This is complete implementation of CRUD operations in mySql using GRPC-GO

1. Create a folder grpc-sql (your choice) in desktop
   1. In this I will be having the folders which all use this functionalities if SQL in GRPC
2. Now create a subfolder in grpc-sql with name company
   1. This contains my server, client,proto,database,views folders
3. Open command prompt and cd to grpc-sql folder
4. Run the below command’s
   1. go mod init example.com/grpc-sql
   2. go mod edit -replace example.com/grpc-sql=../grpc-sql
5. create folders with names as given below along with the given files
   1. company\_server
      1. server.go
   2. company\_client
      1. client.go
   3. companypb
      1. companypb.proto
   4. database
      1. database.go
      2. createTable.sql
   5. views

syntax = "proto3";

package company;

option go\_package = "companypb";

message Company{

    int64 id = 1;

    string name = 2;

    string person = 3;

}

service CompanyService{}

* + 1. views.go

1. now, inside the companypb.proto file add the side code
2. Now from the grpc-sql directory in command prompt run the below command:
   1. protoc company/companypb/company.proto --gogo\_out=plugins=grpc:.
3. Inside the createTable.go insert the following code

DROP TABLE IF EXISTS company;

CREATE TABLE company(

    id INT AUTO\_INCREMENT NOT NULL,

    name  VARCHAR(500) NOT NULL,

    creator  VARCHAR(500) NOT NULL,

     PRIMARY KEY (`id`)

);

INSERT INTO company

(name,creator)

VALUES

("apple","steve jobs"),

("zip2","elon musk");

1. Open the mysql CLI
2. Create data base company with below command and use that database
   1. create database company
   2. use company
3. Now from that database run following command to create a company table
   1. source C:/Users/hp/Desktop/Go/grpc-sql/company/database/companyTable.sql;
4. Now add the following code in server.go just to check every thing
   1. package main
   2. import (
   3. "context"
   4. "fmt"
   5. "log"
   6. "net"
   7. "os"
   8. "database/sql"
   9. "github.com/go-sql-driver/mysql"
   10. "example.com/grpc-sql/company/companypb"
   11. "google.golang.org/grpc"
   12. )
   13. func main() {
   14. fmt.Println("Get Ready! server is starting")
   15. lis, err1 := net.Listen("tcp", "0.0.0.0:50051")
   16. if err1 != nil {
   17. log.Fatalf("Failed to listen: %v", err1)
   18. }
   19. // ------------------------------------------------------------------------
   20. // Database
   21. // Capture connection properties.
   22. cfg := mysql.Config{
   23. User:   os.Getenv("DBUSER"),
   24. Passwd: os.Getenv("DBPASS"),
   25. Net:                  "tcp",
   26. Addr:                 "127.0.0.1:3306",
   27. DBName:               "company",
   28. Params:               map[string]string{},
   29. AllowNativePasswords: true,
   30. }
   31. // Get a database handle.
   32. var err error
   33. db, err = sql.Open("mysql", cfg.FormatDSN())
   34. if err != nil {
   35. log.Fatal(err)
   36. }
   37. pingErr := db.Ping()
   38. if pingErr != nil {
   39. log.Fatal(pingErr)
   40. }
   41. fmt.Println("Connected to the database!")
   42. //--------------------------------------------------------------------------
   43. opts := []grpc.ServerOption{}
   44. s := grpc.NewServer(opts...)
   45. companypb.RegisterCompanyServiceServer(s, &server{})
   46. if err := s.Serve(lis); err != nil {
   47. log.Fatalf("Failed to serve : %v", err)
   48. }
   49. }
5. Now from the command prompt , set the BDUSER and DBPASS which we use to connect to db from below commands
   1. set DBUSER=root
   2. set DBPASS=<root password for mysql >
6. Now run the below command to check from grpc-sql directory
   1. go run company/company\_server/server.go
7. If everything works good and database is connected then, skip to next Step. If there is error in database connection then look tutorial from below link
   1. <https://go.dev/doc/tutorial/database-access>
8. Similar to server, to check the client working, paste the below code into client.go file
   1. package main
   2. import (
   3. "context"
   4. "fmt"
   5. "log"
   6. "example.com/grpc-sql/company/companypb"
   7. "google.golang.org/grpc"
   8. )
   9. func main() {
   10. fmt.Println("Hello I'm Client")
   11. opts := grpc.WithInsecure()
   12. cc, err := grpc.Dial("localhost:50051", opts)
   13. if err != nil {
   14. log.Fatalf("Could not connect : %v", err)
   15. }
   16. defer cc.Close()
   17. c := companypb.NewCompanyServiceClient(cc)
   18. fmt.Println("Client created : ", c)
   19. }
9. Now, run below code from grpc-sql folder in command prompt, to check the client and if there’s any error look at the GRPC course in udemy
   1. go run company/company\_client/client.go
10. Now, if everything is good, you’re good with all boiler plate codes
11. Now we need to add rpc services
12. First go to companypb.proto and add below code.

syntax = "proto3";

package company;

option go\_package = "companypb";

message Company{

    int64 id = 1;

    string name = 2;

    string person = 3;

}

message GetRequest {

    int64 id =1;

