

CSCI235 Database Systems
Assignment 2
Published on 4 May 2026

Important

It is not allowed to use Generative AI when working on the tasks included in this assignment.

Scope

This assignment includes the tasks related to database transaction processing, design of NoSQL database, data manipulations on NoSQL database, query processing and data transformations on NoSQL database.

The outcomes of this assignment are due on **Friday 29 May, 2026, 8.00 pm (sharp)**.

Please read very carefully information listed below.

This assignment contributes to 15% of the total evaluation in a subject CSCI235.

A submission procedure is explained at the end of specification.

This assignment consists of 5 tasks and specification of each task starts from a new page.

A submission marked by Moodle as "late" is treated as a late submission no matter how many seconds it is late.

A policy regarding late submissions is included in the subject outline.

A submission of compressed files (zipped, gzipped, rared, tared, 7-zipped, lhzed, ... etc) is not allowed. The compressed files will not be evaluated.

All files left on Moodle in a state "Draft (not submitted) " will not be evaluated.

An implementation that does not compile due to one or more syntactical and/or run time errors scores no marks.

It is expected that all tasks included within **Assignment 2** will be solved **individually without any cooperation** with the other students. If you have any doubts, questions, etc. please consult your lecturer or tutor during lab classes or office hours. Plagiarism will result in a **FAIL** grade being recorded for the assessment task.

Task 1 (3 marks)

Processing database transactions at READ COMMITTED isolation level

It is recommended to complete this task in week 9.

Consider the following SQL scripts:

```
/* script1.sql */
UPDATE ORDERS
SET freight = freight + 0.5
WHERE shipped_date < '1997-01-01';
```

```
UPDATE ORDERS
SET freight = freight + 1.5
WHERE shipped_date > '1997-07-01';
COMMIT;
```

```
/* script2.sql */
UPDATE ORDERS
SET freight = freight + 0.5
WHERE shipped_date > '1997-05-01';
```

```
UPDATE ORDERS
SET freight = freight + 1.5
WHERE shipped_date < '1997-04-01';
COMMIT;
```

(1) (1 mark)

Assume that both scripts `script1.sql` and `script2.sql` are concurrently processed at READ COMMITTED isolation level.

Discover a sample concurrent execution of the scripts listed above, such that one of the scripts is automatically aborted and rolled back by a database system.

When visualizing a concurrent execution use a technique of two-dimensional diagrams presented to you during the lecture classes, for example, see a presentation 15 Transaction Processing in PostgreSQL DBMS.

(2) (1 mark)

Explain why the concurrent execution of the scripts proposed in step (1) ends up with an automatic abort and rollback of one of the scripts. Save the outcomes of the steps (1) and (2) in a file `solution1.pdf`.

(3) (1 mark)

To verify the correctness of your solution, use the scripts `orders-create.sql` and `orders-load.sql` to create and to load data into a sample relational table ORDERS. Then, open two psql windows and simulate the concurrent execution of UPDATE statements included in `script1.sql` and `script2.sql` that verifies a

solution proposed in a step (1). When ready Copy/Paste the contents of both psql windows into a text file `solution1.rpt` and save the file.

Deliverables

- (1) A file `solution1.pdf` with:
 - visualization of a sample concurrent execution of the scripts `script1.sql` and `script2.sql` where one of the scripts is automatically aborted and rolled back,
 - the detailed explanations on why one of the scripts is automatically aborted and rolled back during a concurrent execution of both scripts.
 - (2) A file `solution1.rpt` with a report from the simulated concurrent execution of UPDATE statements included in `script1.sql` and `script2.sql` that verifies a solution proposed and explained in a file `solution1.pdf`.
-

Task 2 (4 marks)

Processing database transactions at **READ COMMITTED** isolation level

It is recommended to complete this task in week 9.

