DuckDB - The SQLite for Analytics

Mark Raasveldt & Hannes Mühleisen
Postdoc in Database Architectures group
Before that: PhD student
Before that: Master student

I like the CWI
I like databases
In our group we work on RDBMS

- Relational Database Management Systems

Many of you have probably used a RDBMS

- Taught in schools, used everywhere
RDBMS have been around forever (in CS terms)

1970: Edgar F. Codd presented the relational model

A Relational Model of Data for Large Shared Data Banks

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

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 rela-
1974:

SEQUEL: A STRUCTURED ENGLISH QUERY LANGUAGE

by

Donald D. Chamberlin
Raymond F. Boyce

IBM Research Laboratory
San Jose, California

ABSTRACT: In this paper we present the data manipulation facility for a structured English query language (SEQUEL) which can be used for accessing data in an integrated relational data base. Without resorting to the concepts of bound variables and quantifiers SEQUEL identifies a set of simple operations on tabular structures, which can be shown to be of equivalent power to the first order predicate calculus. A SEQUEL user is presented with a consistent set of keyword English templates which reflect how people use tables to obtain information. Moreover, the SEQUEL user is able to compose these basic templates in a structured manner in order to form more complex queries. SEQUEL is intended as a data base sublanguage for both the professional programmer and the more infrequent data base user.
That’s it!

Since 1974 little has changed:

SQL/relational model are **still** used everywhere

What have database researchers been doing!?
SQL is declarative

- You describe **what** you want to see
- Not **how** to get the data

DBMS **decides** the best way of answering queries
While SQL has not changed, DBMS have!

Database community has been optimizing these systems for **half a century**

Original DBMS were made for computers with **kilobytes** of memory

Modern DBMS are made for computers with **terabytes** of memory (or clusters of computers)
- Data has been getting increasingly hyped/popular
- Main reason: storing/analyzing data is cheap now
  - You can store and analyze gigabytes on standard laptops/desktops
This new trend is called **data science**

Overloaded term

Data science = Statistics on a macbook
Data Scientists work with **data**

*Our field:* 50+ years perfecting RDBMS

**Obviously** they use RDBMS!

... right?
No!

- They use them if forced...
  - Data is in a RDBMS
- Prefer to avoid using them
“If your data fits in memory there is no advantage to putting it in a database: it will only be slower and more frustrating”
- Hadley Wickham
Maybe they just don’t need RDBMS technology

What do data scientists actually use?
Data Science Frameworks and Libraries (multiple answers)

- NumPy: 62%
- Pandas: 51%
- Matplotlib: 46%
- SciPy: 38%
- SciKit-Learn: 31%
- TensorFlow: 25%
- Keras: 15%
- Seaborn: 15%
- NLTK: 13%
- Gensim: 4%
- Theano: 3%
- Other: 5%
- None: 27%

The 10 most downloaded R packages in 2019

<table>
<thead>
<tr>
<th>Package</th>
<th>Downloads through July 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>rlang</td>
<td>6 million</td>
</tr>
<tr>
<td>Rcpp</td>
<td>5.6</td>
</tr>
<tr>
<td>dplyr</td>
<td>5</td>
</tr>
<tr>
<td>tibble</td>
<td>4.9</td>
</tr>
<tr>
<td>ggplot2</td>
<td>4.7</td>
</tr>
<tr>
<td>magrittr</td>
<td>4.7</td>
</tr>
<tr>
<td>glue</td>
<td>4.2</td>
</tr>
<tr>
<td>pillar</td>
<td>4.1</td>
</tr>
<tr>
<td>data.table</td>
<td>3.9</td>
</tr>
<tr>
<td>R6</td>
<td>3.9</td>
</tr>
</tbody>
</table>
What do pandas, NumPy, dplyr, tibble and data.table have in common?

- They perform **database operations**
- Joins, aggregates, basic data transformations, filters, etc.
- Data scientists **need** RDBMS functionality!
Let’s go back to Hadley’s quote

“If your data fits in memory there is no advantage to putting it in a database: it will only be slower and more frustrating”

- Hadley Wickham

Why does he think that?

