CLOUDERA

Bridging the Gap between SQL and R

Ian Cook
@ianmcook
ian@cloudera.com
@ianmcook


benjikat/Shutterstock
50
CLOUDERA
@ianmcook

A Relational Model of Data for Large Shared Data Banks

E. F. CODD
IBM Research Laboratory, San Jose, California

Future users of large data banks must be protected from having to know how the data is organized in the machine (the internal representation). A prompting service which supplies such information is not a satisfactory solution. Activities of users at terminals and most application programs should remain unaffected when the internal representation of data is changed and even when some aspects of the external representation are changed. Changes in data representation will often be needed as a result of changes in query, update, and report traffic and natural growth in the types of stored information.

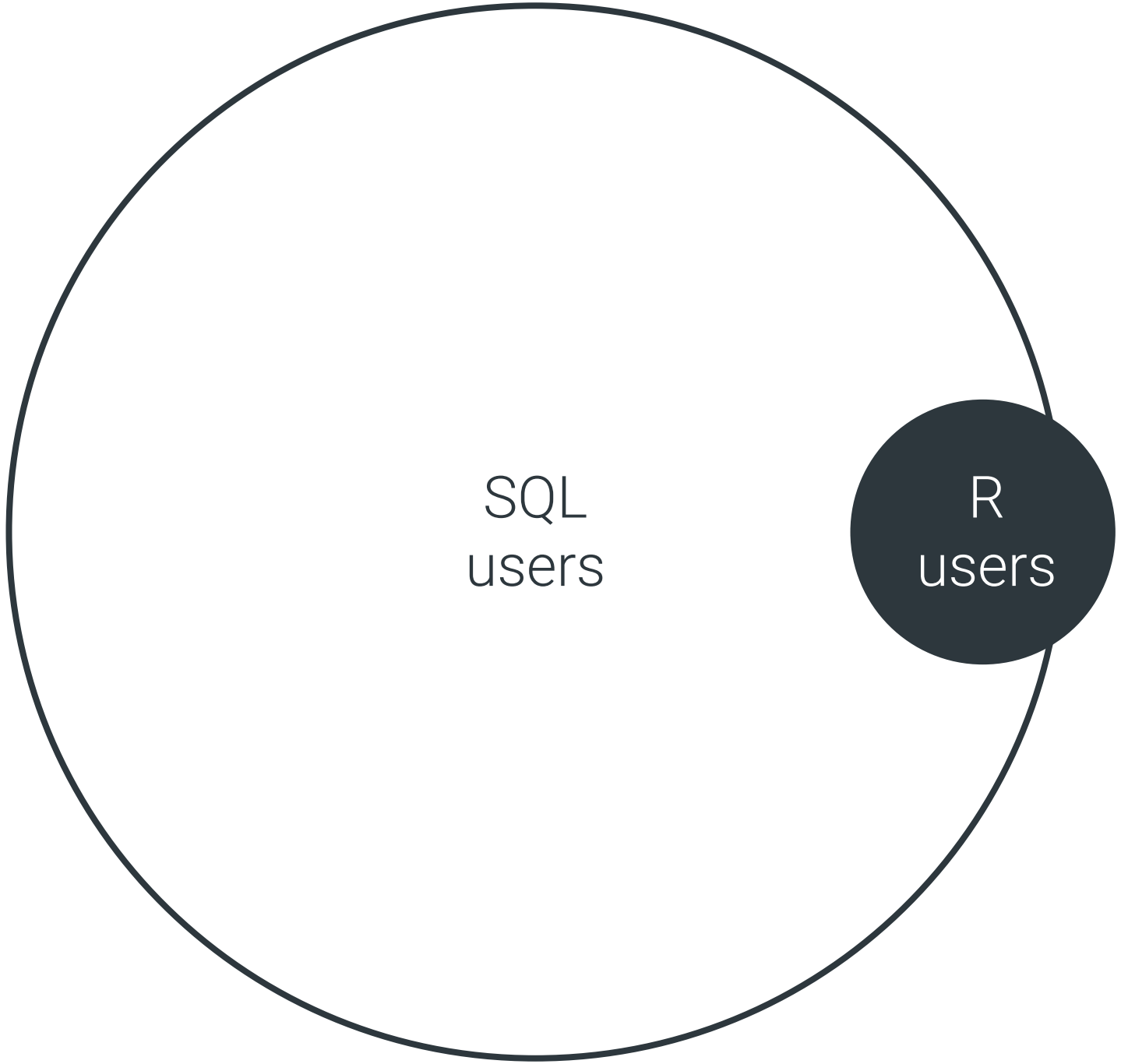
The relational view (or model) of data described in Section 1 appears to be superior in several respects to the graph or network model [3, 4] presently in vogue for non-inferential systems. It provides a means of describing data with its natural structure only—that is, without superimposing any additional structure for machine representation purposes. Accordingly, it provides a basis for a high level data language which will yield maximal independence between programs on the one hand and machine representation and organization of data on the other.

