Databases & SQL

Fundamentals of Computing and Data Display

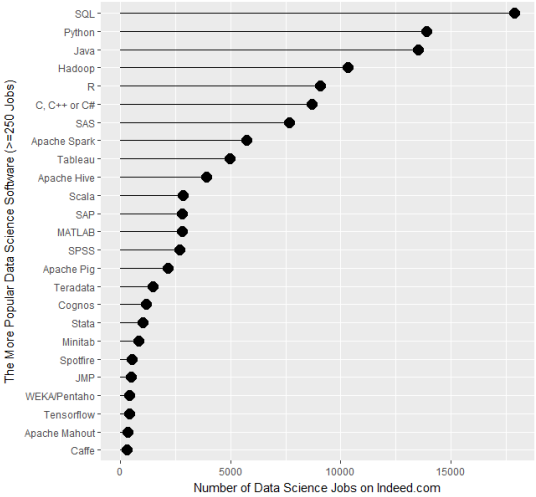
Christoph Kern Ruben Bach

(c.kern, r.bach)@uni-mannheim.de

Outline

Introduction

Figure: Number of data science jobs for popular software packages[1]



[1] http://r4stats.com/articles/popularity/

Introduction

Why using databases?

1. Volume
   - R loads all data (`.Rdata`) into memory
   - Physical limitation: RAM (e.g. 4, 8, 16 gigabytes)
2. Speed
   - Searching, filtering big data inefficient w/o indices
3. Consistency
   - Data might need to be updated, accessed by multiple users
   - Data project might involve transactions between server and client

Types of databases

A database[2]

- "...is a structured collection of data about entities and their relationships"

A database management system (DBMS)[2]

- "...is a software suite designed to safely store and efficiently manage databases, and to assist with the maintenance and discovery of the relationships that database represents"

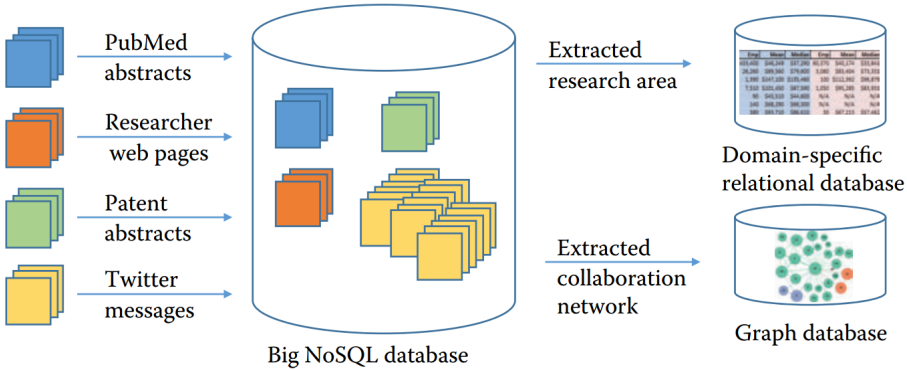
Types of DBMSs

- Relational DBMSs
  - Store large **tabular** data
  - Organize multiple tables linked by keys
  - Communicate via SQL (Structured Query Language)
- NoSQL DBMSs
  - Store (extra-)large amounts of **unstructured** data

[2] Foster et al. 2017

Types of databases

Figure: Combining different DBMS in a research project[3]



[3] Foster et al. 2017

SQL Basics

SQL Basics

Standard SQL

- ANSI standard, several versions (starting with SQL-86)

Some implementations (SQL dialects)

- SQLite
  - `https://sqlite.org/`
- MySQL
  - `https://www.mysql.com/`
- PostgreSQL
  - `https://www.postgresql.org/`
- Proprietary RDBMS
  - Microsoft SQL Server, Oracle

