Modular Queries and Unit Testing

Georgios Gousios
Deparment of Software Technology
Delft University of Technology
g.gousios@tudelft.nl
http://www.gousios.org/
@gousiosg

and

Diomidis Spinellis
Department of Management Science and Technology
Athens University of Economics and Business
dds@aueb.gr
http://www.dmst.aueb.gr/dds
@CoolSWEng

2017-05-23

Real-world queries

- Complex
- Expensive
- Example: leader contribution

Project selection

- Examine data for a period of exactly one year
- Eliminate projects that started their life within the examined period
- Select projects having at least ten watchers and at least 20 commits made by at least seven different people

Project leaders

- Member's contribution:
- Commits
- Commit comments
- Issue events
- Issue comments
- Pull request comments
- Leader: the individual with 20% more project contributions than the one with second highest

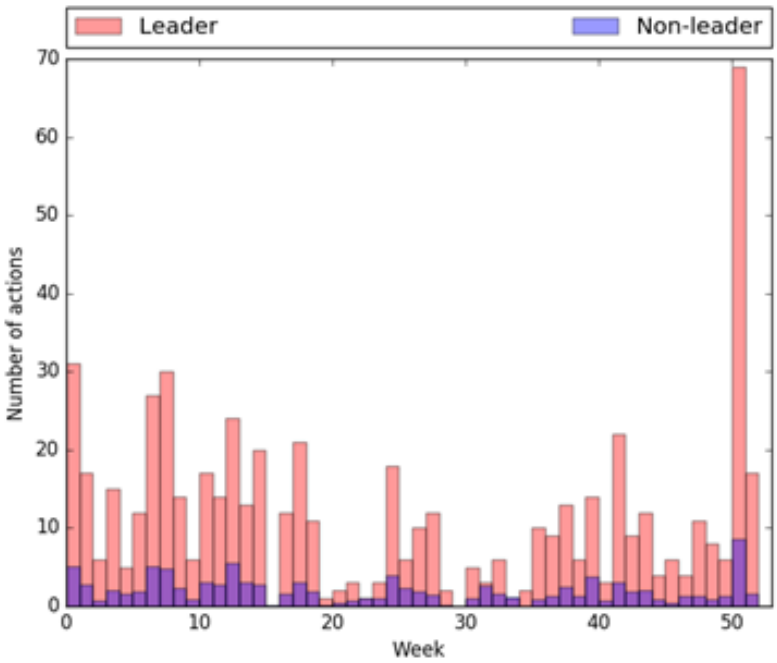

Figure 1: Leader vs member contributions

Real-world queries can be complex

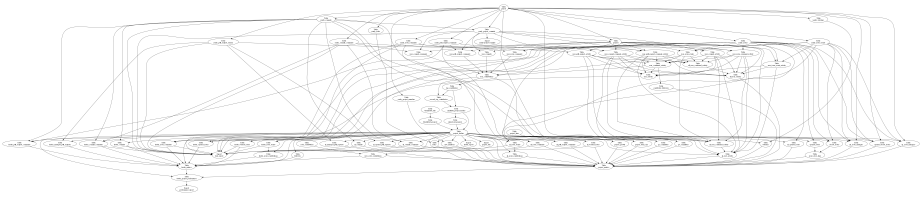

Figure 2: SQL query dependency graph

(2600 lines of SQL)

Real-world queries can be expensive

```
May 14 14:37 yearly_commits
May 14 16:34 yearly_project_commits
May 15 00:02 projects
May 15 00:04 user_commit_activity
May 15 01:57 yearly_commit_comments
[22 lines omitted]
May 15 03:28 commits_with_comments
May 15 04:50 contributor_followers
[42 lines omitted]
May 15 08:49 q1_committers
May 15 10:02 q1_issue_managers
[7 lines omitted]
May 15 10:51 nl_issues_leader_comments
```

What can we do?

- Modularize
- Incremental construction
- Unit testing
- Execution checkpoints
- Reuse

Approaches

- Oracle/DB2/PostgreSQL/… materialized views
- Justin Swanhart's MySQL Flexviews
- Make-based simple-rolap

Example task

1. Choose repositories that have forks (*A*)
2. from *A*, exclude repos that never received a PR (*B*)
3. from *B*, exclude repos that were inactive *recently* (*C*)
4. from *C*, exclude repos that have fewer than 100 stars (*D*)
5. Obtain number of files and lines of code for each project

We could then apply further criteria (e.g. programming language, build system, number of contributors, etc).

