Manual

Panagiotis Toloudis

April 30, 2023

Contents

1 connection.ts

Async Function Connect

Description

The async function `connect` connects to a MongoDB database based on the user mode specified. It checks the user mode passed as a parameter and retrieves the corresponding MongoDB URL from environment variables. It then creates a new `MongoClient` instance and connects to the MongoDB database. If the connection is successful, it returns a `db` object representing the connected database. If the connection fails, it returns `null`.

Parameters

- `userMode`: A string that specifies the user mode. It can be one of the following: 'dev', 'Admin', 'User', or 'Guest'.

Returns

- `db`: A database object representing the connected MongoDB database if the connection is successful, `null` otherwise.

Example Usage

```
% Connect to the MongoDB database as a 'dev' user
const db = await connect('dev');
% Do something with the db object
...
% Disconnect from the MongoDB database
await disconnect();
```

Async Function Disconnect

Description

The async function `disconnect` disconnects from the MongoDB database by closing the connection.

Returns

- Nothing

Example Usage

```
% Disconnect from the MongoDB database
await disconnect();
```

Must import the Db

2 insert.ts

2.1 insert_one

```
async function insert_one(db: Db, collectionName: string, data: any)
```

This function takes in a database, a collection name, and some data, and it inserts the data into the collection.
Parameters:

- **db** – *Db* – the database object
- **collectionName** – *string* – the name of the collection you want to insert into
- **data** – *any* – the data to be inserted

Returns: The return code of the function.

- **0** – success
- **-1** – invalid data
- **-2** – something is null
- **-3** – duplicate entry
- **-4** – permission denied

2.2 insert_many

`async function insert_many(db: Db, collectionName: string, data: any)`

This function takes in a database, a collection name, and an array of data, and it inserts the data into the database.

Parameters:

- **db** – *Db* – the database object
- **collectionName** – *string* – the name of the collection you want to insert into
- **data** – *any* – the data to be inserted

Returns: The return codes are being returned.

- **0** – success
- **-1** – invalid data
- **-2** – something is null

2.3 Helper Functions

2.3.1 isDrugData

`function isDrugData(data: any): data is DrugData`

This function checks whether the data has the properties of DrugData.

Parameters:

- **data** – *any* – the data that we're checking

Returns: A boolean value.

2.3.2 isPdbNoDups

`function isPdbNoDups(data: any): data is pdb_no_dups`

This function checks whether the data has the properties of pdb_no_dups.

Parameters:

- **data** – *any* – the data that we're checking

Returns: A boolean value.

2.3.3 isPdbSeq

`function isPdbSeq(data: any): data is pdb_seq`

This function checks whether the data has the properties of pdb_seq.

Parameters:

- **data** – *any* – the data that we're checking

Returns: A boolean value.

This function is used to insert multiple documents into the specified collection in a database. It takes in three arguments:

- `db`: a `Db` object representing the database
- `collectionName`: a string representing the name of the collection that the data will be inserted into
- `data`: an array of documents that will be inserted into the collection

The function will iterate over the array of data and insert each document individually. It checks if the data is valid for the collection name provided. If the data is not valid, the function returns -1. If any of the arguments are null, the function returns -2.

The return value of the function is a numeric code indicating the status of the insert operation:

- 0: success
- -1: invalid data
- -2: null argument(s)

If an error occurs during the insertion, the error is logged to the console and the program is exited with an exit code of 1.

3 remove.ts

The following is the manual for the MongoDB module that exports the functions remove_id(), remove_many(), remove_all(), remove_one(), and findremove():

`remove_id(db: Db, collectionName: string, query: any): Promise<number>`

This function removes a single document from a collection in a database that matches the specified query. The return value is the number of documents deleted.

- db (type: Db): The database object.
- collectionName (type: string): The name of the collection to remove the document from.
- query (type: any): The query to find the document to delete.
- Return Value (type: Promise¡number¿): The number of documents deleted.

`remove_many(db: Db, collectionName: string, query: any): Promise<number>`

This function removes all documents from a collection in a database that match the specified query. The return value is the number of documents deleted.

- db (type: Db): The database object.
- collectionName (type: string): The name of the collection to remove the documents from.
- query (type: any): The query to find the documents to delete.
- Return Value (type: Promise¡number¿): The number of documents deleted.

`remove_all(db: Db, collectionName: string): Promise<number>`

This function removes all documents from a collection in a database. The return value is the number of documents deleted.