A further advantage of the relational view is that it forms a sound basis for treating derivability, redundancy, and consistency of relations—these are discussed in Section 2. The network model, on the other hand, has spawned a number of confusions, not the least of which is mistaking the derivation of connections for the derivation of relations (see remarks in Section 2 on the "connection trap").

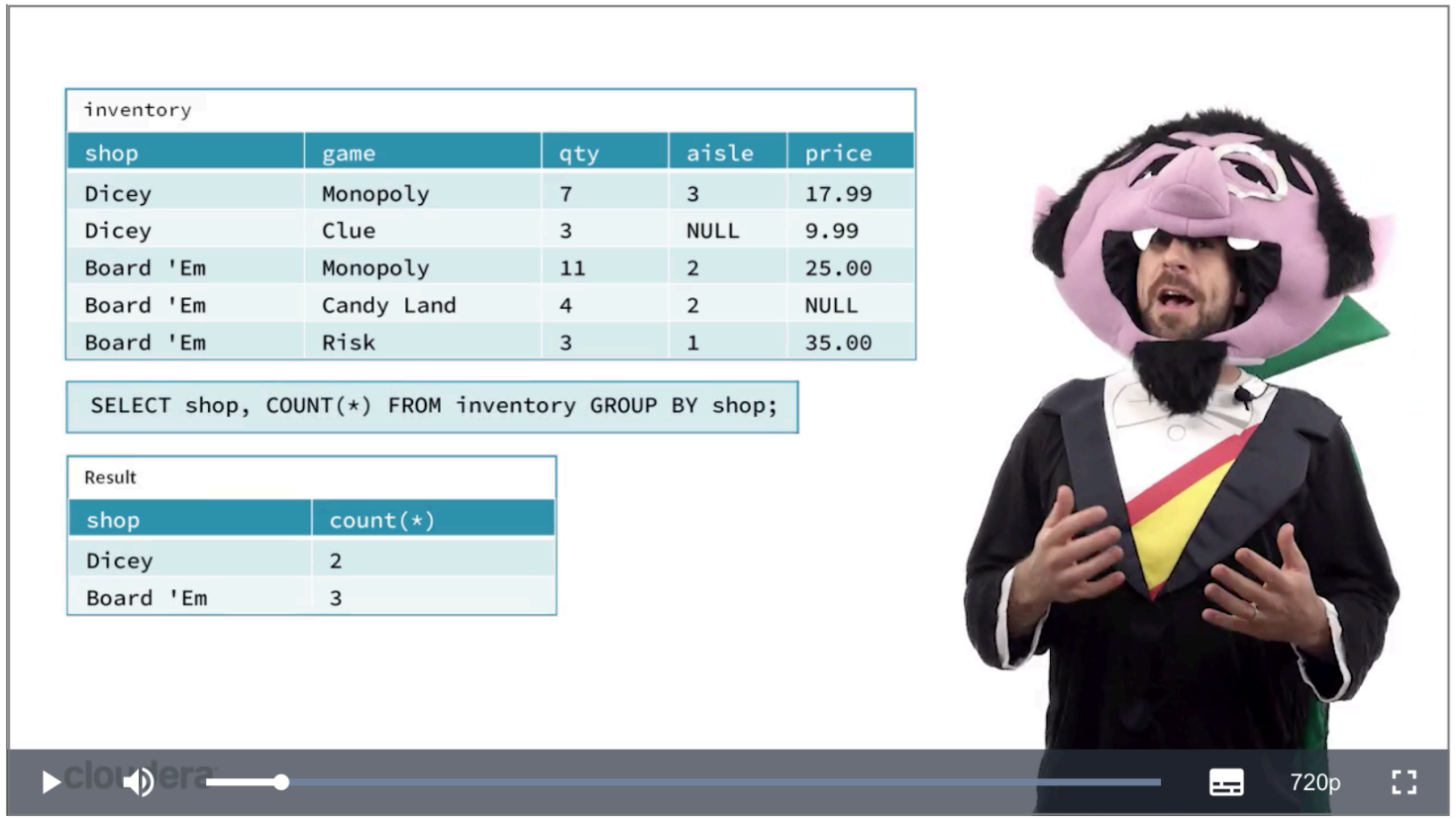
Finally, the relational view permits a clearer evaluation of the scope and logical limitations of present formatted

