



Gagnasöfn og SQL

Hjálmtýr Hafsteinsson
Tölvunarfræði HÍ



Efni námskeiðs

- Almennt um gagnasafnskerfi
- SQLite gagnasafnskerfið
- Einfaldar SQL fyrirspurnir
- Meðhöndlun gagna
- Samsöfnun (*aggregates*), hópun (*group by*)
- Töfluskilgreiningar, skorður (*constraints*)



Gagnasafnskerfi

- Gagnasafn er safn gagna á skipulögðu formi
- Gríðarlega útbreidd notkun:
 - Fjármálagögn banka
 - Birgðabókhald verslana
 - Facebook, Twitter, Amazon, ...
 - ...
- Nær öll vefsetur byggja á gagnasafnskerfum
 - Vefsíður "búna til" upp úr gagnasafni



Eiginleikar gagnasafnskerfa

- Kostir
 - Geyma gögn á öruggan hátt
 - Hraðvirk leit að gögnum
 - Aðgangur frá mörgum notendum samtímis
 - Gögn geymd á skipulögðu formi
- Gallar
 - Flókin og dýr hugbúnaður
 - Henta ekki fyrir lítið gagnamagn
 - Gögn geymd á skipulögðu formi

Venslagagnasöfn

- Byggir á stærðfræðihugtakinu **vensl** (*relation*)

Dæmi um tvístæð (*binary*) vensl:

$\{(1, 2), (1, 3), (2, 3), (1, 4), (2, 4), \dots\}$

Þetta eru venslin: " $x < y$ "

Hvert stak
kallast *tvennd*

Annað dæmi:

$\{(Jón, 895-4321), (Gunná, 555-1234), \dots\}$

Þetta eru venslin: " x **hefur símanúmer** y "



Grunnmengi (*domain*)

- Hvert stak í tvennd (eða n -d) kemur úr mengi
 - Í venslunum $\{ (1, 2), (1, 3), \dots \}$
 - Bæði stökin koma úr mengi jákvæðra heiltalna
 - Í venslunum $\{ (\text{Jón}, 895-4321), \dots \}$
 - Fyrri stakið úr mengi mannanafn (eða einhverjum hópi)
 - Seinna stakið úr mengi löglegra símanúmera
- Aðeins stök úr menginu geta verið í þessu sæti

Töflur og vensl

- Vensl eru oft táknuð sem töflur

Nafn	Sími
Jón Gunna ⋮	895-4321 555-1234 ⋮

Nafn mengis
sem þessi
stök koma úr

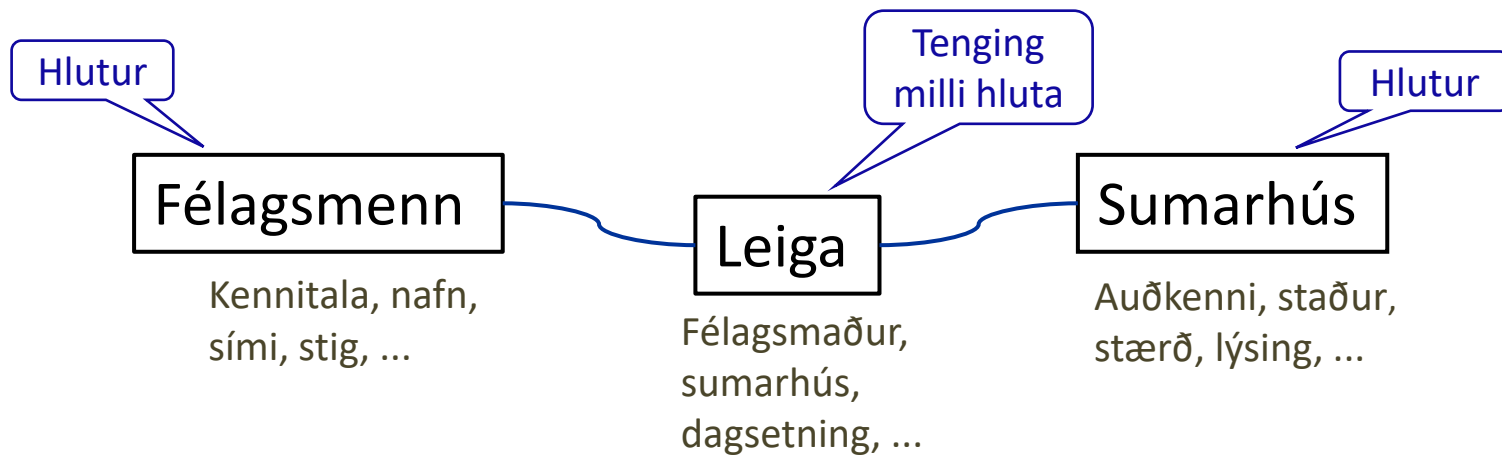
Dálkur

Lína

Ath: Taflan er **mengi** lína

Hönnun gagnasafna

- Gögn eru annað hvort upplýsingar um hlut eða upplýsingar um tengingar hluta
 - Hvorutveggja geymt sem vensl (töflur)