- db (type: Db): The database object.
- collectionName (type: string): The name of the collection to remove all documents from.
- Return Value (type: Promise¡number¿): The number of documents deleted.

`remove_one(db: Db, collectionName: string, query: any): Promise<number>`

This function removes a single document from a collection in a database that matches the specified query. The return value is the number of documents deleted.

- db (type: Db): The database object.
- collectionName (type: string): The name of the collection to remove the document from.
- query (type: any): The query to find the document to delete.
- Return Value (type: Promise¡number¿): The number of documents deleted.

`findremove(db: Db, collectionName: string, query: any): Promise<any>`

This function finds and removes a single document from a collection in a database that matches the specified query. The return value is the data that was removed from the collection.

- db (type: Db): The database object.
- collectionName (type: string): The name of the collection to query and remove the document from.
- query (type: any): The query to find the document to remove.
- Return Value (type: Promise¡any¿): The data removed from the collection.

4 update.ts

The update.ts module provides four functions for updating documents in a MongoDB database: update_one(), update_many(), update_all(), and replace_one(). Each function takes a database object, the name of the collection to update, a query to find the document(s) to update, an update operation, and optional options.

4.1 update_one()

The update_one() function updates a single document that matches the given query in the specified collection. The function takes the following parameters:

db: The MongoDB database object. collectionName: The name of the collection to update. query: The query to find the document to update. update: The update operation to apply to the matching document. options: Optional update options. Default is an empty object. The function returns the number of updated documents, either 1 or 0.

4.2 update_many()

The update_many() function updates multiple documents that match the given query in the specified collection. The function takes the following parameters: db: The MongoDB database object. collectionName: The name of the collection to update. query: The query to find the documents to update. update: The update operation to apply to the matching documents. options: Optional update options. Default is an empty object. The function returns the number of documents updated.

4.3 update_all()

The update_all() function updates all documents in the specified collection. The function takes the following parameters:

db: The MongoDB database object. collectionName: The name of the collection to update. update: The update operation to apply to all documents in the collection. options: Optional update options. Default is an empty object. The function returns the number of documents updated.

4.4 replace_one()

db: the database object collectionName: The name of the collection to update. query: The query to find the document to update. update: the updated document The number of documents modified.

5 find.ts

The functions have different purposes and parameters:

- Read_Data: reads all documents from a collection and returns an array of objects.
- Read_Data_2: reads all documents from a collection that match a given query and returns an array of objects.
- Read_Data_3: reads all documents from a collection that match a given query and projection and returns an array of objects.
- Read_Data_Col: reads a single column from a collection and returns an array of the values.
- Read_Data_Col_2: reads a single column from a collection that match a given query and returns an array of the values.
- Read_Data_Col_3: reads a single column from a collection that match a given query and projection and returns an array of the values.
- getDistinctValues: gets the distinct values of a field from a collection that match a given query and returns an array of the values.

The functions use the mongodb module and receive a Db object and a collection name as parameters. Some functions receive additional parameters such as a query, projection, or column name.

6 my_interface.ts

6.1 interface DrugData { ... }

This defines a TypeScript interface named DrugData. It has two properties, drugName and condition, both of type string.

6.2 interface pdb_no_dups { ... }

This defines a TypeScript interface named pdb_no_dups. It has several properties, such as structureId, classification, experimentalTechnique, and resolution, all of which have specific data types.

6.3 interface pdb_seq { ... }

This defines a TypeScript interface named pdb_seq. It has several properties, including structureId, chainId, sequence, and residueCount, all of which have specific data types.

6.4 interface DrugData_id { ... }

This defines a TypeScript interface named DrugData_id. It has three properties, _id, drugName, and condition, all of which have specific data types. The _id property is of type ObjectId and is used to uniquely identify documents in the database.

6.5 interface pdb_no_dups_id { ... }

This defines a TypeScript interface named pdb_no_dups_id. It has several properties, including_id, structureId, classification, experimentalTechnique, and resolution, all of which have specific data types. The _id property is of type ObjectId and is used to uniquely identify documents in the database.

6.6 export interface pdb_seq_id { ... }

This defines a TypeScript interface named pdb_seq_id. It has several properties, including _id, structureId, chainId, sequence, and residueCount, all of which have specific data types. The _id property is of type ObjectId and is used to uniquely identify documents in the database.

7 querys.ts

7.1 For Drugsdb

The code contains a set of functions related to a Drugs Database. Below is a manual of each function:

7.1.1 The functions are:

- NameCondition(db: Db, name:string): This function takes in a database object and a drug name as parameters, and returns the condition that the drug treats. It uses the Read_Data_Col_2 function from the find module to read data from the "Drugs" collection.