A database programmer implemented the following anonymous PL/pgSQL block.

```
DECLARE
    avg_freight ORDERS.freight%TYPE;

BEGIN
    SELECT AVG(freight)
    INTO avg_freight
    FROM ORDERS
    WHERE order_id IN (383,384,385);

    UPDATE ORDERS
    SET freight = (freight + avg_freight)/2
    WHERE order_id = 384;

    COMMIT;
END;
/
```

Running this PL/pgSQL block concurrently at the **READ COMMITTED** isolation level may lead to data corruption if it interacts with another active transaction.

(1) 2 marks

Provide an example when the concurrent processing at isolation level **READ COMMITTED** of PL/pgSQL block listed above together with another active transaction corrupts a database. It is your task to invent another active transaction.

When visualizing the concurrent processing use a technique of two-dimensional diagrams presented to you during the lecture classes, for example, see a presentation 15 Transaction Processing in PostgreSQL DBMS.

(2) 2 marks

Provide the detailed explanations on why in the example proposed in a step (1) the execution of anonymous PL/pgSQL block listed above corrupts a database when it is concurrently processed together with another active transaction at isolation level **READ COMMITTED**.

Deliverables

A file `solution2.pdf` with

- an example where the concurrent processing at isolation level **READ COMMITTED** of anonymous PL/pgSQL block listed above and another transaction corrupts a database,

- the detailed explanations why a sample concurrent processing of database transactions proposed in a step (1) corrupts a database when it is performed at isolation level READ COMMITTED.
-

Preparations

Perform the following actions before solving the problems included in Task 3, Task 4, and Task 5.

Start VirtualBox and import ova image of a Virtual Machine with MongoDB 8.2.2 Community Edition available on Moodle in a section Software Resources.

Start Virtual machine Ubuntu24.04.3-MongoDB8.2.2-07-DEC-2025. All login information is available on Moodle.

Open Terminal window.

To start MongoDB server use
`sudo systemctl start mongod`
in Terminal widow.

To check a status of MongoDB server use
`sudo systemctl status mongod`
in Terminal widow

To stop MongoDB server use
`sudo systemctl stop mongod`
in Terminal widow. Do not stop MongoDB server now.

To start mongosh command line client use
`mongosh`
in Terminal window

Download to your virtual machine a file `orders.zip`. Move the file to the current folder from where you started `mongosh` client.

Unzip a file `orders.zip` to the current folder.

You should get the files `customers.js` and `suppliers.js`.

Start `mongosh` command line client.

To create a collection `orders` and to load the documents into the collection, process the scripts `customers.js` and `suppliers.js` at `>` prompt of `mongosh` client in the following way.

```
load("customers.js");  
load("suppliers.js");
```

You can use the links: [A conceptual schema of a collection orders](#) and [A logical schema of a collection orders](#) to view a conceptual and logical schema of a collection orders.

Next, use the methods

```
db.orders.find().count() and  
db.orders.find()
```

to count the total number of the documents in a collection `orders` and to list all documents.

Next, try few simple queries.

For example, to list the contents of document with an identifier "CACTU" process a method:

```
db.orders.find({"_id":"CACTU"})
```

For example, to list information about the customers and the orders submitted by a customer who has a customer code "FAMIA" process a method:

```
db.orders.find({"CUSTOMER.customer code":"FAMIA"})
```

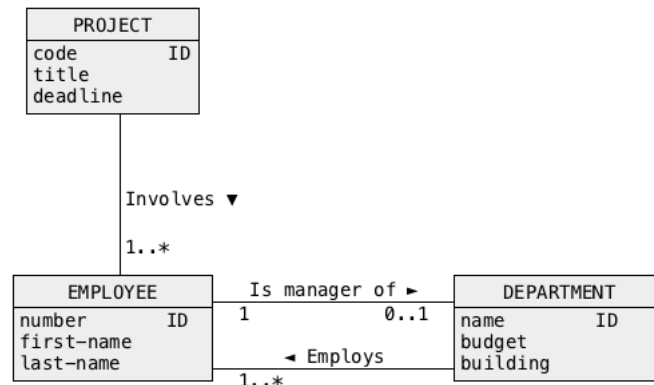
No report is expected from the implementation of actions performed so far.