Do you know SQL?

Did you learn SQL before R?


SQL
users
R
users

coursera Explore What do you want to learn? Log In Join for Free



Analyzing Big Data with SQL

Cloudera

4.9 (99 ratings) | 3.2K Students Enrolled

Enroll for Free

Course 2 of 3 in the Modern Big Data Analysis with SQL Specialization

dplyr ~ SQL

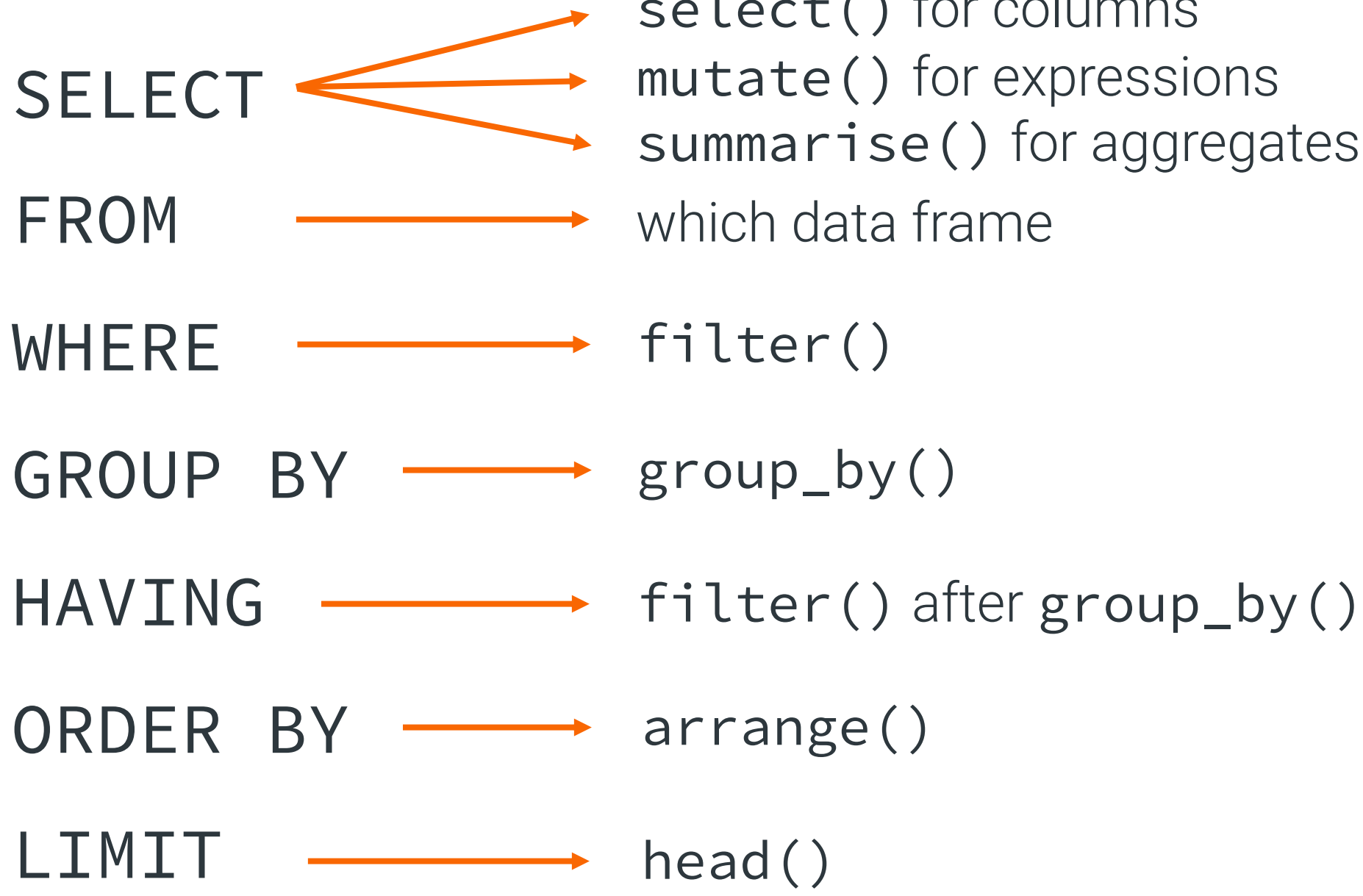
✅ Each variable forms a column

✅ Each observation forms a row

✅ Each type of observational unit forms a table

☝ Tidy data

☝ Third normal form


SELECT
select() for columns
mutate() for expressions
summarise() for aggregates
FROM
which data frame
WHERE
filter()
GROUP BY
group_by()
HAVING
filter() after group_by()
ORDER BY
arrange()
LIMIT
head()

dbplyr

dplyr → SQL

```
> iris_db <- dbplyr::tbl_memdb(iris, name = "iris_db")
> iris_db %>%
    group_by(Species) %>%
    summarise(n = n())

# Source:   lazy query [?? x 2]
# Database: sqlite 3.30.1 [:memory:]
  Species        n
  <chr>      <int>
1 setosa        50
2 versicolor    50
3 virginica     50

> iris_db %>%
    group_by(Species) %>%
    summarise(n = n()) %>%
    show_query()

<SQL>
SELECT `Species`, COUNT() AS `n`
FROM `iris_db`
GROUP BY `Species`
```

SQL → dplyr ?

Package ‘sqldf’

June 28, 2017

Version 0.4-11

Date 2017-06-23

Title Manipulate R Data Frames Using SQL

Author G. Grothendieck <ggrothendieck@gmail.com>

Maintainer G. Grothendieck <ggrothendieck@gmail.com>

Description The sqldf() function is typically passed a single argument which is an SQL select statement where the table names are ordinary R data frame names. sqldf() transparently sets up a database, imports the data frames into that database, performs the SQL select or other statement and returns the result using a heuristic to determine which class to assign to each column of the returned data frame. The sqldf() or read.csv.sql() functions can also be used to read filtered files into R even if the original files are larger than R itself can handle. 'RSQLite', 'RH2', 'RMySQL' and 'RPostgreSQL' backends are supported.

