VE 482 Lab 4 Report

519370910084 Lan Wang

Database

Database system installation

- **What are the most common database systems?**

  Oracle, MySQL, Microsoft SQL Server, PostgreSQL, MongoDB

- **Briefly list the pros and cons of the three most common ones.**
  - Oracle

    pros:

    - Advanced and powerful

    cons:

    - Expensive
    - Need better hardware to support it
  - MySQL

    pros:

    - Free
    - Lots of functionality
    - Can work with other databases

    cons:

    - Lack of quality support
    - Lack of some common features of other database systems
  - Microsoft SQL Server

    pros:

    - Fast and stable
    - Reduce resources use by adjusting performance levels
    - Can access visualizations on mobile
    - Can work with other Microsoft products

    cons:

    - Expensive
    - May gobble resources
    - May face problems when using SSIS to import files
- Create an empty SQLite database.

  ```
  sqlite3 l4.db
  ```

- Use the SQLite shell to prepare two empty tables for each of your .csv file.

```
CREATE TABLE db
(
    hash TEXT NOT NULL,
    name TEXT NOT NULL,
    comment TEXT NOT NULL
);
CREATE TABLE time_stamp
(
    hash TEXT NOT NULL,
    name TEXT NOT NULL,
    dates TEXT,
    tstamp INT
);
```

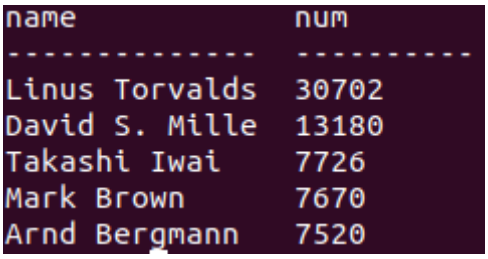
- Import each .csv file in its corresponding SQLite table.

```
.separator "|"
.import db.psv db
.import timestamp.psv time_stamp
```

Database queries

- Who are the top five contributors to the Linux kernel since the beginning?

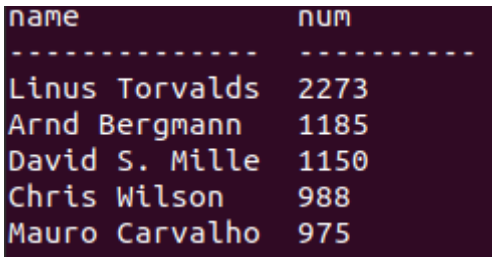
```
SELECT name, COUNT(hash) AS num FROM time_stamp GROUP BY name ORDER BY
COUNT(hash) DESC LIMIT 5;
```



- Who are the top five contributors to the Linux kernel for each year over the past five years?

  2016:

```
SELECT name, COUNT(*) AS num FROM time_stamp WHERE dates BETWEEN '2016-01-
01' AND '2016-12-31' GROUP BY name ORDER BY num DESC LIMIT 5;
```



  2017:

```
SELECT name, COUNT(*) AS num FROM time_stamp WHERE dates BETWEEN '2017-01-
01' AND '2017-12-31' GROUP BY name ORDER BY num DESC LIMIT 5;
```

```
name            num
--------------  ----------
Linus Torvalds  2288
David S. Mille  1420
Arnd Bergmann   1123
Chris Wilson    1028
Arvind Yadav    827
```

2018:

```
SELECT name, COUNT(*) AS num FROM time_stamp WHERE dates BETWEEN '2018-01-
01' AND '2018-12-31' GROUP BY name ORDER BY num DESC LIMIT 5;
```

```
name            num
--------------  ----------
Linus Torvalds  2163
David S. Mille  1405
Arnd Bergmann   919
Christoph Hell  818
Colin Ian King  798
```

2019:

```
SELECT name, COUNT(*) AS num FROM time_stamp WHERE dates BETWEEN '2019-01-
01' AND '2019-12-31' GROUP BY name ORDER BY num DESC LIMIT 5;
```

```
name            num
--------------  ----------
Linus Torvalds  2380
David S. Mille  1205
Chris Wilson    1170
YueHaibing      929
Christoph Hell  911
```

2020:

```
SELECT name, COUNT(*) AS num FROM time_stamp WHERE dates BETWEEN '2020-01-
01' AND '2020-12-31' GROUP BY name ORDER BY num DESC LIMIT 5;
```

```
name            num
--------------  ----------
Linus Torvalds  1886
David S. Mille  923
Christoph Hell  806
Mauro Carvalho  770
Chris Wilson    644
```

No commits in 2021.

