Introduction to PostgreSQL

Relational databases & SQL

Andreas Scharmüller

AG Landscape Ecology

2021-11-13 (updated: 2021-11-14)

What is SQL?

- SQL - Structured Query Language
- Database software
- Standard: ISO/IEC 9075 (since 1987)

The typical query

```
SELECT
  city_name,
  country,
  population
FROM cities
WHERE country = 'Austria'
```

R

```
require(data.table)
require(dplyr)
```

Why SQL?

Thomas J. Leeper
@thosjleeper

It’s now clear to me that SQL should probably be taught as the first data science language anyone learns.

Sure you’ll obviously need some other language but SQL is an interface to rectangular and relational data of any size that will also always be conceptually helpful.

270 9:54 PM - Oct 26, 2020

52 people are talking about this

SQL Software

FOSS

Proprietary

Installation

Installation

Windows

https://www.postgresql.org/download/windows

Mac OS X

https://www.postgresql.org/download/macosx

Linux (Debian-based)

```
sudo sh -c 'echo "deb http://apt.postgresql.org/pub/repos/apt $(lsb
wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.as
sudo apt-get update
sudo apt-get -y install postgresql
```

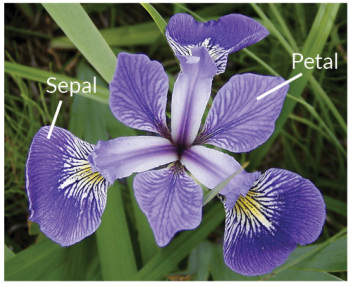
Queries

Queries - The typical query

```
SELECT
  city_name,
  country,
  population
FROM cities
WHERE country = 'Austria'
```

Queries - Example data set

- R - iris dat set
- Dimensions: 150, 5



Iris Versicolor


Iris Setosa


Iris Virginica

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa

Queries - SELECT

```
SELECT
  sepal_length,
  species AS new_name
FROM iris
```

```
SELECT * -- wildcard
FROM iris
LIMIT 10
```

Schema

```
SELECT *
FROM my_schema.iris
LIMIT 10
```

PostgreSQL adressing

```
database.schema.table.column
```

Queries - WHERE

```sql
SELECT
  sepal_length,
  species AS fancy_name
FROM iris
WHERE species = 'setosa'
  AND sepal_length < 5
```

R-equivalent

```r
# data.table
data.table::setDT(iris)
iris[ Species == 'setosa' & Sepal.Length < 5 ]
# dplyr
iris %>%
  dplyr::filter(Species == 'setosa' & Sepal.Length < 5)
```

Different syntax, same logic!

Queries - GROUP BY

```sql
SELECT
  species,
  count(species) AS n,
  max(sepal_length) AS max_sl,
  avg(sepal_length) AS mean_sl
FROM iris
GROUP BY species
```

R-equivalent

```r
# data.table
dt = data.table::as.data.table(iris)
dt[ ,
    .(.N,
      max_sl = max(Sepal.Length)),
    Species ]
# dplyr
iris %>%
  group_by(Species) %>%
  summarise(n = n(),
            max_sl = max(Sepal.Length))
```

