegar aðferðir til að skilgreina tengsl upplýsinga
   Formal methods for defining data relationships
5. Formlegar aðferðir til að brjóta upp töflur
   Formal methods for breaking up tables

Fallákveður (functional dependencies)

- Helsta tækið sem við höfum til að skilgreina tengsl milli gagna eru fallákveður (functional dependencies, FD)
- Eigindi (dálkar, attributes) $\bar{A} = A_1, \ldots, A_n$ ákvarða (functionally determine) eigindi $\bar{B} = B_1, \ldots, B_m$ í venslum R ef allar n-dir í R sem hafa sömu gildi fyrir $A_1, \ldots, A_n$ hafa einnig sömu gildi fyrir $B_1, \ldots, B_m$
- Við táknum þetta með rithættinum (notation)

$$A_1, \ldots, A_n \rightarrow B_1, \ldots, B_m$$

eða

$$\bar{A} \rightarrow \bar{B}$$

Dæmi

title	year	length	studioName	starName
Star Wars	1977	124	Fox	Carrie Fisher
Star Wars	1977	124	Fox	Harrison Ford
Star Wars	1977	124	Fox	Mark Hamill
Empire Strikes Back	1980	111	Fox	Harrison Ford
Terms of Endearment	1983	132	MGM	Debra Winger
Terms of Endearment	1983	132	MGM	Jack Nicholson
The Usual Suspects	1995	106	MGM	Kevin Spacey

- Hvaða fallákveður gilda um þessi vensl?

  What FD‘s might hold here?

Dæmi

Númer	Heiti	Nafn	Notandanafn	Námsleið	Hópur	Dagur	Tími
TÖl303G	Gagnasafnsfræði	Nonni	abcd	Eðlisfræði	d4	Mið	15:50-17:20
TÖl303G	Gagnasafnsfræði	Sigga	bcde	Stærðfræði	d3	Mið	08:20-09:50
TÖl303G	Gagnasafnsfræði	Sigga	cdef	Tölvunarfræði	d2	Fim	15:50-17:20
TÖL304G	Forritunarmál	Sigga	cdef	Tölvunarfræði	d1	Mán	13:20-14:50

- Hvaða fallákveður gilda um þessi vensl?

  What FD‘s might hold here?

(Mögulegir) Lyklar – (Candidate) Keys

- Eigindi $\bar{A} = A_1, \dots, A_n$ mynda **mögulegan lykil (candidate key)** í venslum R ef

  A set $\bar{A} = A_1, \dots, A_n$ of attributes form a **candidate key** in relation R if

1. $\bar{A} \rightarrow B_i$ gildir fyrir alla dálka (öll eigindi) B_i í venslunum

   $\bar{A} \rightarrow B_i$ holds for all columns (all attributes) B_i in the relation

2. Ekkert eiginlegt hlutmengi eigindanna í $\bar{A}$ ákvarðar alla dálka

   No proper subset of the attributes in $\bar{A}$ determines all columns

- Safn af dálkum er **yfirlykill (superkey)** ef það inniheldur mögulegan lykil

  A collection of columns is a **superkey** if it contains a candidate key

  - (Mögulegur) lykill er því áreiðanlega yfirlykill – A (candidate) key is therefore a superkey
  - Yfirlykill er **ekki endilega** mögulegur lykill – A superkey is not necessarily a candidate key

Reglur um fallákveður – Rules for FD‘s

- Gegnvirkni: Fyrir vensl $R(A, B, C)$ gildir:
  Transitivity: For a relation $R(A, B, C)$ we have:

  Ef $A \rightarrow B$ og $B \rightarrow C$ þá $A \rightarrow C$

  If $A \rightarrow B$ and $B \rightarrow C$ then $A \rightarrow C$

- Skipting/sameining: Ef $\bar{B} = \{B_1, \ldots, B_m\}$ er safn eiginda þá eru eftirfarandi fullyrðingar jafngildar:

  Splitting/Combining: if $\bar{B} = \{B_1, \ldots, B_m\}$ is a collection of attributes then the following are equivalent:

  $$\bar{A} \rightarrow \bar{B}$$

  og/and

  $$\bar{A} \rightarrow B_1 \wedge \cdots \wedge \bar{A} \rightarrow B_m$$

Fáfengilegar fallákveður – Trivial FD‘s

- Fallákveða $\bar{A} \rightarrow \bar{B}$ er fáfengileg (trivial) ef $\bar{B} \subseteq \bar{A}$

  An FD $\bar{A} \rightarrow \bar{B}$ is trivial if $\bar{B} \subseteq \bar{A}$

- Allar fáfengilegar fallákveður eru sannar (auðvitað – það er fáfengileg fullyrðing)

  All trivial FD‘s are true (that is a trivial statement)

- Fallákveðan $\bar{A} \rightarrow \bar{B}$ er því jafngild fallákveðu $\bar{A} \rightarrow \bar{C}$ þar sem:

  The FD $\bar{A} \rightarrow \bar{B}$ is therefore equivalent to an FD $\bar{A} \rightarrow \bar{C}$ where:

$$\bar{C} = \bar{B} - \bar{A}$$

  eða, eins og sumir skrifa/or, as some write:

$$\bar{C} = \bar{B} \setminus \bar{A}$$

Lokun eigindamengis – Closure of a set of attributes

- Ef $\bar{A}$ er safn (mengi) eiginda og F er safn af fallákveðum fyrir vensl R, þá er **lokun** $\bar{A}$ mengi þeirra eiginda $\bar{B}$ í venslunum R þannig að F leiðir til þess að $\bar{A} \rightarrow \bar{B}$

  If $\bar{A}$ is a collection (set) of attributes and F is a collection of FD‘s for a relation R, then the **closure** of $\bar{A}$ is the collection of attributes $\bar{B}$ in relation R such that F implies that $\bar{A} \rightarrow \bar{B}$

- Lokun $\bar{A}$ er táknuð með rithættinum $\bar{A}^+$

  The closure of $\bar{A}$ is denoted by $\bar{A}^+$

Dæmi um útreikning lykils – Computing a key

- Gerum ráð fyrir venslum $R(A, B, C)$ með fallákveður $AB \rightarrow C$ og $C \rightarrow B$
- Eiginleikinn A kemur hvergi fyrir í hægri hlið fallákveðu og hlýtur því að vera hluti allra mögulegra lykla
- Lokunin af A er hins vegar $\{A\}^+ = \{A\}$, sem er ekki allt eiginleikamengið, A er því ekki lykill
- Hins vegar eru lokanirnar af AB og AC allt eigindamengið: $\{A, B\}^+ = \{A, C\}^+ = \{A, B, C\}$
- Bæði AB og AC eru því yfirlyklar (superkeys) og mögulegir lyklar (candidate keys) og eru því lyklar fyrir venslin R

Dæmi um útreikning lykils – Computing a key

- Gerum ráð fyrir venslum $R(A, B, C, D, E, F, G, H, I, J)$ með fallákveður $AB \rightarrow C$, $BD \rightarrow EF$, $AD \rightarrow GH$, $A \rightarrow I$ og $H \rightarrow J$
- Eiginleikarnir ABD koma hvergi fyrir í hægri hlið fallákveðu og hljóta því að vera hluti allra mögulegra lykla
- Lokunin af ABD er $\{A, B, D\}^+ = \{A, B, C, D, E, F, G, H, I, J\}$, sem er allt eiginleikamengið
- ABD er því yfirlykill (superkey) og mögulegur lykill (candidate key) og er því lykill – reyndar sá eini sem kemur til greina