Task 3 (3 marks)

Logical design and implementation of BSON documents

It is recommended to complete this task in week 10.

Consider the following conceptual schema of a sample database, that contains information about employees working on projects, departments employing employees, and managers of departments.



(1) 2 marks

Transform a conceptual schema given above into a logical schema of BSON documents. To draw a logical schema of BSON documents use a graphical notation presented in the lecture slides 26 BSON Design. You can use UMLet-15.1 with graphical widget for CSCI235 to draw a logical schema. A link to UMLet-15.1 with graphical widgets for CSCI235 is available on Moodle in a section Software Resources. Use CSCI235Palette to draw a logical schema. Save a logical schema in a file solution3.bmp.

An important objective of the design is to maximize the size and complexity of hierarchical structures in a database and in the same moment to eliminate any redundancies from a database.

(2) 1 mark

Next, implement the MongoDB methods that insert into a collection (a name of collection is up to you) the documents that contain information about one project, that involves two employees and about one department that employs the same two employees. Include information about a manager of a department. The documents must contain meaningful data and the types of values associated with the keys must be consistent with the meanings of the keys. Use MongoDB command line client `mongosh` to process the methods in the Terminal window.

When ready create a report from processing of MongoDB methods listed in a step (2). To do so copy the contents of Terminal window and paste it into a file solution3.rpt.

Please note, that a file `solution3.rpt` MUST contain the listings of all processed methods and it MUST contain the listings of all results from the processing. A report that does not contain the listing of the processed methods and results scores no marks.

Deliverables

A file `solution3.bmp` with a logical schema of BSON documents,

A file `solution3.rpt` with a report from processing of MongoDB methods that insert the documents into a collection and list the inserted documents.

Please remember that:

- a report without the listings of processed methods and without the results from processing scores no marks,
 - a report that contains any kind of syntactical and/or processing errors scores no marks.
-

Task 4 (2.5 marks)

Data manipulations

It is recommended to complete this task in week 11.

In this task use a sample database created in the **Preparations** section of this specification.

Use MongoDB methods that perform and data retrievals and data manipulations listed below. Note, that each task may need application of the several methods.

(1) 0.5 mark

Use a `find()` method to display a customer code, city, region, and country of a customer who has a customer code `FISSA`.

Next, change a location of a customer who has a customer code `FISSA` from Madrid to Barcelona and region to Catalonia.

Next, use a `find()` method again to display a customer code, city, region, and country of a customer who has a customer code `FISSA`.

(2) 0.5 mark

Use a `find()` method to display customer code, country, and order ids of all orders submitted by the customers from France.

Next, rename a key `"submits"` to `"submits orders"` for all customers from France.

Next, use a `find()` method again to display customer code, country, and order ids of all orders submitted by the customers from France.

(3) 0.5 mark

Use a `find()` method to display order id and freight of order with order id equal to 356.

Next, decrease a value of freight by 10% in an order with order id 356.

Next, use a `find()` method again to display order id and freight of order with order id equal to 356.

(4) 0.5 mark

Use a `find()` method to display the orders of a customer with a customer code `LAUGB`.

Next, append to a customer with a customer code `LAUGB` the following information about a new order: `"order id" : 999, "freight" : 10.5`.

Next, use a `find()` method again to display the orders of a customer with a customer code LAUGB.

(5) 0.5 mark

Use a `find()` method to display a complete description of a customer who submitted an order with an order id 333.

Next, remove information about a contact title from the description of a customer who submitted an order with an order id 333.

Next, use a `find()` method again to display a complete description of a customer who submitted an order with an order id 333.

When ready create a report from processing of MongoDB methods listed in the steps (1) to (5). To do so copy the contents of Terminal window and paste it into a text file `solution4.rpt`.

