

CSCI235 Database Systems
Assignment 1
Published on 2 March 2026

Important

The solutions of all tasks included in Assignment 1 must be your original work. The use of Generative AI technology is not allowed.

Scope

This assignment includes the tasks related to database normalization, indexing of relational tables and programming of a relational database server.

The outcomes of this assignment are due on **Friday 1 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.

Each task has its recommended completion week.

It is also recommended to solve the problems before attending the laboratory classes in order to efficiently use supervised laboratory time.

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 1** will be solved **individually without any cooperation** with the other students. If you have any doubts, questions, etc.

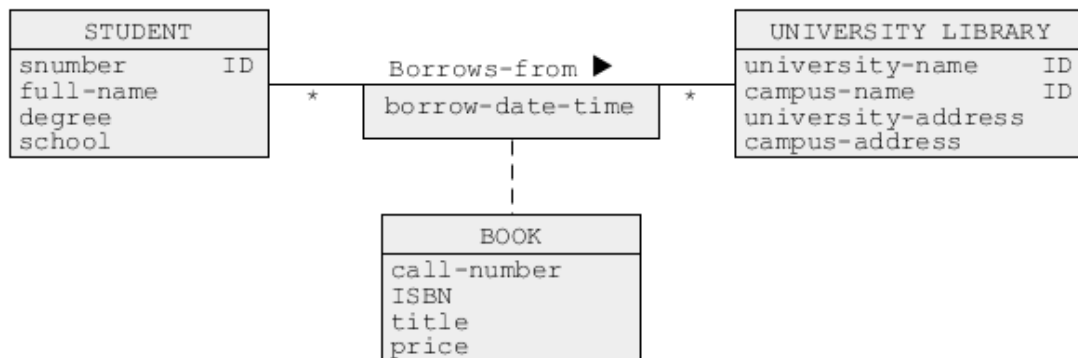
please consult your lecturer or tutor during lab classes or office hours. Plagiarism will result in a **FAIL** grade being recorded for the assessment task.

Task 1 (3 marks)

Normalization of relational schemas

It is recommended to complete this task in week 5.

A database designer created the following conceptual schema.



Next, a database designer applied a step of logical design and converted a conceptual schema given above into the following relational schemas.

STUDENT(snumber, full-name, degree, school)
PK = (snumber)

UNIVERSITY-LIBRARY(university-name, campus-name,
University-address, campus-address)
PK = (university-name, campus-name)

BORROWSFROM(snumber, university-name, campus-name,
borrow-date-time, call-number, ISBN, title, price)
PK = (snumber, university-name, campus-name,
borrow-date-time)
FK1 = (snumber) references STUDENT(snumber)
FK2 = (university-name, campus-name) references
UNIVERSITY-LIBRARY(university-name, campus-name)

(1) 0.5 mark

For each one of the relational schemas obtained in a step (1) find all functional dependencies valid in each schema.

(2) 0.5 mark

For each one of the relational schemas obtained from a step (1) and functional dependencies obtained from a step (2) find the minimal keys.

(3) 1 mark

Apply the definitions of normal forms, functional dependencies and minimal keys to find the highest normal form valid for each one of the relational schemas obtained from a step (1).

(4) 1 mark

Decompose all relational schemas not in BCNF into the relational schemas in BCNF.

Deliverables

A file `solution1.pdf` with a report from the outcomes of normalization of relational schemas. A report must include a list of functional dependencies found, complete derivations of minimal keys, complete identifications of the highest normal form valid in each schema and decompositions into BCNF whenever it is necessary.

Task 2 (3 marks)

Indexing of relational tables

It is recommended to complete this task in week 6.

Prologue

Download the files `dbschema.bmp`, `dbcreate.sql`, `dbload.sql`, `dbcount.sql`, `dbload.sql`, `dbcount.sql` and `dbdrop.sql` included in a section Assignment 1 on Moodle. A script `dbdrop.sql` can be used to drop a sample database. A script `dbcreate.sql` can be used to create a sample database. A script `dbload.sql` can be used to load a sample database. A script `dbcount.sql` can be used to count the total number of rows in each table.