}

message GetResponse{

    Company company=1;

}

service CompanyService{

    // This gives the company details from its ID

    rpc Get (GetRequest) returns (GetResponse){};

}

1. The Company message we created to used as the structure to store the companies details like, id, name, person which are stored as id, name and creator in DB
2. The GET rpc is defined to get the row (company details) from its ID. So, GetRequest is an id and GetResponse is a Company structure that we defined before.
3. Repeat Step – 7
4. After creating the proto files, lets create a local structure just to use within the server, and database.
5. Inside the views folder create a comp.go file and add following code
   1. package views
   2. import (
   3. "example.com/grpc-sql/company/companypb"
   4. )
   5. type Comp struct {
   6. ID      int64
   7. Name    string
   8. Creator string
   9. }
   10. func (c \*Comp) ToCompany() \*companypb.Company {
   11. res := &companypb.Company{
   12. Id:     c.ID,
   13. Name:   c.Name,
   14. Person: c.Creator,
   15. }
   16. return res
   17. }
6. This is just the local data structure that we are going to use (but its just an optional, instead we can just import company from companypb in all files and just use them).
7. There is a small method for that structure, ToCompany which converts the Comp local structure to the Company structure that we created in Companypb (package)

**Note: Make sure that the method functions are all Starting with Capital letters, if they they cannot be used in other files and cannot be exported.**

1. Let’s define the Get function in the server.go which we created in company.proto file

(to know the arguments and return types of GET function , go to company.pb.go file and search for server interface and copy them)

1. Go to server.go file and add the below function code just above the main function and the following imports
   1. import (
   2. "context"
   3. "fmt"
   4. "log"
   5. "net"
   6. "os"
   7. "database/sql"
   8. "github.com/go-sql-driver/mysql"
   9. "example.com/grpc-sql/company/companypb"
   10. "example.com/grpc-sql/company/database"
   11. "example.com/grpc-sql/company/views"
   12. "google.golang.org/grpc"
   13. )

func (\*server) Get(ctx context.Context, req \*companypb.GetRequest) (\*companypb.GetResponse, error) {

    fmt.Println("Get function is invoked")

    id := req.GetId()

    row, err := database.SelectRow(id, db)

    if err != nil || row == nil {

        return nil, err

    }

    res := &companypb.GetResponse{

        Company: row,

    }

    return res, nil

}

1. In the get function in server, we first get the id from the request and we directly call the SelectRow function in the database which we are going to define in next step (which takes an int64 as id and returns the row with that id in Companypb.Company format). Then we check, if there is an error, or no row is returned (no row of that ID) then we return a nil and the error. If not we just cover the Response in Companypb.GetResponse structure and return.
2. Now, add the following code into the database.go file to add the SelectRow function into the database package
   1. package database
   2. import (
   3. "database/sql"
   4. "fmt"
   5. "example.com/grpc-sql/company/companypb"
   6. "example.com/grpc-sql/company/views"
   7. )
   8. // SelectRow queries for the album with the specified ID.
   9. func SelectRow(id int64, db \*sql.DB) (\*companypb.Company, error) {
   10. // A Company to hold data from the returned row.
   11. com := views.Comp{}
   12. row := db.QueryRow("SELECT \* FROM company WHERE id = ?", id)
   13. err := row.Scan(&com.ID, &com.Name, &com.Creator)
   14. if err != nil {
   15. if err == sql.ErrNoRows {
   16. return nil, fmt.Errorf("CompanyByID %d: no such Company", id)
   17. }
   18. return nil, fmt.Errorf("CompanyByID %d: %v", id, err)
   19. }
   20. res := com.ToCompany()
   21. return res, nil
   22. }
3. Here, we first import the packages, like companypb and views, then we first define the SelectRow function which returns a companypb.company structure along with error
4. We first initialise the local structure comp and do a select query with id and scan the row, and check for errors and finally, convert the comp structure to a companypb.company structure using ToCompany method we defines in comp.go file in views and return with nil error
5. If there’s error in the function, refer the below link
   1. <https://go.dev/doc/tutorial/database-access#single_row>
6. Now, the server side is completed.
7. Now go to the client.go file and add following code to the boilerplate code thate we cerated earlier
   1. At top of main function
   2. package main
   3. import (
   4. "context"
   5. "fmt"
   6. "log"
   7. "example.com/grpc-sql/company/companypb"
   8. "google.golang.org/grpc"
   9. )
   10. Below main function
   11. //This function is to get the company of specific id
   12. func getCompany(c companypb.CompanyServiceClient) {
   13. fmt.Println("Starting to do getById function ...")
   14. req := &companypb.GetRequest{
   15. Id: 2,
   16. }
   17. res, err := c.Get(context.Background(), req)
   18. if err != nil {
   19. log.Fatal(err)
   20. }
   21. fmt.Println("Response from the server : ", res)
   22. }
8. Now inside the main function call the getCompany(c) function to get the row of id mentioned in this function.
9. Similarly process from step-20 need to be repeated for remaining CRUD operations also with appropriate datatypes for request and responses.
10. For queries qhere is no return use below link
    1. <https://go.dev/doc/database/change-data>
11. For querying use
    1. <https://go.dev/doc/database/querying>