Kostir venslalíkansins



ENDUR
MENNTUN

- Einfalt gagnsætt líkan
 - Allt er töflur!
- Traustar stærðfræðilegar undirstöður
 - Venslareikningur, mengi
- Passar oftast vel við raunveruleg gögn
- Hraðvirkar útfærslur
 - Nær öll gagnasafnskerfi í dag

SQLite

- Frítt einfalt venslagagnasafnskerfi
 - Hefur nær allar SQL skipanir
 - Mjög auðvelt í uppsetningu
 - Innan við 1MB að stærð!
- Notað mjög víða
 - Innbyggt í Firefox, iPhone, Android, Skype, Photoshop, iTunes, ...



SQLite

- Heimasíða þess:
<http://www.sqlite.org>
- Náði í SQLite og sýnisgagnasafn: <https://hjalmtyr.github.io/SQL1/>
 - Upplýsingar vegna útleigu á sumarhúsum:

felagar	upplýsingar um félagsmenn
sumarhus	upplýsingar um sumarhús
leigur	upplýsingar um leigur

SQLite skipanaskel

- Við notum SQLite í gegnum skipanaskel
 - Leyfir okkur að einbeita okkur að SQL
 - Öll stærri gagnasafnskerfi hafa þannig viðmót
 - Allir "alvöru" notendur gagnasafnskerfa nota skipanalínuviðmót!
- Það eru til grafísk viðmót (*GUI*) fyrir SQLite
 - Listi af þeim er á heimasíðu námskeiðs
 - Þau gefa betri yfirsýn yfir gagnasafn með mörgum töflum

- Ná í SQLite og sýnisgagnasafn
 - Windows: tvær skrá - MacOS: aðeins gagnasafnsskrá
 - Vista í nýju skráarsafni (t.d. **D:\SQL**)
- Keyra **sqlite3.exe** (eða **sqlite3**)
- Opna sýnisgagnasafn:
 - Nota SQLite skipunina:
.open sumarhus.db
- Prófa nokkrar feitletraðar skipanir á blaðinu

Ef þetta gengur illa þá getið
þið prófað sqliteonline.com

Það er vefviðmót, engin uppsetning,
sumt auðveldara, annað flóknara

SQL fyrirspurnarmálið



- SQL hannað hjá IBM ~1972
- Byggir á fræðilegu líkani fyrir vensl
- Inniheldur margar gerðir skipana
 - Ein aðalskipun: **SELECT**
 - Aðrar skipanir vinna með töflur og gögn:
 - Búa til, breyta og eyða töflum
 - Setja inn, breyta og eyða gögnum
 - Breyta skipulagi gagnanna

SQL fyrirspurnir

- SQL fyrirspurnir segja hvaða gögn við viljum, ekki hvernig þau eru fundin
 - Skipunin skilgreinir **mengi** gagnanna sem við viljum fá

Hvað viltu?

"Ég ætla að fá 12" Ítalskan BMT í hvítu brauði með öllu grænmeti nema jalapeno"

Hvernig á
að fá það?

"Taktu hvítt brauð, skerðu það, náðu síðan í salami, pepperoni og skinku, ..."

Fyrirspurnir



- **SELECT** skipunin nær í innihald tafla

select **nafn** **from** **felagar**;

Dálkur í töflunni

Taflan heitir
felagar

Muna eftir
semíkommu

Fyrirspurnir

- Getum fengið fleiri dálka

```
select nafn, inng_ar from felagar;
```



Teljum upp
dálkanöfn



Fyrirspurnir í SQLite

- Útkoman er ekki sérlega flott:

```
Gunnar | 2017  
Erla | 1994  
...
```

- Getum látið SQLite setja úttakið í dálka með nöfnum dálkanna:

```
.mode column
```

Ath: Skipanir til SQLite byrja á **punkti** og enda **ekki** á semíkommu

Fyrirspurnir



ENDUR
MENNTUN

- Getum fengið alla dálka með *

```
select * from felagar;
```

Ekki ráðlegt að nota * í raunverulegri notkun.
Vitum þá ekki hversu marga dálka við fáum.
Töflur breytast gjarnan yfir tíma.