RDB setup

Set up a database

- `CREATE DATABASE dbname;`
- `CREATE TABLE tblname (...);`
  - Requires a list of **definitions**, i.e. name and type per column
  - Allows to specify **constraints** per column
  - Should include the specification of a `PRIMARY KEY` (unique ID)
  - May allow adding indices with `INDEX` or via keys
- `LOAD DATA`, `COPY`, `INSERT`, `UPDATE`
  - Loading to and updating data in a table

```
> CREATE TABLE db.example (
> entity_id integer NOT NULL,
> outcome_date date,
> outcome character varying,
> ybirth numeric);
```

RDB setup

List and select database(s), list and describe tables

- `SHOW DATABASES;`
- `USE dbname;`
- `SHOW TABLES;`
- `DESCRIBE tblname;`

SQL queries

SQL queries

- `SELECT ... FROM`
  - Necessary clauses in every `SELECT` query
  - Select all columns via * or list names
  - Modify column names in query result with `AS`
  - Apply functions with statements preceding `AS`

```
> --A very simple query;
> SELECT *
> FROM example
> LIMIT 500;
> --Another example;
> SELECT COUNT(*) AS n_rows
> FROM example
```

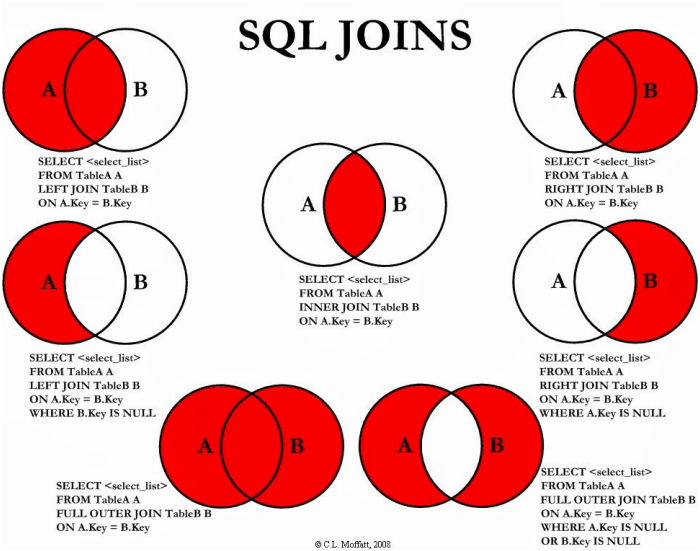
SQL queries

- `WHERE`
  - Filter rows of **original table**
  - Combine conditions with `AND`, `OR`
- `GROUP BY`
  - Aggregate rows by groups
  - Needs aggregation functions after `SELECT`
    - e.g. `COUNT`, `SUM`, `MAX`, `AVG`
- `HAVING`
  - Filter rows of **result set**
  - Same syntax as `WHERE`
- `ORDER BY`
  - Order results based on column(s)

SQL queries

- `JOIN`
  - DBMSs store data on different levels in separate tables
  - Relationships formalized with (foreign) keys
  - Types, syntax of joining varies among SQL flavors
- Types of joins (PostgreSQL)
  - `INNER JOIN`
  - `OUTER JOIN`
  - `LEFT JOIN`
  - `RIGHT JOIN`
  - ...

SQL queries


SQL JOINS
A
B
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
© C.L. Moffatt, 2008

SQL queries

Subqueries

- (Inner) queries can be nested within a (outer) query
- Allows to "stack" multiple operations
- Result set from inner query is passed to outer query
- Dimension of result must match outer operation
  - e.g., WHERE with an equal condition requires a one cell result

```
> --A query within a query;
> SELECT * FROM example
> WHERE ybirth = (SELECT MIN(ybirth)
>                 FROM example
>                 );
```

SQL queries

Order of clauses in a query

1. `SELECT`
2. `FROM`
3. `JOIN`
4. `WHERE`
5. `GROUP BY`
6. `HAVING`
7. `ORDER BY`
8. `LIMIT`

SQL & R

SQL & R

Connecting to a database from within R

- A common DataBase Interface: `DBI`
  - Generic interface ("front-end") for accessing an arbitrary RDBMS
  - Communication with specific RDBMS provided by drivers ("back-end")
- (Some) DBI-compliant driver packages
  - `RSQLite`
  - `RMySQL`
  - `RPostgreSQL`
  - `odbc`

