2 Branches 193 Tags Go to file Go to file Code

k1LoW Merge pull request #597 from k1LoW/update-goreleaser

22cf3b3 · 3 days ago

.github	Update goreleaser-action	3 days ago
.goreleaser	Update .goreleaser.yml	3 days ago
cmd	fix typo `whethre-> whether`	last month
cmdutil	Update pkgs	5 months ago
config	Address lint errors	4 days ago
coverage	Fix test	last year
datasource	Merge branch 'main' of github.c…	last month
ddl	bonsai	5 months ago
dict	Fix linter settings	2 years ago
drivers	Merge pull request #594 from …	4 days ago
img	Add logo	4 years ago
output	made Referenced table section…	3 weeks ago
sample	Fix	2 weeks ago
schema	fix typo `colums-> columns`	last month
scripts	Fix docker image build pipeline	2 years ago
testdata	made Referenced table section…	3 weeks ago
testutil	add view to the test schema	3 weeks ago
version	[tagpr] prepare for the next rele…	4 days ago
.editorconfig	bonsai	4 years ago
.gitignore	Add `--include/--exclude` opti…	2 years ago
.golangci.yml	golangci-lint timeout 5m	8 months ago
.octocov.yml	Use octocov	2 years ago
.tagpr	Fix CD pipeline	5 months ago
CHANGELOG.md	[tagpr] update CHANGELOG.md	4 days ago
CREDITS	Update pkgs	5 months ago
Dockerfile	Copy ca-certificates.crt from bui…	2 years ago
LICENSE	cobra init github.com/k1LoW/tbls	6 years ago
Makefile	Embed tzdata	last month

About

tbls is a CI-Friendly tool for document a database, written in Go.

#mysql #markdown #bigquery #continuous-integration #sqlite #excel #dynamodb #postgresql #documentation-tool #snowflake #plantuml #mariadb #redshift #sqlserver #database-schema #mermaid #hacktoberfest #spanner #er-diagram #database-document

Readme
MIT license
Activity
3.3k stars
23 watching
162 forks
Report repository

Releases 193

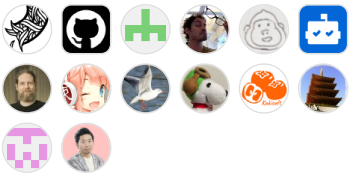
v1.76.1 Latest
4 days ago

+ 192 releases

Packages 1

tbls

Contributors 56



+ 42 contributors

Languages

Go 96.8% **Makefile** 2.8%
Other 0.4%

README.md	Fix some typos in README	3 months ago
client_secrets.json.gpg	Fix decrypt secrets	5 years ago
docker-compose.yml	Fix mount position	last year
go.mod	chore(deps): bump the depend…	3 weeks ago
go.sum	chore(deps): bump the depend…	3 weeks ago
main.go	Update go-mssqldb	2 years ago
use	Fixed position of a comment in …	last year

README MIT license

`tbls` (is pronounced /'teɪbl̩z/.) is a CI-Friendly tool for document a database, written in Go.

Key features of `tbls` are:

- **Document a database automatically in GFM format. Output database schema in many formats.**
- **Single binary = CI-Friendly.**
- **Support many databases.**
- **Work as linter for database**

Table of Contents

        - Support Datasource
    - Document path
    - Document format
    - ER diagram
    - Filter tables
    - Lint
    - Comments
    - Relations
    - Viewpoints
    - Dictionary
    - Personalized Templates
    - Required Version
- Expand environment variables
- Output formats
- Command arguments
- Environment variables

Quick Start

Document a database with one command.

```
$ tbls doc postgres://dbuser:dbpass@hostname:5432/dbname
```

Using docker image.

```
$ docker run --rm -v $PWD:/work -w /work ghcr.io/k1low/tbls doc postgres://dbuser:dbpass@hostname:5432/dbname
```

Install

deb:

```
$ export TBLS_VERSION=X.X.X
$ curl -o tbls.deb -L https://github.com/k1LoW/tbls/releases/download/v$TBLS_VERSION/tbls_$TBLS_VERSION-1_amd64.deb
$ dpkg -i tbls.deb
```