Open a file `dbschema.bmp` and familiarize yourself with a conceptual schema of sample database.

To perform the actions listed below it is recommended to use `psql` client available within pgAdmin 4.

Connect to one of PostgreSQL database server and process a script `dbcreate.sql` to create the relational tables. Next, process a script `dbload.sql` to load data into the relational tables. Then, you can use a script `dbcount.sql` to familiarize yourself with the names of relational tables and the total number of rows in each table. When a sample database is created and loaded a script `dbcount.sql` returns the following results.

```
bank | branch | employee | account | transaction
-----+-----+-----+-----+-----
    21 |      41 |      101 |      43 |          88
(1 row)
```

While connected to the same PostgreSQL database server, process the following SQL statement to list a query processing plan for a given `SELECT` statement.

```
EXPLAIN
SELECT bank_name, hq_city
FROM BANK
WHERE hq_city = 'Sydney';
```

You should get the following results (if you used `psql`).

```
              QUERY PLAN
-----
Seq Scan on bank  (cost=0.00..1.26 rows=1 width=196)
  Filter: ((hq_city)::text = 'Sydney'::text)
(2 rows)
```

A line **Seq Scan on bank** included in a query processing plan above indicates that a database system plans to sequentially scan (read data block by data block) entire relational table APPLICANT to process the query.

Next, use CREATE INDEX statement listed below to create an index on a column city in a relational table APPLICANT.

```
CREATE INDEX BANK_IDX ON BANK(hq_city);
```

Next, turn off sequential scan from query processing plans. We have to do it because a sample database is very small and a query optimizer will always use sequential scan instead of an index. To turn off sequential scan process the following statement.

```
SET enable_seqscan=OFF;
```

Again, process the following SQL statement to list a query processing plan for the same SELECT statement as before.

```
EXPLAIN
SELECT bank_name, hq_city
FROM BANK
WHERE hq_city = 'Sydney';
```

You should get the following results (if you used psql).

```

              QUERY PLAN
-----
Index Scan using bank_idx on bank  (cost=0.14..8.15 rows=1 width=196)
    Index Cond: ((hq_city)::text = 'Sydney'::text)
(2 rows)
```

This time a database system plans to use an index BANK_IDX created a moment ago to process the same query. Note, a line **Index Scan using bank_idx on bank** in a query processing plan given above. It means that a database system plans to vertically traverse an index BANK_IDX to find the identifiers of row where value of attribute hq_city = 'Sydney'.

Conclusions

EXPLAIN statement of SQL can be used to get information about a processing plan created by a query processor for a given SELECT statement. A query processing plan provides information on whether and index created earlier will be used for processing of SQL statement. We can use EXPLAIN statement to check whether an index created to speed up SELECT statement will be used for processing of the statement.

To drop an index, process a statement:

```
DROP INDEX BANK_IDX;
```

As a simple exercise, repeat the same sequence of actions for the following EXPLAIN statement.

```
EXPLAIN
SELECT position, COUNT(*)
FROM EMPLOYEE
GROUP BY position;
```

Next create an index with the following statement.

```
CREATE INDEX EMPLOYEE_IDX ON EMPLOYEE(position);
```

And, then process the same EXPLAIN statement again.

```
EXPLAIN
SELECT position, COUNT(*)
FROM EMPLOYEE
GROUP BY position;
```

Compare the query processing plans obtained without and with an index.

To drop the indexes process the following statements.

```
DROP INDEX BANK_IDX;
DROP INDEX EMPLOYEE_IDX;
```

No report is expected from processing of SQL statements given above.

Problem

Your task is to find what indexes should be created to speed up processing of SELECT statements listed below. You are expected to create one index for one SELECT statement. To simplify the problem, assume that any index used by a query optimizer to speed up processing of SELECT statement, will do. It simply means that you do not need to find the best index.