Please note, that a file `solution4.rpt` MUST contain the listings of all processed methods and it MUST contain the listings of all results from the processing. A report that does not contain the listing of the processed methods and/or results scores no marks.

Deliverables

A file `solution4.rpt` with a report from processing of MongoDB methods that perform the data manipulations and data retrievals listed above.

Please remember that:

- a report without the listings of processed methods and without the results from processing scores no marks,
 - a report that contains any kind of syntactical and/or processing errors scores no marks.
-

Task 5 (2.5 marks)

Query processing and data transformation with aggregation framework

It is recommended to complete this task in week 12.

Before implementation of task 5 you MUST refresh a sample database used in task 4. To do so drop a collection `orders` in the following way.

```
db.orders.drop();
```

Next, re-create a collection `orders`. To re-create collection `orders` and to load the documents into the collection, process the scripts `customers.js` and `suppliers.js` at `>` prompt of `mongosh` client in the following way.

```
load("customers.js");  
load("suppliers.js");
```

No report is expected from the implementation of actions performed so far.

Use MongoDB aggregation framework to implement the queries and data transformations listed below. Note, that some of the problems listed below may need the applications of several methods.

(1) 0.5 mark

Find the total number of orders shipped to the countries `France` or `Brazil`. Display a result in a format `{"total_orders":integer-value}`.

(2) 0.5 mark

Find all orders shipped via `United Package`. Save the results into a collection `shippedvia` that consists of the documents like `{"order_id": 111, "employee_id": 1}`.

Next, display in a format without document identifiers all documents in a collection `shippedvia`.

(3) 0.5 mark

Find the total number of orders per each city where the customers are located at. List the results in a format `{"total_orders":integer-value, "city":city-name}`. List only the cities with the total number of orders greater than 3.

(4) 0.5 mark

Save information about a customer code, city, and country of all customers from `France` or `Brazil` into a collection `FraBra`.

Next, display in a format without document identifiers the contents of a collection `FraBra`.

(5) 0.5 mark

Find 5 largest values of freight from all orders. List the results in a format `{"order id": integer-value, "freight":integer-value}`.

When ready create a report from processing of MongoDB methods listed in the steps from (1) to (5). To do so copy the contents of Terminal window and paste it into a file `solution5.rpt`.

Please note, that a file `solution5.rpt` MUST contain the listings of all processed methods and it MUST contain the listings of all results from the processing. A report that does not contain the listing of the processed methods and results scores no marks.

Deliverables

A file `solution5.rpt` with a report from processing of MongoDB methods that perform the data retrievals and/or data transformations.

Please remember that:

- a report without the listings of processed methods and without the results from processing scores no marks,
 - a report that contains any kind of syntactical and/or processing errors scores no marks.
-

Submission

Submit the files **solution1.pdf**, **solution1.rpt**, **solution2.pdf**, **solution3.bmp**, **solution3.rpt**, **solution4.rpt** and **solution5.rpt** through Moodle in the following way:

- (1) Access Moodle at **<http://moodle.uowplatform.edu.au/>**
- (2) To login use a **Login** link located in the right upper corner the Web page or in the middle of the bottom of the Web page
- (3) When logged select a site **CSCI235 (S126) Database Systems**
- (4) Scroll down to a section **Assignment 2**
- (5) Click at a link **In this place you can submit the outcomes of Assignment 2**
- (6) Click at a button **Add Submission**
- (7) Move a file **solution1.pdf** into an area **You can drag and drop files here to add them**. You can also use a link **Add...**
- (8) Repeat a step (7) for the files **solution1.rpt**, **solution2.pdf**, **solution3.bmp**, **solution3.rpt**, **solution4.rpt** and **solution5.rpt**.
- (9) Click at a button **Save changes**
- (10) Click at a button **Submit assignment**
- (11) Click at the checkbox with a text attached: **By checking this box, I confirm that this submission is my own work** to confirm the authorship of your submission.
- (12) Click at a button **Continue**

End of specification