Röð úttaks

- Úttakið kemur í "einhverri röð"
 - Líklega eftir því hvenær gögnin voru sett inn
- Línurnar eru stök í **mengi**
 - Stök í mengi hafa enga sérstaka röð
- Til að raða úttakinu notum við **order by**:

```
select nafn, inng_ar from felagar  
order by inng_ar;
```

Röð úttaks



- Hægt að raða eftir mörgum dálkum

```
select nafn, inng_ar from felagar  
order by inng_ar, nafn;
```

Hér er raðað fyrst eftir **inng_ar** í hækkandi röð
og síðan í stafrófsröð eftir nafni innan hvers árs



Röð úttaks

- Sjálfgefið er að raðað sé í hækkandi röð
 - Getum raðað í lækkandi röð með **desc**

```
select nafn, stig from felagar  
order by stig desc;
```

Hér koma hæstu stigin fyrst

desc er stytting á orðinu "descending"
Hægt að nota **asc** fyrir hækkandi röð

Margir röðunardálkar

- **desc** (eða **asc**) á aðeins við dálkinn sem það stendur við

```
select nafn, stig, inng_ar from felagar  
order by stig desc, inng_ar;
```

Hæstu stig fyrst og síðan í hækkandi röð
eftir inngönguári ef stig þau sömu



Takmarka fjölda lína

- Stundum viljum við ekki fá allar línur
 - Getum takmarkað fjöldann með **limit**

```
select nafn, stig from felagar  
order by stig desc  
limit 3;
```

Ath: Línuskipting
breytir engu

Sýnir þá þrjá félaga sem hafa flest stig

Ath: Ekkert vit í að nota **limit** nema með **order by**

Sleppa línunum



- Stundum viljum við ekki fyrstu 3 línurnar, heldur næstu 3
 - Bætum þá `offset` við `limit`

```
select nafn, stig from felagar  
order by stig desc  
limit 3 offset 3;
```

`offset x` sleppir
`x` fyrstu línunum

Sýnir þá þrjá félagi sem eru í fjórða,
fimmta og sjötta sæti yfir flest stig



- Sýna sumarhús í hækkandi röð eftir fjölda rúma og lækkandi röð eftir stærð
- Sýna upplýsingar um stærsta sumarhúsið
- Sýna nöfn og inngönguár þriggja nýjustu félagsmannanna
- Sýna dagsetningu næstnýjustu leigunnar

Velja út línur

- Notum skilyrði í **where**-hluta

```
select nafn, stig from felagar  
where stig > 400;
```

Sýnir nafn og stig þeirra sem hafa fleiri en 400 stig

```
select * from felagar  
where nafn='Gunnar' ;
```

Sýnir alla dálka þeirra sem heita Gunnar

Velja út línur



- Fleiri dæmi:

```
select * from felagar  
where nafn != 'Gunnar';
```

Ath.: Notum einfaldar
gæsalappir fyrir strengi

```
select * from felagar  
where inng_ar >= 2000;
```

Allir sem gerðust félagar á þessari öld

Flóknari skilyrði

- Sameinum skilyrði með **and** og **or**

```
select * from felagar  
    where stig >= 200 and stig <= 400;
```

Allir með stig á bilinu 200 til 400

```
select * from felagar  
    where stig between 200 and 400;
```

Jafngild skipuninni að ofan, en sjaldan notað, því ekki augljóst hvort gildin 200 og 400 séu með eða ekki

Flóknari skilyrði

- Hægt að nota útreikning í skilyrðum

```
select * from felagar  
  where stig > 300  
    and 2024-inng_ar > 10;
```

Í SQLite er hægt að fá núverandi ártal með
`strftime('%Y', 'now')`

Dálítið mismunandi milli gagnasafnskerfa
hvernig núverandi dagsetning er fengin

Möguleg vandamál

- Finna þá félaga með fleiri en 400 stig sem gengu í félagið 2016 eða 2017

```
select * from felagar
      where inng_ar = 2016
            or inng_ar = 2017
            and stig >= 400;
```

Úttak:

48	Gunnar	107	450	2017
64	Helga	112	55	2016

en Helga er aðeins
með 55 stig!



Hvert er vandamálið?

- Virkinn **and** hefur hærri forgang en **or**
 - þurfum að nota sviga til að fá rétta útkomu

```
select * from felagar
      where (inng_ar = 2016
             or inng_ar = 2017)
      and stig >= 400;
```

Þetta er svipað og í segðinni $5 - 2 * 3$

Útkoman er $5 - 6 = -1$, en ekki $3 * 3 = 9$



- Sýna sumarhús með fleiri en 6 rúm
- Sýna leigur á árinu 2024
- Sýna alla félaga í Reykjavík eða Kópavogi sem hafa minna en 200 stig
- Sýna þann félaga sem býr utan Reykjavíkur sem hefur mestan fjölda stiga



Reglulegar segðir (*regular expressions*)

- Hægt að nota ***algildisstafi*** (*wildcards*)
 - Notum þá með orðinu **like**

```
select * from sumarhus  
where stadur like 'Husaf%';
```