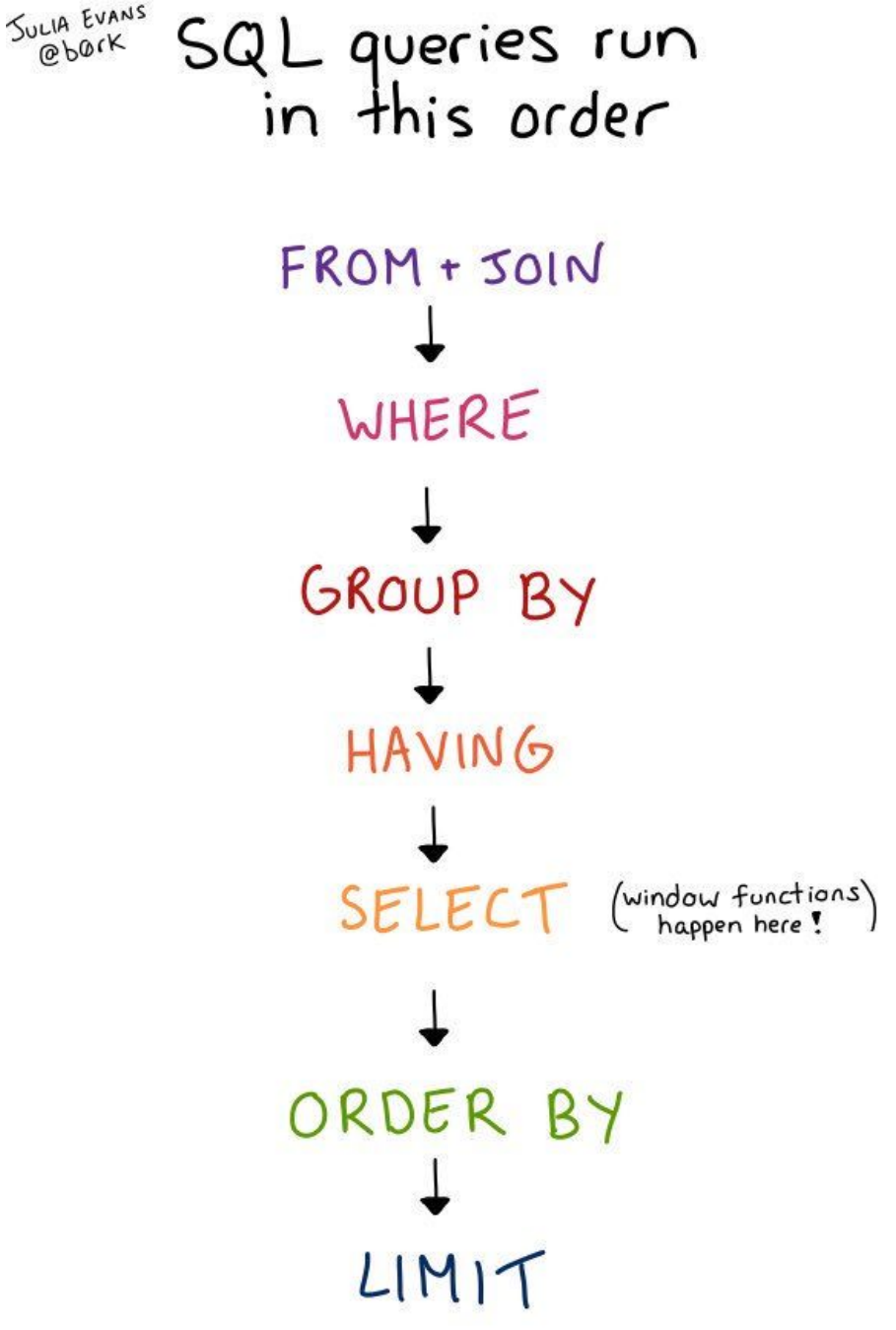
Queries - ORDER BY

```sql
SELECT
  species,
  count(species) AS n,
  max(sepal_length) AS max_sl,
  avg(sepal_length) AS mean_sl
FROM iris
GROUP BY species
ORDER BY max_sl DESC -- ASC
```