Environment setup

- Clone and install github.com/dspinellis/rdbunit
- Clone github.com/dspinellis/simple-rolap

Project setup

Create a `Makefile` with the following contents:

```
export RDBMS?=sqlite
export MAINDB?=rxjs-ghtorrent
export ROLAPDB?=stratsel
export DEPENDENCIES=rxjs-ghtorrent.db

include ../../Makefile

rxjs-ghtorrent.db:
        wget https://github.com/ghtorrent/tutorial/raw/master/rxjs-ghtorrent.db
```

Repositories with forks

Create a file `forked_projects.sql`

```
-- Projects that have been forked

create table stratsel.forked_projects AS
  select distinct forked_from as id from projects;
```

Run it

```
$ make
rm -f ./.depend
sh ../..//mkdep.sh >./.depend
```

```
mkdir -p tables
sh ../..//run_sql.sh forked_projects.sql >tables/forked_projects
```

Run it again

```
$ make
make: Nothing to be done for 'all'.
```

Yes, but is it correct?

Create a file `forked_projects.rdbu`

```
BEGIN SETUP

projects:
id      forked_from
1       15
2       15
3       10
4       NULL

END

INCLUDE CREATE forked_projects.sql

BEGIN RESULT
stratsel.forked_projects:
id
15
10
END
```

Run the tests

```
$ make test
../..//run_test.sh
not ok 1 - forked_projects.rdbu: test_stratsel.forked_projects
1..1
```

Houston, we have a problem!

Step-by–step debugging

```
$ rdbunit --database=sqlite forked_projects.rdbu >script.sql
```

```
$ sqlite3
SQLite version 3.8.7.1 2014-10-29 13:59:56
sqlite> .read script.sql
not ok 1 - forked_projects.rdbu: test_stratsel.forked_projects
1..1

sqlite> select * from test_stratsel.forked_projects;
15
10

sqlite> select count(*) from test_stratsel.forked_projects;
3
```

Correct the error

```
-- Projects that have been forked

create table stratsel.forked_projects AS
  select distinct forked_from as id from projects
  where forked_from is not null;
```

Test again

```
$ make test
rm -f ./.depend
sh ../..//mkdep.sh >./.depend
../..//run_test.sh
ok 1 - forked_projects.rdbu: test_stratsel.forked_projects
1..1
```

Bingo!

Exclude projects with no PRs

Create a file `pr_projects.sql`

```
-- Projects that have been forked and have a PR

create table stratsel.pr_projects AS
  select distinct forked_projects.id from stratsel.forked_projects
  inner join issues on issues.repo_id = forked_projects.id;
```

Run it (incrementally)

```
$ make
rm -f ./.depend
sh ../..//mkdep.sh >./.depend
mkdir -p tables
sh ../..//run_sql.sh pr_projects.sql >tables/pr_projects
```

Test a little

Create a file `pr_projects.rdbu`

```
# Projects that have at least one PR associated with them

BEGIN SETUP

stratsel.forked_projects:
id
1
2
3
4

issues:
id      repo_id
15      1
16      1
17      4
END

INCLUDE CREATE pr_projects.sql

BEGIN RESULT
stratsel.pr_projects:
id
1
4
END
```

Run tests

```
$ make test
../..//run_test.sh
ok 1 - forked_projects.rdbu: test_stratsel.forked_projects
1..1
```

```
ok 1 - pr_projects.rdbu: test_stratsel.pr_projects
1..1
```

Exclude projects with no recent commits

Create a file `recent_commit_projects.sql`

```
-- Projects with recent commits

create table stratsel.pr_projects AS
  select distinct pr_projects.id
  from stratsel.pr_projects
  left join commits
  on commits.project_id = pr_projects.id
  where created_at > '2017-01-01';
```

Test a little (really now?)

Create a file `recent_commit_projects.rdbu`

```
# Projects that have a recent commit associated with them

BEGIN SETUP
stratsel.pr_projects:
id
1
2
3
4

commits:
id      project_id      created_at
15      1               '2017-05-12'
16      1               '2010-01-01'
16      2               '2017-01-02'
16      2               '2017-01-03'
17      4               '1970-01-01'
END

INCLUDE CREATE recent_commit_projects.sql

BEGIN RESULT
stratsel.recent_commit_projects:
id
1
```

```
2
END
```

Run tests

```
$ make test
../..//run_test.sh
ok 1 - forked_projects.rdbu: test_stratsel.forked_projects
1..1
ok 1 - pr_projects.rdbu: test_stratsel.pr_projects
1..1
Error: near line 25: table pr_projects already exists
Error: near line 38: no such table: test_stratsel.recent_commit_projects
1..1
../../Makefile:79: recipe for target 'test' failed
make: *** [test] Error 1
```

???