Táknið % parast á móti
0 eða fleiri stöfum

Sýnir öll sumarhús með staðsetningu sem byrjar á 'Husaf'

Reglulegar segðir

- Annað dæmi:

```
select * from felagar  
      where nafn like '%i%';
```

Sýnir alla félagar með nöfn sem innihalda stafinn i

- Hvað með?

```
select * from felagar  
      where nafn like '%';
```

Reglulegar segðir



- Táknið `_` passar við nákvæmlega einn staf

```
select * from felagar  
where nafn like '____';
```

Sýnir alla félagar með 4ra stafa nöfn

4 _ tákn

- Finna nöfn með 5 eða fleiri stafi:

```
select * from felagar  
where nafn like '_____%';
```

5 _ tákn

Reglulegar segðir



- Viljum finna félaga með nöfn sem eru 4 stafir eða styttri

– Ein leið:

```
select * from felagar
      where nafn like '_' or
            nafn like '__' or
            ...;
```

– Betri leið:

```
select * from felagar
      where nafn not like '____%';
```



Hástafir/lágstafir í **like**

- Sjálfgefið er að **like** geri ekki greinarmun á hástöfum og lágstöfum

- Hægt að breyta því:

```
pragma case_sensitive_like = on;
```

- Þá skilar þessi skipun engri niðurstöðu:

```
select * from felagar  
where nafn like 'gunnar';
```

pragma er óstöðluð skipun, sem hægt er að nota til að breyta hegðun SQLite á ýmsa vegu

Æfingar



ENDUR
MENNTUN

- Sýnið öll sumarhús með textann "vatn" í nafninu
- Sýnið allar leigur í júlí, óháð ári
- Sýnið alla félagsmenn með nafn sem endar á "a" og hafa meira en 200 stig

Aðrar skipanir í SQL

- Að búa til og eyða töflum

```
create table felagar (  
    fnr integer,  
    nafn char(30),  
    postnr char(3),  
    stig integer,  
    inng_ar integer);
```

Nafn dálks

Nafn töflunnar

Tegund dálks

```
drop table felagar;
```

Eyðir út töflunni og
öllu sem í henni er

Aðrar skipanir í SQL

- Setja inn gögn

```
insert into felagar  
  (fnr, nafn, postnr, stig, inng_ar)  
values (55, 'Axel', '108', 50, 2024);
```

Má sleppa því að telja upp dálkana ef öll gildi til staðar

Ef ekki er sett gildi í einhvern dálk þá
verður hann tómur (þ.e. **NULL**)

Aðrar skipanir í SQL



- Eyða gögnum

```
delete from felagar  
where fnr = 55;
```

Eyðir út öllum línum sem uppfylla skilyrðið

– Ef skilyrðið vantar ... **pá er öllum línum eytt!**

```
delete from felagar;
```

Aðrar skipanir í SQL



- Breyta gögnum

```
update felagar  
  set postnr = '101'  
  where fnr = 31;
```

Breytir öllum línum sem uppfylla skilyrðið

```
update felagar  
  set stig = stig - 50  
  where nafn = 'Rakel';
```

- Hækkið stigin hjá öllum félagsmönnum um 50
- Búið til töfluna **tilraun** með dálkunum **a** (heiltala) og **b** (10 stafa texti)
 - Setjið eina línu inní töfluna **tilraun**
 - Skoðið töfluna með **select**
 - Eyðið töflunni



Innflutningur gagna

- Gagnasafnskerfi hafa líka sérstakar skipanir til að hlaða inn gögnum
 - Mismunandi skipanir milli kerfa, ekki hluti af SQL
 - Gögn oftast á CSV-formi (Comma Separated Values)
 - Aðskilnaðartákn geta verið: , ; : | **TAB**
 - Nú að verða algengara að nota XML
 - Oftast hraðvirkara en að nota margar **insert**-skipanir

Gagnainnflutningur í SQLite

- SQLite hefur skipunina **.import** til að lesa gögn inní töflu

```
.import gogn.csv felagar
```

Nafn á skrá

Nafn á töflu

- Skráin þarf að nota rétt aðskilnaðartákn (|)
 - Hægt að breyta því með skipuninni **.separator**

```
.separator ;
```