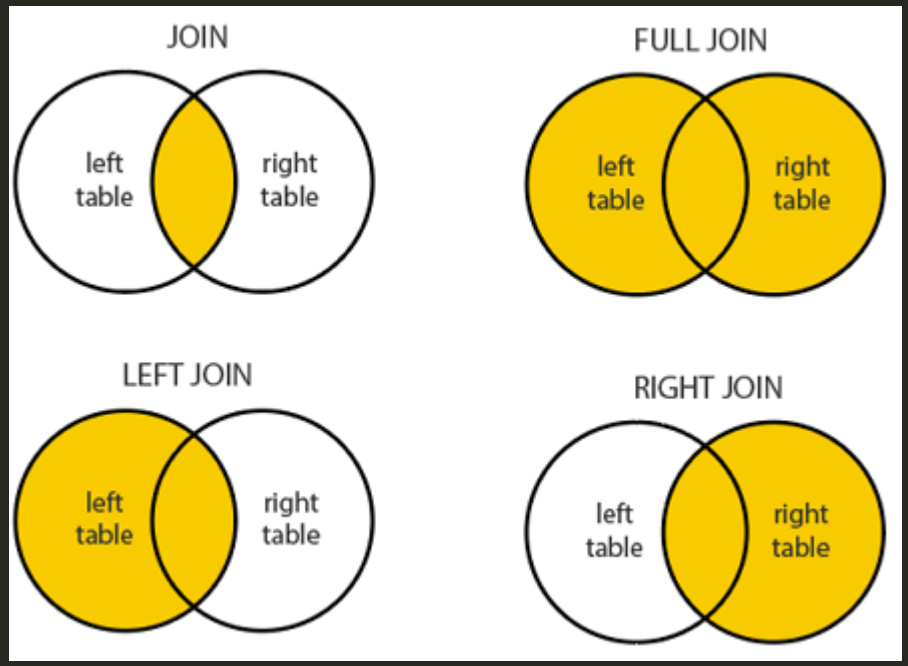
Queries - HAVING

Same as `WHERE`, but only after `GROUP BY`.

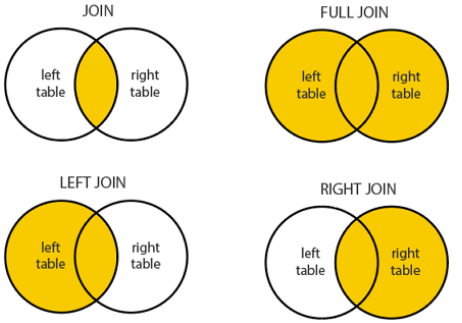
```
SELECT
  species,
  count(species) AS n,
  max(sepal_length) AS max_sl,
  avg(sepal_length) AS mean_sl
FROM iris
GROUP BY species
HAVING species = 'versicolor'
ORDER BY max_sl DESC -- ASC
```


JULIA EVANS
@b0rk
SQL queries run in this order
FROM + JOIN
WHERE
GROUP BY
HAVING
SELECT
(window functions happen here!)
ORDER BY
LIMIT

Queries - JOINs



Queries - JOINs



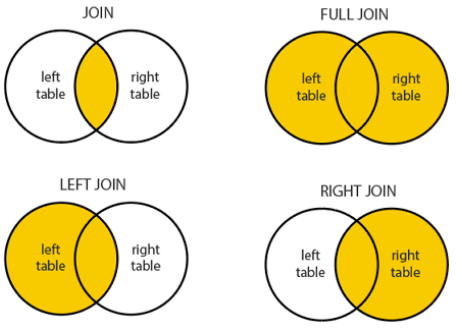
LEFT JOIN

```
SELECT *
FROM my_table1 tab1
LEFT JOIN my_table2 tab2 ON tab1.id = tab2.id
```

RIGHT JOIN

```
SELECT *
FROM my_table1 tab1
RIGHT JOIN my_table2 tab2 ON tab1.id = tab2.id
```