RPM:

```
$ export TBLS_VERSION=X.X.X
$ yum install https://github.com/k1LoW/tbls/releases/download/v$TBLS_VERSION/tbls_$TBLS_VERSION-1_amd64.rpm
```

Homebrew:

```
$ brew install k1LoW/tap/tbls
```

MacPorts:

```
$ sudo port install tbls
```

aqua:

```
$ aqua g -i k1LoW/tbls
```

Manually:

Download binary from releases page

go install:

```
$ go install github.com/k1LoW/tbls@latest
```

Docker:

```
$ docker pull ghcr.io/k1low/tbls:latest
```

On GitHub Actions:

```
# .github/workflows/doc.yml
name: Document

on:
  push:
    branches:
      - main

jobs:
  doc:
    runs-on: ubuntu-latest
    steps:
      -
        name: Checkout .tbls.yml
        uses: actions/checkout@v3
      -
        uses: k1low/setup-tbls@v1
      -
        name: Run tbls for generate database document
        run: tbls doc
```

GitHub Actions for tbls is here.

Temporary:

```
$ source <(curl https://raw.githubusercontent.com/k1LoW/tbls/main/use)
```

```
$ curl -sL https://raw.githubusercontent.com/k1LoW/tbls/main/use > /tmp/use-tbls.tmp && . /tmp/use-tbls.tmp
```

Getting Started

Document a database

Add `.tbls.yml` (or `tbls.yml`) file to your repository.

```
# .tbls.yml

# DSN (Database Source Name) to connect database
dsn: postgres://dbuser:dbpass@localhost:5432/dbname

# Path to generate document
# Default is `dbdoc`
docPath: doc/schema
```

> **Notice:** If you are using a symbol such as `#` `<` in database password, URL-encode the password

Run `tbls doc` to analyzes the database and generate document in GitHub Friendly Markdown format.

```
$ tbls doc
```

Commit `.tbls.yml` and the document.

```
$ git add .tbls.yml doc/schema
$ git commit -m 'Add database document'
$ git push origin main
```

View the document on GitHub.

Sample document

users

Description

Users table

Columns

Name	Type	Default	Nullable	Children	Parents	Cc
id	integer	nextval('users_id_seq'::regclass)	false	user_options posts comments comment_stars administrator.blogs logs		
username	varchar(50)		false			
password	varchar(50)		false			
email	varchar(355)		false			ex. user@e
created	timestamp without time zone		false			
updated	timestamp without time zone		true			

Constraints

Name	Type	Definition
users_username_check	CHECK	CHECK ((char_length((username)::text) > 4))
users_pkey	PRIMARY KEY	PRIMARY KEY (id)
users_username_key	UNIQUE	UNIQUE (username)
users_email_key	UNIQUE	UNIQUE (email)

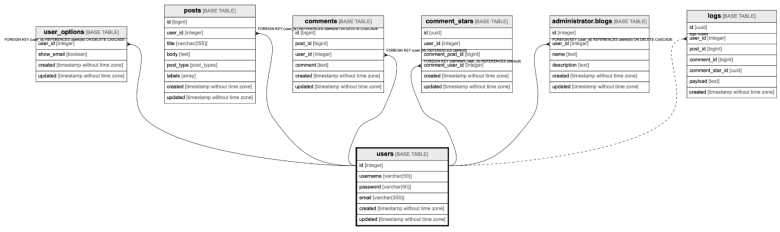
Indexes

Name	Definition
users_pkey	CREATE UNIQUE INDEX users_pkey ON public.users USING btree (id)
users_username_key	CREATE UNIQUE INDEX users_username_key ON public.users USING btree (username)
users_email_key	CREATE UNIQUE INDEX users_email_key ON public.users USING btree (email)

Triggers

Name	Definition
update_users_updated	CREATE TRIGGER update_users_updated AFTER INSERT OR UPDATE ON public.users FOR EACH ROW EXECUTE PROCEDURE update_updated()

Relations



Generated by tbls

Diff database and (document or database)

Update database schema.

```
$ psql -U dbuser -d dbname -h hostname -p 5432 -c 'ALTER TABLE users ADD COLUMN phone_number varchar(15);'
Password for user dbuser:
ALTER TABLE
```

