TravisTorrent

En la tabla travis_torrent tenemos un registro por cada job de un build de Travis. Los atributos que tiene son los siguientes:

- tr_build_id: identificador del build.
- tr_job_id: identificador del job.
- tr_build_number: número de build dentro de ese proyecto.
- gh_project_name: nombre del proyecto en github.
- gh_is_pr: indica si el build se lanzó como parte de una pull request.
- gh_pr_created_at: fecha de la pull request, en utc.
- gh_pull_req_num: id de la pull request en github.
- gh_lang: lenguaje dominante en el repositorio, según github.
- git_merged_with: (si gh_is_pr es true) indica cómo se resolvió (merge button, commits_in_master, fixes_in_commit, unknown o cadena vacía)
- git_branch: rama sobre la que se lanzó el build
- gh_num_commits_in_push: números de commits incluidos en el push que lanzó el build.
- gh_commits_in_push: lista de commits incluidos en el push que lanzó el build.
- git_prev_commit_resolution_status: estado del build anterior (puede ser no_previous_found, build_found o merge_found, este último cuando se ha realizado un merge y no se puede decidir cuál de los padres seguir)
- git_prev_built_commit: commit que lanzó built anterior.
- tr_prev_build: id del build lanzado por git_prev_built_commit
- gh_first_commit_created_at: timestamp del primer commit del push que lanzó el build.
- gh_team_size: personas que comitearon o con pull requests mergeadas desde 3 meses antes de la fecha del build.
- git_all_built_commits: lista de commits hasta commit del build anterior o hasta un commit de un merge. (mirar diferencia con git_commits_in_push)
- git_num_all_built_commits: número de commits en git_all_built_commits
- git_trigger_commit: commit que lanzó el build.
- tr_virtual_merged_into: commit de la rama que travis mergea cuando hace tests de pull requests
- tr_original_commit: commit a partir del que se hizo el build. puede ser un commit virtual (no siempre es igual a git_trigger_commit)
- gh_num_issue_comments: si git_commit está enlazado a una pr, número de comentarios en esa pr.
- gh_num_commits_comments: número de comentarios en los commits de git_all_built_commits.
- gh_num_pr_comments: sí gh_is_pr es true, número de comentarios en la pull requests (code reviews).
- git_diff_src_churn: número de líneas de producción cambiados en git_all_builts_commits.
- git_diff_test_churn: número de líneas de test cambiadas en git_all_built_commits.
- gh_diff_files_added: número de ficheros añadidos en git_all_built_commits.
- gh_diff_files_deleted: número de ficheros eliminados en git_all_built_commits.
- gh_diff_files_modified: número de ficheros modificados en git_all_built_commits.

- gh_diff_tests_added: número de tests añadidos en git_all_built_commits.
- gh_diff_tests_deleted: número de tests eliminados en git_all_built_commits.
- gh_diff_src_files: número de ficheros de código cambiados en git_all_built_commits.
- gh_diff_dic_files: número de ficheros de documentación cambiados en git_all_built_commits.
- gh_diff_other_files: número de archivos que no sean de código o documentación cambiados en git_all_built_commits.
- gh_num_commits_on_files_touched: número de commits en los ficheros cambiados en git_all_built_commits desde que se lanzó el build y hasta 3 meses atrás.
- gh_sloc: número de líneas de código en todo el repositorio.
- gh_test_lines_per_kloc: número de líneas de test por cada 1000 líneas de código (gh_sloc).
- gh_test_cases_per_kloc: número de asserts por cada 1000 líneas de código (gh_sloc).
- gh_by_core_team_member: indica si el commit fue autorizado por un "core team member", usuario que tiene código commiteado en los últimos 3 meses.
- gh_description_complexity: si el build es una pr, nº palabras en el título y la descripción de la pr.
- gh_pushed_at: timestamp del push que lanzó el build (dado por github).
- gh_build_started_at: timestamp del push que lanzó el build (dado por travis).
- tr_status: estado del build proporcionado por travis (passed, failed, errored, canceled, started)
- tr_duration: duración del build.
- tr_jobs: ids de los jobs del build.
- tr_log_lan: lenguaje de programación principal, extraído del análisis del log del build.
- tr_log_status: estado del build, extraído del análisis del log del build (ok, broken, cancelled, unknown).
- tr_log_setup_time: tiempo, en segundos, antes de comenzar la script pase (build actual).
- tr_log_frameworks: frameworks de tests extraídos en el análisis del log del build.
- tr_log_bool_test_ran: indica si se ejecutaron los tests.
- tr_log_bool_tests_failed: indica si los tests fallaron.
- tr_log_num_tests_ok: número de tests exitosos.
- tr_log_num_tests_failed: número de tests fallados.
- tr_log_num_tests_run: número de tests lanzados.
- tr_log_num_tests_skipped: número de tests saltados.
- tr_log_tests_failed: nombres de los tests fallados.
- tr_log_testduration: duración de la ejecución de los tests, en segundos.
- tr_log_builddduration: duración del build completo, en segundos.

