Codenotary

Codenotary immudb cheat sheet

January 2023
Cheat sheet version 1.0
immudb version 1.4.1

Basic information

Immudb main page
https://immudb.io/
Immudb documentation
https://docs.immudb.io/master/
Playground
https://play.codenotary.com/

Releases page
https://github.com/codenotary/immudb/releases
Docker Hub
https://hub.docker.com/r/codenotary/immudb

CLI download / install

Download immudb
https://github.com/codenotary/immudb/releases
Get the link by copy pasting it from releases page
Rename the file and make it executable.
Download immuadmin and immuclient - repeat the same steps as for immudb.

Docker Setup

Pull one image from DockerHub (Docker has to be installed locally)
$ docker pull codenotary/immudb
To have a persistent volume storage run
$ docker storage create immudb_storage
Start the immudb database engine
$ docker run -p 3322:3322 -p 9497:9497 -p 5432:5432 -v immudb_storage:/var/lib/immudb -d --name immudb codenotary/immudb:latest
Additional ports: 9497 for Prometheus, 5432 for Pgserver

Health metrics

Immudb exposes many health metrics via Prometheus. For full info on immudb health monitoring see:
https://docs.immudb.io/master/production/monitoring.html

CLI setup and help

Before starting work do a preliminary setup:
- Start immudb (with Docker just run `docker run` cmd)

$ ./immudb -d
- Login (default u/p is immudb/immudb), if on Docker use docker exec -it immudb immuadmin login immudb

$ ./immuadmin login immudb
- create the database 'mydb'

$ ./immuadmin database create mydb
- create the first user 'user1' for database 'mydb'

$ ./immuadmin user create user1 readwrite mydb
You can find more help about immuadmin:
$ ./immuadmin help or $ ./immuadmin <cmd> help
Start immuclient session
$ ./immuclient
If you work on Docker immudb, download the immuclient tool to the machine where Docker runs. Later use this:
$ IMMUCLIENT_IMMUDB_ADDRESS=0.0.0.0 IMMUCLIENT_IMMUDB_PORT=3322 IMMUCLIENT_USERNAME=user1 IMMUCLIENT_PASSWORD=<password> IMMUCLIENT_DATABASE=mydb ./immuclient
You are in interactive mode. First login as the new user
- login user1

Switch to the new database
- use mydb

You are all set to work! Remember that help is at hand
- help or <cmd> --help

Get the current tx and hash of the entire database. You can later use this hash to check if the database changed:
- current

KV - Create

Set value for a key *location*
- set location France

Set value in a secure way for a key *country*
- safeset country Germany

Single or double quotes become part of the key or value.

KV - Read

Get value for a key *location*
- get location

Get value in a secure way for a key *country*
- safeget country

Get history of values for key *country*
- history country

Get keys and values of all keys that start with 'c'
- scan c

KV - Delete

Mark a value as deleted
- delete country

Please note that in the immutable database immudb no data is actually deleted, it is only marked so.

SQL – Table management

List databases
- query select * from databases();

List tables (also works: > tables)
- query select * from tables();

Create a new table
- exec create table people(id integer, name varchar[10], salary integer, primary key id)

Please note that the size of any field is optional, is in square brackets [] and the table must have a primary key definition.
Additional clauses supported:
- create table if not exists <table_name>
- <field> <type> not null
- <field> <type> auto_increment

Codenotary immudb cheat sheet

January 2023
Cheat sheet version 1.0
immudb version 1.4.1

Check what columns the table has (also: > describe <table>):

- query select * from columns('people');

Check what indexes are there in a table:

- query select * from indexes('people');

SQL – manipulate data

Insert data into a table. You can insert multiple values.

- exec insert into product (prod_id, name, price) values (1, 'Laptop',200), (2, 'Mobile', 100);

Update data

- exec update products set name='Goto' where prod_id=3

Upsert (insert and update if data is already there)

- exec upsert into products(prod_id, name, price) values (3, 'Foto', 500)

Delete selected rows

- exec delete from products where prod_id=4;

SQL – query data

Select data from a table

- query select prod_id from products

Ordering is possible only by primary key

- query select * from products order by prod_id;

Time travel – see the database before a specific tx

- query select * from products before tx 22;

SQL – Filtering (WHERE)

Multiple conditions (AND, OR, NOT)

- query select * from products where price > 300 and name = 'Mobile'

`Like` operator (based on golang regexp)

- query select * from products where name like 'M'

`In` operator

- query select * from products where name in ('Goto','Laptop')

SQL - Joins

Inner join. The word *inner* is optional.

- query select p.name, p.salary, e.nationality from people p inner join employees e on p.id = e.id;

SQL – Indexes

Currently, index creation is only supported on tables that haven't been written into yet. Indexing a column is necessary for grouping and aggregation to work.

You can create index only on columns of types: integer, varchar and blob. Field length has to be set and the length cannot exceed 256.

Create an index on a single column. You can use an optional clause *index if not exists on*

- exec create index on products(prod_id);

Create a composite index:

- exec create index on customers(country, ip);

Unique index prevents insertion of duplicates:

- exec create unique index on customers(email);

SQL - Aggregation

Use basic aggregation (MIN, MAX, AVG also supported)

- query select sum(price) as sum, count(*) as num from products;

Grouping is possible together only with 'order by' clause provided there is an index on this column

- query select sum(price) as sm from products group by name order by name

Data types supported

Name	Description
INTEGER	Signed 64-bit integer value
BOOLEAN	either TRUE or FALSE
VARCHAR	UTF8-encoded text
BLOB	sequence of bytes
TIMESTAMP	datetime value with microsecond precision

Supported functions

now() function returns timestamp of transaction creation time.

- exec insert into events(log_id, name, load_time) values (5, 'Inny', now());

The cast() function can be used to convert a string or an integer to a timestamp value.

- upsert into events(log_id, name, load_time) values (7, 'Key', cast('2023-01-01' AS TIMESTAMP))

SDKs

Go standard library:

https://docs.immudb.io/master/develop/sql/sqlstdlib.html

Pgsql protocol

https://docs.immudb.io/master/develop/sql/pg.html

SDK:

https://docs.immudb.io/master/connecting/sdks.html
https://docs.immudb.io/master/develop/reading.html

Java SDK

https://github.com/codenotary/immudb4j