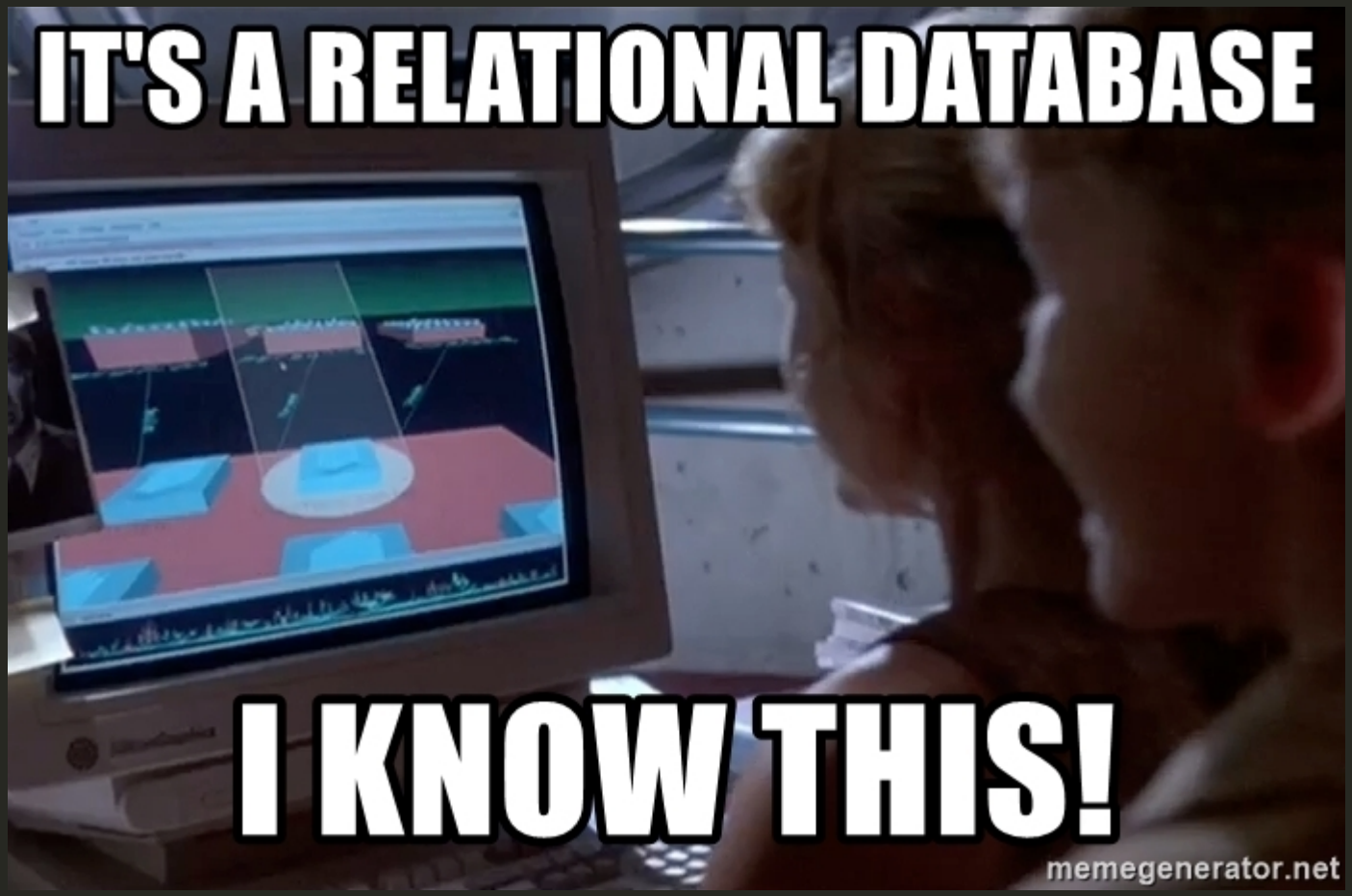
Queries - JOINs



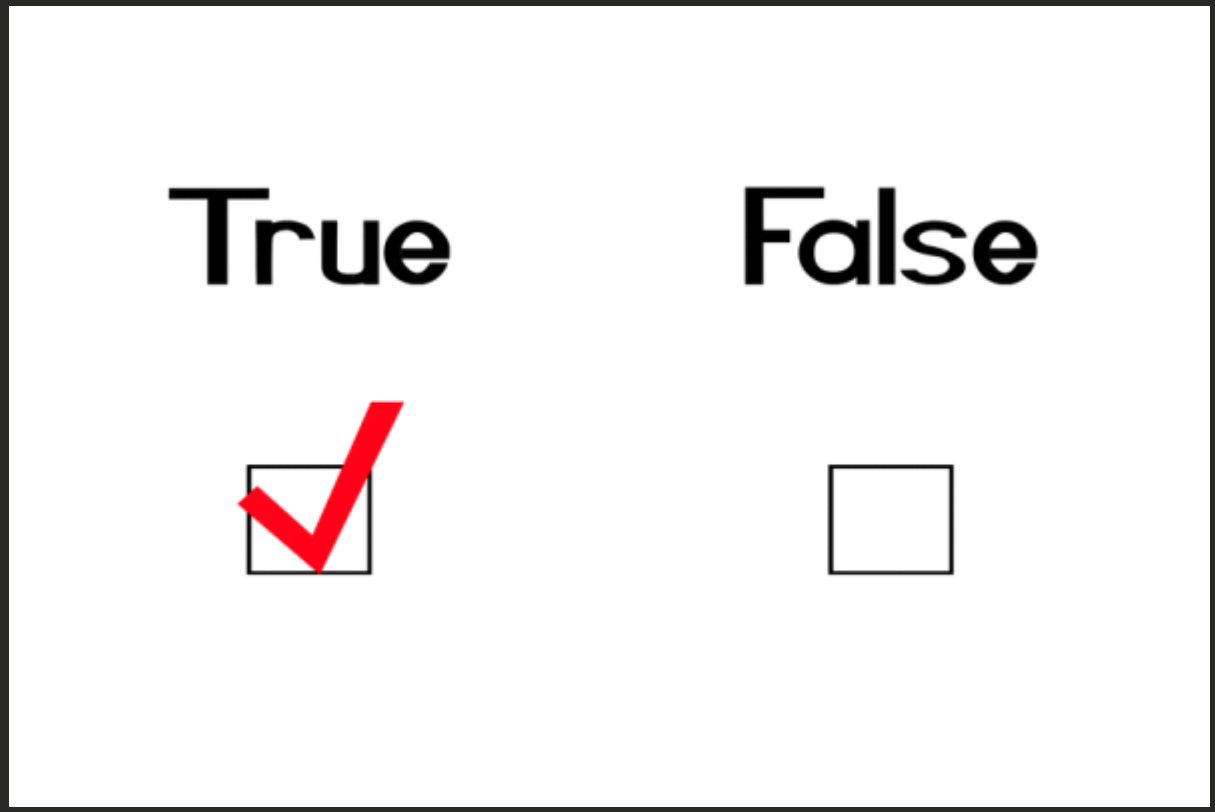
INNER JOIN

```
SELECT *
FROM my_table1 tab1
INNER JOIN my_table2 tab2 ON tab1.id = tab2.id
```

Queries - JOINs



Data types



Data types

- more complex compared to R!
  https://www.postgresql.org/docs/13/datatype.html

name	**size**	**description**	**possible values**	**~R-type**
boolean	1 byte	yes, no	true, false	logical
text	variable	string	all	character
integer	4 bytes	typical choice for integer	-2147483648 to +2147483647	integer
bigint	8 bytes	large-range integer	-9223372036854775808 to +9223372036854775807	integer64
numeric	variable	float, **exact**	up to 131072 digits before the decimal point, 16383 after	numeric
double precision	8 bytes	float	15 decimal digits precision	numeric
date	date	date	4713 BC to 5874897 AD	date
geometry	variable	Geographic information		sfc

Special symbols

Smbol	Description
--	One-line comment
/* Comment this */	Multi-line comment
*	Wildcard, meaning *all*
;	End query