Correct table name in query

```
-- Projects with recent commits

create table stratsel.recent_commit_projects AS
  select distinct pr_projects.id
  from stratsel.pr_projects
  left join commits
  on commits.project_id = pr_projects.id
  where created_at > '2017-01-01';
```

Run tests again

```
$ make test
rm -f ./.depend
sh ../..//mkdep.sh >./.depend
../..//run_test.sh
ok 1 - forked_projects.rdbu: test_stratsel.forked_projects
1..1
ok 1 - pr_projects.rdbu: test_stratsel.pr_projects
1..1
ok 1 - recent_commit_projects.rdbu: test_stratsel.recent_commit_projects
1..1
```

Incremental build

```
$ make
mkdir -p tables
sh ../..//run_sql.sh recent_commit_projects.sql >tables/recent_commit_projects
```

An aha moment

- PR means “pull request” not “problem report”
- Rewrite `pr_projects.sql`

```
-- Projects that have been forked and have a PR

create table stratsel.pr_projects AS
  select distinct forked_projects.id from stratsel.forked_projects
 inner join pull_requests on pull_requests.base_repo_id = forked_projects.id;
```

Run make again

Notice that only dependent tables get built

```
$ make
mkdir -p tables
sh ../..//run_sql.sh pr_projects.sql >tables/pr_projects
mkdir -p tables
sh ../..//run_sql.sh recent_commit_projects.sql >tables/recent_commit_projects
```

Exclude projects with no recent issues

Test and run

```
$ make test
rm -f ./.depend
sh ../..//mkdep.sh >./.depend
../..//run_test.sh
ok 1 - forked_projects.rdbu: test_stratsel.forked_projects
1..1
ok 1 - pr_projects.rdbu: test_stratsel.pr_projects
1..1
ok 1 - recent_commit_projects.rdbu: test_stratsel.recent_commit_projects
1..1
ok 1 - recent_issue_projects.rdbu: test_stratsel.recent_issue_projects
1..1
$ make
mkdir -p tables
```

```
sh ../..//run_sql.sh recent_issue_projects.sql >tables/recent_issue_projects
```

Count number of project stars

Create a file `project_stars.sql`

```
-- Projects with recent issues and the number of stars

create table stratsel.project_stars AS
  select recent_issue_projects.id as id, count(*) as stars
  from stratsel.recent_issue_projects
  left join watchers
  on watchers.repo_id = recent_issue_projects.id
  group by recent_issue_projects.id;
```

Corresponding test

Create a file `project_stars.rdbu`

```
# Projects with recent issues and the number of stars

BEGIN SETUP
stratsel.recent_issue_projects:
id
1
2
3

watchers:
repo_id
1
1
2
END

INCLUDE CREATE project_stars.sql

BEGIN RESULT
stratsel.project_stars:
id      stars
1       2
2       1
3       0
END
```

Run test

```
../..//run_test.sh
ok 1 - forked_projects.rdbu: test_stratsel.forked_projects
1..1
ok 1 - pr_projects.rdbu: test_stratsel.pr_projects
1..1
not ok 1 - project_stars.rdbu: test_stratsel.project_stars
1..1
ok 1 - recent_commit_projects.rdbu: test_stratsel.recent_commit_projects
1..1
ok 1 - recent_issue_projects.rdbu: test_stratsel.recent_issue_projects
1..1
```

Fix query

Count only non-null rows

```
-- Projects with recent issues and the number of stars

create table stratsel.project_stars AS
 select recent_issue_projects.id as id, count(watchers.repo_id) as stars
  from stratsel.recent_issue_projects
  left join watchers
  on watchers.repo_id = recent_issue_projects.id
  group by recent_issue_projects.id;
```

Run test again

```
$ make test
rm -f ./.depend
sh ../..//mkdep.sh >./.depend
../..//run_test.sh
ok 1 - forked_projects.rdbu: test_stratsel.forked_projects
1..1
ok 1 - pr_projects.rdbu: test_stratsel.pr_projects
1..1
ok 1 - project_stars.rdbu: test_stratsel.project_stars
1..1
ok 1 - recent_commit_projects.rdbu: test_stratsel.recent_commit_projects
1..1
ok 1 - recent_issue_projects.rdbu: test_stratsel.recent_issue_projects
1..1
```

Add and test popular projects

```
$ make test
../..//run_test.sh
ok 1 - forked_projects.rdbu: test_stratsel.forked_projects
1..1
ok 1 - popular_projects.rdbu: test_stratsel.popular_projects
1..1
ok 1 - pr_projects.rdbu: test_stratsel.pr_projects
1..1
ok 1 - project_stars.rdbu: test_stratsel.project_stars
1..1
ok 1 - recent_commit_projects.rdbu: test_stratsel.recent_commit_projects
1..1
ok 1 - recent_issue_projects.rdbu: test_stratsel.recent_issue_projects
1..1
$ make
mkdir -p tables
sh ../..//run_sql.sh project_stars.sql >tables/project_stars
mkdir -p tables
sh ../..//run_sql.sh popular_projects.sql >tables/popular_projects
```