  Parameters: db (Db): The database object name (string): The name of the drug Return: The condition of the drug

- ConditionNames(db: Db, condition: string): This function takes in a database object and a condition as parameters, and returns an array of drug names that are used to treat the given condition. It also uses the Read_Data_Col_2 function from the find module to read data from the "Drugs" collection.

  Parameters: db (Db): The database object condition (string): The condition Return: An array of drug names

- ConditionNamesPrt(db: Db, condition: string, size: number): This function takes in a database object, a condition, and a size as parameters, and prints out the drug names that are used to treat the given condition in batches of the specified size. It uses the Read_Data_Col_2 function from the find module to read data from the "Drugs" collection.

  Parameters: db (Db): The database object condition (string): The condition size (number): The size of the batch Return: 0

- ConditionNamesAll(db: Db): This function takes in a database object as a parameter, and prints out the distinct values of the condition field in the "Drugs" collection in the database. It uses the getDistinctValues function from the find module to retrieve the distinct values.

  Parameters: db (Db): The database object Return: An array of distinct condition values

- ConditionNamesAllPrt(db: Db, size: number): This function takes in a database object and a size as parameters, and prints out the distinct values of the condition field in the "Drugs" collection in the database in batches of the specified size. It uses the getDistinctValues function from the find module to retrieve the distinct values.

  Parameters: db (Db): The database object size (number): The size of the batch Return: 0

- alternativeName(db: Db, name: string): This function takes in a database object and a drug name as parameters, and returns a list of alternative drug names that treat the same condition as the input drug. It first retrieves the condition of the input drug using the NameCondition function, then retrieves the drug names that treat the same condition using the Read_Data_Col_2 function from the find module.

  Parameters: db (Db): The database object name (string): The name of the drug Return: An array of alternative drug names

- alternativeNamePrt(db: Db, name: string, size: number): This function takes in a database object, a drug name, and a size as parameters, and prints out the alternative drug names that treat the same condition as the input drug in batches of the specified size.

7.2 For protein no dups

This is a collection of functions that extract and print information from a MongoDB database. Each function takes a MongoDB database object and some parameters related to the information to extract, and returns a result.

7.2.1 The functions are:

- pdbName(db: Db, name: string): This function takes a database object and a PDB name and returns the data from the pdb_no_dups collection in the database for the given PDB name.

- classfNames(db: Db, name: string): This function takes a database object and a classification name, finds all the structures with that classification name, and prints them out.

- classfNamesPrt(db: Db, name: string, size: number): This function takes a database object, a classification name, and a size, finds all the structures with that classification name, and prints them out in groups of the given size. It waits for input to continue to the next group of structures.

- classfNamesAll(db: Db): This function takes a database object, finds all the distinct classification names in the database, and prints them out.

- classfNamesAllPrt(db: Db, size: number): This function takes a database object and a size, finds all the distinct classification names in the database, and prints them out in groups of the given size. It waits for input to continue to the next group of structures.

- mmType(db: Db, name: string): This function takes a database object and a macromolecule type name, finds all the structures with that macromolecule type, and prints them out.

- mmTypePrt(db: Db, name: string, size: number): This function takes a database object, a macromolecule type name, and a size, finds the first size structures with that macromolecule type, and prints out their structureIds in groups of the given size. It waits for input to continue to the next group of structures.

- mmTypeAll(db: Db): This function takes a database object, finds all the distinct macromolecule types in the database, and prints them out.

- mmTypeAllPrt(db: Db, size: number): This function takes a database object and a size, finds all the distinct macromolecule types in the database, and prints them out in groups of the given size. It waits for input to continue to the next group of structures.

- residueCount: Takes a database and a name as input, finds all the structureIds that have the name as their residueCount, and returns them.