Connect

Connect

Command Line Interface

```
psql -d 'my_db' -c 'SELECT * FROM my_table'
```

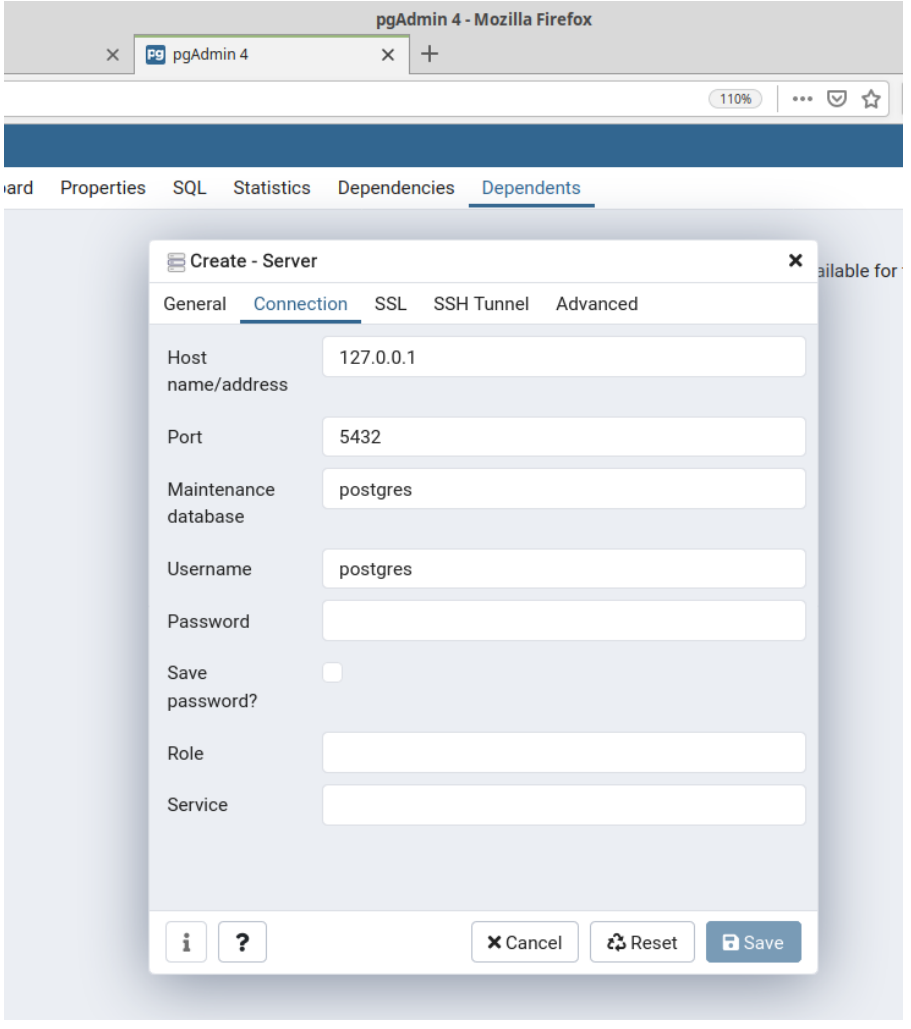
R, python, etc. clients

```
require(RPostgreSQL)
```