`tbls diff` shows the difference between database schema and generated document.

```
$ tbls diff
diff postgres://dbuser:*****@hostname:5432/dbname doc/schema/README.md
--- postgres://dbuser:*****@hostname:5432/dbname
+++ doc/schema/README.md
@@ -4,7 +4,7 @@

 | Name | Columns | Comment | Type |
 | ---- | ------- | ------- | ---- |
-| [users](users.md) | 7 | Users table | BASE TABLE |
+| [users](users.md) | 6 | Users table | BASE TABLE |
 | [user_options](user_options.md) | 4 | User options table | BASE TABLE |
 | [posts](posts.md) | 8 | Posts table | BASE TABLE |
 | [comments](comments.md) | 6 | Comments<br>Multi-line<br>table<br>comment | BASE TABLE |
diff postgres://dbuser:*****@hostname:5432/dbname doc/schema/users.md
--- postgres://dbuser:*****@hostname:5432/dbname
+++ doc/schema/users.md
@@ -14,7 +14,6 @@
 | email | varchar(355) |  | false |  |  | ex. user@example.com |
 | created | timestamp without time zone |  | false |  |  |  |
 | updated | timestamp without time zone |  | true |  |  |  |
-| phone_number | varchar(15) |  | true |  |  |  |

 ## Constraints
```

And, `tbls diff` support for diff checking between database and other database

```
$ tbls diff postgres://dbuser:*****@local:5432/dbname postgres://dbuser:*****@production:5432/dbname
```

> **Notice:** `tbls diff` shows the difference Markdown documents only.

Re-generating database documentation

Existing documentation can re-generated using either `--force` or `--rm-dist` flag.

`--force` forces overwrite of the existing documents. It does not, however, remove files of removed tables.

```
$ tbls doc --force
```

`--rm-dist` removes files in docPath before generating the documents.

```
$ tbls doc --rm-dist
```

Lint a database

Add linting rule to `.tbls.yml` following

```
# .tbls.yml
lint:
  requireColumnComment:
```

```
    enabled: true
    exclude:
      - id
      - created
      - updated
  columnCount:
    enabled: true
    max: 10
```

Run `tbls lint` to check the database according to `lint:` rules

```
$ tbls lint
users.username: column comment required.
users.password: column comment required.
users.phone_number: column comment required.
posts.user_id: column comment required.
posts.title: column comment required.
posts.labels: column comment required.
comments.post_id: column comment required.
comment_stars.user_id: column comment required.
post_comments.comment: column comment required.
posts: too many columns. [12/10]
comments: too many columns. [11/10]

11 detected
```

Measure document coverage

`tbls coverage` measure and show document coverage (description, comments).

```
$ tbls coverage
Table                       Coverage
All tables                  16.1%
 public.users               20%
 public.user_options        37.5%
 public.posts               35.3%
 public.comments            14.3%
 public.comment_stars       0%
 public.logs                12.5%
 public.post_comments       87.5%
 public.post_comment_stars  0%
 public.CamelizeTable       0%
 public.hyphen-table        0%
 administrator.blogs        0%
 backup.blogs               0%
 backup.blog_options        0%
 time.bar                   0%
 time.hyphenated-table      0%
 time.referencing           0%
```

Continuous Integration

Continuous integration using tbls.

1. Commit the document using `tbls doc` .
2. Update the database schema in the development cycle.
3. Check for document updates by running `tbls diff` or `tbls lint` in CI.
4. Return to **1**.

Example: Travis CI

```
# .travis.yml
language: go

install:
```

```
  - source <(curl -sL https://raw.githubusercontent.com/k1LoW/tbls/main/use)
script:
  - tbls diff
  - tbls lint
```

> **Tips:** If your CI based on Debian/Ubuntu (`/bin/sh -> dash`), you can use the following install command `curl -sL https://raw.githubusercontent.com/k1LoW/tbls/main/use > use-tbls.tmp && . ./use-tbls.tmp && rm ./use-tbls.tmp`

> **Tips:** If the order of the columns does not match, you can use the `--sort` option.