SQL & R

`DBI` workflow

1. `dbDriver()`
   - Create driver for particular RDBMS
2. `dbConnect()`
   - Connect to RDB by providing name, location, log-in credentials
3. `dbGetQuery()`
   - Execute SQL querys
4. `dbDisconnect()`

- Other (useful) functions
  - Separate send and fetch: `dbSendQuery()` and `dbFetch()`
  - List tables in database: `dbListTables()`

SQL & d(b)plyr

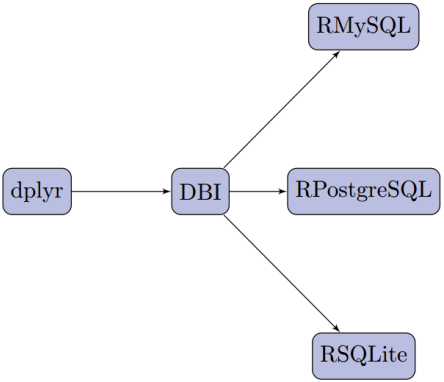
Use `dplyr` on top of a `DBI` connection: `dbplyr`

1. `dbConnect()`
   - Connect to a RDB
2. `tbl()`
   - Map table in database to object in R
3. `dplyr`
   - Use tables as if they were data frames in R!
   - Pull data to a local tibble: `collect()`

```
# Pulling a table into R with dbplyr
> exp_sub <- example %>%
> filter(ybirth == 1984) %>%
> collect() %>%
> mutate(outcome_f = as.factor(outcome))
```

SQL & d(b)plyr

Figure: Dependencies of SQL-related R packages[4]



[4] Baumer et al. 2017

SQL & d(b)plyr

Translating pipelines into SQL

- `show_query()`
  - Shows SQL equivalent of pipeline code
  - Not all R functions supported
- `translate_sql()`
  - Runs translation between R and SQL functions
  - `https://cran.r-project.org/web/packages/dbplyr/vignettes/sql-translation.html`

```
# Translate basic expressions
> translate_sql(mean(ybirth))
<SQL> avg("ybirth") OVER ()
```

SQL & d(b)plyr

Table: Comparison of SQL and dplyr syntax[5]

	SQL	R (dplyr)

Filter by rows & columns	`SELECT col1, col2`
	`FROM a`
	`WHERE col3 = 'x'`
		`a %>%`
		`filter(col3 == "x") %>%`
		`select(col1, col2)`

Aggregate by rows	`SELECT id, sum(col1)`
	`FROM a`
	`GROUP BY id`
		`a %>%`
		`group_by(id) %>%`
		`summarize(sum(col1))`

Combine two tables	`SELECT *`
	`FROM a`
	`JOIN b ON a.id = b.id`
		`a %>%`
		`inner_join(b, by =`
		`c("id" = "id"))`

[5] Baumer et al. 2017

Resources

- Online Learning
  - `https://www.datacamp.com/courses/intro-to-sql-for-data-science`
  - `https://hackr.io/tutorials/learn-sql`
- DBs & Rstudio
  - `https://db.rstudio.com/`
- Cheatsheet
  - `http://files.zeroturnaround.com/pdf/zt_sql_cheat_sheet.pdf`

References

Baumer, B. S., Kaplan, D. T., Horton, N. J. (2017). *Modern Data Science with R*. Boca Raton, FL: Chapman & Hall/CRC Press.

Foster, I., Ghani, R., Jarmin, R. S., Kreuter, F., and Lane, J. (Eds.). (2017). *Big Data and Social Science: A Practical Guide to Methods and Tools*. Boca Raton, FL: CRC Press Taylor & Francis Group.