- residueCountStr: Takes a database, a name, and a size as input, calls Read_Data_Col_2 to retrieve the structureIds for the given name, and prints out the structureIds in groups of the given size.
- id_residueCount: Takes a database and a name as input, calls Read_Data_2 to retrieve the residueCount for the given name, and calls residueCount to find and return the structureIds for that residueCount.
- id_residueCountStr: Takes a database, a name, and a size as input, calls Read_Data_2 to retrieve the residueCount for the given name, and calls residueCountStr to print out the structureIds for that residueCount.
- id_ph: Takes a database and a name as input, calls Read_Data_2 to retrieve the phValue for the given name, and returns it.
- ph_names: Takes a database and a name as input, finds all the structureIds that have the given name in the phValue field, and returns them.
- ph_namesStr: Takes a database, a name, and a size as input, calls Read_Data_Col_2 to retrieve the structureIds for the given phValue, and prints out the structureIds in groups of the given size.
- ph_width: Takes a database, a base pH value, and a width as input, finds all the structures in the database that have a ph value within the width of the base ph value, and returns their structureIds.
- ph_widthStr: Takes a database, a base pH value, a width, and a size as input, calls Read_Data_Col_2 to retrieve the structureIds for the structures with a pH value within the given range, and prints out the structureIds in groups of the given size.
- idResolution: Given a database and a structure ID, returns the resolution of that structure.
- resolution_names: Given a database and a resolution value, returns an array of structure IDs of structures that have that resolution.
- resolution_namesStr: Similar to resolution_names, but prints the structure IDs in groups of a given size, and waits for user input between groups.
- resolution_width: Given a database, a base resolution, and a width, returns an array of structures that have a resolution within the range of base-width to base+width.
- resolution_widthStr: Similar to resolution_width, but prints the structure IDs in groups of a given size, and waits for user input between groups.
- idResolution: Given a database and a structure ID, returns the resolution of that structure.
- resolution_names: Given a database and a resolution value, returns an array of structure IDs of structures that have that resolution.
- resolution_namesStr: Similar to resolution_names, but prints the structure IDs in groups of a given size, and waits for user input between groups.
- resolution_width: Given a database, a base resolution, and a width, returns an array of structures that have a resolution within the range of base-width to base+width.
- resolution_widthStr: Similar to resolution_width, but prints the structure IDs in groups of a given size, and waits for user input between groups.

7.3 For protein seq

This is a set of functions that interact with a database of protein structure data, specifically the pdb_seq collection.

7.3.1 The functions are:

- Seqid. Given a database object db and a PDB file name, it returns the sequence of the protein.
- seq_id. Given a database object db and a sequence name, it prints out the structure IDs of the proteins with that sequence.
- seq_idStr. It is similar to seq_id, but it only prints out the structure IDs in batches of a certain size (specified by the size parameter), prompting the user to continue after each batch.

- mmType2. Given a database object db and a macromolecule type name, it returns the structure IDs of the proteins with that macromolecule type.
- mmType2Str. It is similar to seq_idStr, but it only prints out the structure IDs of proteins with the specified macromolecule type.
- mmType2All. Given a database object db, it prints out the distinct values of the macromolecule type field in the pdb_seq collection.
- mmType2AllStr. It is similar to mmType2All, but it only prints out the distinct values in batches of a certain size, prompting the user to continue after each batch.
- residueCount2. Given a database object db and a residue count name, it returns the structure IDs of the proteins with that residue count.
- residueCountStr. It is similar to seq_idStr and mmType2Str, but it only prints out the structure IDs of proteins with the specified residue count.
- id_residueCount2(db: Db, name: string) Tis asynchronous function takes a database connection object (db) and a string (name) as inputs. It queries the pdb_seq table in the database to retrieve the residueCount value of the protein with the given name. It then passes this value to the residueCount function to calculate the total residue count of the protein. Finally, it returns the calculated residue count value. The function returns a promise that resolves to an integer value representing the residue count of the protein.
- id_residueCountStr2(db: Db, name: string, size: number) This asynchronous function takes a database connection object (db), a string (name), and a number (size) as inputs. It queries the pdb_seq table in the database to retrieve the residueCount value of the protein with the given name. It then passes this value and the size argument to the residueCountStr function to calculate the total residue count of the protein and return a formatted string containing the residue count and size values. Finally, it returns the formatted string. The function returns a promise that resolves to a string value.

7.4 For join the protein no dups and seq

This is a collection of functions that extract and print information from a MongoDB database. Each function takes a MongoDB database object and some parameters related to the information to extract, and returns a result.

7.4.1 The functions are:

joinPro(db: Db, name: string) This asynchronous function takes a database connection object (db) and a string (name) as inputs. It queries the pdb_no_dups and pdb_seq tables in the database to retrieve information related to the protein with the given name. It then returns an array containing two arrays of objects, where the first array contains data from the pdb_no_dups table and the second array contains data from the pdb_seq table. The function returns a promise that resolves to an array value.id_residueCount2(db: Db, name: string)