Configuration

Name

`name:` is used to specify the database name of the document.

```
# .tbls.yml
name: mydatabase
```

Description

`desc:` is used to specify the database description.

```
# .tbls.yml
desc: This is My Database
```

Labels

`labels:` is used to label the database or tables.

label database:

```
# .tbls.yml
labels:
  - cmdb
  - analytics
```

label tables:

```
# .tbls.yml
comments:
  -
    table: users
    labels:
      - user
      - privacy data
```

label columns:

```
# .tbls.yml
comments:
  -
    table: users
    columnLabels:
      email:
        - secure
        - encrypted
```

DSN

`dsn:` (Data Source Name) is used to connect to database.

```
# .tbls.yml
dsn: my://dbuser:dbpass@hostname:3306/dbname
```

Support Datasource

tbls supports the following databases/datasources.

PostgreSQL:

```
# .tbls.yml
dsn: postgres://dbuser:dbpass@hostname:5432/dbname
```

```
# .tbls.yml
dsn: pg://dbuser:dbpass@hostname:5432/dbname
```

When you want to disable SSL mode, add "?sslmode=disable" For example:

```
dsn: pg://dbuser:dbpass@hostname:5432/dbname?sslmode=disable
```

MySQL:

```
# .tbls.yml
dsn: mysql://dbuser:dbpass@hostname:3306/dbname
```

```
# .tbls.yml
dsn: my://dbuser:dbpass@hostname:3306/dbname
```

When you want to hide AUTO_INCREMENT clause on the table definitions, add "?hide_auto_increment". For example:

```
dsn: my://dbuser:dbpass@hostname:3306/dbname?hide_auto_increment
```

MariaDB:

```
# .tbls.yml
dsn: mariadb://dbuser:dbpass@hostname:3306/dbname
```

```
# .tbls.yml
dsn: maria://dbuser:dbpass@hostname:3306/dbname
```

SQLite:

```
# .tbls.yml
dsn: sqlite:///path/to/dbname.db
```

```
# .tbls.yml
dsn: sq:///path/to/dbname.db
```

BigQuery:

```
# .tbls.yml
dsn: bigquery://project-id/dataset-id?creds=/path/to/google_application_credentials.json
```

```
# .tbls.yml
dsn: bq://project-id/dataset-id?creds=/path/to/google_application_credentials.json
```

To set `GOOGLE_APPLICATION_CREDENTIALS` environment variable, you can use

1. `export GOOGLE_APPLICATION_CREDENTIALS` or `export GOOGLE_APPLICATION_CREDENTIALS_JSON`
2. Add query to DSN
    - `?google_application_credentials=/path/to/client_secrets.json`
    - `?credentials=/path/to/client_secrets.json`
    - `?creds=/path/to/client_secrets.json`

Required permissions: `bigquery.datasets.get` `bigquery.tables.get` `bigquery.tables.list`

Also, you can use impersonate service account using environment variables below.

- `GOOGLE_IMPERSONATE_SERVICE_ACCOUNT` : Email of service account
- `GOOGLE_IMPERSONATE_SERVICE_ACCOUNT_LIFETIME` : You can use impersonate service account within this lifetime. This value must be readable from https://github.com/k1LoW/duration .

Cloud Spanner:

```
# .tbls.yml
dsn: spanner://project-id/instance-id/dbname?creds=/path/to/google_application_credentials.json
```

To set `GOOGLE_APPLICATION_CREDENTIALS` environment variable, you can use

1. `export GOOGLE_APPLICATION_CREDENTIALS` or `export GOOGLE_APPLICATION_CREDENTIALS_JSON`
2. Add query to DSN
    - `?google_application_credentials=/path/to/client_secrets.json`
    - `?credentials=/path/to/client_secrets.json`
    - `?creds=/path/to/client_secrets.json`

Also, you can use impersonate service account using environment variables below.

- `GOOGLE_IMPERSONATE_SERVICE_ACCOUNT` : Email of service account
- `GOOGLE_IMPERSONATE_SERVICE_ACCOUNT_LIFETIME` : You can use impersonate service account within this lifetime. This value must be readable from https://github.com/k1LoW/duration .