Hér eftir er búist við aðskilnaðartákninu ; í innlesnum skráum

Gagnaútflutningur í SQLite

- Notum **select**-skipun til að búa til gagnaskrá á CSV-formi

Stilla úttak
úr **select**

Úttak í skrá

Úttak aftur
á skjáinn

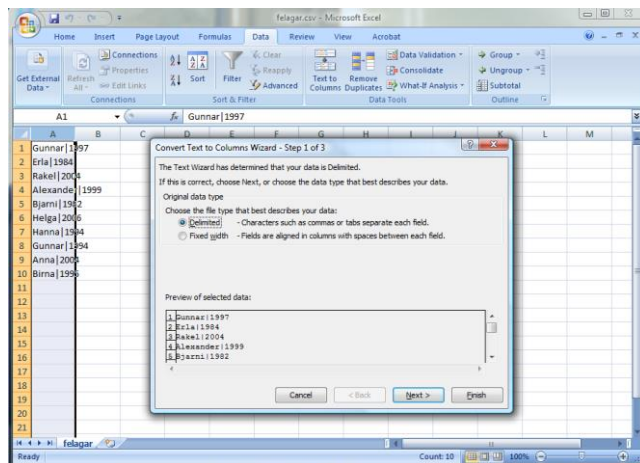
```
.headers off  
.mode list  
.separator |  
.output felagar.csv  
select * from felagar;  
.output
```

Ýmsir fleiri möguleikar
í **.mode**-skipuninni:

- **.mode column**
- **.mode table**
- **.mode csv**
- **.mode box**

Gögn úr SQLite

- Opnið skránnu **felagar.csv** í Excel
 - Fara í "*Data*" valmynd
 - Velja fyrsta dálk skjalsins og smella á "*Text to Columns*"
 - Velja svo "*Delimited*"
 - og svo táknid " | "





Skipanaskrár í SQLite

- Algengt að búnað séu til skrár með SQL skipunum til að skilgreina töflur
- Skipunin **.read** í SQLite les og framkvæmir slíka skrá

.read sumarhus.sql

Sjá þessa skrá á

<https://hjalmtyr.github.io/SQL1/>



Skipanaskrá í SQLite

- Skipunin **.dump** skrifar allt gagnasafnið út
 - Til að skrifa það í skipanaskrá þarf að nota **.output**

```
.output sumarhus1.sql  
.dump  
.output
```

- Einnig hægt að skrifa út einstakar töflur:

```
.dump felagar
```

- Náið í skipanaskránnu **countries.sql**
- Lesið hana inní SQLite
- Skoðið nýju töfluna **countries**
- Setjið innihald töflunnar í CSV-skrá
- Flytjið CSV-skránnu inn í Excel
- Tæmið töfluna **countries** (með **delete**)
- Hlaðið inn í hana úr CSV-skránni

Samsöfnun gagna (*aggregates*)



- Viljum stundum finna heildarupplýsingar um gögn

```
select avg(stig)
  from felagar;
```

- Getum fengið meðaltal yfir hluta gagnanna:

```
select avg(stig) from felagar
  where innng_ar < 2010;
```

Samsöfnun gagna

- Summa yfir dálk

```
select sum(stig) from felagar;
```

- Hágildi og lággildi

```
select max(stig) from felagar  
       where postnr < '170';
```

```
select min(inng_ar) from felagar;
```

Samsöfnun gagna

- Telja allar línur

```
select count(*) from felagar;
```

- Telja gildi í dálki

```
select count(inng_ar) from felagar;
```

- Telja ólík gildi í dálki

```
select count(distinct inng_ar)  
from felagar;
```



Hópun gagna (*group by*)

- Viljum finna meðalfjölda stiga eftir póstnúmeri

- Gætum gert nokkrar fyrirspurnir:

```
select avg(stig) from felagar  
where postnr = '101';
```

og síðan eins fyrir '107', '110', o.s.frv.

- Betra að búa til hópa með **group by** og finna meðaltal innan hvers hóps

```
select postnr, avg(stig) from felagar  
group by postnr;
```



Reglur um hópun

- Aðeins hægt að sýna dálka sem koma fyrir í **group by**-hlutanum

```
select postnr, nafn, avg(stig) from felagar  
group by postnr;
```

Hvaða gildi ætti nafn að hafa fyrir tiltekið pósthúmer?

Þetta er reyndar leyft í SQLite!
Hver er útkoman?



Reglur um hópun

- Má nota fleiri en einn dálk í **group by**
 - Þá er hópað á alla dálkana

```
select postnr, inng_ar, avg(stig)
  from felagar
 group by postnr, inng_ar;
```

Fyrir hvert ólíkt gildi á (póstnúmer, inngönguár)
er fundinn meðalstigafjöldi félaga með þau gildi

Hópun og röðun

- Oft er úttakið raðað eftir hópum

```
select postnr, avg(stig) from felagar  
group by postnr;
```

Gefur úttak í röð eftir póstnúmerum

- Þetta fer eftir útfærslu
 - Til að vera viss um röðun þarf að nota **order by**

```
select postnr, avg(stig) from felagar  
group by postnr  
order by postnr desc;
```



Velja úr hópa

- Viljum ekki sýna alla hópa
 - Veljum línur inní hópa með **where**
 - Veljum hópa til að sýna með **having**

```
select postnr, avg(stig) from felagar  
group by postnr  
having postnr < '170';
```