- What is the most common “commit subject”?

```
SELECT comment, COUNT (*) AS num FROM db GROUP BY comment ORDER BY num DESC
LIMIT 1;
```

```
comment                                                    num
---------------------------------------------------------  ----------
Merge git://git.kernel.org/pub/scm/linux/kernel/git/davem/net  670
```

- On which day is the number of commits the highest?

```
SELECT DATE(dates) AS date, COUNT(*) AS num FROM time_stamp GROUP BY date
ORDER BY num DESC LIMIT 1;
```

```
date        num
----------  ----------
2008-01-30  1031
```

- Determine the average time between two commits for the five main contributor.

```
SELECT
    (SELECT MAX(tstamp) FROM time_stamp WHERE name='Linus Torvalds')-
(SELECT MIN(tstamp) FROM time_stamp WHERE name='Linus Torvalds') AS 'max-
min',
    COUNT(tstamp) AS num
FROM time_stamp WHERE name='Linus Torvalds';
```

Then for Linus Torvalds:

```
max-min     num
----------  ----------
487552654   30702
```

$t = (max - min)/(num - 1) = 15880.68$ s

For other people, just replace the name, and we can get:

David S. Miller:

$t = 487045012/13179 = 35092.23$ s

Takashi Iwai:

$t = 489001082/7725 = 63301.11$ s

Mark Brown:

$t = 459628018/7669 = 59933.24$ s

Arnd Bergmann:

$t = 479764856/7519 = 63807.00$ s

Debugging

1. **How to enable built-in debugging in `gcc`?**

```
gcc -g
```

2. **What is the meaning of GDB?**

   GNU project debugger

3. **Compile the master branch of you `mumsh` with debugging enabled.**

   In `CMakeLists.txt`, add `-g` to `CMAKE_C_FLAGS`:

```
set(CMAKE_C_FLAGS "-g -Wall -Wextra -Werror -pedantic -Wno-unused-result")
```

Basic GDB usage

1. **Find the homepage of the GDB project.**

   https://www.gnu.org/software/gdb/

2. **What languages are supported by GDB?**

   - Ada
   - Assembly
   - C
   - C++
   - D
   - Fortran
   - Go
   - Objective-C
   - OpenCL
   - Modula-2
   - Pascal
   - Rust

3. **What are the following GDB commands doing:**

   • `backtrace`: Print the backtrace of the entire stack. From current frame -> its caller -> trace up the stack.
   • `where`: Print where the program is, i.e. where the segmentation fault occurred
   • `finish`: It will run to the end of the current function and display return value.
   • `delete`: Remove all breakpoints.
   • `info breakpoints`: Displays information about breakpoints.

4. **Search the documentation and explain how to use conditional breakpoints.**

   ```
   break test.c:9 # set breakpoint at line 9 in test.c
   # Assuming this is the 7th breakpoint
   condition 7 count>=MAX # set condition
   ```

5. **What is `-tui` option for GDB?**

   It uses text windows to show various informations.

6. **What is the "reverse step" in GDB and how to enable it. Provide the key steps and commands.**

   First use `record` to activate the reverse execution.

   Commands:

   ```
   reverse-continue [ignore-count]
   # Beginning at the point where your program last stopped, start executing in
   reverse. Reverse execution is similar to normal debugging process, despite
   the direction.
   reverse-finish
   # Go to the point where the current function was called.
   reverse-step [count]
   # Run the program backward until control reaches the start of a different
   source line.
   reverse-stepi [count]
   # Run backward to the beginning of the previous line executed in the current
   (innermost) stack frame.
   reverse-next [count]
   ```

```
# Run backward to the beginning of the previous line executed in the current
(innermost) stack frame.
reverse-nexti [count]
# Executes a single instruction in reverse, except that called functions are
“un-executed” atomically.
set exec-direction
# Set the direction of target execution.
```