We have been optimizing them for 50 years!
Let’s run a data science workflow
With and without a database system
And see what happens!
Data Science Workflow

- Load data from CSV file
- Perform preprocessing
  - **ETL**: Extract Transform Load
- Run analysis
Data Science Workflow

Our example:

- Voter data from North Carolina
  - 360MB in CSV format

Preprocessing:

- Remove entries with missing phone number or house number
- Compute correlation between phone number and house number
Data Science Workflow

Pure R

First, install required packages:

```r
install.packages(c("data.table", "dplyr"))
```
Pure R

Then run the analysis

```r
# load data from CSV file
ncvoter <- fread('ncvoter.csv', sep='\t')
# filter entries with missing phone/house number
ncvoter %>% filter(!is.na(phone_num), !is.na(house_num)) -> ncvoter_filtered
# perform the correlation
cor(ncvoter_filtered[['phone_num']], ncvoter_filtered[['house_num']])
[1] -0.002638091
```
PostgreSQL

Install the database server

- Not trivial without package manager!
- Download sources and compile

```
$ export PGDATA=ncvoters
$ initdb
$ pg_ctl -D ncvoterdb -l logfile start
```
PostgreSQL

Next up: install client

```r
install.packages("RPostgres")
```
Data Science Workflow

 PostgreSQL

PostgreSQL

------------------------------------------------------------------------ ANTICONF ERROR ------------------------------------------------------------------------

Configuration failed because libpq was not found. Try installing:

* deb: libpq-dev (Debian, Ubuntu, etc)
* rpm: postgresql-devel (Fedora, EPEL)
* rpm: postgresql8-devel, psstgresql92-devel, postgresql93-devel, or postgresql94-devel (Amazon Linux)
* csw: postgresql_dev (Solaris)
* brew: libpq (OSX)

If libpq is already installed, check that either:

(i) 'pkg-config' is in your PATH AND PKG_CONFIG_PATH contains a libpq.pc file; or
(ii) 'pg_config' is in your PATH.

If neither can detect, you can set INCLUDE_DIR and LIB_DIR manually via:

R CMD INSTALL --configure-vars='INCLUDE_DIR=... LIB_DIR=...'

That didn't work

Need server installed and in PATH to run client
library(RPostgres)

con <- DBI::dbConnect(RPostgres::Postgres(), host='localhost', dbname='postgres')
dbSendQuery(con, "CREATE TABLE ncvoters(...);")
dbSendQuery(con, paste0("COPY ncvoters FROM ", getwd(), "/", "ncvoter.csv" (FORMAT 'csv', DELIMITER '|', QUOTE '"', NULL ''))")
dbSendQuery(con, "DELETE FROM ncvoters WHERE phone_num IS NULL OR house_num IS NULL")
ncvoter_filtered <- dbReadTable(con, "ncvoters")
cor(ncvoter_filtered[['phone_num']], ncvoter_filtered[['house_num']])
Hadley is right

For data science workflows, database systems are frustrating and slow

What are the problems?

How can we fix them?
Problems

- Difficult to install
- Difficult to setup/maintain
- Slow data transfer to/from client
- Poor interface with client application
- External state: cannot copy/paste!
- Client/Server Architecture
- Good for multiple users/separate machine
- Pointless for single-user
- Makes install/setup much more difficult
Installation needs to be easy

- `pip install` for Python
- `install.packages` for R
RDBMS Problems

- Transfer efficiency is crucial
- SQL is not sufficient for data science
- Data scientists use R/Python packages
- Connectors not built for bulk transfer
SQL is a nice language...

But not as interface for clients!