(1) 0.5 mark

```
SELECT position, br_country, br_city
FROM EMPLOYEE;
```

(2) 0.5 mark

```
SELECT *
FROM BANK
WHERE hq_country = 'Australia' AND
      hq_city= 'Newcastle';
```

(3) 0.5 mark

```
SELECT bank_name, account_type, avg(balance)
FROM ACCOUNT
GROUP BY bank_name, account_type;
```

(4) 1 mark

```
SELECT bank_name
FROM BANK
WHERE bank_name LIKE 'Bank of%' OR hq_country = 'USA';
```

(5) 0.5 mark

```
SELECT account_num, balance, bank_name
FROM ACCOUNT
WHERE balance > 0.0;
```

Implement SQL script `solution2.sql`, such that for each one of SELECT statements given above the script:

- creates an index.
- finds and lists a query processing plan for SELECT statement with an index,
- drops an index.

When ready process SQL script file `solution2.sql` and save a report from processing in a file `solution2.rpt`.

Your report must include a listing of all SQL statements processed and all feedback from processing of SQL statements.

To achieve that, put the statements

```
\set ECHO all
SET enable_seqscan=OFF;
```

at the beginning of your script `solution2.sql`.

When ready start command line tool `psql` and process SQL script file `solution2.sql` in `psql` window. Then, save a report from processing in a text file `solution2.rpt`. To save a report select the entire contents of `psql` window and Copy&Paste it into a file `solution2.rpt`. Save a file `solution2.rpt`.

Your report must include a listing of all SQL statements processed and all feedback from processing of SQL statements. No marks will be granted for the reports without listing of SQL statements processed and/or without feedback from processing of SQL statements.

Make sure that your report contains no errors. No marks will be granted for a report with errors.

Deliverables

A file `solution2.rpt` with a report from processing of a script file `solution2.sql` that lists query processing plans after indexing. A report must have no errors and it must list all SQL statements processed and it must list all feedback from processing of SQL statements.

Task 3 (3 marks)

Extending database sever functionality with a stored function

It is recommended to complete this task during a laboratory class in week 7.

This task applies to a sample database created in Task 2.

Download a template for SQL script available in a file `solution3.sql` and insert into the template the implementations of actions listed below. All implemented actions must directly follow a respective comment with a specification of an action.

(1) 0.5 mark

First, the script modifies the structures of a sample database such it would be possible to store information about the total number of employees located in each country. The best design is expected in this step. Remember to enforce the appropriate consistency constraints.

(2) 0.5 mark

Next, the script modifies the contents of a sample database with information about the total employees located in each country. Consider only the countries where at least one bank branch is located. List the names of countries together with the total number of employees located in each country.

(3) 1.5 mark

Implement a stored function that returns the total number of employees working in all branches located in a given country. Pass a name of country through an input parameter of a stored function.

(4) 0.5 mark

Remove information about two employees from a sample database and update location of one employee. Use a stored function implemented in the previous statement in an `UPDATE` statement that modifies information about the total number of employees located at each country. Display information about the total number of employees located in each country.

You must use a template provided in a file `solution3.sql`. One mark (1.0) will be deducted for not using the template.

When ready start command line tool `psql` and process SQL script file `solution3.sql` in `psql` window. Then, save a report from processing in a text file `solution3.rpt`. To save a report select the entire contents of `psql` window and Copy&Paste it into a file `solution3.rpt`. Save a file `solution3.rpt`.

Your report must include a listing of all SQL statements processed and all feedback from processing of SQL statements. No marks will be granted for the reports without listing of SQL statements processed and/or without feedback from processing of SQL statements.

Make sure that your report contains no errors. No marks will be granted for a report with errors.

Deliverables

A file `solution3.rpt` with a report from processing of a script `solution3.sql`. A report must have no errors and it must list all PL/pgSQL and SQL statements processed.