Meðalfjöldi stiga eftir póstnúmeri í Reykjavík

Velja úr hópa

- Getum valið hópa með flóknari skilyrðum

Finna meðalfjölda stiga eftir póstnúmerum í Reykjavík með a.m.k. tvo félaga

Velur línur
inn í hópana

```
select postnr, avg(stig) from felagar  
where postnr < '170'  
group by postnr  
having count(*) >= 2;
```

Velur hópa
til birtingar

- Sýna heildarfjölda daga í útleigu fyrir hvert sumarhús
- Sýna fjölda leiga eftir dagafjölda (þ.e. helgar- eða vikuleiga)
- Sýna meðalfjölda stiga eftir inngönguári, ef fleiri en einn á því ári

Form **select** skipunar



```
SELECT <dálkar eða útreikningur>  
  FROM <tafla>  
 WHERE <skilyrði>  
 GROUP BY <dálkar>  
 HAVING <skilyrði>  
 ORDER BY <dálkar>  
 LIMIT <tala> OFFSET <tala>;
```

Almennt form **select** skipunar



```
[ WITH [ RECURSIVE ] with_query [, ...] ]  
SELECT [ ALL | DISTINCT [ ON ( expression [, ...] ) ] ]  
    * | expression [ [ AS ] output_name ] [, ...]  
    [ FROM from_item [, ...] ]  
    [ WHERE condition ]  
    [ GROUP BY expression [, ...] ]  
    [ HAVING condition [, ...] ]  
    [ WINDOW window_name AS ( window_definition ) [, ...] ]  
    [ { UNION | INTERSECT | EXCEPT } [ ALL ] select ]  
    [ ORDER BY expression [ ASC | DESC | USING operator ] [ NULLS { FIRST | LAST } ] [, ...] ]  
    [ LIMIT { count | ALL } ]  
    [ OFFSET start [ ROW | ROWS ] ]  
    [ FETCH { FIRST | NEXT } [ count ] { ROW | ROWS } ONLY ]  
    [ FOR { UPDATE | SHARE } [ OF table_name [, ...] ] [ NOWAIT ] [...] ]
```

where *from_item* can be one of:

```
[ ONLY ] table_name [ * ] [ [ AS ] alias [ ( column_alias [, ...] ) ] ]  
( select ) [ AS ] alias [ ( column_alias [, ...] ) ]  
with_query_name [ [ AS ] alias [ ( column_alias [, ...] ) ] ]  
function_name ( [ argument [, ...] ] ) [ AS ] alias [ ( column_alias [, ...] | column_definition [, ...] ) ]  
function_name ( [ argument [, ...] ] ) AS ( column_definition [, ...] )  
from_item [ NATURAL ] join_type from_item [ ON join_condition | USING ( join_column [, ...] ) ]
```

and *with_query* is:

```
with_query_name [ ( column_name [, ...] ) ] AS ( select )
```

```
TABLE { [ ONLY ] table_name [ * ] | with_query_name }
```

Önnur SQLite viðmót

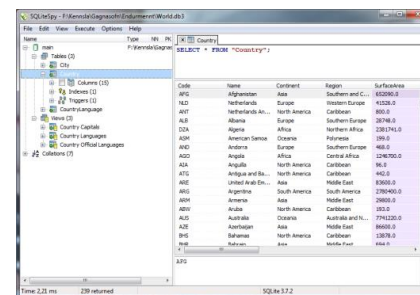
- Skipanaskelin er einföld og þægileg

- en þarf að muna skipanir
- erfiðara að fá yfirlit yfir gagnasafnið

```
C:\Windows\system32\cmd.exe - sqlite3 sumarhus.db
fmr      nafn      postnr  stigi  leiga_ar
48       Gunnar    107     450    1997
12       Erla      200     125    1984
11       Heiði     112     310    2004
12       Alexander 112     200    1977
9        Björni    101     370    1922
64       Helga     112     55     2006
21       Rannsóla 220     500    1974
24       Gunnar    107     330    1994
45       Anna      200     110    2004
45       Birna     112     215    1995
sqlite> select nafn, days from felagar natural join leigur;
nafn      days
Gunnar    2007-07-18
Gunnar    2010-07-10
Erla      2010-02-12
Alexander 2008-06-20
Alexander 2010-08-02
Helga     2010-10-15
Helga     2010-07-09
Anna      2007-11-08
Anna      2010-07-01
sqlite>
```

- Grafísk viðmót

- líkari venjulegum forritum
- en óstöðluð og hafa mism. möguleika





DB Browser for SQLite

- Opinn og frír hugbúnaður
- Til fyrir Windows, Mac, Linux, FreeBSD
- Nokkuð örar uppfærslur
- Margvíslegir eiginleikar:
 - Innflutningur/útflutningur gagna
 - Gandálfar til að búa til töflur og fleira
 - Auðvelt að breyta stillingum gagnasafns
 - Hægt að sjá lista yfir allar framkvæmdar SQL skipanir