Amazon Redshift:

```
# .tbls.yml
dsn: redshift://dbuser:dbpass@hostname:5432/dbname
```

```
# .tbls.yml
dsn: rs://dbuser:dbpass@hostname:5432/dbname
```

Microsoft SQL Server:

```
# .tbls.yml
dsn: mssql://DbUser:SQLServer-DbPassw0rd@hostname:1433/testdb
```

```
# .tbls.yml
dsn: sqlserver://DbUser:SQLServer-DbPassw0rd@hostname:1433/testdb
```

```
# .tbls.yml
dsn: ms://DbUser:SQLServer-DbPassw0rd@localhost:1433/testdb
```

Amazon DynamoDB:

```
# .tbls.yml
dsn: dynamodb://us-west-2
```

```
# .tbls.yml
dsn: dynamo://ap-northeast-1?aws_access_key_id=XXXXXxxxxxxxXXXXXXX&aws_secret_access_key=XXXXXxxxxxxxXXXXXXX
```

To set AWS credentials, you can use

1. Use default credential provider chain of AWS SDK for Go
2. Add query to DSN
    - `?aws_access_key_id=XXXXXxxxxxxxXXXXXXX&aws_secret_access_key=XXXXXxxxxxxxXXXXXXX`

Snowflake (Experimental):

```
---
# .tbls.yml
dsn: snowflake://user:password@myaccount/mydb/myschema
```

See also: https://pkg.go.dev/github.com/snowflakedb/gosnowflake

MongoDB:

```
# .tbls.yml
dsn: mongodb://mongoadmin:secret@localhost:27017/test
```

```
# .tbls.yml
dsn: mongodb://mongoadmin:secret@localhost:27017/test?sampleSize=20
```

If a field has multiple types, the `multipleFieldType` query can be used to list all the types.

```
# .tbls.yml
dsn: mongodb://mongoadmin:secret@localhost:27017/test?sampleSize=20&multipleFieldType=true
```

JSON:

The JSON file output by the `tbls out -t json` command can be read as a datasource.

```
---
# .tbls.yml
dsn: json://path/to/testdb.json
```

HTTP:

```
---
# .tbls.yml
dsn: https://hostname/path/to/testdb.json
```

```
---
# .tbls.yml
dsn:
  url: https://hostname/path/to/testdb.json
  headers:
    Authorization: token GITHUB_OAUTH_TOKEN
```

GitHub:

```
---
# .tbls.yml
dsn: github://k1LoW/tbls/sample/mysql/schema.json
```

Document path

`tbls doc` generates document in the directory specified by `docPath: .`

```
# .tbls.yml
# Default is `dbdoc`
docPath: doc/schema
```

Document format

`format:` is used to change the document format.

```
# .tbls.yml
format:
  # Adjust the column width of Markdown format table
  # Default is false
  adjust: true
  # Sort the order of table list and columns
  # Default is false
  sort: false
  # Display sequential numbers in table rows
  # Default is false
  number: false
  # The comments for each table in the Tables section of the index page will display the text up to the first double
  # Default is false
  showOnlyFirstParagraph: true
  # Hide table columns without values
  # Default is false
  hideColumnsWithoutValues: true
  # It can be boolean or array
  # hideColumnsWithoutValues: ["Parents", "Children"]
```

ER diagram

`tbls doc` generate ER diagram images at the same time.

```
# .tbls.yml
er:
  # Skip generation of ER diagram
  # Default is false
  skip: false
  # ER diagram image format (`png`, `jpg`, `svg`, `mermaid`)
  # Default is `svg`
  format: svg
  # Add table/column comment to ER diagram
  # Default is false
  comment: true
  # Hide relation definition from ER diagram
  # Default is false
  hideDef: true
  # Show column settings in ER diagram. If this section is not set, all columns will be displayed (default).
  showColumnTypes:
    # Show related columns
    related: true
    # Show primary key columns
    primary: true
  # Distance between tables that display relations in the ER
  # Default is 1
  distance: 2
  # ER diagram (png/jpg) font (font name, font file, font path or keyword)
```

```
  # Default is "" ( system default )
  font: M+
```

It is also possible to personalize the output by providing your own templates. See the Personalized Templates section below.