GitHubTorrent

users

- name: nombre de usuario.
- company: compañía a la que pertenecen.
- login:
- created_at: fecha en la que se crearon.
- type: USR o ORG, para usuarios individuales u organizaciones.
- fake: indica si es un usuario real o no.
- email: correo electrónico
- deleted: indica si se ha borrado.
- long, lat, country_code, state y city: datos de geolocalización.

projects

- url: dirección del repositorio.
- owner_id: usuario que lo creó.
- Name y description: nombre y descripción del proyecto.
- language: lenguaje de programación principal.
- created_ad: fecha de creación.
- forked_from: ID del proyecto del que se ha hecho fork.
- deleted: indica si se ha borrado el proyecto.
- updated_at: indica última fecha en la que se actualizó el proyecto.

project_members

- user_id, repo_id: ID del usuario y del proyecto
- created_at
- ext_ref_id

Nota: según GHTorrent, a partir de Noviembre de 2014 GitHub deshabilitó una funcionalidad de la API que provoca que esta tabla no sea del todo precisa.

project_languages

- project_id: ID del proyecto.
- language: lenguaje de programación que utiliza.
- bytes: suma de los bytes de los archivos de ese lenguaje.
- created_at: última fecha en la que se comprobó el lenguaje para un proyecto.

commits

- sha: código sha del commit
- author_id y commiter_id: ID del usuario que realizó el commit (me falta comprobar la diferencia entre ambos, suele ser el mismo valor)
- project_id: ID del proyecto al que se asoció en primer lugar el commit (puede tener más, por eso existe otra tabla project_commits).
- created_at: fecha del commit.

commit_parents

- commit_id y parent_id: ID de cada commit y de su commit padre.

project_commits

- commit_id y project_id: ID del commit y del proyecto asociado al mismo.

commit_comments (comentarios sobre commits individuales, sin PR asociada)

- commit_id: ID del commit del que se hace el comentario.
- user_id: ID del usuario que hace el comentario.
- body: texto del comentario.
- line y position: línea y posición del código al que hace referencia el comentario.
- comment_id: ¿? (no hay una tabla de comments, no sé a qué hará referencia). Supongo que será ID único del comentario proporcionado por GitHub
- created_at: fecha en la que se hizo el comentario.

pull_requests

- head_repo_id y head_commit_id: ID del proyecto y el commit sobre el que queremos hacer merge.
- base_project_id y base_commit_id: ID del proyecto y el commit que queremos mergear.
- pullreq_id: identificador único de la PR dado por GitHub.
- intra_branch: indica si el repositorio head y base son el mismo.

pull_request_history

- pull_request_id: ID de la PR.
- created_at: fecha en la que se hizo la acción.
- actor_id: usuario que realizó la acción.
- action: 5 posibles acciones sobre la PR. (opened, closed, merged, reponed (PR abierta después de ser cerrada) y syncrhonize (commits añadidos o eliminados en repositorio base).

pull_request_commits

- commit_id y pull_request_id: ID del commit y de la PR a la que pertenece.

pull_request_comments (comentarios sobre un commit asociado a una PR)

- pull_request_id: ID de la PR sobre la que se hace el comentario.
- user_id: ID del usuario que realiza el comentario.
- comment_id: ¿? (no hay una tabla de comments, no sé a qué hará referencia). Supongo que será ID único del comentario proporcionado por GitHub
- position: posición sobre la que se hace el comentario.
- body: texto del comentario.
- commit_id: ID del commit sobre el que se realiza el comentario
- created_at: fecha del comentario.

issues

- repo_id: ID del repositorio que tiene el issue asociado.
- reporter_id: ID del usuario que reporta el error.
- assignee_id: ID del usuario al que se le asigna el issue.
- issue_id: identificador único proporcionado por GitHub.
- pull_request: por cada PR, GitHub crea el issue correspondiente. Este campo indica si procede de una PR.
- pull_request_id: ID de la PR enlazada con el issue.

issue_events

- event_id: ID del evento.
- issue_id: ID del issue.
- actor_id: ID del usuario que realiza el evento.
- created_at: fecha de la acción.
- action: acción realizada. Posibles valores:
    - subscribed: usuario se suscribe para recibir notificaciones.
    - mentioned: usuario mencionado por otro usuario.
    - closed: issue se cierra.
    - referenced: issue se referencia con un commit.
    - assigned: issue se asigna a un usuario.
    - reopened: issue cerrada se vuelve a abrir.
    - unsubscribed: usuario se desuscribe.
    - merged: PR enlazada con la issue se mergea.
    - head_ref_cleaned, head_ref_deleted y head_ref_restored
- action_specific: ID del commit cuando la PR asociada a la issue se ha cerrado, mergeado o referenciado.

issue_comments

- issue_id: ID del issue.
- user_id: ID del usuario que realiza el comentario.
- comment_id: ¿?
- created_at: fecha del comentario

Nota: no parece que esté el cuerpo de los comentarios de los issues, por lo que parece que esta tabla es inútil.

repo_labels

- repo_id: ID del repositorio
- name: nombre de la etiqueta

issue_labels

- label_id: ID de la etiqueta
- issue_id: ID del issue al que se le asigna la etiqueta.