Embedding SQL as strings is unnatural

- No syntax highlighting

Whole class of security vulnerabilities:

- SQL injection
- SQL as a string is not easily composable
- dplyr allows us to chain operations:

```r
ncvoter %>%
  filter(!is.na(phone_num), !is.na(house_num)) %>%
  select(phone_num+1)
```

- Easily inspect intermediates:

```r
ncvoter %>%
  filter(!is.na(phone_num), !is.na(house_num)) %>%
  head
```
△ SQL: temporary tables

```sql
dbSendQuery(con, "CREATE TEMPORARY TABLE ncvoters_filtered AS SELECT * FROM ncvoters WHERE phone_num IS NULL OR house_num IS NULL")
dbGetQuery(con, "SELECT * FROM ncvoters_filtered LIMIT 10")
```
library(data.table)
library(dplyr)

# load data from CSV file
ncvoter <- fread('ncvoter.csv', sep='\t')
# filter entries with missing phone/house number
ncvoter %>% filter(!is.na(phone_num), !is.na(house_num)) -> ncvoter_filtered
# perform the correlation
# cor(ncvoter_filtered[['phone_num']], ncvoter_filtered[['house_num']])
RDBMS Problems

- Cannot copy/paste!

```r
library(RPostgres)

con <- DBI::dbConnect(RPostgres::Postgres(), host='localhost',
dbname='postgres')
dbSendQuery(con, "CREATE TABLE ncvoters(...);")
dbSendQuery(con, paste0("COPY ncvoters FROM " , getwd(), "/",
"ncvoter.csv" (FORMAT 'csv', DELIMITER '|', QUOTE '\"', NULL '')))
dbSendQuery(con, "DELETE FROM ncvoters WHERE phone_num IS NULL OR
house_num IS NULL")
ncvoter_filtered <- dbReadTable(con, "ncvoters")
cor(ncvoter_filtered[["phone_num"]], ncvoter_filtered[["house_num"]])
```
Current RDBMS are not a good fit for data science

Why do we (database researchers) care?
Data science is popular
Popular = money $$
Data scientists are re-inventing RDBMS

- Poorly!

- No optimizers
- Huge intermediates
- No out-of-memory computation
- No parallelism
- No scalability
RDBMS Problems

- RDBMS can solve all of these problems
  - And more!
- Current RDBMS are “slow and frustrating”
- But they don’t need to be!
DuckDB
an Embeddable Analytical RDBMS
Why “Duck” DB?

Hannes used to own a pet duck
DuckDB: The SQLite for Analytics

Core Features

- Simple installation
- Embedded: no server management
- Single file storage format
- Fast analytical processing
- Fast transfer between R/Python and RDBMS
Simple Installation & Usage

- Many lessons learned from SQLite
- Embedded system
- Zero external dependencies
  - All dependencies are inlined

Amalgamation (`duckdb.cpp & duckdb.hpp`)

Integrated into package managers:

- `pip install duckdb`
- `install.packages("duckdb")`
Simple Installation & Usage

- DuckDB does not rely on any external state
  - e.g. separate config files, environment variables

Single-File Storage Format

- Files are easier to handle for users than directories
  - See: Microsoft Office
import duckdb

# open database file
db = duckdb.open('cities.db')

# filter on cities with a high population
fcities = db.table('cities').filter('population > 1000')
# print a snapshot
fcities.show()
# now perform a grouped aggregate
fcities.aggregate('country, sum(population)')

Composable Interface
Composable Interface

```python
import duckdb

df = pandas.read_csv('cities.csv')
# directly scan a pandas dataframe
duckdb.from_df(df).filter('population > 1000')
```
Efficient ETL Workloads

Traditional systems are not optimized for ETL:

- Bulk updates
- Bulk deletions
- Adding/removing columns

E.g. a common query is replacing NULL values:

```
UPDATE tbl
SET val=NULL
WHERE val=-999;
```

This can update the entire column!
Efficient Snapshotting for Undo

ETL workflows are generally trial-and-error

When a mistake is made, user has to start over

Typically involves reloading from CSV file again

This can take a very long time!
Efficient Snapshotting for Undo

- RDBMS supports ROLLBACK/versioning of data
  - Current systems not optimized for many different versions of entire columns
- DuckDB has MVCC that is optimized for bulk updates/deletes to support this use case
DuckDB is free and open-source
Currently in pre-release (v0.1.9)

We have a website: www.duckdb.org
Source Code: https://github.com/cwida/duckdb

Feel free to try it
And send us a bug report if anything breaks!