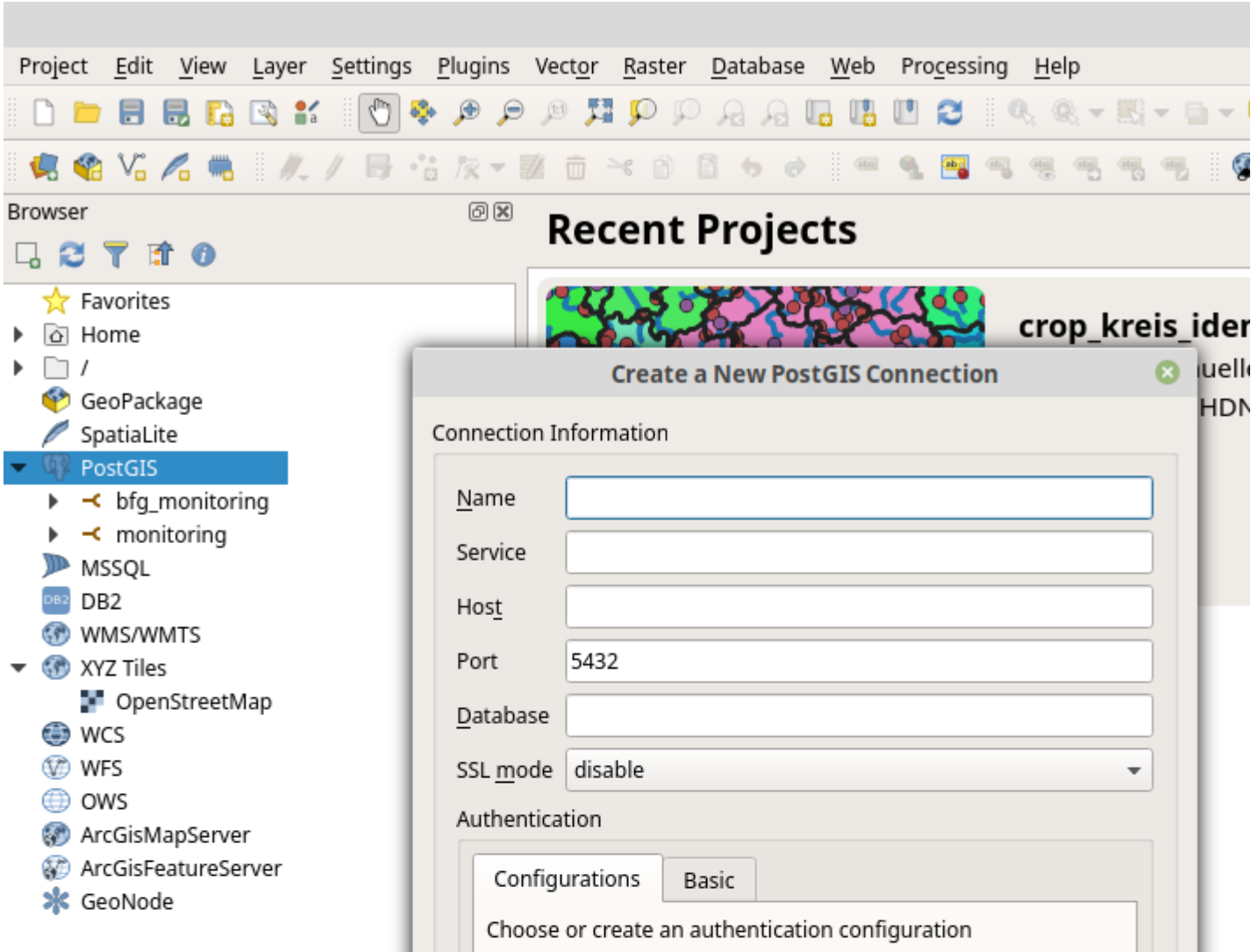
Graphical User Interfaces

- pgAdmin4
- dbeaver
- QGIS

Connect - pgAdmin



Connect - QGIS

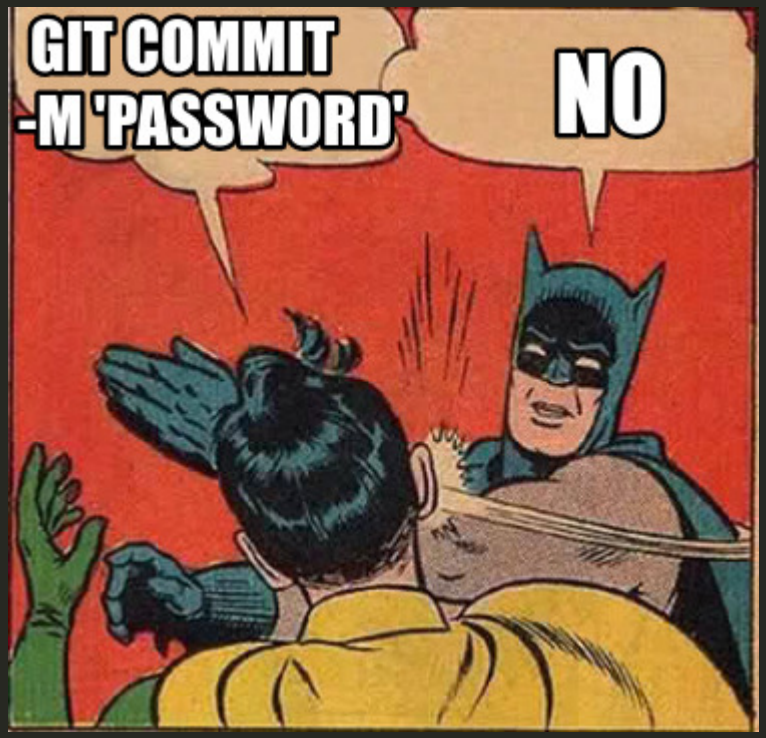


Connect - R

Read

```
require(RPostgreSQL)

drv = dbDriver('PostgreSQL')
con = DBI::dbConnect(drv,
                     host = '127.0.0.1',
                     port = 5432,
                     dbname = 'my_db',
                     user = 'my_user',
                     password = 'my_password')

q_get = "SELECT *
         FROM my_table
         LIMIT 10"

dat = dbGetQuery(con, q) # a data.frame

dbDisconnect(con)
dbUnloadDriver(drv)
```

Careful, credentials!