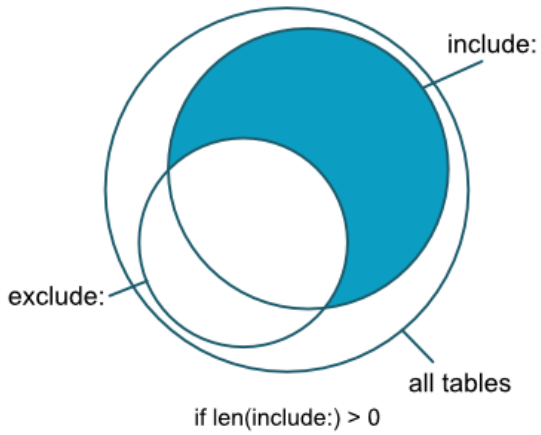
Lint

`tbls lint` work as linter for database.

```
# .tbls.yml
lint:
  # require table comment
  requireTableComment:
    enabled: true
    # all commented, or all uncommented.
    allOrNothing: false
  # require column comment
  requireColumnComment:
    enabled: true
    # all commented, or all uncommented.
    allOrNothing: true
    # exclude columns from warnings
    exclude:
      - id
      - created_at
      - updated_at
    # exclude tables from warnings
    excludeTables:
      - logs
      - comment_stars
  # require index comment
  requireIndexComment:
    enabled: true
    # all commented, or all uncommented.
    allOrNothing: false
    # exclude indexes from warnings
    exclude:
      - user_id_idx
    # exclude tables from warnings
    excludeTables:
      - logs
      - comment_stars
  # require constraint comment
  requireConstraintComment:
    enabled: true
    # all commented, or all uncommented.
    allOrNothing: false
    # exclude constrains from warnings
    exclude:
      - unique_user_name
    # exclude tables from warnings
    excludeTables:
      - logs
      - comment_stars
  # require trigger comment
  requireTriggerComment:
    enabled: true
    # all commented, or all uncommented.
    allOrNothing: false
    # exclude triggers from warnings
    exclude:
      - update_count
    # exclude tables from warnings
    excludeTables:
      - logs
      - comment_stars
  # require table labels
  requireTableLabels:
    enabled: true
    # all commented, or all uncommented.
```

```
    allOrNothing: false
    # exclude tables from warnings
    exclude:
      - logs
  # find a table that has no relation
  unrelatedTable:
    enabled: true
    # all related, or all unrelated.
    allOrNothing: true
    # exclude tables from warnings
    exclude:
      - logs
  # check max column count
  columnCount:
    enabled: true
    max: 10
    # exclude tables from warnings
    exclude:
      - user_options
  # require columns
  requireColumns:
    enabled: true
    columns:
      -
        name: created
      -
        name: updated
        exclude:
          - logs
          - CamelizeTable
  # check duplicate relations
  duplicateRelations:
    enabled: true
  # check if the foreign key columns have an index
  requireForeignKeyIndex:
    enabled: true
    exclude:
      - comments.user_id
  # checks if labels are in BigQuery style ( https://cloud.google.com/resource-manager/docs/creating-managing-labels#
  labelStyleBigQuery:
    enabled: true
    exclude:
      - schema_migrations
  # checks if tables are included in at least one viewpoint
  requireViewpoints:
    enabled: true
    exclude:
      - schema_migrations
```

Filter tables



`include:` and `exclude:` are used to filter target tables from `tbls *`.

```yaml
# .tbls.yml
include:
  - some_prefix_*
exclude:
  - some_prefix_logs
  - CamelizeTable
```

`lintExclude:` is used to exclude tables from `tbls lint`.

```yaml
# .tbls.yml
lintExclude:
  - CamelizeTable
```

Filter logic

1. Add tables from include
2. Remove tables from exclude
   - Check for include/exclude overlaps
   - If include is more specific than exclude (i.e. `schema.MyTable > schema.*` or `schema.MyT* > schema.*`), include the table(s). If include is equally or less specific than exclude, exclude wins.
3. Result

Comments

`comments:` is used to add table/column comment to database document without `ALTER TABLE`.