DB Browser for SQLite - D:\OneDrive - Háskóli Íslands\Endurmennt\SQL\okt2019-\sumarhus.db

File Edit View Tools Help

New Database Open Database Write Changes Revert Changes Open Project Save Project Attach Database Close Database

Database Structure Browse Data Edit Pragma Execute SQL

SQL 1

```
1 select * from felagar
```

	fnr	nafn	postnr	stig	inng_ar
1	48	Gunnar	107	450	2007
2	12	Erla	200	125	1994
3	31	Rakel	112	330	2004
4	32	Alexander	112	200	2009
5	8	Bjarni	101	370	1992
6	64	Helga	112	55	2006

Result: 10 rows returned in 22ms
At line 1:
select * from felagar

Edit Database Cell

Mode: Text

112

Type of data currently in cell: Text / Numeric
3 char(s)

Apply

DB Schema

Name	Type	Schema
Tables (3)		
felagar	CREATE TABLE felagar(fnr integer, nafn	
leigur	CREATE TABLE leigur(fnr integer, husnr	
sumarhus	CREATE TABLE sumarhus(husnr integer,	
Indices (0)		
Views (0)		
Triggers (0)		

SQL Log Plot DB Schema

UTF-8

- SQLite í vafra
- Einfalt viðmót til að vinna með SQLite gagnasöfn
- Byggir á [sql.js](https://sql.js.org/)
 - C-kóði fyrir SQLite þýddur yfir í Javascript
- Ágætt til að æfa SQL
- Engin uppsetning á tölvu

SQL OnLine IDE

+

← → ↻ 🏠 🖨️ 🔒

https://sqliteonline.com

110% 🌐 ⭐ 🛡️ 🏠 ☰

☰

File

Owner DB

▶ Run

📄 Export

📄 Import

Sign in

📧

⚙️

sumarhus.db

Table

felagar

leigur

sumarhus

SQLite

MariaDB

PostgreSQL

MS SQL

BUY

ADS

sumarhus.db

1 SELECT * FROM felagar

i	fnr	nafn	postnr	stig	innng_ar
48		Gunnar	107	450	2017
12		Erla	200	125	1994
31		Rakel	112	330	2014
32		Alexander	112	200	2009
8		Bjarni	101	370	2002
64		Helga	112	55	2016
71		Hanna	220	500	2014
24		Gunnar	107	330	1999
85		Anna	200	110	2014
45		Birna	112	215	2015

📄

📊

📄



NULL gildi

- SQL leyfir dálkum að hafa sérstök NULL gildi
 - Ef gildið er óþekkt, t.d. fæðingardagur
 - Ef gildið á ekki við, t.d. nafn á maka
- Merkingin er að gildi vanti
 - Fáum þau ef gildi vantar í **insert**-skipun

```
insert into felagar (fnr, nafn, inng_ar)  
values (66, 'Helgi', 2024);
```



NULL gildi í SQLite

- SQLite sýnir tóman streng fyrir **NULL** gildi

66|Helgi|||2024

Útkoma úr **select**-skipun
eftir síðustu innsetningu

- Hægt að sýna **NULL** gildi:

`.nullvalue NULL`

Texti sem við ákveðum

Samanburður með **NULL**

- Allar aðgerðir þar sem annað gildið er **NULL** gefa gildið **NULL**

Ef x er **NULL** þá er gildið á $(x + 3)$ líka **NULL**

- Í samanburði þar sem annað gildið er **NULL** verður útkoman sanngildið **Óþekkt**
 - Höfum sanngildin **Satt** og **Ósatt**, nú eru þrjú sanngildi

Sanngildið *Óþekkt*



ENDUR
MENNTUN

- Nú þurfa rökaðgerðirnar AND, OR og NOT að ráða við gildið *Óþekkt*
 - Helstu breytingar:

Satt AND *Óþekkt*

gefur *Óþekkt*

Ósatt AND *Óþekkt*

gefur *Ósatt*

Ósatt OR *Óþekkt*

gefur *Óþekkt*

Satt OR *Óþekkt*

gefur *Satt*

NOT *Óþekkt*

gefur *Óþekkt*



NULL í `select` skipunum

- `select` skilar öllum þeim línum þar sem skilyrðið í `where`-hluta er ***Satt***
- Hvað með þegar skilyrðið er ***Óþekkt***?
 - Sú lína er ekki með (þ.e. eins og gildið væri ***Ósatt***)

```
select * from felagar  
where stig < 200;
```

Skilar ekki línu sem hefur **NULL** í `stig`

Dæmi um NULL

```
select * from felagar  
where stig = 200 or stig != 200;
```

Þetta ættu að vera allir, en fáum
samt ekki þá sem hafa **NULL** í stig

```
select * from felagar  
where 0*stig = 0;
```

Ef stig er **NULL** þá hefur $0*stig$
gildið **NULL**, sem er ekki = 0

Að finna **NULL** gildi

- Hvernig finnum við hvort lína hafi gildið **NULL** í dálki?

```
select * from felagar  
where stig = NULL;
```

Þetta virkar ekki!