Connect - R

Separate file for credentials

`~/credentials.R`

```
host = '127.0.0.1'
port = 5432
user = 'my_user'
password = 'my_password'
```

Connect - R

Read

```
source('credentials.R') # credentials

drv = dbDriver('PostgreSQL')
con = DBI::dbConnect(drv,
                     host = host,
                     port = port,
                     dbname = 'mydb',
                     user = user,
                     password = password)

q_get = "SELECT *
         FROM my_table
         LIMIT 10"

dat = dbGetQuery(con, q) # a data.frame

dbDisconnect(con)
dbUnloadDriver(drv)
```

Connect - R

Write

```
source('credentials.R') # credentials

drv = dbDriver('PostgreSQL')
con = DBI::dbConnect(drv,
                     host = host,
                     port = port,
                     dbname = 'mydb',
                     user = user,
                     password = password)

dbWriteTable(con,
             name = c('schema', 'tbl'),
             value = iris, # data set
             overwrite = TRUE,
             row.names = FALSE)

dbDisconnect(con)
dbUnloadDriver(drv)
```

Connect - R

Send

```
source('credentials.R') # credentials

drv = dbDriver('PostgreSQL')
con = DBI::dbConnect(drv,
                     host = host,
                     port = port,
                     dbname = 'mydb',
                     user = user,
                     password = password)

q_send = "ALTER TABLE my_table ADD COLUMN col_new text;"
dbSendQuery(con, q_send)

dbDisconnect(con)
dbUnloadDriver(drv)
```

Differences SQL & R

Differences

PostgreSQL

- database
- data on disk
  - unlimited rows, 250-1600 columns
  - (partly) slow

R

- programming language
- data in memory
  - limited
  - fast

More Queries

Queries - SELECT CASE WHEN

```
SELECT
  CASE
    WHEN column1 < 10
    THEN 'small'
    ELSE 'large'
  END AS column1_categories,
  column2
FROM my_table
LIMIT 1e3
```

R-equivalent

```
data.table::fcase()
dplyr::case_when()
```

Queries - CEATE TABLE

```
CREATE new_table (
  SELECT
    tab1.id PRIMARY KEY,
    tab1.column1
  FROM my_table1 tab1
  INNER JOIN my_table2 tab2 ON tab1.id = tab2.id
)
```

Queries - INSERT INTO

```
INSERT INTO films (code, title, did, date_prod, kind)
    VALUES ('T_601', 'Yojimbo', 106, '1961-06-16', 'Drama');
```

Queries - UPDATE

```
UPDATE my_table SET
  col = 4
WHERE col = 3;
```

Material

PostgreSQL: https://www.postgresql.org/docs/13/index.html

freeCodeCamp: https://www.freecodecamp.org/news/sql-and-databases-full-course

Data Camp: https://www.datacamp.com/courses/introduction-to-sql

Julia Evans (@b0rk): https://wizardzines.com/comics/sql-query-order

YouTube

Slides

- OLAT
- https://andschar.github.io/teaching/PostgreSQL-intro.html

Made with

- https://github.com/rstudio/rmarkdown
- https://github.com/yihui/knitr
- https://github.com/yihui/xaringan

Introduction to Git & GitHub

Thank you for your attention!

Material: https://andschar.github.io/teaching

Andreas Scharmüller

Quantitative Landscape Ecology
iES Landau, Institute for Environmental Sciences
University of Koblenz-Landau
University of Strasbourg

@andschar
andschar@protonmail.com