For example, you can add comment about VIEW TABLE or SQLite tables/columns.

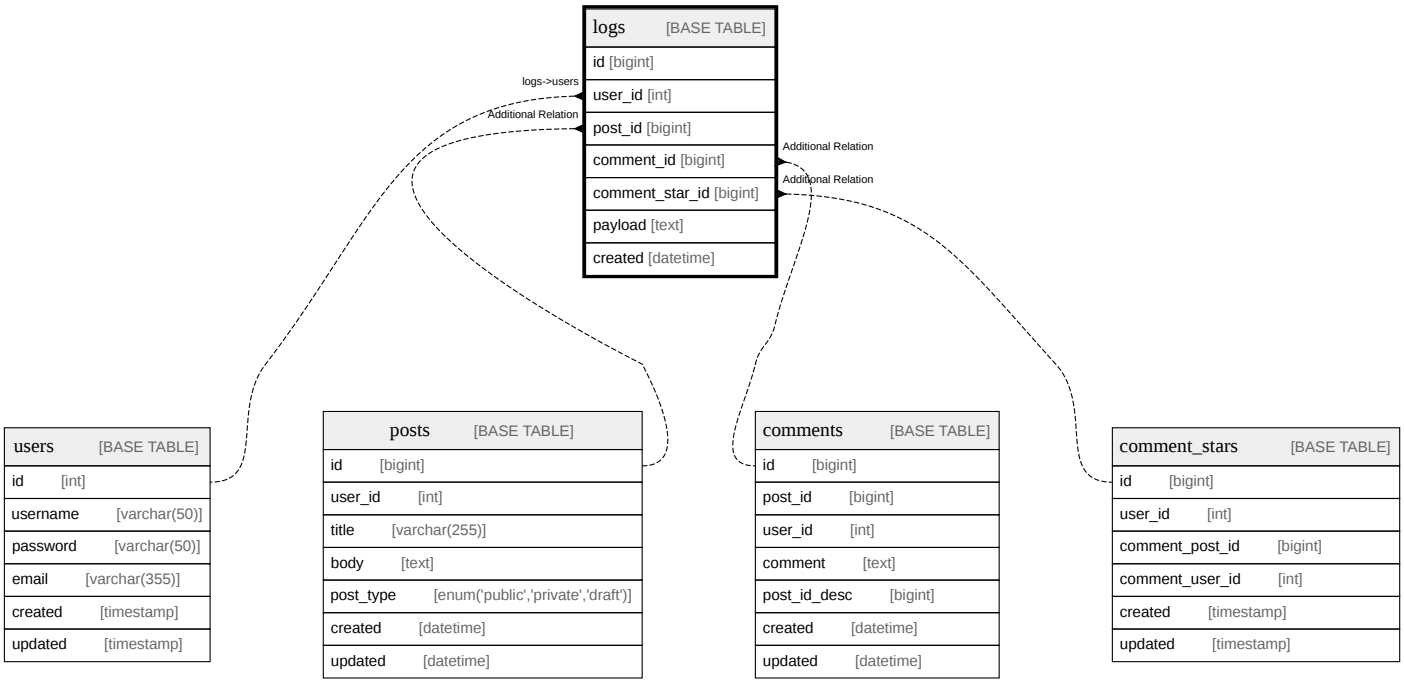
```yaml
# .tbls.yml
comments:
  -
    table: users
    # table comment
    tableComment: Users table
    # column comments
    columnComments:
      email: Email address as login id. ex. user@example.com
    # labels for tables
    labels:
      - privary data
      - backup:true
  -
    table: post_comments
    tableComment: post and comments View table
    columnComments:
      id: comments.id
      title: posts.title
      post_user: posts.users.username
      comment_user: comments.users.username
      created: comments.created
      updated: comments.updated
  -
    table: posts
    # index comments
    indexComments:
      posts_user_id_idx: user.id index
    # constraints comments
    constraintComments:
      posts_id_pk: PRIMARY KEY
    # triggers comments
    triggerComments:
      update_posts_updated: Update updated when posts update
```

Relations

`relations:` is used to add or override table relation to database document without `FOREIGN KEY`.