Where do we stand?

```
$ make graph.png
../..//dep2dot.sed .depend >graph.dot
dot -Tpng graph.dot -o graph.png
```

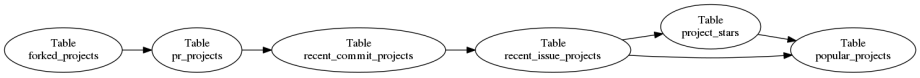

Figure 3: SQL query simple dependency graph

Obtain repo data

- Write out repository URLs
- Create script to analyze repos
- Import back results for further analysis

Repository URLs

Create a file `project_urls.sql`

```sql
-- URLs of popular projects
select projects.id, 'https://github.com/' || substr(url, 30) as url
from stratsel.popular_projects
left join projects
on projects.id = popular_projects.id;
```

Unit test

- Use INCLUDE SELECT
- No table name in result

```
# Projects that have a recent commit associated with them

BEGIN SETUP
stratsel.popular_projects:
id
1
2

projects:
id      url
1       'https://api.github.com/repos/foo'
2       'https://api.github.com/repos/bar'
3       'https://api.github.com/repos/foobar'
END

INCLUDE SELECT project_urls.sql

BEGIN RESULT
id      url
1       'https://github.com/foo'
2       'https://github.com/bar'
END
```

Run tests

```
$ make test
rm -f ./.depend
sh ../..//mkdep.sh >./.depend
../..//run_test.sh
ok 1 - forked_projects.rdbu: test_stratsel.forked_projects
1..1
ok 1 - popular_projects.rdbu: test_stratsel.popular_projects
1..1
ok 1 - pr_projects.rdbu: test_stratsel.pr_projects
```

```
1..1
ok 1 - project_stars.rdbu: test_stratsel.project_stars
1..1
ok 1 - project_urls.rdbu: test_select_result
1..1
ok 1 - recent_commit_projects.rdbu: test_stratsel.recent_commit_projects
1..1
ok 1 - recent_issue_projects.rdbu: test_stratsel.recent_issue_projects
1..1
```

Run queries

- Result is in `reports`
- Named after the query

```
$ make
rm -f ./.depend
sh ../..//mkdep.sh >./.depend
mkdir -p reports
sh ../..//run_sql.sh project_urls.sql >reports/project_urls.txt
$ cat reports/project_urls.txt
1|https://github.com/ReactiveX/rxjs
```

Script to analyze repos

```
#!/bin/sh
# Create a CSV file with files and lines per project

set -e

mkdir -p clones data
cd clones

file_list()
{
  git ls-tree --full-tree -r --name-only "$@" HEAD
}

while IFS=\| read id url ; do
  git clone --bare "$url" $id
  cd $id
  nfiles=$(file_list | wc -l)
  nlines=$(file_list -z |
    xargs -0 -I '{}' git show 'HEAD:{}' |
    wc -l)
```

```
  echo "$id,$nfiles,$nlines"
  cd ..
done <../reports/project_urls.txt >../data/metrics.csv
```

Import data (SQLite)

```
Create file `project_metrics.sql`
create table stratsel.project_metrics (id integer,
  files integer, lines integer);
.separator ","
.import data/metrics.csv stratsel.project_metrics
```

Import data (MySQL)

```
Create file `project_metrics.sql`
create table stratsel.project_metrics (id integer,
  files integer, lines integer);
LOAD DATA LOCAL INFILE 'data/metrics.csv'
INTO TABLE stratsel.project_metrics
FIELDS TERMINATED BY ',';
```

Add dependencies

```
tables/project_metrics: data/metrics.csv

data/metrics.csv: reports/project_urls.txt project_metrics.sh
        sh project_metrics.sh
```

Run make

```
$ make
sh project_metrics.sh
Cloning into bare repository '1'...
remote: Counting objects: 30680, done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 30680 (delta 2), reused 2 (delta 2), pack-reused 30671
Receiving objects: 100% (30680/30680), 73.95 MiB | 10.29 MiB/s, done.
Resolving deltas: 100% (23964/23964), done.
Checking connectivity... done.
mkdir -p tables
sh ../..//run_sql.sh project_metrics.sql >tables/project_metrics
```

Other goodies

- `make tags`
- `make sorted-dependencies`
- `make clean`

Takeaways

- Split analysis into many small queries
- Use simple-rolap to automate process
- *Errare humanum est*
- Code a little, test a little
- Use rdbunit to express tests
- Write additional processing as scripts
- Use *make* dependencies to tie everything together

Thank you!

- The leader selection case was done in collaboration with Niki Panteli and Lefteris Angelis.
- The research described has been partially carried out as part of the CROSSMINER Project, which has received funding from the European Union's Horizon 2020 Research and Innovation Programme under grant agreement No. 732223.