ByteCompile true

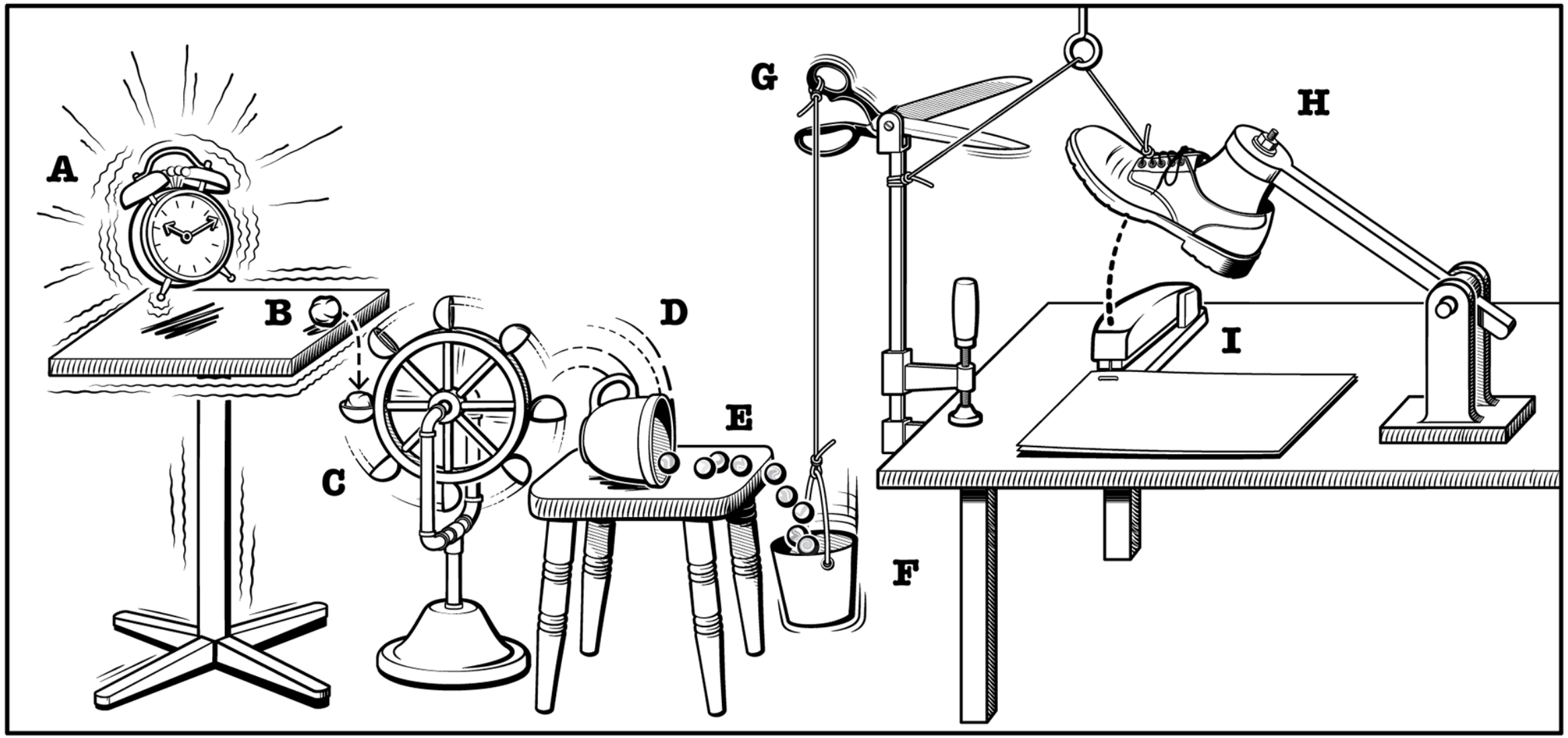
Depends R (>= 3.1.0), gsubfn (>= 0.6), proto, RSQLite

```
> library(sqldf)
> sqldf("SELECT Species, COUNT(*) AS n FROM iris GROUP BY Species")
     Species  n
1     setosa 50
2 versicolor 50
3  virginica 50
```