Task 4 (3 marks)

Implementation of automatic modification of derived data with a row trigger

It is recommended to complete this task during a laboratory class in week 8.

This task applies to a sample database created in Task 2.

Download a template for SQL script available in a file `solution4.sql` and insert into the template the implementations of actions listed below. All implemented actions must directly follow a respective comment with a specification of an action.

(1) 0.5 mark

First, the script modifies the structures of a sample database such it would be possible to store information about the total number of employees in each bank. Remember to enforce the appropriate consistency constraints.

(2) 0.5 mark

Next, the script modifies the contents of a sample data with information about the total number of all employees in each bank.

(3) 0.5 mark

Next, the script lists information about the total number of all employees in each bank.

(4) 1 mark

Next, the script creates a **row trigger**, that automatically updates information about the total number of all employees in each bank in each bank whenever an employee leaves a bank.

(5) 0.5 mark

Next, the script comprehensively tests the trigger. A test must include at least one case when an employee leaves a bank and at least one case when all employees leave a bank. Then, the script must list information about the total number of employees in each bank.

You must use a template provided in a file `solution4.sql`. One mark (1.0) will be deducted for not using the template.

When ready start command line tool `psql` and process SQL script file `solution4.sql` in `psql` window. Then, save a report from processing in a text file `solution4.rpt`. To save a report select the entire contents of `psql` window and Copy&Paste it into a file `solution4.rpt`. Save a file `solution4.rpt`.

Your report must include a listing of all SQL statements processed and all feedback from processing of SQL statements. No marks will be granted for the reports without listing of SQL statements processed and/or without feedback from processing of SQL statements.

Make sure that your report contains no errors. No marks will be granted for a report with errors.

Deliverables

A file `solution4.rpt` with a report from processing of a script `solution4.sql`. A report must have no errors and it must list all PL/pgSQL and SQL statements processed.

Task 5 (3 marks)

Implementation of automatic modification of derived data with a statement trigger

It is recommended to complete this task during a laboratory class in week 8.

Download a template for SQL script available in a file `solution5.sql` and insert into the template the implementations of actions listed below. All implemented actions must directly follow a respective comment with a specification of an action.

(1) 0.5 mark

First, the script modifies the structures of a sample database such it would be possible to store information about the total number of employees in each bank. Remember to enforce the appropriate consistency constraints.

(2) 0.5 mark

Next, the script modifies the contents of a sample database with information about the total number of all employees in each bank.

(3) 0.5 mark

Next, this script list information about the total number of all employees in each bank.

(4) 1 mark

Next, the script creates a **statement trigger**, that automatically updates information about the total number of all employees in each bank in each bank whenever a new employee is employed by a bank.

(5) 0.5 mark

Next, the script comprehensively tests the trigger. A test must include at least one case when a new employee is employed by a bank. Then, the script must list information about the total number of employees in each bank.

You must use a template provided in a file `solution5.sql`. One mark (1.0) will be deducted for not using the template.

When ready start command line tool `psql` and process SQL script file `solution5.sql` in `psql` window. Then, save a report from processing in a text file `solution5.rpt`. To save a report select the entire contents of `psql` window and Copy&Paste it into a file `solution5.rpt`. Save a file `solution5.rpt`.

Your report must include a listing of all SQL statements processed and all feedback from processing of SQL statements. No marks will be granted for the reports without listing of SQL statements processed and/or without feedback from processing of SQL statements.

Make sure that your report contains no errors. No marks will be granted for a report with errors.

Deliverables

A file `solution5.rpt` with a report from processing of a script `solution5.sql`. A report must have no errors and it must list all PL/pgSQL and SQL statements processed.

Submission

Submit the files **solution1.pdf**, **solution2.rpt**, **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 1**
- (5) Click at a link **In this place you can submit the outcomes of Assignment 1**
- (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 **solution2.rpt**, **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, ...** in order to confirm the authorship of your submission.
- (12) Click at a button **Continue**

End of specification