You can create ER diagrams with relations without having foreign key constraints.

```yaml
relations:
  -
    table: logs
    columns:
      - user_id
    parentTable: users
    parentColumns:
      - id
    # Relation definition
    # Default is `Additional Relation`
    def: logs->users
  -
    table: logs
    columns:
      - post_id
    parentTable: posts
    parentColumns:
      - id
  -
    table: logs
    columns:
      - comment_id
    parentTable: comments
    parentColumns:
      - id
  -
    table: logs
    columns:
      - comment_star_id
    parentTable: comment_stars
    parentColumns:
      - id
```



Override relations

If you want to override an existing relation, set the `override:` to `true`.

```yaml
relations:
  -
    table: posts
    columns:
      - user_id
    cardinality: zero or one
    parentTable: users
    parentColumns:
      - id
    parentCardinality: one or more
    override: true
    def: posts->users
```

Automatically detect relations

`detectVirtualRelations:` if enabled, automatically detect relations from table and column names.

```yaml
detectVirtualRelations:
  enabled: true
  strategy: default
```

`default` strategy:

```yaml
detectVirtualRelations:
  enabled: true
  strategy: default
```

- some_table.user_id -> users.id
- some_table.post_id -> posts.id

`singularTableName` strategy:

```yaml
detectVirtualRelations:
  enabled: true
  strategy: singularTableName
```

- some_table.user_id -> user.id
- some_table.post_id -> post.id

Dictionary

`dict:` is used to replace title/table header of database document

```yaml
# .tbls.yml
---
dict:
  Tables: テーブル一覧
  Description: 概要
  Columns: カラム一覧
  Indexes: INDEX一覧
  Constraints: 制約一覧
  Triggers: トリガー
  Relations: ER図
  Name: 名前
  Comment: コメント
  Type: タイプ
  Default: デフォルト値
  Children: 子テーブル
  Parents: 親テーブル
  Definition: 定義
  Table Definition: テーブル定義
```

Personalized Templates

It is possible to provide your own templates to personalize the documentation generated by `tbls` by adding a `templates:` section to your configuration. For example:

```
templates:
  dot:
    schema: 'templates/schema.dot.tmpl'
    table: 'templates/table.dot.tmpl'
  puml:
    schema: 'templates/schema.puml.tmpl'
    table: 'templates/table.puml.tmpl'
  md:
    index: 'templates/index.md.tmpl'
    table: 'templates/table.md.tmpl'
```

A good starting point to design your own template is to modify a copy the default ones for Dot, PlantUML and markdown.

Required Version

The `requiredVersion` setting defines a version constraint string. This defines which version of tbls can be used in the configuration.

```
requiredVersion: '>= 1.42, < 2'
```

Expand environment variables

All configuration values can be set by expanding the environment variables.

```
# .tbls.yml
dsn: my://${MYSQL_USER}:${MYSQL_PASSWORD}@hostname:3306/${MYSQL_DATABASE}
```

Viewpoints

Viewpoints of your database schema based on concerns of your domain and add description to them. You can also define groups of tables within viewpoints.

```
# .tbls.yml

viewpoints:
  -
    name: comments on post
    desc: Users can comment on each post multiple times and put a star on each comment.
    tables:
      - users
      - posts
      - comments
      - comment_stars
      - post_comments
      - post_comment_stars
    groups:
      -
        name: Comments
        desc: Tables about comments
        tables:
          - posts
          - comments
          - post_comments
      -
        name: Stars
        desc: Tables about stars
        tables:
          - comment_stars
          - post_comment_stars
```

Output formats

`tbls out` output in various formats.

Markdown:

```
$ tbls out -t md -o schema.md
```

DOT:

```
$ tbls out -t dot -o schema.dot
```

PlantUML:

```
$ tbls out -t plantuml -o schema.puml
```

Mermaid:

```
$ tbls out -t mermaid -o schema.mmd
```

Image (svg, png, jpg):

```
$ tbls out -t svg --table users --distance 2 -o users.svg
```

JSON:

```
$ tbls out -t json -o schema.json
```

> **Tips:** `tbls doc` can load `schema.json` as DSN.