Martin Lehmann/Shutterstock
CLOUDERA
@ianmcook

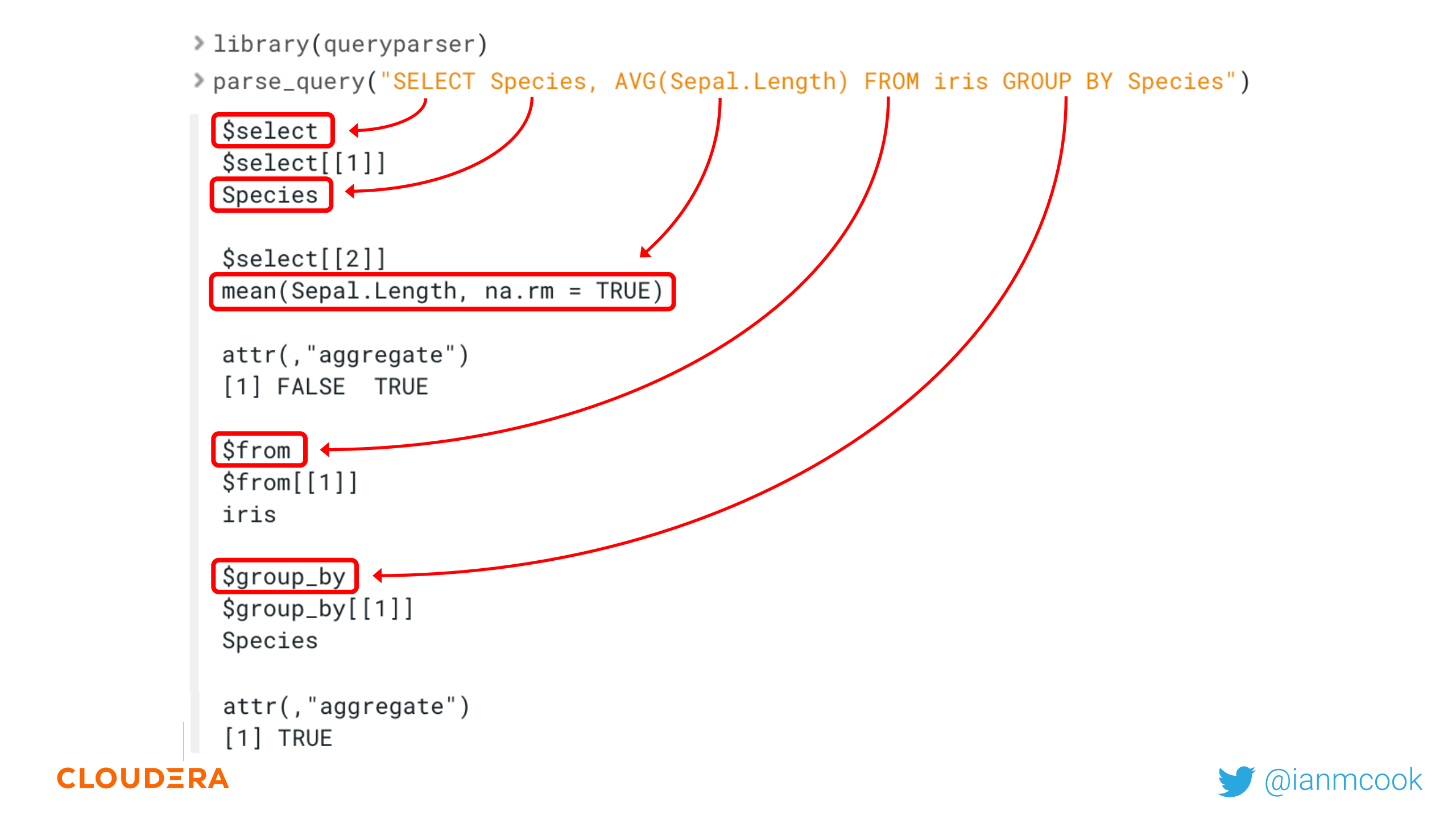
1. Parse SQL query

2. Apply manipulations


A
B
C
D
E
F
G
H
I

Translate SQL queries into unevaluated R expressions

```
> library(queryparser)
> parse_query("SELECT Species, AVG(Sepal.Length) FROM iris GROUP BY Species")
$select
$select[[1]]
Species

$select[[2]]
mean(Sepal.Length, na.rm = TRUE)

attr(,"aggregate")
[1] FALSE  TRUE

$from
$from[[1]]
iris

$group_by
$group_by[[1]]
Species

attr(,"aggregate")
[1] TRUE
```



queryparser

- Has no dependencies; implemented as pure R code
- Includes an option to use tidyverse functions in translated R expressions

```
> parse_query("SELECT COUNT(*) AS n FROM iris", tidyverse = TRUE)

$select
$select$n
dplyr::n()
```