Ef annað gildið í samanburði hefur gildið **NULL** þá er útkoman **Óþekkt**, ekki **Satt**

```
select * from felagar  
where stig is NULL;
```

Sérstakur samanburður
sem skilar **Satt** eða **Ósatt**



Töfluskilgreiningar og **NULL**

- Leyfum ekki sumum dálkum að vera **NULL**
 - Látum vita af því í skilgreiningu töflunnar

```
create table felagar (  
    fnr integer not NULL,  
    nafn char(30) not NULL,  
    postnr char(3),  
    stig integer,  
    inng_ar integer);
```

Ráðum því sjálf
hvaða dálkar mega
ekki vera NULL

- Setjið línu inn í töfluna **felagar** með engum stigum.
 - Sýnið félaga í röð eftir stigum. Koma **NULL** með?
 - Finnið meðalfjölda stiga hjá félagsmönnum. Er **NULL** gildið með?
- Breytið skilgreiningu töflunnar **sumarhus** þannig að **fermetrar** megji ekki vera **NULL**
 - Reynið síðan að setja inn **NULL** þar

Heilleiki gagna (*data integrity*)



- Mjög mikilvægt að gögn í gagnasafninu séu rétt
 - Erfitt að eiga við gölluð gögn í gagnasafninu
 - Gefa rangar niðurstöður
 - Erfitt að finna og leiðrétta á öllum stöðum
 - Betra að koma í veg fyrir að röng/gölluð gögn fari inní safnið í upphafi
 - Staðreyna inntak, t.d. vartöluprófa kennitölur í inntaki
 - Nota skorður (*constraints*) á töflur



Lyklar (*keys*)

- Lykill er **dálkur** (eða safn dálka) sem **ákvarðar línu einkvæmt**
 - Í töflunni **felagar**:
 - postnr** er ekki lykill (margar línur með sama gildi)
 - nafn** er ekki lykill (margir geta heitið sama nafni)
 - fnr** er lykill (við búum dálkinn til þannig!)
- Ræðst af eðli gagnanna hvort dálkur sé lykill
 - Er kennitala lykill?
 - Hvað með Gervimaður útlönd (010130-7789)?

Samsettir lyklar

- Í töflum fyrir tengsl á milli hluta eru lyklar oft samsettir úr lyklum hlutanna
 - Lykill í **leigur** er samsettur úr **fnr**, **husnr** og **dags**
 - Ekki nóg að vita bara **fnr** og **husnr**
- Þurfum sjálf að ákveða hvort tiltekin svið myndi lykil
 - Til dæmis, ef enginn má leigja bústað oftari en einu sinni þá er {**fnr**, **husnr**} lykill fyrir **leigur**

Aðallykill (**primary key**)

- Getum látið gagnasafnskerfið vita um lykla í skilgreiningu töflunnar

```
create table felagar (  
    fnr integer primary key,  
    ...  
);
```

Hver tafla getur aðeins
haft einn **primary key**

```
create table leigur (  
    ...  
    primary key (fnr, husnr, dags));
```

Skorður (*constraints*)

- Skilgreining á aðallykli er dæmi um skorðu
 - Aðrar skorður:
 - unique** - segir að dálkurinn sé einkvæmur
 - not null** - segir að dálkurinn verði að hafa gildi
 - default** - gefur sjálfgefið gildi á dálkinn
 - check (skilyrði)** - tryggir að *skilyrði* sé uppfyllt

Dæmi um skorður



```
create table felagar(  
    fnr integer primary key not null,  
    nafn char(30) not null,  
    postnr char(3),  
    stig integer default 0,  
    inng_ar integer check(inng_ar > 1900),  
    constraint pnr_check (postnr > '100'  
                           and postnr <= '999')  
);
```

Aðallykill

Sjálfgefið gildi

Almenn skorða

Skorða með nafni



Hvað næst?

SQL fyrirspurnarmálið

- Framhaldsnámskeið hjá EHÍ:
 - Töflutengingar (*join*)
 - Undirfyrirspurnir (*subqueries*)
 - Notkun á mengjavirkjum (*set operators*)
 - Sýndartöflur, vísar, hönnun gagnasafna, ...
- Kennsluefni á Vefnum:
 - SQL kennsluefni á heimasíðu
 - Háskólanámskeið um Gagnasafnsfræði