- Performs semantic (not literal) translations

```
> parse_query("SELECT SUM(1) AS n FROM iris", tidyverse = TRUE)

$select
$select$n
dplyr::n() * 1
```

queryparser

- Is secure by default

```
> parse_query("SELECT * FROM iris WHERE system('rm -rf /')")
Error: Unrecognized function or operator: system
```

How queryparser works

1. Split query into clauses
2. Split clauses into strings containing SQL expressions
3. Manipulate strings to change them from SQL expressions to R-ish expressions
4. Parse the R-ish expressions with `str2lang()`
5. Apply a magical combination of quoting, substitution, unquoting, deparsing, string manipulation, and reparsing
6. Manipulate the AST
7. Check for validity
8. Return list

Query R data frames with SQL SELECT statements

```
> library(tidyquery)
> query("SELECT Species, COUNT(*) AS n FROM iris GROUP BY Species")
# A tibble: 3 x 2
  Species        n
  <fct>      <int>
1 setosa        50
2 versicolor    50
3 virginica     50
```

```
> iris %>%
    filter(Species != "setosa") %>%
    query("SELECT Species, COUNT(*) AS n GROUP BY Species") %>%
    arrange(desc(Species))
```

```
# A tibble: 2 x 2
  Species        n
  <fct>      <int>
1 virginica     50
2 versicolor    50
```

How tidyquery works

1. Call `queryparser::parse_query()`
2. Determine which sequence of dplyr verbs to call
3. Use dplyr with rlang to evaluate the expressions in the context of the data frame

Hadley Wickham @hadleywickham · Jan 17

Replying to @ianmcook

I just realised that you should have called tidyquery "rylpbd" 🤣

2 1 6

Current Limitations

queryparser and tidyquery do not support:

- Subqueries
- Unions
- SQL-89-style (implicit) join notation
- Joins of three or more tables
- WITH clause (common table expressions)
- OVER expressions (window or analytic functions)
- Non-ASCII characters in queries

github.com/ianmcook/queryparser/issues

github.com/ianmcook/tidyquery/issues

```
> library(dtplyr)
> iris_dt <- lazy_dt(iris)
> query("SELECT Species, COUNT(*) AS n FROM iris_dt GROUP BY Species")

Source: local data table [?? x 2]
Call:   `_DT1`[, .(n = .N), keyby = .(Species)]

  Species        n
  <fct>      <int>
1 setosa        50
2 versicolor    50
3 virginica     50

# Use as.data.table()/as.data.frame()/as_tibble() to access results
```

```
> show_dplyr("SELECT Species, COUNT(*) AS n FROM iris GROUP BY Species")
```

```
iris %>%
  group_by(Species) %>%
  summarise(n = dplyr::n()) %>%
  ungroup()
```

```
> library(nycflights13)
> show_dplyr(
  "SELECT origin, dest,
     COUNT(*) AS num_flights,
     SUM(seats) AS num_seats
   FROM flights f LEFT JOIN planes p
     ON f.tailnum = p.tailnum
   WHERE distance BETWEEN 300 AND 400
   GROUP BY origin, dest
   HAVING num_flights > 100
   ORDER BY num_seats DESC
   LIMIT 6")
```

```
flights %>%
  left_join(planes, by = "tailnum", suffix = c(".f", ".p"), na_matches = "never") %>%
  rename(f.year = "year.f", p.year = "year.p") %>%
  filter(dplyr::between(distance, 300, 400)) %>%
  group_by(origin, dest) %>%
  filter(dplyr::n() > 100) %>%
  summarise(num_flights = dplyr::n(), num_seats = sum(seats, na.rm = TRUE)) %>%
  ungroup() %>%
  arrange(dplyr::desc(num_seats)) %>%
  head(6)
```

Ian Cook
@ianmcook
ian@cloudera.com
@ianmcook

CLOUDERA